SEPRAN

SEPRA ANALYSIS

STANDARD PROBLEMS

GUUS SEGAL

SEPRAN STANDARD PROBLEMS

January 2015

Ingenieursbureau SEPRA
Park Nabij 3
2491 EG Den Haag
The Netherlands
Tel. 31 - 70 3871309

Contents

1 Introduction

In this manual it is described, how the standard problems available in SEPRAN must be solved, i.e. which standard elements are present and what their standard problem definition numbers are. Furthermore, the available types of boundary conditions are given and for linear problems, it is indicated whether the large matrix is symmetrical and/or positive definite. The manual contains a very limited number of examples. For a more extended list the reader is referred to the manual SEPRAN EXAMPLES.

For non-linear problems a description of the iteration process is given.
Section 1.1 Gives some general remarks concerning this manual. elements.
Chapter 2 treats some simple diffusion-like equations.
Chapter 3 treats general second order elliptic equations, both stationary and time-dependent. Typical representations are the Laplacian (Poisson) equation, the convection-diffusion equation, the Helmholtz equation and the heat equation.
Elements concerning lubrication are treated in Chapter 4.
Chapter 5 is devoted to mechanical elements.
The available elements for solidification problems are given in Chapter 6.
Chapter 7 treats flow problems governed by the incompressible Navier-Stokes equations.
Spectral elements are the subject of Chapter 8.
In Chapter 10 it is described which methods are available for providing coefficients and other necessary information to the SEPRAN standard elements.

1.1 General remarks concerning this manual

The description of the available standard problems is given in the next chapters. These chapters are subdivided into paragraphs corresponding to one type of equation. Each paragraph contains the following items:

Equation Under this heading the (partial) differential equation is given.

Boundary conditions The types of boundary conditions for which standard elements are available are given.

Remarks concerning the solution of the equation treats special information concerning the solution if needed. For example upwind in case of convection, treatment of non-linearity, or penalty function formulation in case of Navier-Stokes are treated in this part.

Coefficients for the differential equation describes the various coefficients that are available for this type of equation, including information about coordinate system and numerical quadrature rule to be applied.
Coefficients may be provided either in a separate coefficients block or in the structure block. If it is found in a coefficients block corresponding to the equation to be solved, the corresponding value is used. If a specific coefficient is not available it is checked if there is a constant, variable or vector defined with that specific name. That value is used.
Note that in the last case the name must be precisely the coefficients name. However, the input is case independent, so it does not matter whether you use capitals or not.

We distinguish between integer coefficients and real coefficients. The first one is usually meant to make a choice between several options, for example what kind of coordinate system is used. These integer coefficients are either in the form:
name_coefficient = value (integer)
or
name_coefficient = string.
For example

```
integration_rule = 3
coordinate_system = cartesian
```

If the value is a string it must be put within quotes when used in a structure block. In a coefficients block, however, these quotes must be neglected. So the example above is meant for a coefficients block, whereas in the structure block it would be:

```
integration_rule = 3
coordinate_system = 'cartesian'
```

In the next chapters each coefficient is followed by eiher (icoef j) or (coef k). This refers to the internal sequence number of the coefficient and whether it is an integer coefficient (icoef) or a real one (coef).
Furthermore string names like `cartesian` are followed by an integer number between brackets indicating the internal value.
This information is needed for the case that the user provides the coefficients in a user written program or uses an old sepran program.

Coefficients for the natural boundary conditions has the same meaning and syntax as for the differential equation.

Type numbers to be used in the problem input block indicates the name that is used to identify the problem and it also gives the internally used type number which can be used alternatively or in user written programs.

In general for natural boundary conditions it is sufficient to give the boundary elements, but one may also define the type numbers explicitly in the subblock natbouncond. This type number is also given.

Derivatives For this equation one can compute various derivatives or derived quantities by

```
vector = derivatives ( input_vector, icheld=i, options )
```

with `input_vector` the vector from which the derivative must be computed and `icheld` $= i$ a sequence number indicating what type of derivative must be computed.
Alternatively sometimes one might use

```
vector = name_action(input_vector)
```

where `name_action` the name of the specific action is, for example `gradient`.

Solution method In this part it is described how the problem must be solved. For example, which iteration process should be used, etc.

Output In this part the output of some subroutines with correspondence to this standard problem is given. For example, this can be the velocity, the pressure or the stress tensor.

Numbering of unknowns The unknowns in a nodal point are numbered according to a fixed sequence. Under this heading it is given what number corresponds to which unknown.

Element types and problem definition numbers Under this heading the available element shapes are treated and their corresponding type numbers with respect to the input block "PROBLEM". The available element shapes are indicated by a sequence number, which refers to the shape number to be used in the mesh generator. See the input of the mesh generator (Manual SEPRAN INTRODUCTION or SEPRAN USERS MANUAL) for a definition of all the available shape numbers.

With respect to the use of the standard elements described in this manual the following points are of importance:

2 Some simple diffusion-like equations

This Chapter describes a set of very simple equations. All these equation can be considered as a special case of the general second order elliptic equation treated in Section 3.1. The following equations are available:

Laplace See 2.1

Poisson See 2.2

Diffusion See 2.3

convection-diffusion See 2.4

2.1 Laplace equation

This is the most simple of all equations to be solved.

2.1.1 Equation

$$\rho c_p \frac{\partial \phi}{\partial t} - \Delta \phi \;=\; 0. \tag{2.1.1}$$

or written in terms of derivatives:

$$\rho c_p \frac{\partial c}{\partial t} - \sum_{i=1}^{n} \frac{\partial^2 \phi}{\partial x_i^2} \;=\; 0, \tag{2.1.2}$$

with n the dimension of the space R^n (1, 2 or 3) and x_i the i^{th} coordinate direction.
In the stationary case the time derivative disappears.
The minus sign is just to make the resulting system of equations positive, but does not change a thing in the equation.
The matrix corresponding to this equation is symmetric and, except in the case of Neumann boundary conditions on each boundary, also positive definite.

2.1.2 Boundary Conditions

For this equation there are two types of boundary conditions available.

Type 1 (Dirichlet boundary condition) $\phi(\mathbf{x})$ given on some part of the boundary.
This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2 (Neumann or mixed boundary condition)

$$\frac{\partial \phi}{\partial \mathbf{n}} + \sigma \phi = h \qquad (\sigma(x) \geq 0), \tag{2.1.1}$$

on some part of the boundary. $\mathbf{n}$ is the outward normal at the boundary and h a given source term. Hence the first term is the normal derivative at the boundary.
This is a so-called natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma = 0$ and $h = 0$, when there is no need to give any condition on this part of the boundary. For $\sigma = 0$ this is a Neumann boundary condition, otherwise it is a mixed boundary condition.
σ and h may be function of space.

2.1.3 Coefficients for the differential equation

The differential equation itself does not require any coefficients, however, the user may provide the following information either in a coefficients block or in the structure block.

INTEGRATION_RULE $= i$ (icoef3) defines the type of integration rule to be applied.
This is an integer coefficient, with the following possible values for i:

- 0 the rule is chosen by the element itself (Default)
- > 0 the integration rule is defined by the user, see below

COORDINATE_SYSTEM $=$ *name* (icoef4) Type of co-ordinate system.
name is a string parameter with the following possible values:

CARTESIAN (0) Cartesian co-ordinates (x, y, z) (Default)

AXI_SYMMETRIC (1) Axisymmetric co-ordinates (2D grids only) (r, z)

POLAR (2) Polar co-ordinates (1D grids only) (r)

DENSITY $= \rho$ (coef6) density.
Default value: 1

HEAT_CAPACITY $= cp$ (coef7) heat capacity.
Default value: 1

2.1.4 Coefficients for the natural boundary conditions

The non-homogeneous natural boundary conditions require extra input for the coefficients. First of all the same coefficients `integration_rule` and `coordinate_system` as for the differential equation may be used. Besides that we need to prescribe σ and h:

diff_sigma $= \sigma$ (coef6) defines the value of the coefficient σ.

diff_flux $= h$ (coef7) defines the value of the coefficient h.

2.1.5 Type numbers to be used in the problem input block

The use of the Laplace equation is indicated in the problem block by the name

```
laplace
```

or alternatively by

```
type = 700
```

The natural boundary conditions do not have to be indicated by a type number, but internally type 801 is used.

2.1.6 Derivatives

The following types of derivatives may be computed:

ICHELD = 1 $\frac{\partial c}{\partial x_i}$, where x_i is defined by the parameter ix

ICHELD = 2 ∇c
Instead one can also use `vector = gradient(input_vector)` in the structure block.

ICHELD = 3 -∇c
Alternatively one can also use `vector = flux(input_vector)`

2.1.7 Extra information

For all other information like vectors of special structure and integrals the reader is referred to Section 3.1.

2.2 Poisson equation

The Poisson equation is in fact the Laplace equation with a non-zero right-hand side.

2.2.1 Equation

$$\rho c_p \frac{\partial \phi}{\partial t} - \Delta \phi \;=\; f. \tag{2.2.1}$$

2.2.2 Boundary Conditions

See Section 2.2.1

2.2.3 Coefficients for the differential equation

Besides the coefficients mentioned in Section 2.2.1, there is one extra term to prescribe the right-hand side f.

SOURCE $= f$ (coef6) defines the right-hand side. The coefficients density and heat capacity refer to coef7 and coef8 respectively.

2.2.4 Coefficients for the natural boundary conditions

See Section 2.2.4

2.2.5 Type numbers to be used in the problem input block

The use of the Poisson equation is indicated in the problem block by the name

```
poisson
```

or alternatively by

```
type = 701
```

The natural boundary conditions do not have to be indicated by a type number, but internally type 801 is used.

2.2.6 Derivatives

See Section 2.2.6

2.3 Diffusion equation

The diffusion equation is in fact the natural extension of the Poisson equation. The only difference is that it has a diffusion parameter that is not identical to one.

2.3.1 Equation

$$\rho c_p \frac{\partial c}{\partial t} - div(\boldsymbol{A}\nabla c) \;=\; f, \tag{2.3.1}$$

where $\boldsymbol{A}$ may be a scalar, a diagonal matrix or a general symmetric $n \times n$ matrix where n is the dimension of the space (R^n).
Written in terms of derivatives:

$$\rho c_p \frac{\partial c}{\partial t} - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_i}\left(A_{ij}\frac{\partial c}{\partial x_j}\right) \;=\; f. \tag{2.3.2}$$

The matrix corresponding to this equation is symmetric and, except in the case of Neumann boundary conditions on each boundary, also positive definite.

2.3.2 Boundary Conditions

With respect to the Dirichlet boundary conditions the same remarks as in Section 2.1.2 are valid.
The mixed boundary condition in this case reads

$$\sum_{i=1}^{n}((\sum_{j=1}^{n} A_{ij}(x)\frac{\partial c}{\partial x_j})\; n_i) + \sigma(x)c(x) = h(x) \qquad (\sigma(x) \geq 0), \tag{2.3.1}$$

with n_i the component of the normal $\boldsymbol{n}$ in x_i direction.
Although Equation (2.3.1) seems more complicated than Equation (2.1.1) it is in fact the same expression and therefore everything mentioned in Section 2.1.2 is still applicable.

2.3.3 Coefficients for the differential equation

Besides the coefficients mentioned in Section 2.2.3, we need to prescribe the diffusion. If the diffusion $\boldsymbol{A}$ is a scalar we need to give one parameter, if it is a diagonal matrix we have to give n parameters and in the general case 3 (n=2) or 6 (n=3).

diffusion = α (coef6/9/11) defines the diffusion as a scalar.

If the diffusion $\boldsymbol{A}$ is a dagonal matrix we use:

diff_x = α_x (coef6) defines the diffusion in x-direction in case of a diagonal matrix.

diff_y = α_y (coefy) defines the diffusion in y-direction.

diff_z = α_z (coefz) defines the diffusion in z-direction.

In the general case the diffusion coefficients are given in the following way:

diff_xx = A_{11} (coef6).

diff_xy = A_{12} (coef7).

diff_xz = A_{13} (coef8).

diff_yy = A_{22} (coef9).

diff_yz = A_{23} (coef10).

diff_zz = A_{33} (coef11).
All coefficients not given are zero.
The sequence numbers of the other coefficients is:

SOURCE (coef12).

DENSITY (coef13).

HEAT_CAPACITY (coef14).

2.3.4 Coefficients for the natural boundary conditions

See Section 2.1.4.

2.3.5 Type numbers to be used in the problem input block

The use of the Diffusion equation is indicated in the problem block by the name

```
diffusion
```

or alternatively by

```
type = 702
```

The natural boundary conditions do not have to be indicated by a type number, but internally type 801 is used.

2.3.6 Derivatives

ICHELD = 1 $\frac{\partial c}{\partial x_i}$, where x_i is defined by the parameter ix

ICHELD = 2 ∇c
Instead of derivatives one can also use `vector = gradient(input_vector)` in the structure block.

ICHELD = 3 -∇c

ICHELD = 6 $-A\nabla c$, where $\boldsymbol{A}$ is the matrix with coefficients A_{ij}.
`vector = flux(input_vector)` has the same effect.

2.4 Convection-diffusion equation

The convection-diffusion equation is in fact the extension of the diffusion equation with a convection term.

2.4.1 Equation

$$\rho c_p \frac{\partial c}{\partial t} - div(\boldsymbol{A}\nabla c) + \rho c_p \boldsymbol{u} \cdot \nabla c \;=\; f, \tag{2.4.1}$$

with, $\boldsymbol{u}$, a given velocity, rho, the density, c_p, the heat capacity at constant pressure, and all other terms the same as in Section 2.3.1.
Written in terms of derivatives:

$$\rho c_p \frac{\partial c}{\partial t} - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_i}\left(A_{ij}\frac{\partial c}{\partial x_j}\right) + \rho c_p \sum_{i=1}^{n} u_i \frac{\partial c}{\partial x_i} \;=\; f. \tag{2.4.2}$$

The matrix corresponding to this equation is asymmetric except in the case that the velocity is zero everywhere.

2.4.2 Boundary Conditions

See Section 2.3.2

2.4.3 Remarks concerning the solution of the equation (upwind)

If in equation (2.4.2) the convective part $\sum_{i=1}^{n} u_i \frac{\partial c}{\partial x_i}$ dominates the diffusive part $-\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_i}\left(\alpha_{ij}\frac{\partial c}{\partial x_i}\right)$, an improvement of the accuracy may be possible by applying a so-called upwind technique. However, it must be remarked that upwinding not always improves the accuracy and, moreover, in all cases the building of matrices in case of upwinding is more expensive than in the standard case.

The upwinding in SEPRAN is realized by the so-called streamline upwind Petrov-Galerkin method (SUPG), see Brooks and Hughes (1982).

Essential in this method is that next to the standard Galerkin equation an extra term of the following type is added:

$$\int_e (Dc - f)p d\Omega$$

where e is the element. Dc represents the differential equation applied to c and f is the right-hand side. The upwind parameter p is defined by

$$p_i = \frac{h\xi}{2}\frac{\mathbf{u}\cdot\nabla\phi_i}{\parallel \mathbf{u} \parallel}$$

with

h the width of the element in the direction of the flow,

ϕ_i the i^{th} basis function,

$\mathbf{u}$ the velocity,

ξ a choice parameter defining the type of upwinding.

For the definition of ξ we introduce the parameters ϵ, β by $\epsilon = \mathbf{u}^T \alpha \mathbf{u}$, where α is the matrix with elements α_{ij}, and $\beta = \frac{\|\mathbf{u}\| h}{2\epsilon}$.

The following choices of ξ have been programmed:

1 Classical upwind scheme: $\xi = 1$

2 Il'in scheme: $\xi = \coth(\beta) - \frac{1}{\beta}$

3 Doubly asymptotic approximation: $\xi = \begin{cases} \beta/3 & -3 \le \beta \le 3 \\ sign(\beta) & \mid \beta \mid > 3 \end{cases}$

4 Critical approximation: $\xi = \begin{cases} -1 - 1/\beta & \beta < 1 \\ 0 & -1 \le \beta \le 1 \\ 1 - 1/\beta & 1 \le \beta \end{cases}$

5 Hughes approximation: $\xi = \sqrt{\frac{\beta^2}{9+\beta^2}}$

6 Time-dependent approximation: $p_i = \left(\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\|\mathbf{u}\|}{h}\right)^2 + \left(\frac{4\epsilon}{h}\right)^2 \right)^{-\frac{1}{2}} \mathbf{u} \cdot \nabla \phi$

7 Discontinuity capturing 1 of Hughes et al. (DC-1):
Define

$$u_{\|} = \frac{\mathbf{u} \cdot \nabla c}{\| c \|^2}, \tag{2.4.1}$$

where c is the solution to be computed.
Then the upwind basis functions p_i become:

$$p_i = (\tau_1 \mathbf{u} + \tau_2 \mathbf{u}_{\|}) \cdot \nabla \phi_i \tag{2.4.2}$$

with $\tau_1 = \frac{h\xi}{2\|\mathbf{u}\|}$ and $\tau_2 = \frac{h\xi}{2\|\mathbf{u}_{\|}\|}$. In this case ξ is chosen according to the doubly asymptotic approximation.

8 Discontinuity capturing 2 of Hughes et al. (DC-2):
This method is identical to DV-1, however, τ_2 is defined by $\tau_2 = max(0, \tau_{\|} - \tau)$, where $\tau_{\|}$ is the τ_2 defined in DC-1

9 Linear triangular method of Mizukami and Hughes satisfying the maximum principle:
This method, described in Mizukami and Hughes (1985), is only applicable for linear triangles. The method may be extended to linear tetrahedrons.
In this method the upwind basis functions p_i depends on the flow direction as well as the solution. Hence it is a non-linear method. The upwind basis functions are chosen such that the maximum principle is satisfied. This means that in the absence of sources the solution can never be lower than the lowest value on the boundary and never be higher than the highest value on the boundary.
Since the method is non-linear it requires iteration.
To start the iteration one might start for example with the doubly asymptotic approximation. Experiments show that sometimes the iteration shows a so-called flip-flop character. It switches between two different stages without ever converging.
To suppress such behavior two methods may be applied:

- Under-relaxation may be applied. This requires relatively many iterations and the choice of the under-relaxation parameter may be a problem.
  See SEPRAN EXAMPLES Section 3.1.8 for an example.

- A flip-flop mechanism may be triggered.
  In this mechanism an integer flip-flop array is used that keeps track of the various stages of the upwind basis functions. If the direction of the upwind basis function is clear it is kept.
  To set this array method = 10 and to update it use method = 11.
  See the manual SEPRAN EXAMPLES Section 3.1.8 for an example.

12 Time-dependent approximation according to Thornberg and Enquist

From these choices $\xi = 1$ gives the least accurate but smoothest results, the accuracy of the methods 2, 3 and 4 is comparable, method 2 is clearly the most expensive because of the necessity to compute the coth function for each point. Method 6 should be used in time-dependent problems. The methods 7 and 8 are examples of non-linear upwind methods. In the time-independent case they require an iteration, since the solution is part of the upwind parameter. In general the methods with discontinuity capturing give smoother results than the other methods. Especially DC-1 is very smooth, but generally less accurate than DC-2.

2.4.4 Coefficients for the differential equation

Compared to the coefficients described in Section 2.3.3, we need to prescribe the velocity, the density and if necessary information about upwind.
The velocity can be given as a vector with n components per node, or components-wise.

velocity = v (coef12/13/14) defines the velocity as vector $\boldsymbol{v}$, which must have been filled before.

u_velocity = u (coef12) defines the x-component of the velocity.

v_velocity = v (coef13) defines the y-component of the velocity.

w_velocity = w (coef14) defines the z-component of the velocity.
components that are omitted are equal to zero.

SUPG = *name* (icoef2) Type of upwind.
name is a string parameter with the following possible values:

NONE (0) No upwind (Default)

CLASIC (1) Classical upwind scheme

ILIN (2) Il'in scheme

DOUBLY_ASSYMPTOTIC (3) Doubly asymptotic approximation

CRITICAL (4) Critical approximation

HUGHES (5) Hughes approximation

TIME_DEPENDENT (6) Time-dependent approximation

DC1 (7) Discontinuity capturing 1

DC2 (8) Discontinuity capturing 2

MIZUKAMI_MAX (9) Linear triangular method of Mizukami and Hughes

MIZUKAMI_ACT (10) Activate flip-flop

MIZUKAMI_USE (11) Use flip flop

TORNBERG_TIME (12) Time-dependent approximation according to Thornberg and Enquist

TYPE_CONVECTION = *name* (icoef5) Indicates the type of convection term to be applied.
Possible values:

INCOMPRESSIBLE (0) (Default) This is the standard convection term as described above in Equation 2.4.1.

COMPRESSIBLE (1) In this case we use the more general form

$$\text{div } \mathbf{u}c \tag{2.4.1}$$

which reduces to the standard form in case of incompressible flow.

ABSOLUTE (2) A special non-linear form of the convective terms is the case in which the term $\mathbf{u} \cdot \nabla c$ is replaced by the term:

$$|\frac{\partial c}{\partial x}| + |\frac{\partial c}{\partial y}| + |\frac{\partial c}{\partial z}| \tag{2.4.2}$$

NON_LINEAR (3) The convective term may be a function g of ∇c.
At this moment it is assumed that the derivative $\frac{\partial g(\nabla c)}{\partial \nabla c}$ exists and can be computed by the user as function of ∇c and $\mathbf{x}$.
Since this term is non-linear a linearization procedure is necessary. At this moment the standard Newton (-Raphson) method is applied for the linearization.
Furthermore in this case the user must provide a user subroutine FUNCC2 (See Users Manual, Section 3.3.10), in which both the function $g(\nabla c)$ and the derivative $\frac{\partial g(\nabla c)}{\partial \nabla c}$ are computed as function of ∇c and $\mathbf{x}$.
In the manual SEPRAN EXAMPLES Section 3.3.4 it is explained how these equations may be solved.

APPLY_UPWIND = *name* (icoef20) Indicates if upwind must be applied to all terms or to certain terms only.
Possible values:

ALL (0) Applied to all terms (Default)

EXCL_MASS_MATRIX (1) Applied to all terms except the mass matrix.

CONVECTION_ONLY (2) Applied to the convection term only.

The sequence numbers of the other coefficients is:

SOURCE (coef15).

DENSITY (coef16).

HEAT_CAPACITY (coef17).

2.4.5 Coefficients for the natural boundary conditions

See Section 2.1.4.

2.4.6 Type numbers to be used in the problem input block

The use of the Diffusion equation is indicated in the problem block by the name

```
convection_diffusion
```

or alternatively by

```
type = 703
```

The natural boundary conditions do not have to be indicated by a type number, but internally type 801 is used.

2.4.7 Derivatives

See Section 2.3.6.

3 Second order elliptic and parabolic equations

In this chapter we consider several types of elliptic and parabolic equations of second order.
The following Sections are available:

3.1 Second order real elliptic and parabolic equations with one-degree of freedom.
In this section the general second order quasi linear elliptic equation is treated. Due to the presence of a time derivative the corresponding parabolic equation is treated as well.
The number of unknowns per point is 1.

3.3 Second order complex elliptic and parabolic equations with one degree of freedom.
This section has the same purpose as Section 3.1, however, in this case complex unknowns are considered.

3.4 Non-linear equations.
This section is devoted to some special non-linear differential equations.

3.5 δ-type source terms.
This section treats a very special type of source term. It has no general character.

3.6 Second order real elliptic and parabolic equations with two degrees of freedom.
This section has the same purpose as Section 3.1, however, in this case the number of unknowns is equal to two per point.

3.7 Extended second order real linear elliptic and parabolic equations with two degrees of freedom
This section has the same purpose as Section 3.6, however extra terms defining the coupling between the equations are present.

3.1 Second order real linear elliptic and parabolic equations with one degree of freedom

Equation

In this section we consider an extension of the convection diffusion equation 2.4.1:

$$\rho c_p(\frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla c) - \text{ div } (\alpha\nabla c + \gamma) + \beta c = f \tag{3.1.1}$$

i.e.

$$\rho c_p \frac{\partial c}{\partial t} + \rho c_p \sum_{i=1}^{n} u_i \frac{\partial c}{\partial x_i} - \sum_{i=1}^{n} \frac{\partial}{\partial x_i}\left(\sum_{j=1}^{n}\left(\alpha_{ij}\frac{\partial c}{\partial x_j}\right) + \gamma_i\right) + \beta c = f \tag{3.1.2}$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subset I\!R^n$$

In the stationary case $\left(\frac{\partial c}{\partial t} = 0\right)$ (3.1.1) reduces to

$$\rho c_p \mathbf{u}\cdot\nabla c - \text{ div } (\alpha\nabla c) + \beta c = f \tag{3.1.3}$$

In this case the equation is elliptic, otherwise it is parabolic.
The coefficients ρc_p, $\mathbf{u}$, α, β and f may depend on space and time and also of solutions of other problems. The following restrictions for the coefficients are required: **Defect correction**

This element allows for defect correction as described in the Users Manual Section 3.2.8 under the keyword SOLVE. If defect correction is applied the matrix to be solved corresponds to central differences, whereas for the iteration matrix the upwind matrix is used. Hence the input must contain the information with respect to the upwind method. Of course this option makes only sense in case convection terms are present.

3.1.1 Boundary Conditions

Besides the Dirichlet boundary conditions the following types of boundary conditions are available:

Type 2 (Neumann or mixed boundary condition)

$$\sum_{i=1}^{n}((\sum_{j=1}^{n}\alpha_{ij}(x)\frac{\partial c}{\partial x_j}) + \gamma_i)n_i + \sigma(x)c(x) = h(x) \qquad (\sigma(x) \geq 0) \tag{3.1.1}$$

on some part of the boundary. This is a so-called natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma(x) = 0$ and $h(x) = 0$, when there is no need to give any condition on this part of the boundary. For $\sigma = 0$ this is a Neumann boundary condition, otherwise it is a mixed boundary condition.

In the case of a convective term that has been written in the form of a linear compressible convection (icompress=1), one may either use the expression given in 3.1.1 in combination with extra elements of type 801 corresponding to boundary conditions of type 5, or one may use the following expression in stead of 3.1.1

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij}(x)\frac{\partial c}{\partial x_j}n_i - \mathbf{u}\cdot\mathbf{n}c(x) + \sigma(x)c(x) = h(x) \qquad (\sigma(x) \geq 0) \tag{3.1.2}$$

If expression 3.1.2 is used the input is exactly the same as for 3.1.1, which means that only h and σ have to be given and if both are zero, no boundary elements are necessary.

Examples

For the Poisson equation boundary conditions of type 2 are given by: $\frac{\partial c}{\partial n} + \sigma(x)c = h(x)$,
for the convection diffusion equation by: $\epsilon\frac{\partial c}{\partial n} + \sigma(x)c = h(x)$,
for ground water flow by: $k\frac{\partial \phi}{\partial n} + \sigma(x)\phi = h(x)$,
for the Reynolds equation by: $h^3\frac{\partial p}{\partial n} + \sigma(x)p = h(x)$

Type 3

$$\sum_{i=1}^{n}((\sum_{j=1}^{n}\alpha_{ij}(x)\frac{\partial c}{\partial x_j}) + \gamma_i)n_i + \sigma(x)\frac{\partial c}{\partial t} = h(x) \qquad (\sigma(x) \geq 0) \tag{3.1.3}$$

on some part of the boundary,
with $\frac{\partial c}{\partial t} = \nabla c \cdot \mathbf{t}$, with $\mathbf{t}$ the tangential vector.

This is a special type of natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma(x) = 0$ and $h(x) = 0$, in which case this condition reduces to a boundary condition of type 2 on this part of the boundary.

Examples

For the Poisson equation boundary conditions of type 3 are given by: $\frac{\partial c}{\partial n} + \sigma(x)\frac{\partial c}{\partial t} = h(x)$,
for the convection diffusion equation by: $\epsilon\frac{\partial c}{\partial n} + \sigma(x)\frac{\partial c}{\partial t} = h(x)$,
for ground water flow by: $k\frac{\partial \phi}{\partial n} + \sigma(x)\frac{\partial \phi}{\partial t} = h(x)$,
for the Reynolds equation by: $h^3\frac{\partial p}{\partial n} + \sigma(x)\frac{\partial p}{\partial t} = h(x)$

Type 4

$$\frac{\partial c}{\partial n} + \sigma(x)\frac{\partial c}{\partial t} = h(x) \qquad (\sigma(x) \geq 0) \tag{3.1.4}$$

on some part of the boundary,
with $\frac{\partial c}{\partial t} = \nabla c \cdot \mathbf{t}$, with $\mathbf{t}$ the tangential vector and $\frac{\partial c}{\partial n} = \nabla c \cdot \mathbf{n}$, $\mathbf{n}$ denotes the normal vector.

This is another special type of natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma(x) = 0$, α is a diagonal matrix with a constant diagonal and $h(x) = 0$, in which case this condition reduces to a boundary condition of type 2 on this part of the boundary.

Type 5 If convective terms of the form (2.4.1) are used and at a part of the outer boundary both $\mathbf{u} \cdot \mathbf{n} \neq 0$ and c is not prescribed, it is necessary to define extra boundary elements with type number 801. These boundary elements must be considered as additional to other boundary elements at that boundary.
This possibility has only been implemented for two-dimensional elements. It is in that case necessary that the boundary where these boundary elements must be applied is created counterclockwise, since a term involving $\mathbf{u} \cdot \mathbf{n}$ is required and $\mathbf{n}$ is computed from the tangential direction in counter-clockwise direction. If the boundary is created clockwise it is necessary to use $-\mathbf{u}$ instead of $\mathbf{u}$.

Type 6 (Discontinuous boundary condition)
A very special boundary condition is formed by the following boundary condition allowing a jump in the solution. Suppose the region is subdivided in an upper region (u) and a lower region (l) separated by a membrane. Assume furthermore that the solution jumps over this membrane and hence is discontinuous. Furthermore we assume that equation 3.1.1 holds for both the upper part and the lower part and that the coefficients are the same at the membrane. At the membrane we assume the following boundary condition:

$$\sum_{i=1}^{n}((\sum_{j=1}^{n}\alpha_{ij}(x)\frac{\partial c}{\partial x_j}) + \gamma_i)n_i + \sigma(x)(c_u(x) - c_l(x)) = h(x) \tag{3.1.5}$$

c_u means the value at the upper region and c_l the value at the lower region. This boundary condition implies that the values at both sides of the membrane are different. In order to use this boundary condition connection elements as described in the Users Manual Section 2.2 are necessary. These connection elements must connect linear or quadratic line elements in R^2 or surface elements in R^3.
So for example in case of linear elements in R^2 one has to use linear line elements as connection elements, like

```
celmj = curves 1 ( ck, cl )
```

The parameter 1 indicates that it concerns linear elements
So in contrast to other boundary conditions it is not longer possible to use so-called boundary elements, but these elements must be used in the same way as internal elements.
If boundary conditions of this type are applied the resulting matrices are non-symmetrical, which implies that a non-symmetrical storage must be used.
See the manual SEPRAN EXAMPLES Section 3.1.4 for an example.

Remark

Both for boundary conditions of type 3 and type 4 it is necessary that the boundary elements are created counter clockwise in order to fix the direction of the normal in relation with the tangential vector. These boundary conditions are at present only available for linear two-dimensional elements.

3.1.2 Coefficients for the differential equation

Compared to the coefficients described in Section 2.4.2, the following extra coefficients are required.

ZERO_ORDER_COEF = β (coef18) contains the parameter β.

GAMMA =γ (coef19/20/21) contains the vector γ or alternatively one may prescribe the components separately.

X_GAMMA =γ_1 (coef19) first component of γ

Y_GAMMA =γ_2 (coef20) second component of γ

Z_GAMMA =γ_3 (coef21) third component of γ

The sequence number of APPLY_UPWIND is icoef25.

3.1.3 Coefficients for the natural boundary conditions

The non-homogeneous natural boundary conditions require extra input for the coefficients. In the standard case of boundary conditions of type 2 we need the same input as in Section 2.1.4. If other types of natural boundary conditions are used extra input is needed

1 (icoef1) Type of natural boundary condition.
Possible values:

0,2: natural boundary condition of type 2

3: natural boundary condition of type 3

4: natural boundary condition of type 4

5: special boundary condition of type 5 corresponding to convective terms of the shape (2.4.1).

8 α_{11}

9 α_{12}

10 α_{13}

11 α_{22}

12 α_{23}

13 α_{33}

In case of boundary conditions of type 5, which are meant for convective terms of the shape (2.4.1), it is necessary to use boundary elements of type 801 for those outer boundaries where $\mathbf{u} \cdot \mathbf{n} \neq 0$ and c is not prescribed. In that case 8 coefficients are required.
The coefficients 6, 7 and 8 must contain information about the velocity vector **u** according to:
coefficient 6: u_1
coefficient 7: u_2
coefficient 8: u_3

3.1.4 Type numbers to be used in the problem input block

The use of this equation is indicated in the problem block by the name

```
general_elliptic_equation
```

or alternatively by

```
type = 705 (or 800)
```

3.1.5 Derivatives

The first derivatives of the solution as well as the gradient and the flux require the same input as in Section 2.3.6. Extra possibilities are

ICHELD = 5 $\left(\frac{\partial c}{\partial y}, -\frac{\partial c}{\partial x}\right)$ 2D only

ICHELD = 11-20 See 1-10, however, now defined per element

ICHELD = 21 Put a function f defined by the sixth coefficient into the output vector. The only difference with subroutine CREATE is the use of the averaging procedure.

ICHELD = 31-40 See 1-10, however, now derivatives are computed with general weight 1

The output vector is defined as follows:

1:	a vector of the type vector of special structure with 1 unknown per point
2,3,5,6:	a vector of the type vector of special structure with *ndim* unknowns per point
11:	a vector of special structure defined per element with one unknown
12,13,15,16:	a vector of special structure defined per element with *ndim* unknowns

In the cases ICHELD = 6, 16 and 21 the user must define coefficients according to:

ICHELD = 6 and 16 coefficients 6 to 11 must contain the various coefficients α_{ij} just as for the definition of the matrix

ICHELD = 21 coefficient 6 must contain the definition of the function f

Furthermore, the integer parameters 3 and 4 are used in exactly the same way as for the building of the matrix.
Besides that the first integer parameter (ISEQ) may be used to define from which sequence number in array ISLOLD the derivatives must be computed if ISLOLD contains more references to solution vectors.
If ISEQ = 0 or 1 the first vector referenced by ISLOLD is used.

3.1.6 Integrals

If the user wants to compute integrals over the solution, he may use the option INTEGRAL in the input block "STRUCTURE"

The parameter ICHELI in the input block "INTEGRALS" is used to distinguish the various possibilities:

ICHELI=1 $\int\limits_{\Omega} f(x)d\Omega$

ICHELI=2 $\int\limits_{\Omega} f(x)c(x)d\Omega$

ICHELI=2+i $\int\limits_{\Omega} f(x)\frac{\partial c}{\partial x_i}d\Omega \qquad (i= 1,2,3)$

ICHELI=6 $\int\limits_{\Omega} f(x)c_{JDEGFD}d\Omega$

ICHELI=7 $\int\limits_{\Omega} d\Omega$, i.e. the Cartesian volume is computed.
If you need the volume in other coordinate systems, use ICHELI=1 in combination with f=1.

ICHELI=8 $\int\limits_{\Omega} f(x)c^2(x)d\Omega$

ICHELI=11-16 See ICHELI=1 to 6. The same integrals are computed, however, dx is used instead of $d\Omega$, hence the integral is computed in the positive x-direction only.

ICHELI=21-26 See ICHELI=1 to 6. The same integrals are computed, however, dy is used instead of $d\Omega$, hence the integral is computed in the positive y-direction only.

ICHELI=31-36 See ICHELI=1 to 6. The same integrals are computed, however, dz is used instead of $d\Omega$, hence the integral is computed in the positive z-direction only.

In this case $c(x)$ is the vector Vj as indicated by the command INTEGRAL in the input block "STRUCTURE:. The user must define the function $f(x)$ as first coefficient by one of the methods described in 2.2.
For ICHELI = 6, u_{JDEGFD} implies the $JDEGFD^{th}$ component of the degree of freedom u. In general u is a derivative quantity, for example the gradient.

Except for ICHELI=7, the input block "INTEGRALS" expects also some coefficients that may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
For each element group 10 parameters and coefficients must be given. The first 3 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 7 are real coefficients.

These parameters and coefficients are defined as follows:

1	type of numerical integration, see the coefficients for the equation
2	type of co-ordinate system, see the coefficients for the equation
3	not yet used
4	f
5-10	not yet used

3.1.7 Types of vectors of special structure

The following types of vectors are standard for standard elements described in this section as well as Sections 2.1 to 2.4.

IVEC=0 Solution vector.
Contains 1 degree of freedom per point. May be renumbered.

IVEC=1 Vector of special structure with 1 degree of freedom per point.

IVEC=2 Vector of special structure with 2 degrees of freedom per point.

IVEC=3 Vector of special structure with 3 degrees of freedom per point.

IVEC=4 Vector of special structure with 6 degrees of freedom per point.

IVEC=5 Vector of special structure with *ndim* degrees of freedom per point, where *ndim* is the dimension of the space.

IVEC=6 Vector of special structure with 1 degree of freedom per vertex and none in the other points.

IVEC=7 Vector of special structure with 2 degrees of freedom per vertex and none in the other points.

IVEC=8 Vector of special structure with 3 degrees of freedom per vertex and none in the other points.

IVEC=9 Vector of special structure with 6 degrees of freedom per vertex and none in the other points.

IVEC=10 Vector of special structure with *ndim* degrees of freedom per vertex and none in the other points.

IVEC=11 Vector of special structure with 4 degrees of freedom per point.

IVEC=12 Vector of special structure with 5 degrees of freedom per point.

IVEC=13 Vector of special structure with 7 degrees of freedom per point.

IVEC=14 Vector of special structure with 8 degrees of freedom per point.

IVEC=15 Vector of special structure with 9 degrees of freedom per point.

IVEC=16 Vector of special structure with 10 degrees of freedom per point.

3.2 A special right-hand side term for the convection diffusion euqation

Equation

In this section we consider the special case of a extra right-hand-side term of the shape

$$\text{div } (\ T\ \nabla c_i\ ) \tag{3.2.1}$$

where c_i and T are known vectors.

We assume that this term must be added to convection-diffusion type equations as treated in Section 3.1.
The idea is that one of more of such terms must be added to the right-hand side. Each term separately can be constructed in the `structure` block by the statement:

```
rhs_i = right_hand_side, problem = p, seq_coef = i
```

and if more than one of such terms are needed they can be added for example by:

```
rhs = rhs_1+rhs_2
```

The result can be used in a time integration by using

```
   seq_add_rhsd = rhs
```

in the `time_integration` input block.

The type number for this extra term to be used is 816 and the essential boundary conditions must be defined in exactly the same way as for the equation to which this term is added. The values of the boundary conditions are not used.

- Definition of the coefficients for the differential equation:

  This extra term requires 7 input coefficients, where the first 5 have exactly the same meaning as for type 800, defined in Section 3.1.
  The other 2 (coef 6 and 7 respectively) refer to T and c_i, for example:

  ```
  coefficients, sequence_number 3
     coef 6 = T
     coef 7 = c_1
  end
  ```

3.3 Second order complex elliptic and parabolic equations with one degree of freedom

Equation

In this section we consider equations of the following form:

$$\rho c_p \left(\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c \right) - \text{ div } (\alpha \nabla c) + \beta c = f \tag{3.3.1}$$

i.e.

$$\rho c_p \frac{\partial c}{\partial t} + \rho c_p \sum_{i=1}^{n} u_i \frac{\partial c}{\partial x_i} - \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial}{\partial x_i} \left(\alpha_{ij} \frac{\partial c}{\partial x_j} \right) + \beta c = f \tag{3.3.2}$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subset I\!R^n$$

In the stationary case $\left(\frac{\partial c}{\partial t} = 0\right)$ (3.3.2) reduces to

$$\rho c_p \mathbf{u} \cdot \nabla c - \text{ div } (\alpha \nabla c) + \beta c = f \tag{3.3.3}$$

In this case the equation is elliptic, otherwise it is parabolic.
The coefficients ρc_p, $\mathbf{u}$, α, β and f may depend on space and time and also of solutions of other problems. The following restrictions for the coefficients are required:

- $\rho c_p > 0$.
- The coefficients α_{ij}, u_i, β and f may be complex, ρc_p must be real.
  The matrix α with coefficients α_{ij} must be a symmetric matrix.

Typical examples of equations of the form (3.3.1) are

- The Helmholtz equation: $-$ div $(\alpha \nabla \phi) + \beta \phi = f$ i.e.

  $\mathbf{u} = 0$
  $\rho c_p = 0$

Boundary and initial conditions

In the instationary case it is necessary to give an initial condition at $t = 0$.

The following types of boundary conditions are available:

Type 1 (Dirichlet boundary condition) $c(\mathbf{x})$ given on some part of the boundary.
This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2 (Neumann or mixed boundary condition)

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{ij}(x) \frac{\partial c}{\partial x_j} n_i + \sigma(x) c(x) = h(x) \tag{3.3.4}$$

on some part of the boundary. This is a so-called natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma(x) = 0$ and $h(x) = 0$, when there is no need to give any condition on this part of the boundary.
Both σ and h are complex functions.

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input block "MATRIX" of program SEPCOMP.

  In general, two matrices may be created: the mass matrix and the stiffness matrix. The mass matrix is only used for time-dependent problems. This matrix pre-multiplies the discretized time-derivative. The stiffness matrix represents the discretization of the stationary terms in the left-hand side of equation (3.3.1).

  The mass matrix is, in general, positive definite and symmetrical.
  The stiffness matrix is, in general, non-symmetrical except in the case that the velocity $\mathbf{u} = 0$. This matrix is complex.

  The storage scheme corresponding to the mass matrix may be either METHOD = 1 or METHOD = 5. The storage scheme corresponding to the stiffness matrix must be 3 or 7 in case of a symmetrical and 4 or 8 in case of a non-symmetrical matrix.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
  For each element group 5 (1D), 8 (2D) or 12 (3D) parameters and coefficients must be given. These parameters and coefficients are defined as follows:

seq. number	1D	2D	3D
1	α_{11}	α_{11}	α_{11}
2	u_1	α_{12}	α_{12}
3	β	α_{22}	α_{13}
4	f	u_1	α_{22}
5	ρc_p	u_2	α_{23}
6		β	α_{33}
7		f	u_1
8		ρc_p	u_2
9			u_3
10			β
11			f
12			ρc_p

  The coefficients may be zero, constants or functions as described in Section 10.1. They may also depend on precomputed vectors.

- Parameters for subroutine BUILD:

  With respect to subroutine BUILD the parameter IMAS (IINBLD(4)) is of importance. In the stationary case no mass matrix is necessary so IMAS may be chosen equal to zero.
  In the time-dependent case IMAS may be either 1 (diagonal mass matrix), 2 or 3 ("consistent" mass matrix).

  A diagonal mass matrix is only recommended in case of linear elements.

- Computation of derivatives:

  The parameter ICHELD in the input block "DERIVATIVES" is used to distinguish the various possibilities:

  ICHELD = 1 $\frac{\partial \phi}{\partial x_{IX}}$ in the vertices of the element. Hence when $IX = 2 : \frac{\partial \phi}{\partial x_2}$.

ICHELD = 2 $\nabla\phi$ in the vertices of the element.

ICHELD = 3 $-\nabla\phi$ in the vertices of the element.

ICHELD = 4 $k \parallel \nabla(\phi) \parallel^2 =$
$k(\text{Re } [\frac{\partial\phi}{\partial x}])^2 + (\text{Im } [\frac{\partial\phi}{\partial x}])^2 + (\text{Re } [\frac{\partial\phi}{\partial y}])^2 + (\text{Im } [\frac{\partial\phi}{\partial y}])^2 + (\text{Re } [\frac{\partial\phi}{\partial z}])^2 + (\text{Im } [\frac{\partial\phi}{\partial z}])^2$

The output vector is defined as function of ICHELD as follows:

1 for linear elements a vector of the type solution vector with one unknown per point, otherwise a vector of special structure with sequence number 1 (one unknown in each vertex)

2,3 for linear elements a vector of special structure with sequence number 1 with NDIM unknowns per point. Otherwise a vector of special structure with sequence number 2 with NDIM unknowns per vertex.

4 a vector of special structure defined per element (one real unknown per element). No averaging takes place.
This possibility is only available for linear and bilinear elements.

In the case ICHELD = 4 the user must define the coefficient k as first parameter. k must be a real parameter.

- Types of integrals that may be computed

The same values for ICHELI as is Section 3.1 with respect to type number 800 are available. The function $c(x)$ must be real, which means that $c(x)$ may not be the solution of the differential equation itself. However, $c(x)$ may be for example a real derived quantity like the one computed with ICHELD=4.

The input block "INTEGRALS" expects also some coefficients that may be defined by one of the methods described in 10.1, where, in general, the method by SEPCOMP is recommended. For each element group 10 parameters and coefficients must be given. The first 3 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 7 are real coefficients.

These parameters and coefficients are defined as follows:

1	type of numerical integration, see the coefficients for the equation
2	type of co-ordinate system, see the coefficients for the equation
3	not yet used
4	f
5-10	not yet used

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved. In the old version of SEPRAN type numbers are also connected to the type of elements to be used. So for the same equation many more type numbers are necessary. The following type numbers are available:

150 Complex elliptic differential equation for linear triangle in R^2.

151 Complex elliptic differential equation for linear triangle in R^2, axi-symmetric co-ordinates.

152 Boundary condition of type 2 for linear line element in R^2. Must be used with linear triangles or bilinear quadrilaterals.

154 Complex elliptic differential equation for quadratic triangle in R^2.

155 Complex elliptic differential equation for quadratic triangle in R^2, using axi-symmetric co-ordinates.

158 Complex elliptic differential equation for bi-linear quadrilateral in R^2.

159 Complex elliptic differential equation for linear triangle in R^2, axi-symmetric co-ordinates, using axi-symmetric co-ordinates.

160 Complex elliptic differential equation for bi-quadratic quadrilateral in R^2.

161 Complex elliptic differential equation for bi-quadratic quadrilateral in R^2, using axi-symmetric co-ordinates.

163 Complex elliptic differential equation for linear tetrahedron in R^3.

165 Complex elliptic differential equation for tri-linear hexahedron in R^3.

3.4 Non-linear equations

In this section we consider special non-linear elliptic and parabolic equations. In general the equations treated in Sections 3.1 and 3.2 are linear, but they may be treated as non-linear equations by using coefficients that depend on preceding solutions. If a non-linear solver is used, these coefficients may be updated in each iteration step and a type of Picard linearization, also called successive substitution arises.

In this section we shall be considered with elliptic and parabolic equations in which the non-linearity is made explicit. For that reason it is possible to use explicit non-linear solvers and therefore a better convergence may be achieved.

At this moment only a non-linear diffusion problem, where the diffusion coefficient is a function of the norm of the gradient of the solution is available. This type of equations is commonly used in the computation of magnetic fields if a non-linear constitutive relation must be used. See SEPRAN EXAMPLES Section chap-3.3.1 for the details. An example of the use of these elements is given in SEPRAN EXAMPLES Section chap-3.3.2, where the magnetic field in an alternator is computed.

3.4.1 A special non-linear diffusion equation

Equation

$$-\text{ div }(\alpha\nabla\phi)+\beta\phi=f \tag{3.4.1.1}$$

i.e.

$$-\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}\left(\alpha\frac{\partial\phi}{\partial x_j}\right)+\beta\phi=f$$

where $\alpha=\alpha(\mathbf{x},\parallel\nabla\phi\parallel)$, $\quad\parallel\nabla\phi\parallel=\left(\sum_{i=1}^{n}\left(\frac{\partial\phi}{\partial x_i}\right)^2\right)^{\frac{1}{2}}$ $\quad\mathbf{x}=(x_1,x_2,\ldots,x_n)\in\Omega\subset I\!R^n$

Remark

α and β must be larger than zero.

Boundary conditions

The following types of boundary conditions are available:

Type 1 (Dirichlet boundary conditions) $\phi(\mathbf{x})$ given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2 (Neumann or mixed boundary conditions)

$$\alpha\frac{\partial\phi}{\partial n}+\sigma(x)\phi(x)=h(x)\qquad(\sigma(x)\geq 0) \tag{3.4.1.2}$$

on some part of the boundary. This is a so-called natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma(x)=0$ and $h(x)=0$, when there is no need to give any condition on this part of the boundary. In case $\sigma=0$ this is a Neumann boundary condition, otherwise it is called mixed boundary condition.

Solution method

Since the equation is non-linear, a linearization procedure must be applied. The following techniques are available:

- Picard iteration (successive substitution)
- Newton iteration (quadratic convergence)

The iterative procedure is as follows:

(i) Start with an approximation $\mathbf{u}^0$. A good starting value may be for example the solution for α constant, or $\alpha(\mathbf{x},0)$. For the Newton method a start with Picard may be attractive.

(ii) Solve the system of equations.

(iii) Repeat step (ii) until convergence has been achieved: $\parallel\mathbf{u}^{i+1}-\mathbf{u}^i\parallel<\epsilon_{tol}$, where ϵ_{tol} is some tolerance.

Remark: In general the rate of convergence of Picard is linear, whereas Newton is a quadratic converging process. However, the convergence region is usually larger for Picard, than for Newton. Therefore Newton's method is frequently started with one or more Picard iterations. Another method to control convergence is to start with a small value of the source f and to raise this value during the iteration process.

Practical implementation

The most simple way of solving the non-linear problem is to use program SEPCOMP with the option NONLINEAR_EQUATIONS in the input file as described in the Users Manual, Section 3.2.9 and 3.2.3.

If the user creates his own main programs, however, the following steps may be programmed:

i Build the start vector (for example by CREATE).

ii Build the matrix and right-hand side for the non-linear equations linearized by Picard or Newton (subroutine BUILD).

iii Solve the system of equations (subroutine SOLVE).

iv Compute the difference between two succeeding iterations (subroutine MANVEC). When the difference is too large, repeat steps ii, iii and iv.

The steps ii, iii and iv can be carried out with help of subroutines NONLIN and FILNLN.

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input of program SEPCOMP or subroutine SEPSTR or alternatively by the parameter JMETOD in subroutine COMMAT.

  If a Picard method is chosen, the matrix is symmetrical and positive definite. In that case METHOD = 1 or 5 may be chosen. Otherwise (Newton iteration) the matrix is non-symmetrical and METHOD = 2 or 6 must be chosen.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in Section 10.1, where, in general, the method by SEPCOMP of FILCOF is recommended. For each element group 20 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 15 are real coefficients.

  These parameters and coefficients are defined as follows:

1	not yet used
2	not yet used
3	type of numerical integration
4	type of co-ordinate system
5	type of linearization
6	not yet used
7	not yet used
8	not yet used
9	not yet used
10	not yet used
11	not yet used
12	not yet used
13	not yet used
14	not yet used
15	β

16 f
17-20 not yet used

For each element group 20 parameters and coefficients must be given. The parameters 1 to 5 are of integer type the parameters 15-20 are real.
These parameters and coefficients are exactly the same as for the second-order real linear elliptic equation described in Section 3.1. The only difference is that the parameters 6 to 14 are not used and the parameter ITER_TYPE is used as fifth parameter. This parameter defines the type of iteration, according to:

ITER_TYPE = 1 Picard iteration
2 Newton iteration

In order to compute the parameters α and $\frac{\partial \alpha}{\partial \| \nabla \phi \|}$ a subroutine FUNCC2 must be provided of the following shape:

```
SUBROUTINE FUNCC2 ( ICHOIS, X, Y, Z, GRADPH, ALPHA, DALPDG )
IMPLICIT NONE
INTEGER ICHOIS
DOUBLE PRECISION X, Y, Z, GRADPH, ALPHA, DALPDG

  statements to fill ALPHA ( ICHOIS = 1 ) or
  ALPHA and DALPDG ( ICHOIS = 2 )

END
```

Subroutine BUILD gives X, Y, Z and GRADPH a value for each integration point and gives ICHOIS the value of ITER_TYPE.
GRADPH $= \| \nabla \bar{\phi} \| = \left(\sum_{i=1}^{n} \frac{\partial \bar{\phi}^2}{\partial x} \right)^{\frac{1}{2}}$, with $\bar{\phi}$ the result of the preceding iteration or the start vector.
When ICHOIS = 1, ALPHA must get the value $\alpha(\| \nabla \phi \|)$, when ICHOIS = 2, ALPHA must get the value $\alpha(\| \nabla \phi \|)$ and DALPDG the value $\frac{\partial \alpha}{\partial \| \nabla \phi \|}$.

- Definition of the coefficients for the boundary conditions:

The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
For each element group 15 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 10 are real coefficients.

These parameters and coefficients are defined as follows:

1 Type of natural boundary condition
2 Not yet used (must be zero)
3 type of numerical integration
4 type of co-ordinate system
5 not yet used

6 σ
7 h
8-25 not yet used

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients

has to be given.
The coefficients 6 and 7 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors.

With respect to the parameters 1-5 exactly the same choices as in Section 3.1 are available.

- Computation of derivatives:

  The definition of the parameter ICHELD as well as the output vector for the computation of the derivatives is exactly the same as the definition in Section 3.1.

- Types of integrals that may be computed:

  The definition of the parameter ICHELI for the computation of volume integrals is exactly the same as the definition in Section 3.1.

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the special non-linear diffusion equation in this section the following type numbers are available:

803 general type number for the internal elements. Defines the differential equation. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

shape = 11 linear tetrahedron.

shape = 12 quadratic tetrahedron.

shape = 13 trilinear hexahedron.

shape = 14 triquadratic hexahedron.

801 boundary conditions of type 2. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

Previous versions of SEPRAN

In previous versions of SEPRAN equation 3.4.1.1 has been solved with a different type number. These type numbers may still be used, however, it is recommended to use the new type numbers described earlier when creating new input or new programs. In this section we will point out the differences of the previous type number and the present one.

- Definition of the coefficients for the differential equation:

  The old version may only be used in R^2 with linear triangles.
  The coefficient β is not available. The number of parameters is equal to 2 one real and one integer parameter.
  It concerns the following parameters:

  1 integer parameter METHOD

  2 real parameter f

  The integer parameter METHOD defines the type of linearization in the same way as before.

With respect to the computation of integrals and derivatives the same possibilities as described in Section 3.1, previous version of SEPRAN, are available.

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved. In the old version of SEPRAN type numbers are also connected to the type of elements to be used. For that version only the following type number is available:

190 Special non-linear diffusion equation for linear triangle in R^2.

3.5 δ-type source terms

3.5.1 Equation

If in the case of equations as treated in 2.2 to 2.4 or 3.1 the right-hand side is a δ-type source term a special element is necessary.
This source term may be either defined in a point or on a line and is defined such that

$$\int_{\Omega} \delta(\mathbf{x}) d\Omega = \rho,$$

where ρ is a given function.

To use this type of source term the user has to define extra point or line boundary elements.

In each nodal point of these boundary elements the value ρ must be provided by the user.

3.5.2 Coefficients for the differential equation

The user must define the height ρ by the name `deltarhs` (coef1).

3.5.3 Type numbers to be used in the problem input block

The use of the Poisson equation is indicated in the problem block by the name

```
delta_function
```

or alternatively by

```
type = 802
```

3.6 Second order real linear elliptic and parabolic equations with two degrees of freedom

Equation

In this section we consider equations of the following form:

$$\rho^1(\frac{\partial c^1}{\partial t} + \mathbf{u}^1 \cdot \nabla c^1) - \text{ div } (\alpha^1 \nabla c^1) + \beta^1 c^1 = f^1 \tag{3.6.1}$$

$$\rho^2(\frac{\partial c^2}{\partial t} + \mathbf{u}^2 \cdot \nabla c^2) - \text{ div } (\alpha^2 \nabla c^2) + \beta^2 c^2 = f^2 \tag{3.6.2}$$

i.e.

$$\rho^k \frac{\partial c^k}{\partial t} + \rho^k \sum_{i=1}^{n} u_i^k \frac{\partial c^k}{\partial x_i} - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_i}\left(\alpha_{ij}^k \frac{\partial c^k}{\partial x_j}\right) + \beta^k c^k = f^k \tag{3.6.3}$$

$$k = 1, 2; \qquad \mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subset I\!R^n$$

In the stationary case $\left(\frac{\partial c^k}{\partial t} = 0\right)$ (3.6.1,3.6.2) reduces to

$$\mathbf{u}^k \cdot \nabla c^k - \text{ div } (\alpha^k \nabla c^k) + \beta^k c^k = f^k \tag{3.6.4}$$

In this case the equation is elliptic, otherwise it is parabolic.
The coefficients ρ^k, $\mathbf{u}^k$, α^k, β^k and f^k may depend on space and time and also of solutions of other problems. The following restrictions for the coefficients are required:

- $\rho^k > 0$.
- The matrices α^k with coefficients α_{ij}^k must be positive definite symmetric. In the extreme case α^k may be equal to zero in which case the equation is of hyperbolic type.

Remark
The equations 3.6.1, 3.6.2 are identical to the equation 3.1.1 in Section 3.1, with ρc_p replaced by ρ. Furthermore the number of equations is equal to 2 instead of 1, each with its own coefficients. The equations itself are not coupled, however, coupling may be possible by the boundary conditions. Also the coefficients of one equation may depend on the solution of the other equation, but only by using the result of a previous iteration or time-step.
The case of coupled equations is treated in Section 3.7

Upwinding

Upwinding has not yet been implemented.

Defect correction

Defect correction has not yet been implemented.

Boundary and initial conditions

In the instationary case it is necessary to give an initial condition at $t = 0$.

The following types of boundary conditions are available:

Type 1 (Dirichlet boundary condition) $c^k(\mathbf{x})$ given on some part of the boundary.
This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2 (Neumann or mixed boundary condition)

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij}^k(x)\frac{\partial c^k}{\partial x_j}n_i+\sigma^k(x)c^k(x)=h^k(x)\qquad(\sigma^k(x)\geq 0)\tag{3.6.5}$$

on some part of the boundary. This is a so-called natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma^k(x)=0$ and $h^k(x)=0$, when there is no need to give any condition on this part of the boundary. For $\sigma^k=0$ this is a Neumann boundary condition, otherwise it is a mixed boundary condition.

Type 3

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_{ij}^k(x)\frac{\partial c^k}{\partial x_j}(x)n_i+\sigma^k(x)\frac{\partial c^k}{\partial t}=h^k(x)\qquad(\sigma^k(x)\geq 0)\tag{3.6.6}$$

on some part of the boundary,
with $\frac{\partial c^k}{\partial t}=\nabla c^k\cdot\mathbf{t}$, with $\mathbf{t}$ the tangential vector.

This is a special type of natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma^k(x)=0$ and $h^k(x)=0$, in which case this condition reduces to a boundary condition of type 2 on this part of the boundary.

Type 4

$$\frac{\partial c^k}{\partial n}+\sigma^k(x)\frac{\partial c^k}{\partial t}=h^k(x)\qquad(\sigma^k(x)\geq 0)\tag{3.6.7}$$

on some part of the boundary,
with $\frac{\partial c^k}{\partial t}=\nabla c^k\cdot\mathbf{t}$, with $\mathbf{t}$ the tangential vector and $\frac{\partial c^k}{\partial n}=\nabla c^k\cdot\mathbf{n}$, $\mathbf{n}$ denotes the normal vector.

This is another special type of natural boundary condition. In general, boundary elements are necessary, except in the case that $\sigma^k(x)=0$, α^k is a diagonal matrix with a constant diagonal and $h^k(x)=0$, in which case this condition reduces to a boundary condition of type 2 on this part of the boundary.

Remarks

- Both for boundary conditions of type 3 and type 4 it is necessary that the boundary elements are created counter clockwise in order to fix the direction of the normal in relation with the tangential vector. These boundary conditions are at present only available for linear two-dimensional elements.
- Mark that the boundary conditions of type 1 to 4 are identical to the ones given in the manual SEPRAN EXAMPLES Section 3.1.1 and these boundary conditions also relate to each of the unknowns separately. Hence if only these boundary conditions are used in combination with the equations 3.6.1 and 3.6.2, it is better to use the equations in the manual SEPRAN EXAMPLES Section 3.1.1 twice.

Input for the various subroutines

- Definition of the storage scheme:

  See the remarks in the manual SEPRAN EXAMPLES Section 3.1.1. In this case the matrix is only symmetrical if both equations are symmetrical and there is no coupling between the boundary conditions.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
  For each element group 35 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 30 are real coefficients.

  These parameters and coefficients are defined as follows:

  1 not yet used
  2 not yet used
  3 type of numerical integration
  4 type of co-ordinate system
  5 not yet used

  real parameters with respect to equation 1

  6 α_{11}^1
  7 α_{12}^1
  8 α_{13}^1
  9 α_{22}^1
  10 α_{23}^1
  11 α_{33}^1
  12 u_1^1
  13 u_2^1
  14 u_3^1
  15 β^1
  16 f^1
  17 ρ^1
  18-20 not yet used

  real parameters with respect to equation 2

  21 α_{11}^2
  22 α_{12}^2
  23 α_{13}^2
  24 α_{22}^2
  25 α_{23}^2
  26 α_{33}^2

27 u_1^2
28 u_2^2
29 u_3^2
30 β^2
31 f^2
32 ρ^2
33-35 not yet used

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
The coefficients 6-35 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors. Of course, in 1D and 2D not all coefficients are used.
The default values for the real coefficients 6 to 16 and 21 to 31 are zero, for coefficients 17 and 32 (ρ^k), however, the default value is one.

With respect to the parameters 1-5 the same choices may be made as in Section 3.1.1, except for the parts that have not yet been implemented.

- Definition of the coefficients for the boundary conditions:

The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
For each element group 25 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 20 are real coefficients.

These parameters and coefficients are defined as follows:

1 Type of natural boundary condition
2 Not yet used (must be zero)
3 type of numerical integration
4 type of co-ordinate system
5 not yet used

real parameters with respect to equation 1

6 σ^1
7 h^1
8 α_{11}^1
9 α_{12}^1
10 α_{13}^1
11 α_{22}^1
12 α_{23}^1
13 α_{33}^1
14-15 not yet used

real parameters with respect to equation 2

16 σ^2
17 h^2
18 α_{11}^2
19 α_{12}^2
20 α_{13}^2
21 α_{22}^2
22 α_{23}^2
23 α_{33}^2
24-25 not yet used

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
The coefficients 6-25 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors. Of course, in 1D and 2D not all coefficients are used. The coefficients 8-13 and 18-23 are only used for boundary conditions of type 4.

With respect to the parameters 1-5 the same choices as in the manual SEPRAN EXAMPLES Section 3.1.1 are available, except for the parts that have not yet been implemented.

- Parameters for subroutine BUILD:

  See the manual SEPRAN EXAMPLES Section 3.1.1.

- Parameters with respect to the linear solver:

  See the manual SEPRAN EXAMPLES Section 3.1.1.

- Computation of derivatives:

  The computation of derivatives has not yet been implemented.

- Types of integrals that may be computed:

  The computation of integrals has not yet been implemented.

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the second order elliptic equation in this section the following type numbers are available:

805 general type number for the internal elements. Defines the differential equation.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

shape = 11 linear tetrahedron.

shape = 12 quadratic tetrahedron.

shape = 13 trilinear hexahedron.

shape = 14 triquadratic hexahedron.

806 boundary conditions of type 2 to 4. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

3.7 Extended second order real linear elliptic and parabolic equations with two degrees of freedom

Equation

The equations considered in this section are the same as in Section 3.6, however with some extra terms which define a coupling between the two equations. By this extension the equations can be used to solve the biharmonic equation as a set of two second order equations, if the boundary conditions allow so.

$$\rho^1(\frac{\partial c^1}{\partial t} + \mathbf{u}^{11}\cdot\nabla c^1) - \text{ div }(\alpha^{11}\nabla c^1) + \beta^{11}c^1 - \text{ div }(\alpha^{12}\nabla c^2) + \mathbf{u}^{12}\cdot\nabla c^2 + \beta^{12}c^2 = f^1 \quad (3.7.1)$$

$$\rho^2(\frac{\partial c^2}{\partial t} + \mathbf{u}^{22}\cdot\nabla c^2) - \text{ div }(\alpha^{22}\nabla c^2) + \beta^{22}c^2 - \text{ div }(\alpha^{21}\nabla c^1) + \mathbf{u}^{21}\cdot\nabla c^1 + \beta^{21}c^1 = f^2 \quad (3.7.2)$$

i.e.

$$\rho^k\frac{\partial c^k}{\partial t}+\rho^k\sum_{i=1}^{n}u_i^{kk}\frac{\partial c^k}{\partial x_i}-\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}\left(\alpha_{ij}^{kk}\frac{\partial c^k}{\partial x_j}\right)+\beta^{kk}c^k-\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial}{\partial x_i}\left(\alpha_{ij}^{kl}\frac{\partial c^l}{\partial x_j}\right)+\sum_{i=1}^{n}u_i^{kl}\frac{\partial c^l}{\partial x_i}+\beta^{kl}c^l=f^k \quad (3.7.3)$$

$$k=1,2; \qquad \mathbf{x}=(x_1,x_2,\ldots,x_n)\in\Omega\subset I\!R^n \text{ and } l\neq k$$

In the stationary case we have $\left(\frac{\partial c^k}{\partial t}=0\right)$
In this case the equation is elliptic, otherwise it is parabolic.
The coefficients ρ^k, $\mathbf{u}^{kl}$, α^{kl}, β^{kl} and f^k may depend on space and time and also of solutions of other problems. The following restrictions for the coefficients are required:

- $\rho^k > 0$.
- The matrices α^{kk} with coefficients α_{ij}^{kk} must be positive definite symmetric. In the extreme case α^{kk} may be equal to zero.

Upwinding

Upwinding has not yet been implemented.

Defect correction

Defect correction has not yet been implemented.

Boundary and initial conditions See Section 3.6

Input for the various subroutines

- Definition of the storage scheme:

  In general the matrices are not symmetric, only in very special occasions this may be the case.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
  For each element group 65 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 60 are real coefficients.

  These parameters and coefficients are defined as follows:

  1 not yet used
  2 not yet used
  3 type of numerical integration
  4 type of co-ordinate system
  5 not yet used

  real parameters with respect to equation 1, corresponding to unknown c^1

  6 α_{11}^{11}
  7 α_{12}^{11}
  8 α_{13}^{11}
  9 α_{22}^{11}
  10 α_{23}^{11}
  11 α_{33}^{11}
  12 u_1^{11}
  13 u_2^{11}
  14 u_3^{11}
  15 β^{11}
  16 f^1
  17 ρ^1
  18-20 not yet used

  real parameters with respect to equation 2, corresponding to unknown c^2

  21 α_{11}^{22}
  22 α_{12}^{22}
  23 α_{13}^{22}
  24 α_{22}^{22}
  25 α_{23}^{22}
  26 α_{33}^{22}
  27 u_1^{22}
  28 u_2^{22}

29 u_3^{22}
30 β^{22}
31 f^2
32 ρ^2
33-35 not yet used

real parameters with respect to equation 1, corresponding to unknown c^2

36 α_{11}^{12}
37 α_{12}^{12}
38 α_{13}^{12}
39 α_{22}^{12}
40 α_{23}^{12}
41 α_{33}^{12}
42 u_1^{12}
43 u_2^{12}
44 u_3^{12}
45 β^{12}
46-50 not yet used

real parameters with respect to equation 1, corresponding to unknown c^1

51 α_{11}^{21}
52 α_{12}^{21}
53 α_{13}^{21}
54 α_{22}^{21}
55 α_{23}^{21}
56 α_{33}^{21}
57 u_1^{21}
58 u_2^{21}
59 u_3^{21}
60 β^{21}
61-65 not yet used

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
The coefficients 6-65 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors. Of course, in 1D and 2D not all coefficients are used.
The default values for the real coefficients are zero, for coefficients 17 and 32 (ρ^k), however, the default value is one.

With respect to the parameters 1-5 the same choices may be made as in Section 3.1.1, except for the parts that have not yet been implemented.

- Parameters with respect to the linear solver:

  See the manual SEPRAN EXAMPLES Section 3.1.1.

- Computation of derivatives:

  The computation of derivatives has not yet been implemented.

- Types of integrals that may be computed:

  The computation of integrals has not yet been implemented.

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the second order elliptic equation in this section the following type numbers are available:

808 general type number for the internal elements. Defines the differential equation.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

- **shape = 1** linear line element.
- **shape = 2** quadratic line element.
- **shape = 3** linear triangle.
- **shape = 4** quadratic triangle.
- **shape = 5** bilinear quadrilateral.
- **shape = 6** biquadratic quadrilateral.
- **shape = 11** linear tetrahedron.
- **shape = 12** quadratic tetrahedron.
- **shape = 13** trilinear hexahedron.
- **shape = 14** triquadratic hexahedron.

806 boundary conditions of type 2 to 4 see Section 3.6

4 Elements for lubrication theory

In this chapter we discuss elements and examples for lubrication theory.

4.1 The Reynolds equation

In this section we consider elements for lubrication. The Reynolds equation describes the pressure in a film flow, where the film thickness h is assumed very small compared to the size of the bearing. One can show by dimensional analysis that in such a situation the continuity equation and the Navier-Stokes equations describing fluid flow may be approximated by the Reynolds equations. Such an approximation is commonly used in lubrication theory. We shall distinguish between an incompressible (oil) bearing and a compressible (air) bearing. At this moment only elements for the stationary case are available.

4.1.1 Equation

The Reynolds equation in the incompressible case reads:

$$\text{div } (-\frac{h^3}{12\mu}(\nabla p - \rho \mathbf{f}) + \frac{h}{2}(\mathbf{u}_1 + \mathbf{u}_2)) + h_t + k(p - p_0) = 0. \tag{4.1.1}$$

In the compressible case the Reynolds equation is given by:

$$\text{div } (-p\frac{h^3}{12\mu}(\nabla p - \frac{p}{RT}\mathbf{f}) + \frac{ph}{2}(\mathbf{u}_1 + \mathbf{u}_2)) + ph_t + k(p^2 - {p_0}^2) = 0. \tag{4.1.2}$$

Here we have assumed that the temperature is constant, hence an adiabatic situation is considered. The coefficients $h, \rho, h_t, ph_t, \mathbf{f}$ and p_0 may depend on space and time and also of solutions of other problems. k and RT must be constants. The following restrictions for the coefficients are required:

- $\rho, \mu, h, k, RT > 0$.

In a physical context the parameters have the following meaning:

h Film thickness

μ The dynamic viscosity

h_t or ph_t For stationary computations h_t or ph_t representing the time-dependent term

$\frac{\partial h}{\partial t}$ respectively $\frac{\partial ph}{\partial t}$ is treated as a squeeze term.

k A given constant

p_0 The reference pressure

$\mathbf{u}_1$ Velocity of "lower" surface.

$\mathbf{u}_2$ Velocity of "upper" surface.

p The pressure

RT The constant arising from the ideal gas law

Since the Reynolds equation is a special case of the general second order elliptic equation, also the elements described in Section 3.1 may be used.

4.1.2 Generalized Reynolds equation

A more general formulation of the Reynolds equation is due to Dowson (1962):

$$-\text{div } (F_2 \ \nabla p \ - \frac{F_3}{F_0}(\mathbf{U}_2 - \mathbf{U}_1)) = -h \text{ div } (\rho \mathbf{U})_2 \ + \ \int_0^h \frac{\partial \rho}{\partial t} dz \ + \ (\rho w)_2 - (\rho w)_1 \tag{4.1.1}$$

The subscript 1 refers to the bottom face and 2 to the top face.
U defines the velocity of these faces (x and y components).
w is the velocity in vertical direction.
The parameters F_0, F_1, F_2 and F_3 are given by

$$F_0 \ = \ \int_0^h \frac{1}{\eta} \ dz \tag{4.1.2}$$

$$F_1 \ = \ \int_0^h \frac{z}{\eta} \ dz \ = \ \bar{z} F_0 \tag{4.1.3}$$

$$F_2 \ = \ \int_0^h \frac{\rho z}{\eta} (z - \bar{z}) \ dz \tag{4.1.4}$$

$$F_2 \ = \ \int_0^h \frac{\rho z}{\eta} \ dz \tag{4.1.5}$$

$$\mu \ = \ \frac{\eta}{\rho} \tag{4.1.6}$$

$\bar{z}$ denotes the mean value of z, which is equal to $\frac{h}{2}$.

The horizontal velocities **u** can be computed by:

$$\mathbf{u} \ = \ \mathbf{U}_1 + \int_0^z \frac{z}{\eta} \ dz \ \nabla p \ = \ (\frac{\mathbf{U}_2 - \mathbf{U}_1}{F_0} \ - \ \bar{z} \nabla p) \ \int_0^z \frac{1}{\eta} \ dz \tag{4.1.7}$$

If η and ρ are constant these equations reduce to Equation (4.1.1).

The shear stress at bottom ($z = 0$) and top surface ($z = h$) can be derived from the standard formulas for shear stress and the expressions (4.1.7) for the velocity:

$$\tau \ = \ \nabla p (z - \bar{z}) \ = \ \frac{\mathbf{U}_2 - \mathbf{U}_1}{F_0} \tag{4.1.8}$$

4.1.3 Boundary Conditions

The following types of boundary conditions are available:

Type 1: p(**x**) given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2: $-\left(\frac{-h^3}{12\mu}(\nabla p - \rho \mathbf{f}) + \frac{h}{2}(\mathbf{u}_1 + \mathbf{u}_2)\right) \cdot \mathbf{n} + \sigma(\mathbf{x}) p(\mathbf{x}) = g(\mathbf{x}) \qquad (\sigma(\mathbf{x}) \geq 0)$
on some part of the boundary for the incompressible case and
$-\left(p\frac{-h^3}{12\mu}(\nabla p - \frac{p}{RT}\mathbf{f}) + p\frac{h}{2}(\mathbf{u}_1 + \mathbf{u}_2)\right) \cdot \mathbf{n} + \sigma(\mathbf{x}) p(\mathbf{x}) = g(\mathbf{x}) \qquad (\sigma(\mathbf{x}) \geq 0)$
for the compressible case.

This is a so-called natural boundary condition. In general boundary elements are necessary, except in the case that $\sigma(\mathbf{x}) = 0$ **and** $g(\mathbf{x}) = 0$, when there is no need to give any condition on this part of the boundary.

Type 3: Restrictor boundary condition

A restrictor may be simulated by a point boundary condition. The restrictor Q_r flow depends on the difference between the unknown restrictor pressure p_r and the given oil supply pressure p_s according to:

$$Q = \gamma(p_s - p_r) \text{ (linear relation) or}$$

$$Q = \gamma\sqrt{p_s - p_r} \text{ (non-linear relation)}$$

γ denotes the restriction coefficient.

The unknown pressure p_r is equal to the pressure computed by the Reynolds equation in the particular point. Application of this boundary condition, thus introduces a relation between oil supply pressure and local bearing pressure. If the non-linear relation is used the final equations become also non-linear.

4.1.4 Cavitation

In some cases the computed pressure may be below the cavitation pressure (usually 0 in the case of incompressible bearings). In that case we have to satisfy the constraint that the pressure can never be below the cavitation pressure. At this moment SEPRAN has two options to satisfy this constraint:

constraint in linear solver This solution is the most simple one. Instead of solving the system of linear equations in the classical sense, we use the constrained overrelaxation process as described in the Users Manual Section 3.2.8.
A clear disadvantage of this approach is that it is sometimes difficult to choose the parameters such that convergence is achieved.

Kumars mass conserving scheme According to the literature standard solvers for the Reynolds equations are not mass conserving in case of cavitation.
Kumar and booker (1994) describes an algorithm that, according to him guarantees mass conservation. Actually his method is meant for time dependent problems, but at this moment we have only implemented it for the stationary case. Therefore there is no need to define a density.
In this case we need the reaction force, in each step. The reaction force is used to detect if points satisfy $\frac{\partial \rho}{\partial t} < 0$. In fact for points with a positive reaction force this condition is satisfied.
So using this method requires the computation of a reaction force and hence the parameter IBCMAT in the input block matrix must be equal to 1.
If you want to apply Kumars algorithm, you need to add the line `solve_bearing` to the structure block and add a new block `BEARING` to the input file as described in the Users Manual Section 3.2.24.

4.1.5 Coefficients for the differential equation

The coefficients for the differential equation may be either defined in the structure block or alternatively in a coefficients block.
For the standard Reynolds element one may define the following coefficients:

TYPE_OF_BEARING = text (icoef1) Defines the type of bearing. If used in the structure block one must use a string between quotes to define the type otherwise in the coefficients

block the same string without the quotes.
Possible values for text are:

INCOMPRESSIBLE (0)

COMPRESSIBLE (1)

MASS_CONSERVING (2)

Default value: INCOMPRESSIBLE

LINEARIZATION = text (icoef2) Defines the type of linearization to be used in case of a compressible bearing.
Possible values for text are:

PICARD (0) A simple Picard iteration is used.

NEWTON (1) A (quadratic) Newton linearization is used. If it converges it is usually faster than Picard, but Picard is more robust.

Default value: PICARD

INTEGRATION_RULE = i (icoef3) defines the type of quadrature rule (between 0 and 4).

TRANSFORMATION = i (icoef4) i defines the type of transformation from R^3 to R^2.
Possible values:

0 no special transformation

1 (x, y, z) is transformed to (x, y)

2 (x, y, z) is transformed to (y, z)

3 (x, y, z) is transformed to (x, z)

4 (x, y, z) is transformed to (θ, z), where θ is defined by $\theta = \text{atan2}(y, x)$, i.e. θ defines the rotation along the surface of a cylinder.

The reason to define a transformation is that the velocity vectors have only two components and must be defined in the 2D system that is applied. Hence the transformation also defines the velocity vector. In the future this may be extended by defining the velocity vector in a three-dimensional setting.

LAYER_THICKNESS = h (coef6) defines the layer thickness. Usually h is a given vector.

VISCOSITY = μ (coef7) defines the viscosity.

SQUEEZE = ht (coef8) defines the squeeze term.

CONSTANT = k (coef9) defines the constant k in the equation.

REFERENCE_PRESSURE = $p0$ (coef10)

VELOCITY =v (coef11/12) defines the velocity vector of the lower surface.

U_VELOCITY = v_1 (coef11) defines the first component of this vector

V_VELOCITY = v_2 (coef12) defines the second component of this vector Either the complete vector or its components are given as input.

DENSITY = = rho (coef14)

RT =rt (coef15)

FORCE = f (coef16/17) defines an extra force vector if present.

X_FORCE = f_1 (coef16) defines the first component of this vector

Y_FORCE = f_2 (coef17) defines the second component of this vector

UPPER_FACE_VELOCITY (coef19/10) defines the velocity vector of the upper surface.

X_UPPER_FACE_VELOCITY (coef19) defines the first component of this vector

Y_UPPER_FACE_VELOCITY (coef20) defines the second component of this vector Either the complete vector or its components are given as input.

In case of the generalized Reynolds equation it is easier to use the element for the general second order elliptic equation (type 800), as treated in Section 3.1. We also need 20 coefficients, where the first 5 are integer and the last 15 reals.
Only the second and third integer coefficient are used. They have the same meaning for both types of elements The real coefficients must be filled as follows:

6 F_2 or $\frac{h^3}{12\mu}$ for the classical form

7-8 0

9 coef 6, hence identical to coefficient 6.

10-15 0

16 f (right-hand side).
In case of generalized Reynolds, put the term $-h\text{div } \nabla(\rho\mathbf{U})_2$ into this f.

17 ρ.
This term is not used since it refers to the convective terms only, so 0 may be used.

18 $\frac{F_3}{F_0}(u_2 - u_1)$, u is first component of velocity.

19 $\frac{F_3}{F_0}(v_2 - v_1)$, v is second component of velocity.

20 0

4.1.6 Coefficients for the natural boundary conditions

The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.

Input for boundary conditions of type 2

For each element group 15 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 10 are real coefficients. The input for the boundary conditions is exactly the same as in the manual SEPRAN EXAMPLES Section 3.1.1. However, actually only the first 7 parameters should be used for the Reynolds equation. The first one should be chosen equal to 0.

These 7 parameters and coefficients are defined as follows:

1 Type of natural boundary condition (use 0)
2 Not yet used (must be zero)
3 type of numerical integration
4 type of co-ordinate system
5 not yet used

6 σ
7 h

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP or FILCOF this means that no information about these coefficients has to be given.
The coefficients 6-15 may be zero, constants or functions as described in Section 10.1. They may also depend on precomputed vectors.

Input for boundary conditions of type 3

The restriction boundary conditions are implemented by use of a point element. This point element must be introduced as natural boundary condition in the input part of PROBLEM ... END.

For each element group 3 parameters and coefficients must be given. The first parameter is of integer type which means that it must be defined by ICOEFi in the input, the last 2 are real coefficients.

These 3 parameters and coefficients are defined as follows:

1	Type of restriction relation
2	Capillary restriction coefficient γ
3	Oil supply pressure p_s

With respect to the parameter 1 the following choices are available:

0:	linear relation
1:	non-linear relation, with Newton linearization

4.1.7 Parameters with respect to the linear solver

In the case of an incompressible bearing the matrix is not only symmetric but also positive definite.

In the case of a compressible bearing the system of equations is non-linear and hence a non-linear solver must be used. This may be done by program SEPCOMP if the option SOLVE_NONLINEAR_EQUATIONS is used, or by a standard non-linear solver from the Programmers Guide, like subroutine NONLIN.

4.1.8 Derivatives

Depending on the parameter ICHELD in subroutine DERIV various types of derivatives are computed. For a description of possible derivatives and the type of output vector the user is referred to Section 3.1.

Besides the possibilities mentioned in Section 3.1 for this type of element 4 extra values of ICHELD are implemented:

ICHELD=22 the so-called flow per nodal point is computed. The flow in a node is defined by the integral over the equation 4.1.1 or 4.1.2 multiplied by a basis function. In order to compute the flow DERIV must be used with ICHELD = 22, and the coefficients filled in exactly the same way as for subroutine BUILD. The computed flow must be approximately zero in the inner nodal points. To compute the flow through a part of the boundary, it is necessary to integrate the computed flow by the subroutine BOUNIV or by the option BOUNDARY_INTEGRAL in the input for SEPCOMP.

ICHELD=23 Flow vector is computed. This vector with 2 components is defined by (incompressible case):

$$-\frac{h^3}{12\mu}(\nabla p - \rho\mathbf{f}) + \frac{h}{2}(\mathbf{u}_1 + \mathbf{u}_2) \tag{4.1.1}$$

and in the compressible case by:

$$-p\frac{h^3}{12\mu}(\nabla p - \frac{p}{RT}\mathbf{f}) + \frac{ph}{2}(\mathbf{u}_1 + \mathbf{u}_2) \tag{4.1.2}$$

ICHELD=24 Shear stress at bottom surface, defined by

$$\tau = -\frac{h}{2}(\nabla p - \rho\mathbf{f}) + \frac{\mu}{h}(\mathbf{u}_2 - \mathbf{u}_1) \tag{4.1.3}$$

ICHELD=25 Shear stress at top surface.

$$\tau = \frac{h}{2}(\nabla p - \rho\mathbf{f}) + \frac{\mu}{h}(\mathbf{u}_2 - \mathbf{u}_1) \tag{4.1.4}$$

In case of elements of type 800 we need a different set of coefficients to compute the flow and shear stress. In that case 25 coefficients instead of 20 are needed. The first 5 are integer the last 20 are real.
Only the integer coefficients 3 and 4 are used.
The other coefficients must be filled as follows:

6-11	see the generalized Reynolds equation, given before in this section.
12-13	The horizontal components of the bottom velocity
14	0, not yet used
15	F_0
16	right-hand side f
17	0
18	ρ
19-21	see the generalized Reynolds equation, positions 18-20.
22	h
23-24	The horizontal components of the top velocity
25	not yet used (must be 0)

4.1.9 Integrals

If the user wants to compute integrals over the solution he may use subroutine INTEGR.

The type of integrals depends on the parameter ICHELi. For a description of possible integrals the user is referred to Section 3.1. For example to compute the load of the bearing it is necessary to use INTEGR or the option INTEGRAL in the input for SEPCOMP.
Mark that the extended definition of the type of co-ordinate system is also valid for the integrals.

4.1.10 Type numbers to be used in the problem input block

The use of the Laplace equation is indicated in the problem block by the name

```
reynolds
```

or alternatively by

```
type = 325
```

The natural boundary conditions do not have to be indicated by a type number, but internally type 801 is used.
A special type is 304 which is a point element for boundary conditions of type 3.

4.2 Coupled elasticity-flow interaction for a bearing (Reynolds equation coupled with mechanical elements)

In this section we consider the interaction of the oil pressure in the lubrication film and the surrounding material. An example of such an interaction is the elasto-hydrodynamic lubrication of an oil pumping ring seal as shown in Figure 4.2.1. Figure 4.2.2 shows the geometry of the pumping ring.

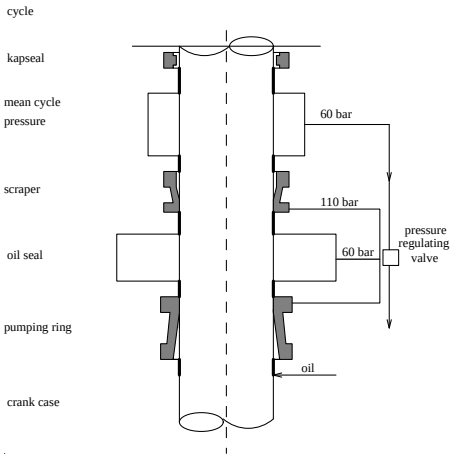


Figure 4.2.1: Assembly of a pumping ring and scraper

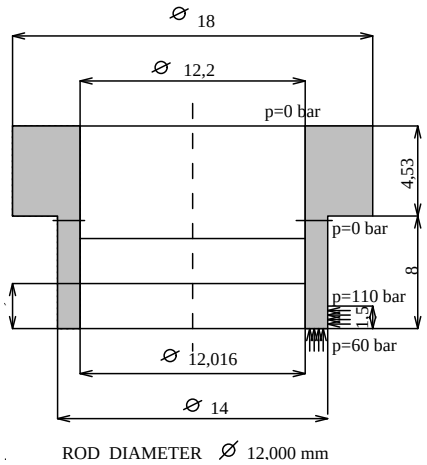


Figure 4.2.2: Geometry of the pumping ring

The pumping ring satisfies the stress-strain relations as described in Section 5.1 (axi-symmetric stress analysis). Due to the oil pressure in the film we have a distributed loading at the inner side of the ring given by:

$$T_r \;=\; -\,pT_z \;=\; -\,\frac{h}{2}\frac{\partial p}{\partial z} - \frac{\eta U}{h} \tag{4.2.1}$$

with h the oil film thickness

p the oil pressure

η the fluid dynamic viscosity, and

U the oil velocity in z-direction.

The film thickness is assumed to be so small that the pressure satisfies the one-dimensional Reynolds equation:

$$\frac{\partial h}{\partial t} - \frac{\partial}{\partial z}[\frac{h^3}{12\eta}\frac{\partial p}{\partial z} - \frac{Uh}{2}] + k(p - p_0) = 0 \tag{4.2.2}$$

with

p_0 the reference pressure and

k a given constant.

In stationary computations we have $\frac{\partial h}{\partial t} = 0$.

The pumping ring may be discretized by the elements given in Section 5.1. In this section we choose for the coupled approach, i.e. the distributed loadings combined with the Reynolds equation are discretized by a line element with 3 degrees of freedom. The first two degrees of freedom are the displacements in r and z direction, the third degree of freedom is the oil pressure. The oil pressure influences the distributed loading at the inner side of the ring, on the other hand the displacement fixes the film thickness h and hence the oil pressure. So we are faced with a strict non-linear problem.
In Section 5.3 the decoupled approach is treated, where the elasticity equations and the Reynolds equation are solved separately in an iteration process.

Boundary conditions for the pressure

The following types of boundary conditions are available:

Type 1: p(**x**) given in a point. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2: $-(\frac{-h^3}{12\eta}\frac{\partial p}{\partial z} - \frac{hU}{2}) = 0$ in an end point.
This is a so-called natural boundary condition. No boundary elements are necessary.

Type 3: $-\frac{-h^3}{12\eta}\frac{\partial p}{\partial z} - \frac{hU}{2} = q_z$ in an end point.
This is a so-called non-homogeneous natural boundary condition, requiring a point boundary element.

Input for the various subroutines (line and point element only

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input of program SEPCOMP or subroutine SEPSTR or alternatively by the parameter JMETOD in subroutine COMMAT.

  The system of equations is non-linear, hence it is necessary to use an iteration procedure to solve the system of non-linear equations. In each step of the linearization process a linearization must be applied. As a result the matrix is not symmetric anymore and METHOD = 2 or 6 must be chosen.

- Solution method:

  In the stationary case we have to solve a system of non-linear equations because of the non-linear line element. For the solution of these equations two linearization techniques are available:

  - Picard linearization (successive substitution)

  - Newton iteration

  In general Newton converges faster than Picard, however, in some problems the convergence region of Picard is larger than that of Newton.

  The iterative procedure is as follows:

  Start with an approximation $\mathbf{u}^0$, p^0. For example one can start with $\mathbf{u}^0 = \mathbf{0}$, $p^0 = 0$.

  Compute the solution of the system of non-linear equations by program SEPCOMP using the option SOLVE_NONLINEAR_EQUATIONS, or by a standard non-linear solver from the Programmers Guide, like subroutine NONLIN.

  For the time-dependent problem we do not only need to linearize the equations but also to discretize in time. For the linearization the same methods as in the stationary case may be used. No iteration is necessary for small time steps Δt. A useful procedure is the following modified θ-method:

$$\frac{h^{n+\theta} - h^n}{\theta \Delta t} - \frac{\partial p}{\partial z}\left[\frac{(h^{n+\theta})^3}{12\eta}\frac{\partial p^{n+\theta}}{\partial z} - \frac{U h^{n+\theta}}{2}\right] + k(p^{n+\theta} - p_0) = 0 \tag{4.2.3}$$

  with

$$h^{n+1} = \frac{1}{\theta} h^{n+\theta} - \frac{1-\theta}{\theta} h^n \tag{4.2.4}$$

$$p^n + 1 = \frac{1}{\theta} p^{n+\theta} - \frac{1-\theta}{\theta} p^n \tag{4.2.5}$$

$$\mathbf{u}^{n+1} = \frac{1}{\theta} \mathbf{u}^{n+\theta} - \frac{1-\theta}{\theta} \mathbf{u}^n \tag{4.2.6}$$

  and

  n the time level, $0 < \theta \leq 1$.

  For $\theta >= \frac{1}{2}$ the θ-method is unconditionally stable. $\theta = \frac{1}{2}$ is the most accurate one (Crank-Nicolson), $\theta = 1$ has the best damping properties.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 6 parameters and coefficients must be given. Parameter 6 is of integer type which means that it must be defined by ICOEFi in the input, the other ones are real coefficients.

  These parameters and coefficients are defined as follows:

1	the diameter of the rod (D)
2	the fluid dynamic viscosity (η)
3	the constant k in the equation

4 the reference pressure p_0
5 the velocity U
6 METHOD

These parameters may be zero, constants or functions as described in Section 10.1. They may also depend on precomputed vectors.

The oil film thickness h^n is computed from: $h^n = r - D/2 + u_r$, where u_r is the displacement in r-direction.

With respect to the parameters the following choices are available:

Possible values:
1: stationary case, Picard iteration, shear stress T_z is taken into account
2: stationary case, Newton iteration, shear stress T_z is taken into account
3: instationary case, Picard iteration, shear stress T_z is taken into account
4: instationary case, Newton iteration, shear stress T_z is taken into account
11: stationary case, Picard iteration, shear stress T_z is not taken into account
12: stationary case, Newton iteration, shear stress T_z is not taken into account
13: instationary case, Picard iteration, shear stress T_z is not taken into account
14: instationary case, Newton iteration, shear stress T_z is not taken into account

Besides these parameters also common block CTIME is used as described in the Programmers Guide Section 7.2. Of these parameters TSTEP and THETA are used explicitly.

- Definition of the coefficients for the boundary conditions:

  The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 1 coefficient must be given.

  This coefficient is defined as follows:

  1 q_z

  At the initial point ($z = z_min$), the coefficient must be equal to $-q_z$, at the end point ($z = z_max$), the coefficient must be equal to q_z.

- Storage of the solution vector:

  u and p in the nodal points.

Definition of type numbers

The type numbers, which are given in the standard input file in the part between PROBLEM and END (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved.

At this moment only one-dimensional elements are available as well as the point element for the boundary conditions.

The following type numbers are available:

302 type number for linear line elements. Defines the differential equation.

303 point element for boundary conditions of type 3.

4.3 Decoupled elasticity-flow interaction for a bearing (Reynolds equation coupled with mechanical elements)

In Section 4.2 the coupled solution of the elasticity equations and the Reynolds equations has been treated. For that coupled approach it was necessary to develop a special line element that contained the Reynolds equation and the boundary conditions for the elasticity equations in one element. A clear disadvantage of this approach that it requires new elements for each shape of element. An advantage, however, is of course, that the convergence may be much faster than in a decoupled approach. Nevertheless we shall consider the decoupled approach in this section.

We consider an elastic medium that is lubricated by some lubricant. A typical example is given in the manual SEPRAN EXAMPLES Section 4.3.1. The pressure in the lubrication film and the deformation of the medium are mutually dependent. The medium will deform due to the hydrostatic pressure in the film and the pressure depends on the local film height. This film height is a function of the deformation.

In order to solve this coupled problem we perform the following algorithm:

Start with a given position of the elastic medium

Compute the corresponding film height

While not converged do

- Solve the Reynolds equation in the lubrication film using the just computed film height
- Solve the elasticity equations using the just computed pressure as a given load
- Recompute the film height from the computed deformation

5 Mechanical elements

In this chapter some mechanical elements are described.
The following sections are available:

5.1 is devoted to standard linear elastic problems.

5.2 treats incompressible or nearly incompressible elasticity.

5.3 is devoted to non-linear elasticity problems.

5.4 deals with (thick) plates.

5.1 Linear elastic problems

In this section linear elastic elements are described. The equations are based on:

- The conservation of momentum.
- A relation between strain and displacements.
- The stress-strain relation.

Incompressible or nearly incompressible elements are the subject of Section (5.2).
Non-linear cases are treated in Section (5.3).
Plate elements can be found in Section (5.4).

In Section (5.1.1) the equations are given.
Boundary and initial conditions are treated in subsection (5.1.2)
The representation of the equations in various co-ordinate systems is given in subsection (5.1.10)
Temperature dependence is the subject of subsection (5.1.3)
Subsection (5.1.4) deals with solution methods.
The definition of the input of the coefficients for elastic equations is given in subsection (5.1.5) and for the natural boundary conditions in subsection (5.1.6)
Subsection (5.1.7) deals with the computation of derived quantities, like gradient, stress, strain and so on.
In subsection (5.1.8) the available type numbers are presented.
Finally in subsection (5.1.9) some information of previous versions of SEPRAN with respect to the linear elastic equations is given.

5.1.1 Linear elastic equations

In this section linear elastic elements are described. It concerns two-dimensional elements (plane stress and plane strain), three-dimensional elements and axi-symmetrical elements.

Equation

The total strain at any point can be defined by the components of strain which contribute to internal work.

Basic equations

$$\frac{\partial^2 \rho \mathbf{u}}{\partial t^2} - \ div\boldsymbol{\sigma} \ = \ \mathbf{F} \tag{5.1.1.1}$$

$\boldsymbol{\sigma}$ denotes the stress tensor and $\mathbf{F}$ a body force acting on a unit volume of material.
In order to express the stresses in displacements it is necessary to define the strain-displacement relations and the constitutive equations, which define a relation between stresses and strains.

This relation is usually expressed as, (see [1]):

$$\boldsymbol{\epsilon} \ = \ \mathbf{B} \ \mathbf{u} \tag{5.1.1.2}$$

Where $\mathbf{B}$ represents the transpose of the divergence operator.

The stress-strain relations are defined by the so-called constitutive equations, which for linear elasticity is given by a direct linear relation. The constitutive relations read in general form:

$$\boldsymbol{\sigma} = \mathbf{D}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon_0}) + \boldsymbol{\sigma_0} \tag{5.1.1.3}$$

$\boldsymbol{\epsilon_0}$ the initial strain and $\boldsymbol{\sigma_0}$ the initial stress. where $\boldsymbol{D}$ denotes the elasticity matrix, $\boldsymbol{\epsilon_0}$ the initial strain and $\boldsymbol{\sigma_0}$ the initial tensor.

In case of a visco-elastic problem we do not use the second order time derivative in Equation (5.1.1.1) but the stress tensor σ is extended with an extra time-dependent term:

$$\sigma = \mu_1 \frac{\partial \epsilon}{\partial t} \ + \ \mu_2 \frac{\partial \mathrm{div}\mathbf{u}}{\partial t}\mathbf{I} \ + \ \mathbf{D}(\epsilon - \epsilon_0) + \sigma_0 \tag{5.1.1.4}$$

Mark that although the equations are formulated in terms of the stress tensor, the unknown in the problems in this section is the displacement. The stress tensor itself is a derived quantity.

Loadings

The user may introduce loadings in the following way:

i body forces by F_x and F_y

ii concentrated loads in user defined points (using point elements)

iii distributed loadings on a part of the boundary given. In the 2D case these body forces are given by:
$$\boldsymbol{\sigma_x n_x} + \boldsymbol{\tau_{xy} n_y} = \boldsymbol{T_x} \qquad \boldsymbol{\sigma_y n_y} + \boldsymbol{\tau_{xy} n_x} = \boldsymbol{T_y}$$
where $\boldsymbol{n_x}$ and $\boldsymbol{n_y}$ denote the cosine of the (unit) normal at the boundary in the x and y-direction respectively.
T_x, T_y denote the component of the boundary force per unit area in x and y-direction respectively.
In the 3D case the trivial extension of these boundary forces may be applied.

5.1.2 Boundary and initial conditions for the linear elastic equations

In the instationary case it is necessary to give an initial displacement and initial velocity at $t = t_0$. However, in the visco-elastic case we have only first order time-derivatives and hence we need only an initial displacement.

The following types of boundary conditions are available:

Type 1 One or two or all components of the displacement given at a boundary

Type 2 External loadings given at a boundary, see *loadings*.
As extra option it is possible to define a linear relation between displacements and loadings at the boundary in the following way:

$$LOAD_i \;=\; T_i \;+\; \sum_{j=1}^{n} \alpha_{ij} u_j \tag{5.1.2.1}$$

In the three-dimensional case the external loads may only be given in the direction of the co-ordinate axis, or if local transformations are used in the direction of the local co-ordinate axis.
In the two-dimensional case the external boundary loads may be given in two different ways. These ways are defined by the parameter ILOAD as follows:

ILOAD=2 the load must be given in the direction of local co-ordinate axis. Normally this means in the x respectively y-direction. In the case of local transforms this means in the direction of the normal and tangential vectors at the boundary.

ILOAD=3 the load must be given by the parameters T and θ, where θ denotes the angle with the x-axis and T the length of the load.

In the axi-symmetrical case also the parameter ILOAD is defined. In that case θ defines the angle with the r-axis.

ILOAD=4 the load must be given in normal and tangential direction. The direction of the element defines the outward normal.

It is in any case necessary to prescribe exactly two boundary conditions at each boundary in 2D and three in 3D. The boundary conditions must be given in two (three) independent directions. However, if at a boundary for one direction no boundary condition is given, this means that at that position the displacement is free and there is external loading. In fact this means that implicitly a natural boundary condition is applied.

5.1.3 Temperature dependence

According to [1], temperature changes will in general result in an initial strain vector. The relation between temperature rise θ and the initial strain depends on the thermal expansion coefficient α and the type of material. The following relations may be used.

Plane stress-isotropic material:

$$\boldsymbol{\epsilon_0} = \begin{bmatrix} \alpha\theta \\ \alpha\theta \\ 0 \end{bmatrix} \tag{5.1.3.1}$$

3D isotropic material:

$$\boldsymbol{\epsilon_0} = \begin{bmatrix} \alpha\theta \\ \alpha\theta \\ \alpha\theta \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.1.3.2}$$

Plane strain isotropic material:

$$\boldsymbol{\epsilon_0} = (1+\nu) \begin{bmatrix} \alpha\theta \\ \alpha\theta \\ 0 \end{bmatrix} \tag{5.1.3.3}$$

Plane stress-orthotropic material:

$$\boldsymbol{\epsilon_0} = \begin{bmatrix} \alpha_1\theta \\ \alpha_2\theta \\ 0 \end{bmatrix} \tag{5.1.3.4}$$

Plane strain-orthotropic material:

$$\boldsymbol{\epsilon_0} = (1+\nu) \begin{bmatrix} \alpha_1\theta \\ \alpha_2\theta \\ 0 \end{bmatrix} \tag{5.1.3.5}$$

3D Axi-symmetrical isotropic material:

$$\boldsymbol{\epsilon_0} = \begin{bmatrix} \alpha\theta \\ \alpha\theta \\ \alpha\theta \\ 0 \end{bmatrix} \tag{5.1.3.6}$$

3D Axi-symmetrical stratified material:

$$\boldsymbol{\epsilon_0} = \begin{bmatrix} \alpha_z\theta \\ \alpha_r\theta \\ \alpha_r\theta \\ 0 \end{bmatrix} \tag{5.1.3.7}$$

with α_z the thermal expansion coefficient in axial direction and α_r in the plane normal to it.

5.1.4 Solution methods for the linear elastic equations

Definition of the storage scheme:

The matrices that are created are symmetric and positive definite. Hence in the input block matrix one can use the keyword `symmetric`, which reduces the amount of storage by a factor 2.

Time dependence

In the time dependent case a mass matrix and a stiffness matrix is formed. These matrices are both symmetric and positive definite.
In the elastic case we are dealing with second order time-derivatives and it is necessary to use a time integration scheme that is suitable for this type of equations. Recommended methods are for example the Newmark scheme of the generalized α scheme, see the Users Manual Section 3.2.15. An example is treated in the manual Examples Section 5.1.3.

In the visco-elastic case we have order time derivatives, which implies that standard methods like Euler Implicit or Crank Nicolson may be used.

5.1.5 Definition of the coefficients for the linear elasticity input

The user may provide the following information either in a coefficients block or in the structure block.

STRESS_STRAIN_RELATION = *name* (icoef2) type of stress-strain relations.
name is a string parameter with the following possible values:

PLANE_STRESS (0) plane stress, isotropic material or 3D stress, isotropic material (Cartesian).
This is the default value.

PLANE_STRAIN (1) plane strain, isotropic material (Cartesian)

AXI_SYMMETRIC_STRESS (2) axi-symmetrical stress, isotropic material

ORTHOTROPIC_PLANE_STRESS (3) plane stress, orthotropic material

ORTHOTROPIC_PLANE_STRAIN (4) plane strain, orthotropic material

STRATIFIED_AXI_SYMMETRIC_STRESS (5) axi-symmetrical stress, stratified material

USER_DEFINED_ELASTICITY_CART (6) user defined elasticity matrix (3x3) (Cartesian)

USER_DEFINED_ELASTICITY_AXI (7) user defined elasticity matrix (4x4) (Axi-symmetric)

INTEGRATION_RULE = i (icoef3) defines the type of integration rule to be applied.
This is an integer coefficient, with the following possible values for i:

0 the rule is chosen by the element itself (Default)

> 0 the integration rule is defined by the user, see below

ELASTIC_MODULUS (coef6) E

POISSONS_RATIO (coef7) ν

THICKNESS (coef27) thickness of the plate: h

FORCE (coef28/29/30) body force

X_FORCE (coef28) F_x (body force in x-direction, or in the axi-symmetrical case r-direction)

Y_FORCE (coef29) F_y (body force in y-direction, or in the axi-symmetrical case z-direction)

Z_FORCE (coef30) F_z (body force in z-direction, 3D only)

INITIAL_STRAIN (coef31 .. 36) initial strain

XX_INITIAL_STRAIN (coef31)

XY_INITIAL_STRAIN (coef32)

XZ_INITIAL_STRAIN (coef33)

YY_INITIAL_STRAIN (coef34)

YZ_INITIAL_STRAIN (coef35)

ZZ_INITIAL_STRAIN (coef36)

INITIAL_STRESS (coef37..42) initial stress

XX_INITIAL_STRESS (coef37)

XY_INITIAL_STRESS (coef38)

XZ_INITIAL_STRESS (coef39)

YY_INITIAL_STRESS (coef40)

YZ_INITIAL_STRESS (coef41)

ZZ_INITIAL_STRESS (coef42)

DENSITY (coef43) density ρ

5.1.6 Definition of the coefficients for the natural boundary conditions input

The user may provide the following information either in a coefficients block or in the structure block.

ILOAD (icoef1) Type of natural boundary condition (ILOAD)

0,2	ILOAD = 2
3	ILOAD = 3
4	ILOAD = 4

STRESS_STRAIN_RELATION = *name* (icoef2) type of stress-strain relations.
name is a string parameter with the same meaning as in Section 5.1.5.

INTEGRATION_RULE = i (icoef3) defines the type of integration rule to be applied.
This is an integer coefficient, with the following possible values for i:

0	the rule is chosen by the element itself (Default)
> 0	the integration rule is defined by the user, see below

X_LOAD (coef6) T_x

Y_LOAD (coef6) T_y

Z_LOAD (coef8) T_z

THICKNESS (coef9) h.
If omitted the value of the differential equation is copied.

5.1.7 Computation of quantities derived from the solution of the linear elastic equations

Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

1 $\frac{\partial u_j}{\partial x_i}$, where x_i is defined by the parameter ix, and j by the parameter JDEGFD.

2 $\nabla\mathbf{u}$

3 -$\nabla\mathbf{u}$

4 *div* **u**

5 *curl* **u**

6 The stress σ, according to 5.1.10.1 to 5.1.10.3

7 The strain ϵ, according to 5.1.10.1 to 5.1.10.3.

12-17 See types 2-7 but now defined per element, i.e. constant values per element. The output vector is vector of special structure defined per element (type 116).

32-37 See types 2-7 but now defined per node per element.
At this moment only ICHELD = 36 and 37 have been implemented.
The output vector is vector of special structure defined per node per element (type 126).

42-47 See types 2-7 but now defined per integration point per element.
At this moment only ICHELD = 46 and 47 have been implemented.
The output vector is vector of special structure defined per integration point per element (type 129).

The output vector is defined as follows:

1,4 a vector of the type solution vector with one unknown per point

2,3 a vector of the type vector of special structure with $ndim^n dim$ unknowns per point

5 a vector of the type solution vector with 0, 1 or 3 unknowns per point if *ndim* is respectively 1, 2 or 3

11 a vector of special structure defined per element with one unknown

6,7 a vector of the type solution vector with six unknowns per point

In the cases ICHELD = 6, 7, 16, 17, 36, 37, 46 and 47 the user must define exactly the same coefficients as for the differential equation, except for the parameter ρ, which is not used.

5.1.8 Type numbers available for the linear elastic equations

The type numbers, which are given in the standard input file in the part between PROBLEM and END (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved.

For the linear elasticity equation in this section the following type numbers are available:

250 general type number for the internal elements. Defines the differential equation.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

shape = 7 7 point triangle. This element is treated as shape 4, i.e. there are no degrees of freedom in the center.

shape = 11 linear tetrahedron.

shape = 12 quadratic tetrahedron.

shape = 13 trilinear hexahedron.

shape = 14 triquadratic hexahedron.

251 boundary conditions of types 2 and 3.
The element may be either a point element for a point load or a boundary element for a distributed load. The boundary elements must be curve elements in 2D and surface elements in 3D. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 0 Point element. (point element for concentrated load)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 3 linear triangle.

shape = 4 quadratic triangle.

shape = 5 bilinear quadrilateral.

shape = 6 biquadratic quadrilateral.

5.1.9 Previous versions of SEPRAN dealing with the linear elastic equations

In previous versions of SEPRAN equation (5.1.1.1) has been solved with different type numbers. These type numbers may still be used, however, it is recommended to use the new type numbers described earlier when creating new input or new programs. In this section we will point out the differences of the previous type numbers and the present one.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 5 or 6 parameters and coefficients must be given.

  These parameters and coefficients are defined as follows:

  1 IGPROB with:
  IGPROB=1 : plane stress
  IGPROB=2 : plane strain
  IGPROB=3 : axi-symmetrical stress

  2 E

  3 ν

  4 F_x or F_r

  5 F_y or F_z

6 h (plane stress and strain only)

- Definition of the coefficients for the boundary conditions:

  The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 4 or 5 parameters and coefficients must be given.

  These parameters and coefficients are defined as follows:

  1 IGPROB

  2 ILOAD. Possible values:

  - **1** The load **T** is given in the co-ordinate direction
  - **3** The load T is given in one direction and the angle θ between this direction and x-direction is given.

  3 T_x or T_r if ILOAD=1 or T if ILOAD=3

  4 T_y or T_z if ILOAD=1 or θ if ILOAD=3

  5 h (plane stress and strain only)

- Computation of derivatives:

  Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

  1: The stress σ, according to 5.1.10.1 to 5.1.10.3

Definition of type numbers

The type numbers, which are given in the standard input file in the part between PROBLEM and END (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved.

For the linear elasticity equation in this section the following type numbers are available:

206 Plane stress or plane strain for linear triangle in R^2.

207 Plane stress or plane strain for bilinear quadrilateral in R^2.

208 Boundary condition of type 2 for type numbers 206 and 207

209 Point element for concentrated load.

5.1.10 Representation of the linear elasticity equations for various coordinate systems

Although the stress tensor, the strain tensor and the displacement vector itself are independent on the co-ordinate system, their representation in various co-ordinate systems varies. The following representations are presently available for this element:

2D Cartesian co-ordinates:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \end{bmatrix} \tag{5.1.10.1}$$

3D Cartesian co-ordinates:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_z}{\partial z} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\ \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \\ \frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} \end{bmatrix} \tag{5.1.10.2}$$

3D Axi-symmetrical co-ordinates:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_r \\ \sigma_z \\ \sigma_\theta \\ \tau_{rz} \end{bmatrix} \quad \mathbf{u} = \begin{bmatrix} u_r \\ u_z \end{bmatrix} \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_r \\ \epsilon_z \\ \epsilon_\theta \\ \gamma_{rz} \end{bmatrix} = \begin{bmatrix} \frac{\partial u_r}{\partial r} \\ \frac{\partial u_z}{\partial z} \\ \frac{u_r}{r} \\ \frac{\partial u_r}{\partial z} + \frac{\partial u_z}{\partial r} \end{bmatrix} \tag{5.1.10.3}$$

The equation in these co-ordinate systems get the following shapes:

2D Cartesian co-ordinates

$$\frac{\partial^2 \rho u_x}{\partial t^2} - \frac{\partial \sigma_x}{\partial x} - \frac{\partial \tau_{xy}}{\partial y} = F_x \tag{5.1.10.4}$$

$$\frac{\partial^2 \rho u_y}{\partial t^2} - \frac{\partial \sigma_y}{\partial y} - \frac{\partial \tau_{xy}}{\partial x} = F_y \tag{5.1.10.5}$$

3D Cartesian co-ordinates

$$\frac{\partial^2 \rho u_x}{\partial t^2} - \frac{\partial \sigma_x}{\partial x} - \frac{\partial \tau_{xy}}{\partial y} - \frac{\partial \tau_{zx}}{\partial z} = F_x \tag{5.1.10.6}$$

$$\frac{\partial^2 \rho u_y}{\partial t^2} - \frac{\partial \tau_{xy}}{\partial x} - \frac{\partial \sigma_y}{\partial y} - \frac{\partial \tau_{yz}}{\partial z} = F_y \tag{5.1.10.7}$$

$$\frac{\partial^2 \rho u_z}{\partial t^2} - \frac{\partial \tau_{zx}}{\partial x} - \frac{\partial \tau_{yz}}{\partial y} - \frac{\partial \sigma_z}{\partial z} = F_z \tag{5.1.10.8}$$

3D Axi-symmetrical co-ordinates

$$\frac{\partial^2 \rho u_r}{\partial t^2} - \frac{\partial \sigma_r}{\partial r} - \frac{\partial \tau_{rz}}{\partial z} - \frac{\sigma_r - \sigma_\theta}{r} = F_r \tag{5.1.10.9}$$

$$\frac{\partial^2 \rho u_z}{\partial t^2} - \frac{\partial \sigma_z}{\partial z} - \frac{\partial \tau_{rz}}{\partial z} - \frac{\tau_{rz}}{r} = F_z \tag{5.1.10.10}$$

Elasticity matrix $\mathbf{D}$

The constitutive equations depend on the type material properties. These properties are translated into the elasticity matrix $\mathbf{D}$. For this element there are several types of elasticity matrices available.

Plane stress-isotropic material:

$$\mathbf{D} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{5.1.10.11}$$

where E denotes Young's modulus and ν Poisson's ratio.

3D isotropic material:

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{5.1.10.12}$$

where E denotes Young's modulus and ν Poisson's ratio.

Plane strain isotropic material:

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{5.1.10.13}$$

Plane stress-orthotropic material:

$$\mathbf{D} = \frac{E_2}{1-n\nu_2^2}\begin{bmatrix} n & n\nu_2 & 0 \\ n\nu_2 & 1 & 0 \\ 0 & 0 & m(1-n\nu_2^2) \end{bmatrix} \tag{5.1.10.14}$$

with $n = \frac{E_1}{E_2}$, $m = \frac{G_2}{E_2}$.

Plane strain-orthotropic material:

$$\mathbf{D} = \frac{E_2}{(1+\nu_1)(1-\nu_1-2n{\nu_2}^2)}\begin{bmatrix} n(1-n{\nu_2}^2) & n\nu_2(1+\nu_1) & 0 \\ n\nu_2(1+\nu_1) & (1-{\nu_1}^2) & 0 \\ 0 & 0 & m(1+\nu_1)(1-\nu_1-2n{\nu_2}^2) \end{bmatrix} \tag{5.1.10.15}$$

with $n = \frac{E_1}{E_2}$, $m = \frac{G_2}{E_2}$.

3D Axi-symmetrical stratified material:

$$\mathbf{D} = \frac{E_2}{(1+\nu_1)(1-\nu_1-2n\nu_2^2)}\begin{bmatrix} 1-{\nu_1}^2 & n\nu_2(1+\nu_1) & n\nu_2(1+\nu_1) & 0 \\ n\nu_2(1+\nu_1) & n(1-n{\nu_2}^2) & n(\nu_1+n{\nu_2}^2) & 0 \\ n\nu_2(1+\nu_1) & n(\nu_1+n{\nu_2}^2) & n(1-n{\nu_2}^2) & 0 \\ 0 & 0 & 0 & m(1+\nu_1)(1-\nu_1-2n{\nu_2}^2) \end{bmatrix} \tag{5.1.10.16}$$

with $n = \frac{E_1}{E_2}$, $m = \frac{G_2}{E_2}$.

3D Axi-symmetrical isotropic material:

$$\mathbf{D} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \tag{5.1.10.17}$$

with $n = \frac{E_1}{E_2}$, $m = \frac{G_2}{E_2}$.

User defined elasticity matrix:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}, \tag{5.1.10.18}$$

where $d_{ij} = d_{ji}$.

Thickness of the plate

In the 2D applications (plane strain and plane stress) the thickness of the solid h may be linear variable in the element, see Figure 5.1.10.1. The finite element equations are integrated over the local thickness of the plate. At this moment the average thickness per element is used for the integration.

Remark
If Poisson's ratio ν approaches 0.5, the standard displacement formulation given in this section fails. In that case it is necessary to use the elements described in Section 5.2, concerning (nearly) incompressible materials.

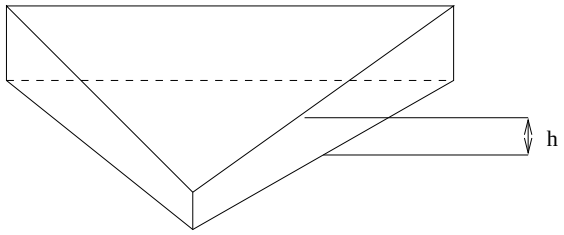


Figure 5.1.10.1: membrane element with thickness h

5.1.11 Old definition of the coefficients for the linear elasticity input

These parameters and coefficients are defined as follows:

1 not yet used

2 type of stress-strain relations

3 type of numerical integration

4 not yet used

5 not yet used

6-26 information about the elasticity matrix depending on the type of stress-strain relations

27 thickness of the plate: h

28 F_x (body force in x-direction, or in the axi-symmetrical case r-direction)

29 F_y (body force in y-direction, or in the axi-symmetrical case z-direction)

30 F_z (body force in z-direction, 3D only)

31-36 initial strain

37-42 initial stress

43 density ρ

44 parameter mu_1 for the visco-elastic model.

45 parameter mu_2 for the visco-elastic model.

The parameters that are not used must be set equal to zero, since this is the default value in case of future extensions.

With respect to these parameters the following choices are available:

2 Type of stress-strain relations (IGPROB)
Possible values:

0 plane stress, isotropic material or 3D stress, isotropic material (Cartesian)

1 plane strain, isotropic material (Cartesian)

2 axi-symmetrical stress, isotropic material

3 plane stress, orthotropic material

4 plane strain, orthotropic material

5 axi-symmetrical stress, stratified material

6 user defined elasticity matrix (3x3) (Cartesian)

7 user defined elasticity matrix (4x4) (Axi-symmetric)

3 Type of numerical integration
Possible values:

0 default value defined by element

6-26 Information about the elasticity matrix
The following values are needed depending on IGPROB:

IGPROB = 0, 1 or 2:
coefficient:

6 E

7 ν

IGPROB = 3:
coefficient:

6 E_1

7 E_2

8 ν_2

9 ν_2

10 G_2

IGPROB = 4, 5:
coefficient:

6 E_1

7 E_2

8 ν_1

9 ν_2

10 G_2

IGPROB = 6:
coefficient:

6 d_{11}

7 d_{21}

8 d_{31}

9 d_{22}

10 d_{23}

11 d_{33}

IGPROB = 7:
coefficient:

6 d_{11}

7 d_{21}

8 d_{31}

9 d_{41}

10 d_{22}

11 d_{32}

12 d_{42}

13 d_{33}

14 d_{43}

15 d_{44}

31-36 Initial strain:
coefficient:

31 $(\epsilon_x)_0$ or $(\epsilon_r)_0$

32 $(\epsilon_y)_0$ or $(\epsilon_z)_0$

33 $(\epsilon_z)_0$ or $(\epsilon_\theta)_0$

34 $(\gamma_{xy})_0$ or $(\gamma_{rz})_0$

35 $(\gamma_{yz})_0$

36 $(\gamma_{zx})_0$

If the initial strain tensor is stored as a vector of special structure defined per element per node (type 126), then this vector should be provided as coefficient 31. Once a vector of type 126 is recognized it is assumed that the whole tensor is stored in this vector and the coefficients 32 to 36 are skipped.
The same is the case for vectors of type 129 (defined per integration point per element).

37-42 Initial stress:
coefficient:

37 $(\sigma_x)_0$ or $(\sigma_r)_0$

38 $(\sigma_y)_0$ or $(\sigma_z)_0$

39 $(\sigma_y)_0$ or $(\sigma_\theta)_0$

40 $(\tau_{xy})_0$ or $(\tau_{rz})_0$

41 $(\tau_{yz})_0$

42 $(\tau_{zx})_0$

If the initial stress tensor is stored as a vector of special structure defined per element per node (type 126), then this vector should be provided as coefficient 37. Once a vector of type 126 is recognized it is assumed that the whole tensor is stored in this vector and the coefficients 38 to 42 are skipped.
The same is the case for vectors of type 129 (defined per integration point per element).

In case of the visco-elastic model we have to increase the integer coefficient 2 by 200.
Hence icoef2 = 200, means visco-elastic model and plane stress.
icoef2 = 201, means visco-elastic model and plane strain.

5.1.12 Old definition of the coefficients for the natural boundary conditions input

The coefficients for the boundary conditions may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
For each element group 25 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 20 are real coefficients.

These parameters and coefficients are defined as follows:

1 Type of natural boundary condition (ILOAD)

2 type of stress-strain relations (IGPROB, see differential equation)

3 type of numerical integration

4 Not yet used (must be zero)

5 not yet used

6 T_x

7 T_y

8 T_z

9 h

10 α_{11}

11 α_{12}

12 α_{13}

13 α_{22}

14 α_{23}

15 α_{33}

16-25 not yet used

Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP or FILCOF this means that no information about these coefficients has to be given.
The coefficients 6-15 may be zero, constants or functions as described in Section 10.1. They may also depend on precomputed vectors. Of course, in 1D and 2D not all coefficients are used. The coefficients 8-13 are only used for boundary conditions of type 4.

With respect to the parameters 1-5 the following choices are available:

1 Type of natural boundary condition.
Possible values:

0,2 ILOAD = 2

3 ILOAD = 3

4 ILOAD = 4

3 Type of numerical integration rules.
Possible values:

0 the rule is chosen by the element itself (Default)

> 0 the integration rule is defined by the user.

5.2 Linear incompressible or nearly incompressible elastic problems

When Poisson's ratio ν approaches 0.5 or when the material becomes incompressible, the standard displacement formulation of elastic problems as described in Section 5.1 fails.
Even when the material becomes nearly incompressible, with $\nu > 0.4$ sometimes problems may be expected, for example for the linear triangle. This is for example the case if we use rubber-like materials.

Once a material becomes (nearly) incompressible it is better to use a so-called mixed formulation. To that it is convenient to separate the mean stress of pressure from the total stress field and treat it as an independent variable.

In this section we shall concentrate ourselves to this kind of problems.

Equation

Basic equations

The mean stress of pressure is given by

$$p = \frac{\sigma_x + \sigma_y + \sigma_z}{3} \tag{5.2.1}$$

Remark:

Equation 5.2.1 is only true for materials where the pressure can be defined by splitting the stress into a hydrostatic and a deviatoric stress. In many cases this is not possible, but for some material models is is, for example for Neo-Hookean and Mooney models.
The rest of the equations in this section, however, remain valid.

The pressure is related to the volumetric strain, ε_v, by the bulk modulus of the material K, for isotropic behaviour:

$$\varepsilon_v = \varepsilon_x + \varepsilon_y + \varepsilon_z = \frac{p}{K} \tag{5.2.2}$$

For an incompressible material $K = \infty$ and the volumetric strain is zero.
The deviatoric strain ε_d is defined by

$$\boldsymbol{\varepsilon}_d = \boldsymbol{\varepsilon} - \frac{\mathbf{m}\varepsilon_v}{3} = (\mathbf{I} - \frac{1}{3}\mathbf{m}\mathbf{m}^T)\boldsymbol{\varepsilon} \tag{5.2.3}$$

with $\mathbf{m}$ given by

$$\mathbf{m}^T = [1, 1, 1, 0, 0, 0] \tag{5.2.4}$$

$\boldsymbol{\varepsilon}_d$ is related in isotropic elasticity to the deviatoric stress $\boldsymbol{\sigma}_d$ by the stress modulus G as

$$\boldsymbol{\sigma}_d = \boldsymbol{\sigma} - \mathbf{m}p = G\mathbf{D}_0\boldsymbol{\varepsilon}_d = G(\mathbf{D}_0 - \frac{2}{3}\mathbf{m}\mathbf{m}^T)\boldsymbol{\varepsilon} \tag{5.2.5}$$

with

$$\mathbf{D}_0 = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.2.6}$$

Mark that G and K are related to E and ν in Section 5.1 by

$$G = \frac{E}{2(1+\nu)} \qquad K = \frac{E}{3(1-2\nu)} \tag{5.2.7}$$

The equations 5.2.1 to 5.2.7 can be rewritten into

$$\frac{\partial \boldsymbol{\rho}\mathbf{u}}{\partial t} - \text{ div } \boldsymbol{\sigma}_d + \nabla p \;=\; \mathbf{F} \tag{5.2.8}$$

and

$$\text{div } \mathbf{u} = \frac{p}{K} \tag{5.2.9}$$

Boundary and initial conditions

With respect to boundary and initial conditions, the same types as in Section 7.1 for the incompressible Navier-Stokes equations are available. This means that it is not possible to prescribe the pressure at the boundary, except implicitly through the normal stress.

Solution method

The equations for the (nearly) incompressible materials have the same structure as the incompressible Navier-Stokes equations, without the convective terms. This means that remarks about the solution method that are made in Section 7.1 for the Navier-Stokes equations are also valid here, except those concerning the convective terms.

In particular this means that the continuity equation 5.2.9 provides extra difficulties, although for K finite the material is not exactly incompressible.
As a consequence, it is not possible to use any arbitrary approximation of pressure and displacement, but only a limited number will produce correct results. Furthermore we can use the same type of solution methods as described in Section 7.1, i.e. either the penalty function method or the integrated method.
In case of a penalty function method it is necessary to define a penalty parameter ϵ.

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input of program SEPCOMP or subroutine SEPSTR or alternatively by the parameter JMETOD in subroutine COMMAT.

  In general, two matrices may be created: the mass matrix and the stiffness matrix. The mass matrix is only used for time-dependent problems. This matrix pre-multiplies the discretized time-derivative. The stiffness matrix represents the discretization of the stationary terms in the left-hand side of equations (5.2.8) and (5.2.9).

  Both the stiffness matrix and the mass matrix are positive definite, which means that METHOD = 1 or 5 may be chosen.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 20 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 15 are real coefficients.

  These parameters and coefficients are defined as follows:

1 not yet used

2 not yet used

3 type of numerical integration

4 type of co-ordinate system

5 not yet used

6-20 information about the problem and the penalty parameter in case a penalty function method is applied

6 density ρ

7 Poisson ration ν

8 Elasticity modulus E

9 F_x (body force in x-direction, or in the axi-symmetrical case r-direction)

10 F_y (body force in y-direction, or in the axi-symmetrical case z-direction)

11 F_z (body force in z-direction, 3D only)

12 penalty function parameter ϵ

13-20 not yet used

The parameters that are not used must be set equal to zero, since this is the default value in case of future extensions.
With respect to these parameters the following choices are available:

4 Type of co-ordinate system Possible values:

0 Cartesian co-ordinates

1 Axi-symmetric co-ordinates

- Definition of the coefficients for the boundary conditions: See Section 7.1 for the boundary conditions

- Parameters for subroutine BUILD:

  With respect to subroutine BUILD the parameter IMAS (IINBLD(4)) is of importance. In the stationary case no mass matrix is necessary so IMAS may be chosen equal to zero.
  In the time-dependent case IMAS may be either 1 (diagonal mass matrix) or 2 ("consistent" mass matrix).

  A diagonal mass matrix is only recommended in case of linear elements.

- Parameters with respect to the linear solver:

  The stiffness matrix is not only symmetric but also positive definite.

- Computation of derivatives:

  Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

  1 $\frac{\partial u_j}{\partial x_i}$, where x_i is defined by the parameter ix, and j by the parameter JDEGFD.

  2 $\nabla \mathbf{u}$

  3 -$\nabla \mathbf{u}$

  4 *div* $\mathbf{u}$

  5 *curl* $\mathbf{u}$

  6 The stress σ

7 The strain ϵ

21-27 See 1 to 7, however, now defined in the vertices instead of the nodal points.

The output vector is defined in exactly the same way as in Section 7.1.

- Types of integrals that may be computed:

  See Section 7.1.

Definition of type numbers

The type numbers, which are given in the standard input file in the part between PROBLEM and END (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved.

For the incompressible linear elasticity equation in this section the following type numbers are available:

260 General type number for the internal incompressible elasticity elements.
This number is restricted to the penalty function formulation only.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 4 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. Internally the displacement is treated as a quadratic polynomial with extra third order term. The displacement in the centroid is internally used, but not available to the user. The internal pressure is defined in the centroid, but also not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains two displacement components in each node.

shape = 5 Linear quadrilateral. This is a so-called Crouzeix-Raviart element that does not satisfy the Brezzi-Babuska condition. The displacement is defined in the vertices.
Since the Brezzi-Babuska condition is not satisfied, the pressure may contain unrealistic wiggles. However, if at some part of the boundary the normal stress is given (or equivalently the normal component of the displacement is not prescribed), there is good chance that the wiggles are not visible. The internal pressure is defined in the centroid, but not available to the user.
The solution vector contains two displacement components in each node.

shape = 6 Bi-quadratic quadrilateral. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid, but not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains two displacement components in each node.

shape = 14 Tri-quadratic hexahedron. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid, but not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains three displacement components in each node.

262 General type number for the internal elements.
This number is restricted to the integrated solution method only.
For this type of element, swirl is not allowed.
Elements of this type may be used in combination with direct methods and iterative methods.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 6 Bi-quadratic quadrilateral. This is a so-called Crouzeix-Raviart element. The internal pressure and its gradient are defined in the centroid and available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two displacement components in each node.
Furthermore in the centroid (Point 9), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: p 4: $\frac{\partial p}{\partial x}$ 5: $\frac{\partial p}{\partial y}$

shape = 7 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. The displacement is treated as a quadratic polynomial with extra third order term. The displacement in the centroid is available to the user. The internal pressure is defined in the centroid and available to the user. This pressure is discontinuous over the element

boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two displacement components in each node.
Furthermore in the centroid (Point 7), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: p 4: $\frac{\partial p}{\partial x}$ 5: $\frac{\partial p}{\partial y}$

shape = 9 Linear quadrilateral. This is a so-called Crouzeix-Raviart element that does not satisfy the Brezzi-Babuska condition. The displacement is defined in the vertices.
Since the Brezzi-Babuska condition is not satisfied, the pressure may contain unrealistic wiggles. However, if at some part of the boundary the normal stress is given (or equivalently the normal component of the displacement is not prescribed), there is good chance that the wiggles are not visible. The internal pressure is defined in the centroid, but not available to the user.
The solution vector contains two displacement components in each vertex. In the centroid only the pressure is defined.

shape = 14 Tri-quadratic hexahedron. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid and available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains three displacement components in each node.
Furthermore in the centroid (Point 14), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: v_3 4: p 4: $\frac{\partial p}{\partial x}$ 6: $\frac{\partial p}{\partial y}$ 7: $\frac{\partial p}{\partial z}$

261 General type number for the internal elements.
This number is restricted to the integrated solution method only.
For this type of element, swirl is not allowed.
The difference with type number 262 is that the gradient of the pressure and the displacement in the centroid are eliminated internally. As a consequence the number of unknowns is reduced considerably.
Elements with this type number may be used in combination with direct and iterative linear solvers. Experiments have shown that sometimes the iterative solvers do not converge. In that case it is advised to use type number 262 instead.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 7 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. The displacement is treated as a quadratic polynomial with extra third order term. The displacement in the centroid is available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two displacement components in each node except the centroid.
In the centroid (Point 7), the pressure is available, but not the pressure gradient.
In this point there is only one unknown (p), but this pressure is the physical degree of freedom with sequence number 3.

With respect to type numbers for the boundary conditions the reader is referred to Section 7.1.

5.3 Nonlinear solid computation

Non-linear solid mechanics problems can be solved either by a Total Lagrange approach or an updated Lagrange approach. In SEPRAN elements for both types of equations are available.
Section (5.3.1) treats elements using the Total Lagrange approach.
Elements using the updated Lagrange approach are treated in Section (5.3.2).

5.3.1 Nonlinear solid computation using a Total Lagrange approach

For nonlinear elasticity geometrical nonlinearity (i.e. large deformation) and material nonlinearity (i.e. nonlinear stress-strain relation) is supported for isotropic and anisotropic stress-strain relations. At this moment 3D Cartesian coordinates and 2D axi-symmetric coordinates are only supported. In both cases a Total-Lagrange approach is used.

For (nearly) incompressibility two methods are supported. The first is reduced integration of compression relations This method is also referred to a Natural Penalty Method in literature Ogden (1984), Peng and Chang (1997), Gielen (1998). The second is a special Total-Lagrange formulation of the Enhanced Stiffness Method Souza et al (1996), Gielen (1998) that suppresses mesh locking in incompressibility.

General theory

Equation

In the case of non-linear solid computations, the situation is more complex than for linear elasticity. It is not longer possible to define the basic equations in terms of a simple partial differential equation at a fixed domain.

In the case of large displacements it is common practice to start with a reference situation at time t_0 and to step gradually in time. During this time-stepping both the load as well as the material properties may change. Furthermore the region is deformed in each step. In the reference situation

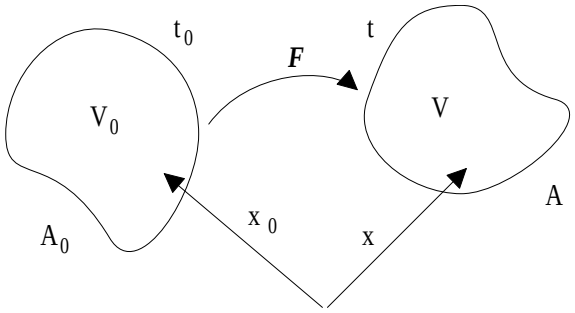


Figure 5.3.1.1: Deforming configuration.

at time t_0, the body volume is $V_0 q$ and the boundary surface A_0. Every point of the body is identified with a space vector $\vec{x}_0$. In the deformed situation at time t these quantities are V, A and $\vec{x}$, respectively. The way the material points deform, as shown in figure 5.3.1.1, is described using the deformation tensor $\boldsymbol{F}$, which is defined as

$$\boldsymbol{F} = (\text{grad}\,_0 \vec{x})^c \tag{5.3.1.1}$$

where ^{c} is a tensor conjugation, and $\text{grad}\,_0$ the gradient with respect to the reference configuration. When neglecting inertial and body forces, the local conservation of momentum can be written as

$$\text{grad} \cdot \boldsymbol{\sigma}^c = \vec{0} \tag{5.3.1.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress in the material. The conservation of moment of momentum can be expressed as:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^c \tag{5.3.1.3}$$

This property is often used to simplify equations. In general, it is impossible to determine the exact solution of equation (5.3.1.2). A differential equation in the form of equation (5.3.1.2) is not a good starting point for the determination of an approximation solution. We use the weighted residuals formulation

$$\int_V \vec{w} \cdot (\text{grad} \cdot \boldsymbol{\sigma}^c) dV = 0 \tag{5.3.1.4}$$

with $\vec{w}$ a weight function. It can be proved that equation (5.3.1.4) equals equation (5.3.1.2) for arbitrary weight function $\vec{w}$. This formulation is weakened by partial integration

$$\int_V (\text{grad}\,\vec{w})^c : \boldsymbol{\sigma} dV = \int_A \vec{w} \cdot \vec{t} dA \tag{5.3.1.5}$$

with $\vec{t} = \boldsymbol{\sigma} \cdot \vec{n}$ and $\vec{n}$ the outer normal of area A. The right-hand side of equation (5.3.1.5) represents the surface traction.

To complete equation (5.3.1.5), we must specify how $\boldsymbol{\sigma}$ relates to $\boldsymbol{F}$ and t, by means of a constitutive equation, and we must choose an appropriate iteration scheme.

Iteration scheme

The iteration scheme has two requirements. Firstly the integration over the unknown volume V and area A must be transformed to a known volume and area. Secondly, due to the large deformations, the equations have to be linearized.
We transform (5.3.1.5) to the reference configuration. This approach is called Total Lagrange. This yields

$$\int_{V_0} (\text{grad}_0\,\vec{w})^c : (\boldsymbol{S} \cdot \boldsymbol{F}^c) dV_0 = -\int_{A_0} (\vec{w} \cdot \boldsymbol{\sigma}_{ext} \cdot \boldsymbol{F}^{-c} \cdot \vec{n}_0 J) dA_0 \tag{5.3.1.6}$$

with $J = \det(\boldsymbol{F})$ and $\boldsymbol{\sigma}_{ext}$ the externally applied stress. The second Piola Kirchoff stress tensor $\boldsymbol{S}$ is defined as:

$$\boldsymbol{S} = \det(\boldsymbol{F})\ \boldsymbol{F}^{-1} \cdot \boldsymbol{\sigma} \cdot \boldsymbol{F}^{-c} \tag{5.3.1.7}$$

The second Piola Kirchoff stress tensor $\boldsymbol{S}$ is a function of the Green-Lagrange strain tensor, that is defined as:

$$\boldsymbol{E} = (\boldsymbol{F}^c \cdot \boldsymbol{F} - \boldsymbol{I})/2 \tag{5.3.1.8}$$

The linearization is accomplished by a Newton-Raphson scheme, which substitutes $\vec{x} = \vec{x}_0 + \vec{u}$ with $\vec{x} = \vec{x}_0 + \vec{u}^* + \delta\vec{u}$. Or, the real configuration is the undeformed configuration ($\vec{x}_0$) plus a estimated displacement ($\vec{u}^*$), plus a deviation displacement ($\delta\vec{u}$). Likewise, $\boldsymbol{S}^* + \delta\boldsymbol{S}$ is substituted for $\boldsymbol{S}$. For reason of convenience we write $\vec{q}$ instead of $\vec{\delta u}$. When second order terms are neglected, equation (5.3.1.6) transforms into

$$\int_{V_0} (\text{grad}_0\,\vec{w})^c : (\boldsymbol{S}^* \cdot (\text{grad}_0\,\vec{q}) + \delta\boldsymbol{S} \cdot \boldsymbol{F}^{*c}) dV_0 = k - \int_{V_0} (\text{grad}_0\,\vec{w})^c : (\boldsymbol{S}^* \cdot \boldsymbol{F}^{*c}) dV_0 \tag{5.3.1.9}$$

with k the right-hand side of equation (5.3.1.6). In the computation the stress estimate $\boldsymbol{S}^*$ and deformation estimate $\boldsymbol{F}^*$ are determined using the configuration estimate $\vec{x}^* = \vec{x}_0 + \vec{u}^*$. The stress increment $\delta\boldsymbol{S}$ is determined from the constitutive behaviour by

$$\delta\boldsymbol{S} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} : \delta\boldsymbol{E} \quad \text{with} \quad \delta\boldsymbol{E} = \boldsymbol{F}^* \cdot \text{grad}_0\,\vec{q} + (\boldsymbol{F}^* \cdot \text{grad}_0\,\vec{q})^c \tag{5.3.1.10}$$

where $\partial\boldsymbol{S}/\partial\boldsymbol{E} =^4 \boldsymbol{M}$ is the tangential material stiffness at strain-state $\boldsymbol{E}^*$ Using the symmetry properties of this fourth order tensor ${}^4\boldsymbol{M}$, the left-hand side of (5.3.1.9) can be rewritten as

$$\int_{V_0} (\text{grad}_0\,\vec{w})^c : \boldsymbol{S}^* \cdot \text{grad}_0\,\vec{q} dV_0 + \int_{V_0} (\text{grad}_0\,\vec{w})^c \cdot \boldsymbol{F}^{*c} :^4 \boldsymbol{M} : \boldsymbol{F}^c \cdot \text{grad}_0\,\vec{q} dV_0 \tag{5.3.1.11}$$

The first term of equation (5.3.1.11) is the stiffness due to the stress-state, often referred to as the *initial or nonlinear stiffness* matrix, and the second term is the stiffness due to the material stiffness, often referred to as the *tangential, linear, or incremental stiffness* matrix.

Discretization

Discretization in this case, is equal to rewriting the equations in index notation. The integration

has to be taken into account with a numerical integration rule (not included in the description, but trivial).

$$({}_L K_{ij}^{IJ} + {}_{NL} K_{ij}^{IJ}) q_j^J = R_i^I \tag{5.3.1.12}$$

$${}_L K_{ij}^{IJ} = F_{ik} \frac{\partial \phi^I}{\partial x_l} M_{klmn} \frac{\partial \phi^J}{\partial x_m} F_{jn} \tag{5.3.1.13}$$

$${}_{NL} K_{ij}^{IJ} = \frac{\partial \phi^I}{\partial x_i} \delta_{ij} S_{jk} \frac{\partial \phi^J}{\partial x_k} \tag{5.3.1.14}$$

$$R_i^I = F_{ij} S_{kj} \frac{\partial \phi^I}{\partial x_j} \tag{5.3.1.15}$$

In fact the problem is reduced to a series of linear elasticity problems, where in each step a new right-hand side and a new matrix is computed.

Theory concerning incompressibility

In the case of (nearly) incompressibility, convergence problems may occur. At this moment two methods are supported: A natural penalty method and an enhanced stiffness method. In both cases the constitutive equation is extended with a compression function, so that:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}_m + \boldsymbol{\sigma}_c \tag{5.3.1.16}$$

The part $\boldsymbol{\sigma}_m$ is specified with the constitutive equation of the general theory. The part $\boldsymbol{\sigma}_c$ is specified with an extra constitutive equation that approximates the conservation of volume. With a proper choice of the extra constitutive equation

$$\sigma_c = -p\boldsymbol{I} \tag{5.3.1.17}$$

With the natural penalty method the convergence is improved if the extra equation is integrated reduced. This is expressed in the integer coefficient 2, with `inonlin=2` or `4`. The compression function must be supplied in the extra subroutine **fncomprs**.

In case of the enhanced stiffness method, the definition of the strain tensor is redefined. First the deformation tensor is splitted into a distortional ($\boldsymbol{F}_d$) and a volumetric ($\boldsymbol{F}_v$) part according to:

$$\boldsymbol{F} = \boldsymbol{F}_d \cdot \boldsymbol{F}_v = \boldsymbol{F}_v \cdot \boldsymbol{F}_d \quad \begin{cases} \boldsymbol{F}_d = J^{-\frac{1}{3}} \boldsymbol{F} \\ \boldsymbol{F}_v = J^{\frac{1}{3}} \boldsymbol{I} \end{cases} \tag{5.3.1.18}$$

For a near-incompressible material, the decomposition of 5.3.1.18 forms the basis for formulating strain interpolation schemes that do not suffer from the over-constraining associated with isoparametric interpolation. The essential idea is to discard the volumetric part of $\boldsymbol{F}_v$ computed from the interpolated displacements and to replace it by a suitable modified volumetric part $\tilde{\boldsymbol{F}}_v$. This modified volumetric deformation field is chosen such that it incorporates less deformation modes than the compatible field. In this way fewer constraints result when the incompressibility limit is approached and locking is overcome. The modified deformation field is constructed by using lower order interpolation or, as in this case, lower order integration as:

$$\tilde{\boldsymbol{F}} = \tilde{\boldsymbol{F}}_d \cdot \tilde{\boldsymbol{F}}_v = \tilde{\boldsymbol{F}}_v \cdot \tilde{\boldsymbol{F}}_d \quad \begin{cases} \tilde{\boldsymbol{F}}_d = \tilde{J}^{-\frac{1}{3}} \tilde{\boldsymbol{F}} \\ \tilde{\boldsymbol{F}}_v = \tilde{J}^{\frac{1}{3}} \boldsymbol{I} \end{cases} \tag{5.3.1.19}$$

with $\tilde{J} = \det(\tilde{\boldsymbol{F}})$.

A modified deformation tensor is constructed with

$$\bar{\boldsymbol{F}} = \boldsymbol{F}_d \cdot \tilde{\boldsymbol{F}}_v = \left(\frac{\tilde{J}}{J}\right)^{\frac{1}{3}} \boldsymbol{F} \tag{5.3.1.20}$$

where $\tilde{\boldsymbol{F}}$ represents the deformation tensor in the centre of the element.

If the system of equations is linearized, a non-symmetric set of matrix equations appears. This method is selected with setting `inonlin=1` or `3` for icoef2, and `1` or `2` for icoef5. In this case the compression function must be incorporated in **fnmateri**. See the remarks of both **fnmateri** and **fncomprs**.

Input for the various subroutines:

- Definition of the storage scheme:

  (See also 5.1) In general the stiffness matrix will be positive definite. The storage scheme is chosen as METHOD = 1 (direct solver) or 5 (iterative solver).

- Definition of the coefficients for the differential equation:

  1 not yet used
  2 type of stress-strain relation
  3 type of numerical integration
  4 type of material model
  5 extra options
  6-15 material coefficients
  16-45 not yet used

  With respect to these parameters the following choices are available:

  2 stress-strain relation:
  The non-linear behaviour must be expressed by the integer coefficient 2 (IGPROB), for which extra possibilities are added: `icoef 2 = igprob + 10*inonlin + 100*ispecial`

  Remark: These definitions have been changed from 1 august 1997 (subroutine ELM250, version 4.0). If you are not sure which version you have, check this subroutine using for example sepvi.

  Possible values for `igprob`:

  0 3D stress-strain relation ship.
  1 reserved for the future.
  2 2D axi-symmetric

  Possible values for `inonlin`:

  1 physical and geometrical nonlinear behaviour. Only globally defined anisotropy is possible.
  The user has to supply a user written function subroutine `fnmateri` to define the material properties.
  2 physical and geometrical nonlinear behaviour, with reduced integrated incompressibility term. The extra equation must be supplied with user written subroutine: `fncomprs`

3 physical and geometrical nonlinear behaviour. Anisotropy with local directions is possible.
The user has to supply two user written subroutines:
`fnmateri` to define the material properties and
`fnlocdir` to define the local anisotropy directions. If the material is isotropic, it makes no sense to use `inonlin=3`; one is advised to use `inonlin=1` instead.

4 physical and geometrical nonlinear behaviour. Anisotropy with local directions is possible. An extra constitutive equation is added, which is reduced integrated. This extra equation must be supplied with user written subroutine: `fncomprs`.

Possible values for `ispecial`:

0 : no specialties.

1 : Muscle contraction. This option is only supported at Eindhoven University of Technology, department of Mechanical Engineering, section Engineering Fundamentals (Contact: Frans van de Vosse or Cees Oomens). Based on method described in Gielen (1998).

3 integration rule:

0 default (2-point Gauss)

1 Newton-Cotes: this integration rule will result in a poor shear response.

2 Gauss 1-point

3 Gauss 2-point

4 material model choice number: In `fnmateri` and `fncomprs` it is termed: `ichoice`. In future these will be used for standard material models. If you do not write your own `fnmateri` and `fncomprs`, the standard routines of Sepran are used. The coefficient has the following definition:
`icoef4 = imateri + 10*icomprs + 100*iadd`

`imateri` Material model number of the default Sepran FNMATERI. See description of FNMATERI for the definitions of the model numbers.

`icomprs` Compression model number of the default Sepran FNCOMPRS. See the description of FNCOMPRS for the definition of the model numbers.

`iadd` If iadd=1 then the stresses and tangential matrix of the FNCOMPRS function is added to the stresses and tangential matrix of FNMATERI. This option is used in combination with the Enhanced Stiffness Method and should not be used in combination with reduced integration.

5 extra options:

0 no extra options (default)

1 de Enhanced Stiffness incompressibility method of the first kind. The matrix will be non-symmetric! METHOD must be 2 (direct solver) or 6 (iterative solver).

2 de Enhanced Stiffness incompressibility method of the second kind. The matrix will be non-symmetric! METHOD must be 2 (direct solver) or 6 (iterative solver).

3 Hydrostatic suppression. **This one is experimental: do not use it unless you know exactly what you are doing!**

6-15 material parameters: These are transferred to `fnmateri` and `fncomprsto` be used in computation of stress estimate and material stiffness. In `fnmateri` they are termed: `matpar(1..10)`.

- Definition of the coefficients for the boundary conditions:
The input is the same as in Section 5.1 for type number 251 for linear elastic analysis.
Note: Natural boundary conditions are described with respect to the *undeformed* configuration, except for the the combination of `icoef1 = 4` and `icoef2 = 10`. In that case the

natural boundary conditions is applied only for normal loads (`coef6`) with respect to the deformed geometry. When this option is used, the old solution (i.e., estimated solution) must be supplied as the first old solution vector, and only boundary element shape 5 (linear quad) is allowed.

- Parameters for subroutine BUILD:
  An old solution must be specified with the total displacement as the first old vector. In case of the specialty option (ispecial=1) a second old solution vector must be supplied.

- **Notes on STRUCTURE**: (if you use structure)
  The solution of a nonlinear solid mechanics problem, happens in a iterative way. In every iteration a correction vector $\vec{q}$ is computed, that must be added to the previous displacement estimate $\vec{u}^*$. This means that there are *two* vectors needed to compute the results: one holds correction the vector $\vec{q}$ (i.e. the solution of the matrix equations), the other the total displacement $\vec{u}$. Be sure that at least the next 3 commands appear in the structure block:

  1. `create_vector, vector = a`: (`a` an integer number)
     This command initializes vector `a`, that will be used as the total displacement vector $\vec{u}$. (Do not forget to give the corresponding `create` block).
  2. `prescribe_boundary_conditions, vector = b` (`b` an integer number)
     The prescribed boundary conditions are placed in the `b` vector, that will also be used to store the incremental displacements $\vec{q}$. (Do not forget to give the corresponding `essential boundary conditions` block).
  3. `solve_nonlinear_system, vector = b` (`b` an integer number)
     This block tells Sepran to solve the nonlinear problem, and store the iteration solutions (i.e., the incremental displacements $\vec{q}$) in vector `b`. (Do not forget to give the corresponding `nonlinear equations` block).

- **Notes on NONLINEAR EQUATIONS**:
  Be sure that you set the iteration method with: `iteration_method = incremental_newton` and to store the total solution in a different vector, with `seqtotal_vector = a` (`a` an integer number). Do not forget to put a `fill_coefficients` command in this block.

- Computation of derivatives:
  Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

  1–5 Standard derivatives

  6 The Cauchy or true stress $\boldsymbol{\sigma}$. If local directions are used, then the stresses are with respect to the local directions.

  7 The Green-Lagrange strain $\boldsymbol{E}$. If local directions are used, then the strains are with respect to the local directions.

  8 Function of det($\boldsymbol{F}$). The user has to supply the subroutine **fndetf**.

User written subroutines

Depending on the parameter `icoef 2`, the user must supply extra user written subroutines. These subroutines must be written according to the following rules:

Subroutine FNMATERI

Description

This routine computes the material behaviour. The user gets the strain estimate $\boldsymbol{E}^*$ and has to return the stress estimate $\boldsymbol{S}^*$ and the material stiffness ${}^4\boldsymbol{M}$. The material stiffness tensor ${}^4\boldsymbol{M}$ is stored as a 6x6 matrix (see also the linear stiffness matrix in Section 5.1). The material parameters for this routine are given by the user as coefficients 6 to 15.

Heading

```
subroutine fnmateri ( ichoice, s, se, eps, detf, matpar, makese )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION S(6), SE(6,6), EPS(6), DETF, MATPAR(10)

LOGICAL MAKESE

ICHOICE this is `icoef 4`.
Mark that this parameter has been included in the version of August 1997. Hence old versions must be updated.

S In this array the user has to store the second Piola-Kirchoff stress tensor. The sequence of the elements is (1,1), (2,2), (3,3), (1,2), (2,3), (3,1)

SE In this array the user has to store the tangential matrix. This matrix is symmetrical, but has to be filled completely by the user. The sequence of the rows and columns is like S.

EPS Contains the Green-Lagrange strains at input. Storage as in S.

DETF Contains the determinant of the deformation gradient at input.

MATPAR Contains the user defined material parameters at input. These are exactly the coefficients 6 to 15 provided by the user at input.

MAKESE If true the tangential matrix SE has to be filled, otherwise, (false) there is no need to fill SE.

Input

The arrays EPS and MATPAR have been filled by the element subroutine.
DETF and MAKESE have got a value.

Output

The user must fill array S and depending on MAKESE also array SE.

Remark

If you do not supply a subroutine FNMATERI, the default subroutine is used. At present the following values of `imateri` have been programmed in the default subroutine:

0 Hookean material. See also linear elasticity.
The following coefficients are required:

6 E

7 ν

1 Orthotropic hookean material. See also linear elasticity.
The following coefficients are required:

6 E_1

7 E_2

8 ν_1

9 ν_2

10 G_1

2 Exponential material, based on strain energy function:

$$W = a_0 \left(\exp(a_1 I_E + a_2 II_E^2 + a_3 E_{11}^2 + a_4(E_{12}^2 + E_{13}^2)) - 1\right)$$

with the first and second invariant of the Green-Lagrange strain tensor, defined as:

$$I_E = \text{trace}(E) \quad II_E = \frac{1}{2}\left(I_E^2 - \text{trace}(E^2)\right)$$

The following coefficients are required:

6 a_0

7 a_1

8 a_2

9 a_3

10 a_4

This model requires a compression model (see FNCOMPRS).

3 Mooney Rivlin material.
The following coefficients are required:

6 c_1

7 c_2

If $c_2 = 0$ this model reduces to Neo-Hookean material. The model requires a compression model (see FNCOMPRS).

Note: The default FNMATERI routine does not check if the coefficients make any sense. The user has to take care that they do.

Subroutine FNCOMPRS

Description

This routine computes the extra constitutive equation, which must be an isotropic functions. The user gets the strain estimate $\boldsymbol{E}^*$ and has to return the stress estimate $\boldsymbol{S}^*$ and the material stiffness ${}^4\boldsymbol{M}$. The material stiffness tensor ${}^4\boldsymbol{M}$ is stored as a 6x6 matrix (see also the linear stiffness matrix in Section 5.1). The material parameters for this routine are given by the user as coefficients 6 to 15. **Note:** If your coefficients depend on coordinates or old solutions, the values are the values computed for the last integration point of the element, not the value in the centre!

Heading

```
subroutine fncomprs ( ichoice, s, se, eps, detf, matpar, makese )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION S(6), SE(6,6), EPS(6), DETF, MATPAR(10)

LOGICAL MAKESE

ICHOICE this is `icoef 4`

S In this array the user has to store the second Piola-Kirchoff stress tensor. The sequence of the elements is (1,1), (2,2), (3,3), (1,2), (2,3), (3,1)

SE In this array the user has to store the tangential matrix. This matrix is symmetrical, but has to be filled completely by the user. The sequence of the rows and columns is like S.

EPS Contains the Green-Lagrange strains at input. Storage as in S.

DETF Contains the determinant of the deformation gradient at input.

MATPAR Contains the user defined material parameters at input. These are exactly the coefficients 6 to 15 provided by the user at input.

MAKESE If true the tangential matrix SE has to be filled, otherwise, (false) there is no need to fill SE.

Input

The arrays EPS and MATPAR have been filled by the element subroutine.
DETF and MAKESE have got a value.

Output

The user must fill array S and depending on MAKESE also array SE. **Note:** S and SE must be isotropic functions of EPS. A sample FNCOMPRS is included in the SEPRAN source. A copy is obtained with the command: `sepget fncomprs`.

Remark

If you don't write your own FNCOMPRS, then the default Sepran subroutine is used. This default routine uses the following definition for the compression function:

$$-p = \frac{1}{\epsilon} h(J) = K h(J)$$

with p the pressure, ϵ the penalty parameter or K the bulk-modulus, h the compression function that depends on J, the relative change of volume. The following model are programmed for the values of `icomprs` given below:

1 $h(J) = J - 1$

2 $h(J) = \log(J)$

3 $h(J) = \frac{1}{2b}(J^{2b} - 1)$, default $b = 1/6$.

4 $h(J) = \frac{1}{b-1}(\frac{1}{J} - \frac{1}{J^b})$, default $b = 10$.

5 $h(J) = \frac{1}{2}J(J^2 - 1)$

The bulk modulus K is `coef11` and b is `coef12`. In case of icomprs=3 or 4, b is made the default value if `coef12`=0E0 or if `coef12` is omitted.

Subroutine FNLOCDIR

Description

This routine determines the rotation matrix in case of anisotropic material with variable anisotropy directions. Constant anisotropy can be taken care of in the material behaviour. The rotation matrix, contains the per column the projection of the local vector $i = 1 \ldots 3$ on the global coordinates x, y, z.

$$R = \begin{bmatrix} n_x^1 & n_x^2 & n_x^3 \\ n_y^1 & n_y^2 & n_y^3 \\ n_z^1 & n_z^2 & n_z^3 \end{bmatrix} \tag{5.3.1.21}$$

Example: If the rotation of Figure 5.3.1.2 has to be programmed, the rotation matrix

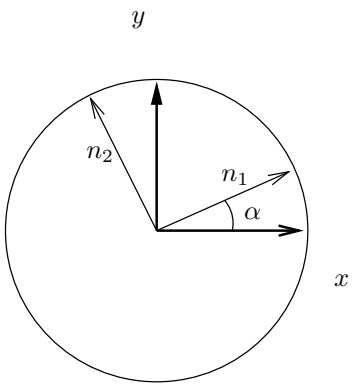


Figure 5.3.1.2: Example for fnlocdir. x and y are the global axes, n_1 and n_2 the local axes. The rotation matrix will hold the projection of the local vectors on the global axes.

is:

$$R = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.3.1.22}$$

Or:

$$R_j^i = (n_i, n_j) \quad i = 1 \ldots 3 \; j = x, y, z \tag{5.3.1.23}$$

with (n_i, n_j) the inner product of n_i with n_j.

Heading

```
SUBROUTINE FNLOCDIR ( POS, DIR, ELGRP )
```

Parameters

INTEGER ELGRP

DOUBLE PRECISION POS(3), DIR(3,3)

POS This is position (x,y,z) at which location the local direction must be given.

DIR In this array the user has to store the rotation matrix.

ELGRP This is the element group number.

Input

POS and ELGRP have been filled by the element subroutines.

Output

The user must fill DIR.

Function FNDETF

Description

For ICHELD=8 the user has to supply the routine FNDETF, where the user can supply a function of DETF. This option is useful in the case of nearly incompressibility to monitor the volume changes, or the compute the pressure given a compressibility relation.

Heading

```
double precision function fndetf ( ichoice, eps, detf, matpar )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION EPS(6), DETF, MATPAR(10)

ICHOICE this is `icoef 4`. **This parameter is added. Update your old fndetf!**

EPS Contains the Green-Lagrange strains at input.

DETF Contains the determinant of the deformation gradient at input.

MATPAR Contains the user defined material parameters at input. These are exactly the coefficients 6 to 15 provided by the user at input.

FNDETF must be given a value by the user.

Input

The arrays EPS and MATPAR and the variable ICHOICE have been filled by the element subroutine.

Output

The user must give FNDETF a value.

Remark

If you do not write a FNDETF routine, the default routine is used. In that case the function returns the pressure according to the pressure definition described in FNCOMPRS.

Definition of type numbers

The type number which are given in the standard input file in the part PROBLEM define the type of differential equation to be solved. For nonlinear elasticity the following type numbers are available:

250 Type number for the internal elements.

shape = 11 linear tetrahedron. (pyramid) Note: As linear tetrahedrons give poor results, they are not tested for all options. Most options will work, but no guarantee is given.

shape = 13 trilinear hexahedron. (brick)

251 Type number for the boundary elements.

shape = 1 In preparation for nonlinear axi-symmetric analysis. See linear elasticity for linear options.

shape = 2,3,4,6 See linear elasticity.

shape = 5 Besides the linear options (See linear elasticity), for this shape also a nonlinear option is supplied for ILOAD=4 and IGPROB=10. In that case the normal loads are applied with respect to the deformed configuration. Don not prescribe any tangential loads with this option. **Note:** If you use this option, be sure that `a = 1` and `b > 1` in the nonlinear equations block. ILOAD = 0 or 2 is in preparation.

5.3.2 Nonlinear solid computation using an Updated Lagrange approach

Introduction

In this section a general nonlinear solid mechanics element in SEPRAN is considered in two and three dimensional space using an updated Lagrange approach. Besides the mathematical formulation also the user input coefficients are defined. In the standard version, SEPRAN offers several types of material behaviour and types of pressure approximation. Also some characteristic test examples are included to demonstrate the use of this element.

This section is written by Tijmen Gunther and is compiled from work by Jurgen de Hart, Raoul van Loon, Chris van Ooijen, Marco Stijnen, Tijmen Gunther, Peter Bovendeerd and Frans van de Vosse of Eindhoven University.

Governing equations

In this section we consider a nonlinear solid mechanics element in two dimensional (plain strain) and three dimensional space. When neglecting body forces, the equations of mass and momentum can be written as:

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} = \vec{0} & \text{in } \Omega(t) \quad (5.3.2.1a) \\ \det(\boldsymbol{F}) - 1 = 0 & \text{in } \Omega(t) \quad (5.3.2.1b) \end{cases}$$

with $\boldsymbol{\sigma}$ the Cauchy stress tensor and $\boldsymbol{F} = (\text{grad}\,_0\vec{x})^T$ the deformation gradient tensor defining the deformation from the reference state Ω_0 with position vectors $\vec{x}_0$ to the current state $\Omega(t)$ with position vectors $\vec{x}(t)$. The Cauchy stress can be written as:

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau} \tag{5.3.2.2}$$

where p is the hydrostatic pressure and $\boldsymbol{\tau}$ the deviatoric stress resulting from deformation. In order to model the deviatoric stress tensor $\boldsymbol{\tau}$ it is necessary to specify the constitutive behaviour.

Constitutive equations

Besides a Neo-Hookean material model also a Moonley-Rivlin model, a composite model and a user defined model is available. These elastic material laws are characterized by:

$$\boldsymbol{\tau} = \boldsymbol{\tau}(\boldsymbol{B}) \tag{5.3.2.3}$$

in which $\boldsymbol{B}$ is the Finger tensor ($\boldsymbol{B} = \boldsymbol{F} \cdot \boldsymbol{F}^T$). For example in the case of an incompressible Neo-Hookean material it holds $\boldsymbol{\tau} = G(\boldsymbol{B} - \boldsymbol{I})$. More general, for isotropic hyperelastic materials the deviatoric stress can be derived from a strain energy function $W = W(I_1, I_2, I_3)$, where I_n is the n-th invariant of $\boldsymbol{B}$. In the incompressible case $I_3 = 1$ the strain energy function reduces to $W = W(I_1, I_2)$. Splitting the Cauchy stress according to equation 5.3.2.2 the extra stress tensor $\boldsymbol{\tau}$ yields:

$$\boldsymbol{\tau} = 2\left[\frac{\partial W}{\partial I_1}\boldsymbol{B} - \frac{\partial W}{\partial I_2}\boldsymbol{B}^{-1}\right] = g_1(I_1, I_2)\boldsymbol{B} - g_2(I_1, I_2)\boldsymbol{B}^{-1} \tag{5.3.2.4}$$

Here the following relation is used:

$$\boldsymbol{\sigma} = \frac{2}{J}\boldsymbol{F} \cdot \frac{dW}{d\boldsymbol{C}} \cdot \boldsymbol{F}^T \tag{5.3.2.5}$$

where $J = \det(F)$ and the Cauchy-Green tensor $\boldsymbol{C}$ is defined as:

$$\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F} \tag{5.3.2.6}$$

Together with appropriate boundary and initial conditions the system of equations can be solved.

Boundary and initial conditions

The following types of boundary conditions are available:

1 Components of the displacement $\vec{v}(\vec{x})$ can be given on (some part of) the boundary. These essential boundary conditions are also called Dirichlet conditions.

2 A load is imposed on (a part of) the boundary. This is a so-called natural boundary condition (also called Neumann condition). Only the normal stress $\boldsymbol{\sigma}_n$ is defined in this element type i.e. in the normal direction of the surface. The tangential components $\boldsymbol{\sigma}_t$ are not defined. For this type of boundary conditions boundary elements are required.

Coordinate system

Although the stress tensor, strain tensor and the displacement vector itself are independent of the coordinate system, their representation may vary in various coordinate systems:

2D Cartesian coordinates:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix}, \qquad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix}, \qquad \vec{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{5.3.2.7}$$

3D Cartesian coordinates:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix}, \qquad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix}, \qquad \vec{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.3.2.8}$$

Solution method

For this type of element the updated Lagrange approach is used. In figure 5.3.2.1 a schematic representation of describing the deformation within an updated Lagrange approach is depicted. Following this approach we can define:

$\boldsymbol{F_n}$ the deformation tensor which describes the deformation of the continuum to the configuration Ω_n at $t = t_n$.

$\boldsymbol{F_\Delta}$ the deformation tensor which describes the deformation from the last known configuration Ω_n to the current configuration Ω_t at $t = t_{n+1}$.

In the Theoretical manual the weak form using an updated Lagrange approach is presented. Whereas in the total Lagrange method a transformation back to the initial configuration Ω_0 is made, the updated Lagrange method uses transformations back to the last known configuration Ω_n. Moreover since the set of equations is nonlinear in terms of the displacement Newton iterations are used to solve the equations. Hence also the linearized weak form of the governing equations (eq. 5.3.2.1b) is presented the Theoretical Manual.

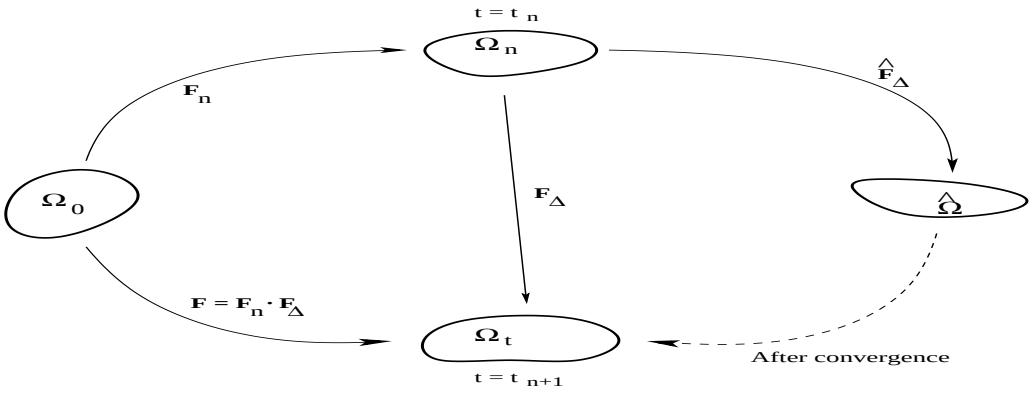


Figure 5.3.2.1: Representation of the updated Lagrange approach.

Definition of the coefficients for the differential equation

The coefficients for the differential equation may be defined by one of the methods described in Section 2.2 of the SEPRAN User Manual, where, in general, the method by SEPCOMP is recommended. For each element group 45 parameters and coefficients must be given.The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the other coefficients are reals.

These parameters and coefficients are defined as follows:

1 Integer ($ifsi\_solution$) refers to sequence number of solution vector for fully coupled fluid structure interaction. More info becomes available soon. Please contact the University of Technology Eindhoven, department of Mechanical Engineering, division MATE.

2 Type of stress-strain relation (IGPROB):

0 Default value for fully $3D$ and plane strain (only $2D$) problems

3 Type of numerical integration (INTRULE):

0 Default, integration rule is defined by element.

1 Newton-Cotes numerical integration.

2 Gauss numerical integration (element dependent).

3 Gauss numerical integration (element dependent).

4 Type of constitutive law (IMATLAW):

1 Compressible elastic solid.

2 Incompressible elastic solid.

3 Compressible Moonley-Rivlin material.

4 Incompressible Moonley-Rivlin material.

10 Composite material.

11 Composite material (with volume-fraction θ).

99 User defined material.

5 User flags (coef = IUSRVC + 100*IUSRFLG)

Coefficients $6-45$ contain information about the elasticity matrix depending on the type of constitutive law (icoef4). The parameters that are not used must be set equal to zero, since this is the default value in case of future extensions. The following values are needed depending on IMATLAW:

IMATLAW $= 1, 2, 4, 10, 11$:

6 Density ρ

7-9 not used

IMATLAW $= 1$:

10 `Shear modulus` G

11 `Bulk modulus` κ

IMATLAW $= 2$:

10 Shear modulus G

IMATLAW $= 4$:

10 material parameter c0

11 material parameter c1

IMATLAW $= 10, 11$:

10-29 see above for matrix material

30 type of constitutive equation for matrix material

31 not used

32 number of fiber layers

33 not used

34 theta (only for ICOEF4 = 11)

35-45 fiber material parameters

IMATLAW $= 99$:

10-29 user material parameters

Definition of the coefficients for the natural boundary conditions

The coefficients for the boundary conditions may be defined by one of the methods described in 2.2 of the SEPRAN User Manual, where, in general, the method by SEPCOMP is recommended. The element type in this case is $type = 210$. For each element group 15 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 10 are real coefficients.

These parameters and coefficients are defined as follows:

1 Type of natural boundary condition (ILOAD)

0 Global system

1 Local system

2 Integer refers to sequence number of solution vector for fully coupled fluid-structure interaction ($IFSI\_SOLUTION$) More info becomes available soon. Please contact the University of Technology Eindhoven, department of Mechanical Engineering, division MATE.

3 Type of numerical integration

0 Default value depending on element shape

1 Newton-Cotes numerical integration

3 Gauss numerical integration

4 not used

5 not used

6 1-component of stress

7 2-component of stress

8 3-component of stress

9 cx

10 cy

11 cz

Coefficients $12 - 15$ are not used yet.

Computations of derivatives:

Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

1 $\frac{\partial u_j}{\partial x_i}$, where x_i is defined by the parameter ix, and j by the parameter $jdegfd$.

2 $\nabla \mathbf{u}$

3 - $\nabla \mathbf{u}$

4 div **u**

5 $curl$ **u**

6 Stress σ (ivec= 11), depending on ix:

$ix = 1$ default (depends on element)

$ix = 2$ tau

$ix = 3$ fiber stress (global)

$ix = 4$ fiber stress (local)

7 Pressure p (ivec= 1)

8 Strain (ivec= 11)

9 Deformation gradient(ivec= 13), depending on ix:

$ix = 1$ nodal points

$ix = 2$ integration points

10 Local direction (ivec=3)

Definition of type numbers:

The type numbers, which are given in the standard input file in the PROBLEM block (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved. The following type numbers are available:

200 General type number for the internal elements. Defines the differential equation. This type is available for the following element shape numbers (See the Users Manual, Section 2.2, table 2.2.1):

shape = 7 extended quadratic triangle

shape = 9 quadratic quadrilateral

shape = 14 quadratic hexahedron

shape = 16 extended quadratic tetrahedron

In the case of element type 200 no divergence part is build.

201 Discontinuous pressure element (Crouzeix-Raviart). Available shapes: See type 200.

202 Continuous pressure element (Taylor-Hood). Available shapes: See type 200 and also shape = 6 (quadratic triangle).

210 General type number for boundary elements. The boundary elements are (curved) line elements in $2D$ and (curved) surface elements in $3D$. This type is available for the following element shape numbers (See the Users Manual, Section 2.2, table 2.2.1):

shape = 2 quadratic line element

shape = $6, 7$ (extended) quadratic triangle

shape = 9 quadratic quadrilateral

Additional topics

As mentioned before SEPRAN offers several types of material behaviour. The material dependent part is specified through the material stress tensor $\boldsymbol{\sigma}$ and the material stiffness matrix D_T. In the Theoretical Manual this topic is elaborated. Both are computed in the subroutines `el83xx.f` in which xx denotes a specific material law, e.g. `el8302` is used for an incompressible Neo-Hookean material behaviour. A more general subroutine is `el8309` (See also Section 5.3.2 in the Theoretical Manual. This subroutine can easily be used for material laws obtained from Strain Energy Density functions W.

5.4 (Thick) plate elements

In this section plate elements are described. Plates require a special approach. The method we use can be found in many text books like Bathe (1982), Zienkiewicz and Taylor (1989), Volume 2, Mohr (1992) and Hughes (1987).

Equation

Starting point of the equations are the 3D linear elasticity equations as given in Section 5.1. Following the Reissner-Mindlin theory we make the following assumptions:

1. The domain $\Omega \in R^3$ can be written as:
   $z \in [-t/2, t/2]; \quad (x, y) \in A \subset R^2$,
   with t the thickness of the plate and A its area with boundary S.
2. $\sigma_{33} = 0$
3. $u_1(x, y, z) = z\Theta_2(x, y), \quad u_2(x, y, z) = -z\Theta_1(x, y)$
4. $u_3(x, y, z) = w(x, y)$

w is the transverse displacement and Θ_α is the rotation vector. It may be interpreted as the rotation of a fiber initially normal to the plate mid surface ($z = 0$).
The direction of the rotation vector is as follows: θ_1 is in the direction of the x-axis and θ_2 in the direction of the y-axis as sketched in Figure 5.4.1.
So we have three unknowns w, Θ_1 and Θ_2. The orientation of the axis is given in Figure 5.4.1.

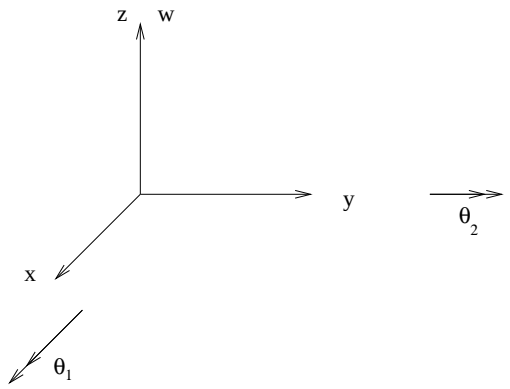


Figure 5.4.1: Orientation of rotation vectors

Constitutive Equation

The constitutive equation can be derived from the linear elasticity using our previous assumptions. In the isotropic case this leads to

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij}, \tag{5.4.1}$$

where λ and μ are the Lamé coefficients and δ_{ij} is the Kronecker delta.
Assumption 2 implies

$$\varepsilon_{33} = \frac{-\lambda}{\lambda + 2\mu}\varepsilon_{\alpha\alpha}, \tag{5.4.2}$$

$$\sigma_{\alpha\beta} = \bar{\lambda}\delta_{\alpha\beta}\varepsilon_{\gamma\gamma} + 2\mu\varepsilon_{\alpha\beta}, \tag{5.4.3}$$

$$\sigma_{\alpha 3} = 2\mu\varepsilon_{\alpha 3}, \tag{5.4.4}$$

where

$$\bar{\lambda} = \frac{2\lambda\mu}{\lambda + 2\mu}. \tag{5.4.5}$$

$\bar{\lambda}$ and μ may be expressed in Young's modulus E and Poisson's ratio ν:

$$\bar{\lambda} = \frac{\nu E}{1-\nu^2}, \tag{5.4.6}$$

$$\bar{\mu} = \frac{E}{2(1+\nu)}. \tag{5.4.7}$$

Strain-displacement relations

From assumptions 3 and 4 we get the following strain-displacement relations:

$$\varepsilon_{1\beta} = u_{(1,\beta)} = z\theta_{(2,\beta)}, \tag{5.4.8}$$

$$\varepsilon_{2\beta} = u_{(2,\beta)} = -z\theta_{(1,\beta)}, \tag{5.4.9}$$

$$\varepsilon_{\alpha 3} = u_{(\alpha,3)} = \frac{-\theta_\alpha + w_{,\alpha}}{2}. \tag{5.4.10}$$

With $u_{(i,j)}$ we mean $\frac{u_{i,j}+u_{j,i}}{2}$ and with $u_{i,j}$: $\frac{\partial u_i}{\partial x_j}$.

Implementation

The stiffness matrix can we written as $\mathbf{k} = \mathbf{k}_b + \mathbf{k}_s$, with

$$\mathbf{k}_b = \int_A \mathbf{B}^{b^T} \mathbf{D}^b \mathbf{B}^b dA \qquad \text{(bending stiffness)}, \tag{5.4.11}$$

$$\mathbf{k}_s = \int_A \mathbf{B}^{s^T} \mathbf{D}^s \mathbf{B}^s dA \qquad \text{(shear stiffness)}, \tag{5.4.12}$$

where the submatrices $\mathbf{B}^b$ and $\mathbf{B}^s$ are given by:

$$\mathbf{B}^b = \begin{bmatrix} 0 & \frac{\partial}{\partial x} & 0 \\ 0 & 0 & \frac{\partial}{\partial y} \\ 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \tag{5.4.13}$$

and

$$\mathbf{B}^b = \begin{bmatrix} \frac{\partial}{\partial x} & -1 & 0 \\ \frac{\partial}{\partial y} & 0 & -1 \end{bmatrix}, \tag{5.4.14}$$

multiplied by the shape (basis) functions..
The matrices $\mathbf{D}^b$ and $\mathbf{D}^s$ follow from integration over z of the classical stiffness matrix and the $\sigma_{\alpha 3}$ term respectively.

Elasticity matrices $\mathbf{D}^b$ *and* $\mathbf{D}^s$

The constitutive equations depend on the type material properties. These properties are translated into the elasticity matrices $\mathbf{D}^b$ and $\mathbf{D}^s$. For this element the following type of elasticity matrices is available.

Isotropic material:

$$\mathbf{D}^b = \frac{Et^3}{12(1-\nu^2)} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{(1-\nu)}{2} \end{bmatrix} \tag{5.4.15}$$

$$\mathbf{D}^s = \frac{Etk}{2(1+\nu)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.4.16}$$

where E denotes Young's modulus and ν Poisson's ratio.
k is a constant to account for the actual non-uniformity of the shearing stress. Following the standard literature we take $k = 5/6$.
At this moment only a distributed loading is allowed as force vector.

Boundary conditions

At this moment only Dirichlet boundary conditions, i.e. transverse displacement or rotations given. If at a boundary a component is not prescribed this implies that a natural boundary condition is available.
The following alternatives per point are available:

- Either the transverse displacement given or the boundary shear force Q = 0 .
- Either the α^{th} component of the rotation vector given or boundary moment $M_\alpha = 0$.

The following types of boundary conditions are common for practical physical situations:

Clamped $w = 0, \Theta_s = 0, \Theta_s = 0$

Free $Q = 0, M_s = 0, M_n = 0$

Simply supported either $w = 0, M_s = 0, M_n = 0$ or $w = 0, \Theta_n = 0, M_n = 0$.
According to Hughes the latter corresponds to thin plate theory but the first one is preferred as curved boundaries are used.

Symmetric $Q = 0, M_s = 0, \Theta_s = 0$

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input of program SEPCOMP or subroutine SEPSTR or alternatively by the parameter JMETOD in subroutine COMMAT.

  The stiffness matrix is positive definite, which means that METHOD = 1 or 5 may be chosen.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP of FILCOF is recommended.
  For each element group 45 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 40 are real coefficients.

  These parameters and coefficients are defined as follows:

  1 not yet used

  2 type of stress-strain relations

  3 type of numerical integration

4 not yet used

5 not yet used

6-26 information about the elasticity matrix depending on the type of stress-strain relations

27 thickness of the plate: h

28 (distributed load in z-direction)

29-30 not yet used

31-36 not yet used

37-42 not yet used

43 not yet used

44 In case the distributed load depends on the displacement, the linear part of this expression ($\frac{\partial F}{\partial w}$), may be subtracted from the right-hand side and added to the matrix. Here F denotes the load and w the displacement.

45 not yet used

The parameters that are not used must be set equal to zero, since this is the default value in case of future extensions.

With respect to these parameters the following choices are available:

2 Type of stress-strain relations (IGPROB)
Possible values:

0 isotropic material (Cartesian)

3 Type of numerical integration
Possible values:

0 default value defined by element

6-26 Information about the elasticity matrix
The following values are needed depending on IGPROB:

IGPROB = 0:
coefficient:

6 E

7 ν

- Definition of the coefficients for the boundary conditions:

  At this moment no coefficients for the boundary conditions are required.

- Parameters with respect to the linear solver:

  The stiffness matrix is not only symmetric but also positive definite.

- Computation of derivatives:

  Depending on the parameter ICHELD in subroutine DERIV the following types of derivatives are computed:

  1 The derivatives $\frac{\partial u_j}{\partial x_i}$, where j refers JDEGFD and i to IX.
  In this case we mean by $\mathbf{u}$ the vector $(w, \theta_1, \theta_2)^T$. IX may take the values 1 and 2.

  6 The stress σ, for $z = h$. This tensor is defined by $(\sigma_{11}, \sigma_{22}, \sigma_{12}, \sigma_{23}, \sigma_{31}, \sigma_{33})$, using the formulas: 5.4.1 to 5.4.4.

7 The strain ϵ, for $z = h$. This tensor is defined by $(\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{12}, \varepsilon_{23}, \varepsilon_{31}, \varepsilon_{33})$, using the formulas: 5.4.8 to 5.4.10.

The output vector is defined as follows:

1 a vector of the type of the solution vector.

6,7 a vector of special structure with six unknowns per point.

In the cases ICHELD = 6 and 7 the user must define exactly the same coefficients as for the differential equation, except for the parameter ρ, which is not used.

Definition of type numbers

The type numbers, which are given in the standard input file in the part between PROBLEM and END (input for SEPCOMP or subroutine SEPSTR) define the type of differential equation to be solved.

For the plate problem in this section the following type numbers are available:

255 general type number for the internal elements. Defines the differential equation.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 5 bilinear quadrilateral. This is the special T1 element described by Hughes and developed to prevent the so-called locking problem.

6 Solidification problems

Solidification problems are examples of free surface problems, where the interface between the solid and liquid part of the region is determined by the solution itself.
In order to solve such kind of problems we may distinguish between *fixed grid* methods and adaptive mesh methods. In the first approach we have a fixed grid that is used for solid and liquid part. The interface intersects the grid somewhere, usually not along edges of elements. In case of an adaptive mesh method the interface is computed explicitly and the mesh is adapted such that the interface is a (internal) boundary of the mesh.

In this chapter we shall follow both approaches.

Fixed grid methods

6.1 Enthalpy approach.
The enthalpy equation is solved instead of the standard temperature equation, to avoid discontinuities at the interface.
At this moment we have to alternatives:
The non-linearity is solved in a non-linear over-relaxation (6.1.1 or
The non-linearity is solved by a quasi Newton approach due to Nedjar et al. (6.1.2.

6.2 The Newton approach of Fachinotti et al.

6.3 The heat capacity method.
An effective heat capacity is computed depending on the enthalpy. This is an alternative for the enthalpy approach.

Adapted grid methods

6.1 A fixed grid method: the enthalpy method

Consider a region in which we have a solid part and a liquid part. The solid is melting gradually. The interface between the solid and liquid is considered to have thickness zero. Physical parameters in the solid part and liquid part may be different.
At the interface we have a latent heat L which describes the extra heat necessary for the melting of the solid.

Equation

The problem is described by the standard heat equation

$$\rho c_p \frac{\partial T}{\partial t} - \text{ div } (\kappa \nabla T) = Q \tag{6.1.1}$$

with

T the temperature,

ρ the density,

c_p the specific heat and

κ the thermal conductivity

So we are dealing with the same type of equation as in Section 3.1.

Boundary and initial conditions

Exactly the same type of boundary and initial conditions as in Section 3.1 may be applied.

Interface conditions

The interface is a special type of boundary. Besides that not only the position of the interface is unknown, we need two boundary conditions instead of one. One of the two is the standard boundary condition, the other one is used to compute the position of the interface.
The following interface conditions are used:

$$\begin{cases} T = T_m & \text{melting temperature.} \quad (6.1.2a) \\ \rho L v_n = \left[\kappa \dfrac{\partial T}{\partial n} \right] & \text{The jump in the flux} = \rho L v_n \quad (6.1.2b) \end{cases}$$

The first interface condition (6.1.2a) makes the temperature at the interface equal to the melting temperature.
The second interface condition (6.1.2b) defines the velocity of the interface as function of the difference of the flux $\kappa \frac{\partial T}{\partial n}$ at both sides of the interface.

Solution method

In order to solve the temperature equation with interface conditions (6.1.2a), (6.1.2b) we use the enthalpy H defined by

$$H(T) = \begin{cases} \int_{T_0}^{T} \rho c_p(T) dT & T \le T_m \\ \int_{T_m}^{T} \rho c_p(T) dT + \rho L & T > T_m \end{cases}, \tag{6.1.3}$$

where T_0 is some reference temperature.

The heat equation (6.1.1) is now replaced by the *enthalpy equation*

$$\frac{\partial H}{\partial t} - \text{ div } (\kappa \nabla T) = Q \tag{6.1.4}$$

Note that this equation contains both the enthalpy H and the temperature T. Since T depends on H in a non-linear way, this is essentially a non-linear time dependent problem. The main advantage is that we do not have to impose the interface conditions, since they are hidden in the formulation. To solve this equation internally a *Kirchoff transformation* is employed, transforming the temperature to a normalized form. The non-linear equations may be solved either by a Newton-method, or in the case that the coefficients are constant for each phase, by a *constrained over-relaxation method*.
At present only the last approach has been implemented.
For more details about the solution method the reader is referred to the Theoretical Manual.

The solution of the enthalpy equation follows the following steps:

$t = 0$
Give the initial temperature.
Compute the initial enthalpy from the temperature
while end time not reached **do**
  $t = t + \Delta t$
  Solve one step of the enthalpy equation by non-linear iteration.
  Compute the temperature from the enthalpy
end while

The non-linear iteration can be carried out in several ways.
In this section we restrict ourselves to a non-linear over-relaxation approach (6.1.1) and a quasi-Newton approach due to Nedjar et al (6.1.2).

6.1.1 Enthalpy approach by non-linear over-relaxation

In the non-linear over-relaxation method, the non-linear system of equations is solved by an over-relaxation method. So the non-linearity is immediately coupled to the solution of the system of equations. The main advantage of this approach is that the system of equations has to be formed only once. Solving and linearization are coupled into one big system.
Disadvantage is that at this moment the method can not be coupled to a standard hear equation. Hence if in one part of the region we have no phase change, whereas in another part we have, the method can not be applied.
Another disadvantage might be that sometimes the over-relaxation method converges slowly. However, in a number of applications this method proved to be fast and simple.
Below we describe the input necessary to run this method.

SEPRAN input At this moment only the input in case of constant coefficients per phase is described. In that case the constrained over-relaxation process is applied per time step.
The user may use program SEPCOMP to solve the problem. In the input file several blocks must be filled in a specific way. Each of these blocks is treated separately.

constants

This block is not necessary, but it makes the input much more easy.

Some reals and vector names may be defined in this block, which are recognized by name by the program. It makes the rest of the input more simple

```
constants
   reals
      rho   = ...
      kappa_s = ...
      kappa_l = ...
      latent_heat = ...
      capacity_s = ...
      capacity_l = ...
      melt_temp = ...
   vector_names
      Temperature
      Enthalpy
end
```

Meaning of these variables

rho	density ρ
kappa_s	thermal conductivity κ in the solid phase
kappa_l	thermal conductivity κ in the liquid phase
latent_heat	latent heat L
capacity_s	specific heat c_p in the solid phase
capacity_l	specific heat c_p in the liquid phase
melt_temp	melting temperature T_m
Temperature	name of temperature vector
Enthalpy	name of enthalpy vector

problem

In this block the user must define the usual input with respect to the temperature equation. Hence, for example, type number 800 must be used.

matrix

This block defines the structure of the matrix.

Since over-relaxation is used it is necessary to use storage method 9, i.e. compact storage defined per row. Hence the block looks like

```
matrix
   method = 9
end
```

coefficients

The coefficients to be filled have the same meaning as in Section 3.1. However, due to the internal transformations applied it is required to make both the diffusion coefficients as well as the parameter ρc_p equal to 1. The actual parameters are used in the transformation. So the input block in R^2 looks like:

```
coefficients
   elgrp1(nparm=20)
      coef6  = 1                # kappa
      coef9  = 1                # kappa
      coef17 = 1                # rho*c
end
```

solve

It is necessary to provide this block, since an over-relaxation method is used. The block must contain at least the following parts

```
solve
   iteration_method = overrelaxation, omega = 1, max_iter = ....
end
```

The choice of iteration method is clear, ω must be set to 1 so that in each time step a new over-relaxation parameter is computed. Otherwise the process may diverge. The maximum number of iterations must be set to a value, for example 1000.

time_integration

This block is necessary for the process. It should look like:

```
time_integration
   method = euler_implicit
   tinit = ...
   tend  = ...
   tstep = ...
end
```

More parameters may be given but the ones shown are necessary. The method has only been tested in combination with *Euler implicit*.

enthalpy_integration

This block is specific for the enthalpy integration. The input is not described in the Users Manual since it is restricted to this particular application.
The following input is recognized:

```
enthalpy_integration [, sequence_number = s]
   seq_enthalpy = i             (Default 1)
   seq_temperature = i          (Default 2)
   seq_time_integration = i     (Default 1)
   seq_boundary_conditions = i  (Default 1)
   kappa_s = k                  (Default 1)
```

```
   kappa_l = k                  (Default kappa_s)
   latent_heat = k              (Default 0)
   capacity_s = k               (Default 1)
   capacity_l = k               (Default capacity_s)
   rho = k                      (Default 1)
end
```

Only the first and last line are mandatory, all others are optional.
Meaning of the various lines:

enthalpy_integration , sequence_number = s
: Necessary statements, opens the input for the enthalpy equation.
May be provided with a sequence number in case more input blocks of this type are used.

seq_enthalpy $= i$, defines the sequence number of the enthalpy vector in the set of solution arrays.
If this line is not given, it is checked if a vector with name `enthalpy` can be found. If so, that is the default, otherwise the default is 1.

seq_temperature $= i$, defines the sequence number of the temperature vector in the set of solution arrays.
If this line is not given, it is checked if a vector with name `temperature` can be found. If so, that is the default, otherwise the default is 2.

seq_time_integration $= i$ defines the sequence number of the block with time integration input.
Default value: 1

seq_boundary_conditions $= i$ defines the sequence number of the block with essential boundary conditions input.
This must contain the essential boundary conditions for the temperature.
Default value: 1

kappa_s $= k$ defines the value of κ in the solid phase.
If this line is not given, it is checked if a constant with name `kappa_s` can be found. If so, that is the default, otherwise an error message is given.

kappa_l $= k$ defines the value of κ in the liquid phase.
If this line is not given, it is checked if a constant with name `kappa_l` can be found. If so, that is the default, otherwise an error message is given.

latent_heat $= k$ defines the value of the latent heat L.
If this line is not given, it is checked if a constant with name `latent_heat` can be found. If so, that is the default, otherwise an error message is given.

capacity_s $= k$ defines the value of c_p in the solid phase.
If this line is not given, it is checked if a constant with name `capacity_s` can be found. If so, that is the default, otherwise an error message is given.

capacity_l $= k$ defines the value of c_p in the liquid phase.
If this line is not given, it is checked if a constant with name `capacity_l` can be found. If so, that is the default, otherwise an error message is given.

rho $= k$ defines the value of the density ρ.
If this line is not given, it is checked if a constant with name `rho` can be found. If so, that is the default, otherwise an error message is given.

structure
This block is also necessary. It defines the structure of the main program. It must contain at least the following lines:

```
structure
   create_vector, vector=%Temperature
```

```
   compute_enthalpy
   start_time_loop
      enthalpy_integration, sequence_number = 1
   end_time_loop
end
```

Explanation:
The first and last line are always required for the structure block.
The second line `create_vector` is necessary to define the initial conditions for the temperature. It refers to an input block `create`.
The third line `compute_enthalpy` computes the enthalpy from the already computed temperature. It uses the information of the input block `enthalpy_integration`.
The next line `start_time_loop` is required to do the time integration. Since in this case we do a special type of time integration, this is the way to deal with it.
The next line `enthalpy_integration` defines how the enthalpy integration must be performed. It uses the information of the input block `enthalpy_integration`. The new temperature and enthalpy are computed.
The next line `end_time_loop` ends the time loop.

Of course the user may extend this block with extra lines for manipulation, printing, plotting and output.

For an example of the use of this input, the reader is referred to Section 6.1.1 of the manual Examples.

6.1.2 Enthalpy approach by quasi-Newton

An alternative of the over-relaxation method is the quasi-Newton approach of Nedjar et al. This method allows for coupled heat equation with the phase change problem and is therefore more flexible. A clear disadvantage is that during each non-linear iteration the matrix must be build and the system of equations be solved. An advantage is that every linear solver can be used, so also the most fastest ones.
In practice this method may take a relatively large number of non-linear iterations, so sometimes it s very expensive.
The SEPRAN input for this case is very similar to that of the over-relaxation method, so we concentrate only at the differences.

SEPRAN input The constants block for both methods can be the same, so no need to repeat.
The following blocks are different.

problem

In this block the user must define the usual input with respect to the enthalpy equation. Since we are dealing with a complete new type of equation type number 800 can not be used anymore, but should be replaced by a new type number: 810.
This type number is only suited for the enthalpy equation. It can be combined with type numbers 800 in parts of the region without phase change, where only the temperature equation must be solved.

matrix

This block defines the structure of the matrix.
Since the standard linear solver is used the storage must be adapted to the solver. A common choice is `method = 5`, which means that a compact symmetric storage is used. The corresponding solver is the preconditioned conjugate gradients method.

coefficients

Since we are dealing with a new type number, also the storage of the coefficients is a little bit different from type 800.
The number of coefficients required is 25.
Coefficients 1 to 5 are again integer.
Coefficients 1 to 4 and 6 to 16 have the same meaning as for type 800. Coefficients 23 to 25 are not yet used and must be zero. (3.1). The following coefficients have a different meaning:

5 iseq_temperature (integer), gives the sequence number of the temperature in the set of solution vectors.
A common choice is `icoef5 = %Temperature`

17 rho (real), defines the density ρ.
A common choice is `coef17 = $rho`

18 capacity_s (real), defines the heat capacity in solid.
A common choice is `coef18 = $capacity_s`

19 capacity_l (real), defines the heat capacity in fluid.
A common choice is `coef19 = $capacity_l`

20 latent_heat (real), defines the latent heat L.
A common choice is `coef20 = $latent_heat`

21 melt_temp (real), defines the melting temperature T_m.
A common choice is `coef21 = $melt_temp`

22 iseq_enthalpy (integer), gives the sequence number of the enthalpy in the set of solution vectors.
A common choice is `icoef22 = %Enthalpy`

A typical example in 1D could be

```
coefficients
   elgrp1(nparm=25)
     icoef3  = 3
     icoef5  = %Temperature
      coef6  = $kappa
      coef17 = $rho
      coef18 = $capacity_s
      coef19 = $capacity_l
      coef20 = $latent_heat
      coef21 = $melt_temp
     icoef22 = %Enthalpy
end
```

solve

The input block solve is standard. An example could be

```
solve
   iteration_method = cg
end
```

time_integration

This block is necessary for the process. Compared to the over-relaxation method it must contain at least the following extra items:

```
non_linear_iteration
max_iter = ...
abs_iteration_accuracy = ...
```

All these parameters are used to define the non-linear iteration process per time step.

enthalpy_integration

This block is specific for the enthalpy integration.
In this case we need less input than for the over-relaxation method, since the matrix is formed by type 810.
The following input is commonly used:

```
seq_time_integration = ...
solution_method = nedjar
seq_coefficients = ...
seq_boundary_conditions = ...
```

The sequence numbers just refer to input blocks, where sequence number 1 is always the default.
solution_method = Nedjar is necessary for the method to decide that this quasi newton iteration is applied.

structure

The structure block can be the same as for the over-relaxation method and will not be repeated here.

For an example of the use of this input, the reader is referred to Section 6.1.2 of the manual Examples.

6.2 The Newton approach of Fachinotti et al.

Equation

The method of Fachinotti et al. uses the temperature formulation. The diffusion of heat is described by:

$$\rho c_{s,l} \frac{\partial T}{\partial t} + \rho L \frac{\partial f_l}{\partial t} - \kappa_{s,l} \Delta T = q, \tag{6.2.1}$$

with T the temperature, ρ the density, c the heat capacity, κ the thermal conductivity and L the latent heat. The subscript s refers to the solid phase and l to the liquid phase. $f_i(T)$ denotes the liquid volume fraction, which in case of isothermal phase-change is equal to the heavyside step function $H(T - T_m)$, with T_m the melting temperature.
The time-derivative of the liquid volume fraction is to be interpreted in the weak sense. Boundary conditions for Equation (6.2.1), are the same as for the heat equation (3.1.2).
Application of the general Galerkin procedure, in which the temperature field is approximated by

$$T(\mathbf{x}, t) \approx \sum_{i=1}^{n} \phi_i(\mathbf{x}) T_i(t), \tag{6.2.2}$$

where ϕ_i is a (linear) basis shape function and T_i is the nodal temperature, in combination with an Euler backward time discretization, yields the following discretized system of (nonlinear) equations (at time level $m + 1$):

$$M^{m+1} \frac{T^{m+1} - T^m}{\Delta t} + \frac{L^{m+1} - L^m}{\Delta t} + S^{m+1} T^{m+1} = q^{m+1}. \tag{6.2.3}$$

Note that both the mass matrix as well as the stiffness matrix are time-dependent (specific heat c is included in the mass matrix; thermal conductivity κ is included in the stiffness matrix).

The matrix and vector entries for the above system are given by (boundary conditions have been omitted):

$$llM_{ij} = \int_\Omega \rho c \phi_i \phi_j d\Omega, \tag{6.2.4}$$

$$S_{ij} = \int_\Omega \nabla \phi_i \cdot (\kappa \nabla \phi_j) d\Omega, \tag{6.2.5}$$

$$L_i = \rho L \int_\Omega \phi_i f_l d\Omega, \tag{6.2.6}$$

$$q_i = \int_\Omega \phi_i q d\Omega. \tag{6.2.7}$$

The distinct feature of the temperature based approach is the use of discontinuous integration in space. The key idea behind discontinuous integration, as for instance described by Fachinotti et al., is that for elements intersected by the moving interface, the integrals (6.2.4 - 6.2.7) are not computed over an element as a whole at once, using for instance an averaged value for the physical parameters, but are instead computed over the liquid and solid subdomains separately.
For details on the evaluation of the integrals and the solution of the non-linear system, the reader is referred to the thesis of John Brusche.

Input for the various subroutines

- Definition of the storage scheme:

  Except for 1D problems it is advised to use an iterative solver. Hence use

  ```
  storage_scheme = compact
  ```

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
  For each element group 20 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 15 are real coefficients.

  These parameters and coefficients are defined as follows:

1	not yet used
2	not yet used
3	type of numerical integration
4	type of co-ordinate system
5	not yet used
6	κ_s
7	κ_l
8	ρ
9	c_s
10	c_l
11	Q, i.e. source term
12	L (latent heat)
13	T_m (melting temperature)
14-20	not yet used

  Parameters that are not yet used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
  The coefficients 6-20 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors.

6.3 The heat capacity method

Equation

$$C^* \frac{\partial T}{\partial t} - \nabla \cdot (k \cdot \nabla T) = q \tag{6.3.1}$$

i.e.
$$C^* \tfrac{\partial T}{\partial t} - \sum_{i=1}^{n} \sum_{j=1}^{n} \tfrac{\partial}{\partial x_i} k \tfrac{\partial T}{\partial x_i} = q$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \Omega \subset I\!R^n$$

where C^* denotes the "effective" heat capacity,
k the thermal conductivity,
T the temperature, and
q a source.

The conditions at the freezing front are:
$T_1 = T_2 = T_f \qquad (t > 0, x = s(t))$
and $k_1 \frac{\partial T_1}{\partial n} - k_2 \frac{\partial T_2}{\partial n} = \rho_1 L \frac{\partial s}{\partial t} \qquad x = s(t)$
where T_1 and k_1 correspond to the solid phase, and T_2 and k_2 to the liquid phase.
L is the latent heat,
T_f the freezing temperature and
s the freezing front position.

The freezing front is accounted for implicitly, and does not have to be specified.

Boundary conditions

The following types of boundary conditions are available:

Type 1: $T(\mathbf{x})$ given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2: $k \frac{\partial T}{\partial n} + \sigma(\mathbf{x}) T(\mathbf{x}) = h(\mathbf{x}) \qquad (\sigma(\mathbf{x}) \geq 0)$
on some part of the boundary. This is a so-called natural boundary condition. In general boundary elements are necessary, except in the case that $\sigma(\mathbf{x}) = 0$ **and** $h(\mathbf{x}) = 0$, when there is no need to give any condition on this part of the boundary.

Solution method

The freezing front may be defined by one temperature T_f, or by a freezing interval $2\Delta T_f$ defined by:

$$T_{f_1} = T_f - \Delta T_f \leq T \leq T_f + \Delta T_f = T_{f_2}$$

where T_{f_1} and T_{f_2} are the solidus and liquidus temperature respectively. The "effective" heat capacity C^* is defined by: $C^* = \frac{\partial H}{\partial T}$
where the enthalpy H satisfies:

$$H(T) = \int_{T_0}^{T} \rho C_1(T) dT \qquad\qquad T < T_{f_1}$$

$$H(T) = H(T_{f_0}) + \int_{T_{f_1}}^{T} [\rho \frac{\partial L(T)}{\partial T} + \rho C_f(T)]dT \qquad T_{f_1} \le T \le T_{f_2}$$

or $$H(T) = H(T_{f_0}) + \rho L \qquad \text{when } \Delta T_f = 0 \text{ and } T = T_f$$

$$H(T) = H(T_{f_1}) + \int_{T_{f_1}}^{T} \rho C_2(T)dT \qquad T > T_{f_2}$$

where ρC_f denotes the heat capacity in the freezing interval.
The heat capacity C^* may be evaluated directly from $\frac{\partial H}{\partial T}$, or in order to prevent overshoot of the freezing front, an approximation may be used. The following approximations are available:
"Lemmon-approximation":

$$C^* = \left[\frac{\left(\frac{\partial H}{\partial x_1}\right)^2 + \left(\frac{\partial H}{\partial x_2}\right)^2 + \left(\frac{\partial H}{\partial x_3}\right)^2}{\left(\frac{\partial T}{\partial x_1}\right)^2 + \left(\frac{\partial T}{\partial x_2}\right)^2 + \left(\frac{\partial T}{\partial x_3}\right)^2} \right]^{\frac{1}{2}}$$

"Del Guidice-approximation":

$$C^* = \frac{\frac{\partial H}{\partial x_1}\frac{\partial T}{\partial x_1} + \frac{\partial H}{\partial x_2}\frac{\partial T}{\partial x_2} + \frac{\partial H}{\partial x_3}\frac{\partial T}{\partial x_3}}{\left(\frac{\partial T}{\partial x_1}\right)^2 + \left(\frac{\partial T}{\partial x_2}\right)^2 + \left(\frac{\partial T}{\partial x_3}\right)^2}$$

After space discretization, the heat equation reduces to:

(*) $\quad \mathbf{C}\frac{\partial \mathbf{T}}{\partial t} + \mathbf{K}\mathbf{T} = \mathbf{F}$
with $\mathbf{C}$ the heat capacity matrix, $\mathbf{K}$ the heat conduction matrix and $\mathbf{F}$ represents the effect of the right hand side and boundary conditions.

The following algorithms can be used for the solution of (*):

(i) Θ-method

$\mathbf{C}^{n+\Theta}\frac{\mathbf{T}^{n+1}-\mathbf{T}^n}{\Delta t_n} + \mathbf{K}^{n+\Theta}(\Theta\mathbf{T}^{n+1} + (1-\Theta)\mathbf{T}^n) = \mathbf{F}^{n+\Theta}$

with $0.5 \le \Theta \le 1$, Δt_n the local time step, and $\mathbf{T}^{n+\Theta} = \Theta\mathbf{T}^{n+1} + (1-\Theta)\mathbf{T}^n$
or: $(\mathbf{C}^{n+\Theta} + \Theta\Delta t_n \mathbf{K}^{n+\Theta})\mathbf{T}^{n+1} = (\mathbf{C}^{n+\Theta} - (1-\Theta)\Delta t_n \mathbf{K}^{n+\Theta})\mathbf{T}^n + \Delta t_n \mathbf{F}^{n+\Theta}$

$\mathbf{C}^{n+\Theta}$ is approximated by : $\mathbf{C}(\Theta\mathbf{T}^{n+1,i} + (1-\Theta)\mathbf{T}^n)$
with $\quad \mathbf{T}^{n+1,0} = (1 + \frac{\Delta t_n}{\Delta t_{n-1}})\mathbf{T}^n - \frac{\Delta t_n}{\Delta t_{n-1}}\mathbf{T}^{n-1}$
Usually only one or two iterations i are necessary.

$\mathbf{K}^{n+\Theta}$ is approximated by $\mathbf{K}(\Theta\mathbf{T}^{n+1,i} + (1-\Theta)\mathbf{T}^n)$

Remark: The matrix $\mathbf{C}$ is lumped, i.e. it is a diagonal matrix.

Practical implementation

(i) initial condition: set $t = 0, \mathbf{T}^0$ and $\mathbf{T}^{-1}$.

(ii) Time stepping algorithm to be performed until the end time has been reached:

a. set Δt_n.
$\mathbf{T}^{n+1,0} = (1 + \frac{\Delta t_n}{\Delta t_{n-1}}\mathbf{T}^n - \frac{\Delta t_n}{\Delta t_{n-1}}\mathbf{T}^{n-1}$ (subroutine MANVEC).

b. Build matrix $\mathbf{K}$ and diagonal matrix $\mathbf{C}$ with the old solution $\mathbf{T}^{n+\Theta,i}$
(subroutine BUILD).
The diagonal matrix $\mathbf{C}$ is stored as a right-hand side vector. The effect of boundary

conditions may **not** be taken into account.

Build right hand side vector, $\mathbf{F}^{n+\Theta}$ including the effect of boundary conditions (subroutine BUILD).

When $\Theta \neq 1$: compute vector $\mathbf{K}^{n+\Theta,i}\mathbf{T}^n$ (subroutine MAVER).

Compute vector $\mathbf{C}^{n+\Theta,i}\mathbf{T}^n - (1-\Theta)\Delta t_n \mathbf{K}^{n+\Theta,i}\mathbf{T}^n + \Delta t_n \mathbf{F}^{n+\Theta}$ (subroutine MAVER + MANVEC).

Compute the matrix $\mathbf{C}^{n+\Theta,i} + \Theta\Delta t_n \mathbf{K}^{n+\Theta,i}$ (subroutine ADDMAT).

Solve the BUILD of equations:
$(\mathbf{C}^{n+\Theta,i} + \Theta\Delta t_n \mathbf{K}^{n+\Theta,i})\mathbf{T}^{n+1,i+1} = (\mathbf{C}^{n+\Theta,i} - (1-\Theta)\Delta t_n \mathbf{K}^{n+\Theta,i})\mathbf{T}^n + \Delta t_n \mathbf{F}^{n+\Theta}$
(Subroutine SOLVE).

c. Repeat step b. until convergence has been achieved:
$\| \mathbf{T}^{n+1,i+1} - \mathbf{T}^{n+1,i} \| < \epsilon$ (subroutine DIFFVC or ANORM).

d. $\mathbf{T}^{n+1} = \mathbf{T}^{n+1,i+1}$
$t^{n+1} = t^n + \Delta t_n$, return to step (ii).

Input for the various subroutines

Subroutine BUILD

(i) Input for the differential equation (matrices **C** and **K**).

The user must specify the parameters METHOD, ICOORD as well as the coefficients C^*, k and q for each standard element. The specification of METHOD and ICOORD must be stored in array IUSER, the specification of q in IUSER and USER. For the parameters C^* and k function subroutines are required. Subroutine FIL100 may be used for the storage in arrays IUSER and USER. The parameter IPARM in FIL100 must be equal to 3. The sequence of the parameters in IUSER is:

METHOD, ICOORD, q.

Meaning of the parameters:

METHOD indicates whether the matrices or the right hand side must be built, and the type of approximation to be used. Possibilities:

1 Only the right hand side f is built. This possibility is only necessary when $q \neq 0$.
>1 The matrices **C** and **K** are built.
2 The "Lemmon" approximation is used for **C**. **K** is evaluated directly.
3 The "Del Guidice" approximation is used for **C**. **K** is evaluated directly.
4 The "Lemmon" approximation is used for **C** and **K**.
5 The "Del Guidice" approximation is used for **C** and **K**.

ICOORD indicates the type of co-ordinates to be used. Possibilities:

1 Cartesian co-ordinates.
2 Axi-symmetric co-ordinates.
q Right hand side function (may be zero).

For the parameters C^* and k three or four function subroutines are required, depending on the value of METHOD:

FNH000 This function subroutine gives the value of enthalpy H as function of the temperature T. Call:

VALUE = FNH000(T),

where T gets a value in the element subroutine and FNH000 gets a value in the function subroutine.

FNC000 This function subroutine gives the value of the heat capacity ρC as function of the temperature T. Call:

VALUE = FNC000(T)

FNK000 This function subroutine gives the value of the heat conductivity k as function of the temperature T. Call:

VALUE = FNK000(T)

When METHOD = 4 or 5 also a function subroutine for the integral of k is needed:

FNK001 This function subroutine gives the value of the integral of the heat conductivity k as function of the temperature T:

FNK001(T) $= \int_{T_0}^{T} k(T)dT.$

Call:

VALUE = FNK001(T)

The functions FNH000. FNC000, FNK000 and FNK001 must be programmed as follows:

```
FUNCTION FNx00. ( T )
IMPLICIT NONE
DOUBLE PRECISION FNx00., T, alpha

statements to give alpha a value

FNx00. = alpha

END
```

In this function subroutine x denotes H, C or K, and . 0 or 1.

(ii) Input for the boundary conditions.

Essential boundary conditions must be filled in the solution vector. This may be done by subroutine BVALUE (User Manual 5.3).
When boundary elements are necessary (boundary conditions of type 2 with $\sigma(\mathbf{x}) \neq 0$ or $g(\mathbf{x}) \neq 0$), the user must specify the coefficients $\sigma(\mathbf{x})$ and $g(\mathbf{x})$ and store this information in the arrays IUSER and USER. Subroutine FIL101 may be used for this storage. The sequence of the parameters in the arrays IUSER and USER is: $\sigma,\ h$.
The parameter IPARM in FIL101 must be equal to 2.

Subroutine COMMAT

The user should set JMETOD = 1 in the call of subroutine COMMAT.

Subroutine SOLVE

IPOS = 1 must be used in the call of subroutine SOLVE.

Subroutine DERIVA

The parameters ICHOIS and IVEC are computed by the subroutine. Parameter IX defines the contents of the output vector IOUTVC. See output.
ICHELD and JDEGFD must be equal to 1.

Output of some subroutines

Subroutine SOLVE: T in the nodal points.

Subroutine DERIVA:
When ICHELD = 1: $\frac{\partial T}{\partial x_{IX}}$ in the vertices of the element. Hence when $IX = 2$: $\frac{\partial T}{\partial x_2}$

Available element types and problem definition numbers

6.1.1 Linear triangle in $I\!R^2$

Differential equation:
element shape number for mesh generation (see 2.2): 3.
Problem definition number: 500

Boundary conditions of type 2:
element shape number for mesh generation (see 2.2): 1.
Boundary problem number : 102

7 Flow problems

In this chapter some elements for flow problems are described.
In order to use these elements it is necessary that you have the special license to run Navier-Stokes elements. Inform at your local installation officer whether you have this license.

The following Sections are available:

7.1 is devoted to isothermal laminar incompressible and weakly compressible flow problems.
Examples for this type of flows can be found in the Introduction Section 7.3 (Flow in T-shaped region using the penalty function method) and in the Users Manual Section 6.3.1 (Flow through a bend). Also in the manual SEPRAN EXAMPLES there are a lot of examples with these elements.

7.2 deals with temperature-dependent laminar incompressible flow problems.

7.3 treats isothermal turbulent incompressible flow.

7.1 The isothermal laminar flow of incompressible or slightly compressible liquids

In this section we consider the laminar flow of incompressible liquids described by the Navier-Stokes equations. These equations consist of three parts:

- The conservation of mass.
- The conservation of momentum.
- The constitutive equation.

In the non-isothermal case these relations must be extended with a temperature equation (see Section 7.2) and in the turbulent case with extra equations for the turbulence modeling (see Section 7.3). In this section we restrict ourselves to isothermal laminar flow.

7.1.1 Navier-Stokes Equations

The set of Navier-Stokes equations and continuity equation is given by:

The conservation of mass
In case of incompressible flow:

$$\text{div } \mathbf{v} = 0 \tag{7.1.1}$$

or alternatively in case of slightly compressible flow:

$$\text{div } \rho\mathbf{v} = 0 \tag{7.1.2}$$

To make things more general it is also allowed to give a right-hand side $\mathbf{f_{div}}$ instead of zero.

Conservation of momentum (Euler-Cauchy equations):

$$\rho[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} + 2\boldsymbol{\Omega}\times\mathbf{v}] + \nabla P - \text{div } \mathbf{t} = \rho\mathbf{f} \tag{7.1.3}$$

The parameters in the Equations (7.1.1) and (7.1.3) have the following meaning:

ρ density of the liquid.

$\mathbf{v}$ velocity.

$\boldsymbol{\Omega}$ angular velocity $\boldsymbol{\Omega} = (0, 0, \omega)$ of rotating co-ordinate system with respect to an inertial system $\mathbf{x}$.

$\mathbf{f}$ external force field (body force).

$\mathbf{t}$ deviatoric stress tensor.

The stress tensor $\boldsymbol{\sigma}$ consists of two parts: the pressure part and the deviatoric stress tensor according to:

$$\boldsymbol{\sigma} = -P\mathbf{1} + \mathbf{t}. \tag{7.1.4}$$

The generalized or reduced pressure P can be written as

$$P = p + \frac{1}{2}\rho(\boldsymbol{\Omega}\times\mathbf{x})\cdot(\boldsymbol{\Omega}\times\mathbf{x}), \tag{7.1.5}$$

where p denotes the hydrostatic pressure. Mark that the quantity P contains the contribution of all conservative forces.

In order to model the deviatoric stress tensor $\mathbf{t}$ it is necessary to add the constitutive equations. These equations depend on the type of material used. In SEPRAN a Newtonian model can be used, but also a number of non-Newtonian models are available. In this section we restrict ourselves to generalized Newtonian fluids.
The following models are available:

- Newtonian model
- Power-law model
- Carreau model
- Plastico-viscous model
- Three user type models

These models are characterized by their constitutive relations.

Constitutive equations:

The constitutive equations have all the form:

$\mathbf{t} \ = \ \eta(\nabla\mathbf{v})(\nabla\mathbf{v} + \nabla\mathbf{v}^T)$

The choice of $\eta(\nabla\mathbf{v})$ defines the model

- Newtonian fluid: η is constant.
- Power-law liquid: $\eta \ = \ \eta_n II^{\frac{n-1}{2}}$
- Carreau liquid: $\eta \ = \ \eta_c(1+\lambda II)^{\frac{n-1}{2}}$
- Plastico-viscous liquid: $\eta \ = \ \eta_{pv}(1+[\frac{s}{\eta_{pv}} II^{-\frac{1}{2}}]^{\frac{1}{n}})^n$
  $\lim_{II\to 0} \frac{1}{2}\mathbf{t} : \mathbf{t} = \lim_{II\to 0} \eta^2 II = s^2$
- user model 1: $\eta \ =$ FNV000$(\nabla\mathbf{v})$
- user model 2: $\eta \ =$ FNV001(II)
- user model 3: $\eta \ =$ FNV002$(x_1, x_2, x_3, v_1, v_2, v_3, II)$

η denotes the dynamic viscosity (> 0) and s the yield stress. The parameters η_n, η_c, η_{pv} and n must be all positive. These parameters including s must be provided by the user.

II denotes the second invariant of the velocity deformation tensor, defined by
$II = \frac{1}{2}\mathbf{A}_1 : \mathbf{A}_1$; $\mathbf{A}_1 = \nabla\mathbf{v} + \nabla\mathbf{v}^T$.

A specially scaled form of the equations

Besides the standard form of the Navier-Stokes equations as treated before it is also allowed to use the following specially scaled form of these equations.
Mark that this form can only be used for Cartesian coordinates in two dimensions.
The Navier-Stokes equations in the special form read:

Continuity

$$cdiv_1\frac{\partial v_1}{\partial x_1} + cdiv_2\frac{\partial v_2}{\partial x_2} \tag{7.1.6}$$

Momentum

$$-cvisc_1\frac{\partial v_1^2}{\partial x_1^2} - cvisc_2\frac{\partial v_1^2}{\partial x_2^2} + cconv_1 v_1\frac{\partial v_1}{\partial x_1} + cconv_2 v_2\frac{\partial v_1}{\partial x_2} + cgrad\frac{\partial p}{\partial x_1} = f_1 \tag{7.1.7}$$

$$-cvisc_3\frac{\partial v_2^2}{\partial x_1^2} - cvisc_4\frac{\partial v_2^2}{\partial x_2^2} + cconv_3 v_1\frac{\partial v_2}{\partial x_1} + cconv_4 v_2\frac{\partial v_2}{\partial x_2} + cgrad\frac{\partial p}{\partial x_2} = f_2 \tag{7.1.8}$$

7.1.2 Boundary Conditions

In the instationary case it is necessary to give an initial condition for the velocity at $t = 0$.

The following types of boundary conditions are available:

Type 1 Components of the velocity $\mathbf{v}(\mathbf{x})$ given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type 2 Stress tensor components given on a part of the boundary. This is a so-called natural boundary condition.
Define σ_n as the stress component in the normal direction of a surface, and $\boldsymbol{\sigma}_t$ as the tangential component. Define v_n as the velocity component in the normal direction of a surface and $\mathbf{v}_t$ as the tangential component. In R^2 $\boldsymbol{\sigma}_t$ and $\mathbf{v}_t$ are scalar quantities.

Type 3 Mass flux given. This boundary condition is usually applied in combination with periodical boundary conditions. If the velocities at opposite boundaries are periodical and the pressure is periodical but with a pressure difference between the two sides, then we need an extra condition to prescribe the difference. A possibility is to give the mass flux. This parameter implicitly prescribes the pressure difference.

Type 4 (discontinuous boundary condition) This is the same type of boundary condition as used in the membrane in Section (3.1). Suppose the region is subdivided in an upper region (u) and a lower region (l) separated by a membrane. Assume furthermore that the solution jumps over this membrane and hence is discontinuous. Furthermore we assume that equation (7.1.3) holds for both the upper part and the lower part and that the coefficients are the same at the membrane.
At the membrane we assume the following boundary conditions:

$$\sigma_n = T_n - C_n(v_n^{upper} - v_n^{lower}) \tag{7.1.1}$$

$$\sigma_t = T_t - C_t(v_t^{upper} - v_t^{lower}) \tag{7.1.2}$$

So there is no combination of tangential and normal components. v^{upper} means the value at the upper region and v^{lower} the value at the lower region. This boundary condition implies that the values at both sides of the membrane are different. In order to use this boundary condition connection elements as described in the Users Manual Section 2.2 are necessary. These connection elements must connect linear or quadratic line elements in R^2 or surface elements in R^3.
So for example in case of quadratic elements in R^2 one has to use quadratic line elements as connection elements, like

```
celmj = curves 2 ( ck, cl )
```

The parameter 2 indicates that it concerns quadratic elements
So contrary to other boundary conditions it is not longer possible to use so-called boundary elements, but these elements must be used in the same way as internal elements.
If boundary conditions of this type are applied the resulting matrices are non-symmetrical, which implies that a non-symmetrical storage must be used.
It is not necessary to have both boundary conditions (7.1.1) and (7.1.2). If only one of the two is used, the coefficients of the other one must be made equal to zero. Of course in that case for that component another boundary condition is required.
For an example of the use of the boundary condition see the manual SEPRAN EXAMPLES Section 7.1.15.1.

The following combinations are permitted:

1. $c_n v_n + \sigma_n$ and $\mathbf{c}_t\mathbf{v}_t + \boldsymbol{\sigma}_t$ given; $\quad \sigma_n = \mathbf{n}\cdot\boldsymbol{\sigma}\cdot\mathbf{n}\,, \quad \boldsymbol{\sigma}_t = \boldsymbol{\sigma}\cdot\mathbf{n} - \sigma_n\mathbf{n}$

2. $c_n v_n + \sigma_n$ given and $\mathbf{v}_t$ given.
3. $\mathbf{c}_t \mathbf{v}_t + \boldsymbol{\sigma}_t$ given and v_n given.
4. periodical boundary conditions and mass flux given.
5. combination with the membrane boundary conditions

n denotes the outward normal on the boundary and **t** the tangential vector. In case (2) $\boldsymbol{\sigma}_t = \mathbf{0}$ and in case (3) $\sigma_n = 0$ may be used, since these values are not utilized. The boundary conditions $\mathbf{v}_t$ given and v_n given are essential boundary conditions. When these boundary conditions are used with **n** and **t** not in the direction of the co-ordinate axis, local transformations are necessary (see the Users Manual, Section 1.2.4).
When $\sigma_n \neq 0$ and/or $\boldsymbol{\sigma}_t \neq \mathbf{0}$, boundary elements for this type are necessary, when $\sigma_n = 0$ (or non-prescribed) and $\boldsymbol{\sigma}_t = \mathbf{0}$ (or non-prescribed) no boundary elements have to be used.

Remark When at no part of the boundary the normal stress is given (in other words if everywhere at the boundary the normal velocity component is prescribed), the pressure P is fixed up to an additive constant. If the penalty function formulation is used this has no consequences, however, if the integrated method in combination with a direct linear solver is used, this means that the pressure must be prescribed in some point.

The following combinations of boundary conditions are frequently used:

1. no-slip condition: $\mathbf{v} = \mathbf{0}$
2. free-slip condition: $v_n = 0$ and $\boldsymbol{\sigma}_t = \mathbf{0}$.
3. symmetry condition: $v_n = 0$ and $\boldsymbol{\sigma}_t = \mathbf{0}$.
4. instream/outstream condition: σ_n given and $\boldsymbol{\sigma}_t$ given.
   At fully developed flow: $\sigma_n = -P$, $\mathbf{v}_t = 0$.
5. Periodical boundary conditions with mass flux given. For examples see the manual SEPRAN EXAMPLES Sections 7.1.9, 7.1.10 and 7.1.11.

Mark that the normal component of the stress $\boldsymbol{\sigma}_n$ is equal to $-P+\eta\frac{\partial u_n}{\partial n}$. As a consequence prescribing the normal stress is usually identical to prescribing the pressure, since in practice frequently either η is small or $\frac{\partial u_n}{\partial n}$ is small at those places where the normal stress is prescribed. In this definition **n** is the outward normal.

In order to prescribe a surface tension one has to prescribe the normal stress by $\sigma_n = \gamma(\frac{1}{R_1} + \frac{1}{R_2})$, with R_1 and R_2 the radii of curvature on the surface. In a 2D Cartesian case we have $R_2 = \infty$ and hence only R_1 is used.

7.1.3 Solution methods for the stationary Navier-Stokes equations

The finite element equations for the incompressible fluid can be written as:

$$\mathbf{M}\frac{\partial \mathbf{u}}{\partial t} + \mathbf{S}(\mathbf{u})\mathbf{u} + \mathbf{N}(\mathbf{u})\mathbf{u} - \mathbf{L}^T\mathbf{p} = \mathbf{F} \qquad \text{(momentum equations).} \tag{7.1.1}$$

$$\mathbf{L}\mathbf{u} = \mathbf{0} \qquad \text{(continuity equation).} \tag{7.1.2}$$

where **u** denotes the discretized velocity.
M denotes the mass matrix.
S denotes the stress matrix; for a generalized Newtonian fluid **S** depends on **u**.
$\mathbf{N}(\mathbf{u})\mathbf{u}$ is the discretization of the convective terms.
$-\mathbf{L}^T\mathbf{p}$ represents the ∇P term.

For the solution of the Navier-Stokes equations the following items are distinguished:

- incompressibility condition.
- non-linearity (due to convective terms and stress tensor in the case of non-Newtonian fluids)
- time dependence. See subsection (7.1.10)

Continuity equation

The equations (7.1.1) and (7.1.2) may in the stationary case be written in compact form as:

$$\begin{bmatrix} \mathbf{S} & \mathbf{L}^T \\ \mathbf{L} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{7.1.3}$$

In this case **S** stands for the combination of viscous terms and convective terms. It is clear that the main diagonal contains a so-called *zero-block* at the main diagonal due to the absence of the pressure in the continuity equation. This *zero-block* may give rise to extra complications and for that reason several solution techniques have been developed. We distinguish between the following three methods:

Integrated method or coupled approach Both the velocity and the pressure are used as degrees of freedom. In this case it is possible that, without precautions, the matrix to be inverted gets zero pivots during the elimination process. This is caused by the fact that the pressure is not present in the continuity equation. Since the continuity equation is the equation that is coupled to the pressure unknowns, this means that a zero diagonal element occurs. If no renumbering is performed, the first pivot during the elimination process may be zero and the LU-decomposition subroutines fail.

The method to overcome this problem is to renumber the unknowns in such a way that first the velocities and then the pressure unknowns are used. However, this type of numbering may result in a very large matrix and is for that reason inefficient.
The alternative is to renumber the unknowns per level as described in the Users Manual, Section 3.2.2, in the input block "PROBLEM". This numbering has been developed especially for Stokes and Navier-Stokes type problems and is at present the most efficient one to be used in case of an integrated method.

If an iterative solver is used to solve the resulting systems of linear equations, both the complete renumbering of unknowns and the renumbering per level may be applied, since in that case the size of the matrix is independent of the ordering. However, a number of experiments indicate that renumbering per level might result in a faster convergence. Since only a limited number of experiments have been carried out, it is too early to state that renumbering per level is superior to complete renumbering and the user is advised to experiment himself in this matter.
The only case where there renumbering of the unknowns does not improve convergence is in the case of *Taylor-Hood* elements with linear pressure and velocity approximation. In this particular case renumbering makes the convergence slower.

The system of linear equations, resulting from the integrated approach, may be solved by a direct or an iterative method. However, there is also the option to use a special block iterative methods, like the *simple* method the augmented Lagrangian method or the LSC method. SIMPLE is variant, of the *simple* method, used in finite differences or finite volumes. An example of how to use these block methods is described the manual SEPRAN EXAMPLES Section 7.1.20.

Examples of the integrated method may for example be found in the manual SEPRAN EXAMPLES Sections 7.1.1, 7.1.3, 7.1.4 and 7.1.8.

Penalty function method or decoupled approach In order to decouple pressure and velocity, thus reducing the system of equations to be solved, the penalty function method may be applied. This method has as extra advantage that there is no problem with zero pivots. The method can be described as follows:
The mass balance equation is perturbed:

$$\epsilon P \ + \ \text{div} \ \mathbf{v} \ = \ 0 \tag{7.1.4}$$

The small number ϵ must be chosen such that $\epsilon P \ = \ O(10^{-6})$. P is eliminated from the momentum balance equations. Hence the velocity $\mathbf{v}$ is considered as degree of freedom and P is computed as a derived quantity. For further details and a convergence proof of the method, see Girault and Raviart (1979). The reduced pressure P may contain all the conservative external force fields. When the body force $\mathbf{f}$ can be written as $\mathbf{f} = \nabla\phi$ then P may be defined as $P \ = \ p \ - \ \frac{1}{2}\rho(\mathbf{\Omega} \times \mathbf{x}) \ \cdot \ (\mathbf{\Omega} \times \mathbf{x}) \ - \ \rho\phi$.
This has consequences for the boundary conditions: for fully developed flow $\sigma_n = -P$.
When at no part of the boundary the normal stress is given, the pressure is fixed up to an additive constant. The penalty method sets this constant such that $\int\limits_{\Omega} P d\Omega \ = \ 0$.

In discretized form the penalty function method reads:

$$\mathbf{M}\frac{\partial \mathbf{u}}{\partial t} \ + \ \mathbf{S}(\mathbf{u})\mathbf{u} \ + \ \mathbf{N}(\mathbf{u})\mathbf{u} \ + \ \frac{1}{\epsilon}\mathbf{L}^T\mathbf{M}_p^{-1}\mathbf{L}\mathbf{u} \ = \ \mathbf{F}, \tag{7.1.5}$$

and the pressure is computed from:

$$\epsilon\mathbf{M}_p\mathbf{p} \ = \ -\mathbf{L}\mathbf{u}. \tag{7.1.6}$$

The matrix $\mathbf{M_p}$ is termed the pressure mass matrix.

A clear advantage of the penalty function method is the reduction of the size of the matrix. For standard two-dimensional problems this means usually a large reduction in computing time.
Disadvantages of the penalty function method are:

- The parameter ϵ must be chosen. For standard problems this is no problem, however, if the pressure becomes large as in very viscous fluids (usually of non-Newtonian type), it may be too difficult to choose a suitable value of ϵ.
- Due to the small parameter ϵ, which appears as $\frac{1}{\epsilon}$ in the matrix, the condition of the system of equations to be solved will be large. As a consequence loss of figures may be possible. More severe is that iterative solution techniques do not longer converge ones the penalty function method is applied. For large two-dimensional and moderate three-dimensional problems this means that the computation time will become very large. In that case the integrated method is recommended.

Examples of the penalty function method may be found in the manual SEPRAN EXAMPLES Sections 7.1.1, 7.1.2 and 7.1.8.

Method of divergence-free basis functions This method is not yet available.

Non-linearity

The non-linear terms (convective terms, generalized Newtonian model) must be linearized in order to solve the equations.
For the convective terms two techniques are available:

- Picard linearization (successive substitution): $(\mathbf{v} \cdot \nabla\mathbf{v})^{n+1} \approx \mathbf{v}^n \cdot \nabla\mathbf{v}^{n+1}$
- Newton iteration: $(\mathbf{v} \cdot \nabla\mathbf{v})^{n+1} \approx \mathbf{v}^n \cdot \nabla\mathbf{v}^{n+1} + \mathbf{v}^{n+1} \cdot \nabla\mathbf{v}^n - \mathbf{v}^n \cdot \nabla\mathbf{v}^n$
- "incorrect" Picard linearization: $(\mathbf{v} \cdot \nabla\mathbf{v})^{n+1} \approx \mathbf{v}^{n+1} \cdot \nabla\mathbf{v}^n$.
  This method is called "incorrect" since it usually does not converge.

For time-dependent problems a linearization per time step is sufficient. For a stationary problem iteration is necessary.

The iterative procedure is as follows:

- Start with an approximation $\mathbf{u}^0$.
  Good starting values are for example:
  a The vector $\mathbf{u}^0 = \mathbf{0}$ in the inner region and $\mathbf{u}^0$ is equal to the boundary conditions for the boundary.
  b The solution of the Stokes equations (convective terms neglected).
  c The solution of the Navier-Stokes equations for a smaller Reynolds number (larger value of the viscosity).
- Compute the solution of the non-linear system of equations. To that end the input block "NONLINEAR_EQUATIONS" must be utilized.

Remarks

For moderate values of the Reynolds number Re Newton's method can be used. Usually 3 to 5 iterations are sufficient for the convergence of the convective terms. The rate of convergence of the generalized Newtonian model decreases for decreasing values of n ($0 < n < 1$).
For larger values of Re it is recommended to use one Picard iteration before Newton's method is applied. When too many iterations are necessary, use a smaller Reynolds number to find a good initial estimate.

The linearization of the stress for the generalized Newtonian liquid requires special attention. For the plastico-viscous model, Picard linearization (successive substitution) is used. This linearization method is also applied to the power-law and Carreau model when $0 < n < 1$. The convergence of this type of iteration can be accelerated with the help of a relaxation factor $1 + \alpha(1 - n)$ with $\alpha \approx 0.4 - 0.6$ (see the description of the input block "NONLINEAR_EQUATIONS" in the Users Manual, with ω replaced by α). For $n > 1$, however, the power-law and Carreau model require the linearization by means of Newton's method. Relaxation is not required. SEPRAN itself automatically selects the correct method. One is strongly advised to perform the first iteration with help of the Newtonian model in order to obtain a reasonable initial estimate.

Types of elements

In general two classes of elements are distinguished:

- Elements of Taylor-Hood type with continuous pressure.
  These pressure is always defined in vertices and as a consequence, it is not possible to use the penalty function method. Hence one can only apply the integrated method.
- Elements of Crouzeix-Raviart type with discontinuous pressure.
  The pressure and possibly its gradient is defined in the centroid of the elements. For post-processing purposes it is always necessary to average the pressure to vertices.
  In this case both the integrated method and the penalty function method may be applied.

Treatment of the boundary condition: mass flux given

The combination of periodical boundary conditions with given mass flux and hence unknown pressure difference is a very special one. There are two implementations available:

- The implementation as one treated in Segal et al (1994) by a penalty function approach.

- The implementation using global unknowns. This approach is treated in Segal et al (1994) for the finite volume method.

The mass flux is given by

$$\int_{\Gamma} \mathbf{u} \cdot \mathbf{n} d\Gamma \ , \tag{7.1.7}$$

where Γ denotes the inflow boundary. Hence the Navier-Stokes equations are solved with the given mass flux as constraint. Corresponding to this constraint we have an unknown pressure jump between the inflow and outflow boundary.

- The penalty function method and can only be used in combination with a direct linear solver.
  For that method it is necessary to define a line element along the complete inflow boundary. This must already been done in the input for program SEPMESH. Along this line element a special boundary element is defined that prescribes the mass flux.
- If the option with global unknowns is used, there is no need to define a line element along the inflow boundary. It is sufficient to define a global unknown (the pressure jump) along the inflow or outflow boundary. Since no penalty parameter is used, it is no longer necessary to use a direct solution method.
  Also in this case a special boundary element is used, but this element is different from that in the penalty function approach.

At the opposite sides it is necessary to use periodical boundary conditions hence in the input of SEPMESH also connection elements must be defined. The pressure, however, contains a pressure difference over the inflow and outflow boundary and this means that we have to be sure that the pressure is not periodical. This may be achieved by excluding the connection elements when computing the pressure in the DERIVATIVES input block. This is only possible in combination with Crouzeix-Raviart elements.
For examples see the manual SEPRAN EXAMPLES Sections 7.1.9, 7.1.10 and 7.1.11.

Treatment of the boundary condition: pressure given

The boundary condition pressure given is never applied directly to the unknowns but always implicitly through the normal stress. This is even the case if Taylor-Hood elements are applied. In that case we have pressure unknowns at the boundary and one might assume that prescribing these pressures has the same effect as giving the normal stress. However, experiments with a simple channel flow, driven by a pressure difference, show that prescribing the pressures lead to completely wrong answers, whereas giving the normal stress produces a good approximation.

Treatment of the boundary condition: surface tension given

A given surface tension is treated as an extra force on the surface where this surface tension is given. As described in the theoretical manual this surface tension involves the radius of curvature, which in a finite element context is replaced by a derivative of the tangential vector along the surface. This is achieved by application of the Gauss divergence theorem for surfaces. As a consequence, at the begin and end point of a free surface curve in R^2 we get two "boundary" terms. If in an end point the normal velocity is prescribed this term vanishes, however, if the normal velocity is free in an end point we have to prescribe this boundary term. Actually this term appears to be equal to the surface tension coefficient γ times the tangential vector of the boundary. So implicitly this defines the so-called contact angle.
In other words, if at a boundary a surface tension is given and if in an end point not the normal velocity is prescribed, it is necessary to prescribe the tangential vector. For example in the case of a jet where you want a horizontal outflow, you must prescribe the tangential vector even though we can not speak about a contact angle in this case. Of course this is not necessary of the curve with the surface tension is closed.
An example of the use of surface tension in combination with a free surface is treated in Sections (7.6) and the manual SEPRAN EXAMPLES Section 7.6.1.

7.1.4 Coefficients for the differential equation

The following coefficients may be filled for the Navier-Stokes equations

MODEL = *name* (icoef2) Type of viscosity model
name is a string parameter with the following possible values:

NEWTONIAN (1) Newtonian liquid. In this case η is the dynamic viscosity.
This is the Default value.

POWER_LAW (2) Power-law liquid. In this case η is the parameter η_n and n the parameter n in the power of the model.

CARREAU (3) Carreau liquid. In this case η is the parameter η_c and n the parameter n in the power of the model. λ is the parameter λ the viscosity model.

PLASTICO_VISCOUS (4) Plastico-viscous liquid. In this case η is the parameter η_{pv} and n the parameter n in the power of the model. λ is the parameter s the viscosity model.

INTEGRATION_RULE = i (icoef3) defines the type of integration rule

COORDINATE_SYSTEM = *name* (icoef4) Type of co-ordinate system.
name is a string parameter with the following possible values:

CARTESIAN (0) Cartesian co-ordinates (x, y, z) (Default)

AXI_SYMMETRIC (1) Axisymmetric co-ordinates (2D grids only) (r, z)

POLAR (2) Polar co-ordinates (1D grids only) (r)

LINEARIZATION = *name* (icoef5) Type of linearization of the convective terms.
name is a string parameter with the following possible values:

NONE (0) Stokes flow, the convective terms $(\mathbf{v} \cdot \mathbf{v})$ are neglected. (Default)

PICARD (1) Linearization by Picard's method.

NEWTON (2) Linearization by Newton's method.

INCORRECT_PICARD (3) Linearization by the "incorrect" Picard method.

PENALTY (coef6) Parameter ε for the penalty function method.
If this parameter is given in a Taylor-Hood element it is neglected.
If this parameter is given in Crouzeix-Raviart element in combination with the integrated method it is used as a kind of artificial compressibility parameter. It might improve the performance in case an iterative solver is used for the system of linear equations.
Mark that this parameter is in fact the inverse of a penalty parameter and hence must have a small value.

DENSITY (coef7) Density ρ.

ANGULAR_VELOCITY (coef8) Angular velocity ω.

FORCE (coef9/10/11) Force vector.

X_FORCE (coef9) First component of body force $\mathbf{f}_1$.

Y_FORCE (coef10) Second component of body force $\mathbf{f}_2$.

Z_FORCE (coef11) Third component of body force $\mathbf{f}_3$.

VISCOSITY (coef12) Viscosity parameter η.

POWER_N (coef13) Viscosity model parameter n.

LAMBDA (coef14) Viscosity model parameter λ.

7.1.5 Coefficients for the natural boundary conditions

The coefficients for the natural boundary conditions of type 2 may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
For each element group 15 coefficients must be given, 5 integer coefficients (1 to 5) and 10 real coefficients.

These coefficients are defined as follows:

1 ILOAD: type of boundary condition
Possible values for ILOAD:

- 0 Prescribed stresses in the direction of the co-ordinate axis or in case of local transformations in the direction of the transformed co-ordinates.
- 1 Prescribed stresses in normal and tangential direction. At this moment this possibility has only been implemented for R^2
- 2 Surface tension given. At this moment this possibility has only been implemented in R^2
- 3 Friction option.
  The boundary condition is in this case $c_t \mathbf{v}_t + \sigma_t = \mathbf{f} \cdot \mathbf{t}$, with $\mathbf{t}$ the tangential vector.
  Usually this is a friction boundary condition with c_t the friction parameter and $\mathbf{f}$ the velocity of the surface along which the fluid flows.
  In fact this is a special case ILOAD=0, however, there are some special issues connected to this option:
  - This boundary condition may only be used in R^3.
  - This boundary condition may only be applied on a surface where a local transformation is defined. The first direction of the local transformation must be the normal direction, the other two directions must be tangential directions. How these tangential directions are defined is of no importance.
  - The vector $\mathbf{f}$ must be given in Cartesian coordinates. By the inner product with the tangential vector (extracted from the local transform) it is transformed in the correct direction.
  - The parameter c_t must be given as coefficient 9, the Cartesian components of the vector $\mathbf{f}$ in coefficients 6 to 8.

2 not yet used

3 Type of numerical integration.

- 0: the rule is chosen by the element itself (Default)
- > 0: the integration rule is defined by the user.

4 Type of co-ordinate system.
Possible values:

- 0 Cartesian co-ordinates
- 1 Axi-symmetric co-ordinates
- 2 Polar co-ordinates

5 not yet used The rest of the coefficients depend on the parameter ILOAD.

If ILOAD=0:

6 T_x (1-component in local co-ordinate system). If the 1-direction is the normal direction, this is the parameter σ_n

7 T_y (2-component in local co-ordinate system). If the 2-direction is the first tangential direction, this is the first component of $\boldsymbol{\sigma}_t$.

8 T_z (3-component in local co-ordinate system). If the 3-direction is the second tangential direction, this is the second component of $\boldsymbol{\sigma}_t$.

9 c_x (1-component of multiplication factor in local co-ordinate system).

10 c_y (2-component of multiplication factor in local co-ordinate system).

11 c_z (3-component of multiplication factor in local co-ordinate system).

12-15 not yet used

If ILOAD=1:

6 T_n, i.e. the normal component of the stress. The normal direction is defined clockwise with respect to the tangential direction. The tangential direction is defined by the direction of the curve from which the boundary elements are constructed.

7 T_t, i.e. the tangential component of the stress.

8-15 not yet used

If ILOAD=2:

6 γ, i.e. the coefficient for surface tension.
On the boundary where you prescribe the surface tension, it is not allowed to give local transformations.
If you want the combination local transformation with surface tension, please contact SEPRA.

7 is only used in case of a point element which is necessary to define the contact angle. It must then contain the t_1 component of the tangential vector in the point. Together with coefficient 8, this defines the tangential vector and hence the contact angle.

8 is only used in case of a point element which is necessary to define the contact angle. It must then contain the t_2 component of the tangential vector in the point.

9-15 not yet used

If ILOAD=3:

6 f_x, i.e. the x-component of the vector $\mathbf{f}$.

7 f_y, i.e. the y-component of the vector $\mathbf{f}$.

8 f_z, i.e. the z-component of the vector $\mathbf{f}$.

9 c_t, i.e. the friction coefficient.

10-15 not yet used

In the two-dimensional case σ_t must be defined in the direction of the curve and σ_n in direction of the outward normal.
For axi-symmetric problems with swirl either the component of $\boldsymbol{\sigma}_t$ perpendicular to the (r,z)-plane is supposed to be zero, or v_ϕ must be prescribed.

Definition of the coefficients for given mass flux:

The boundary condition of type 3 requires 10 coefficients, 5 integer ones (1 to 5) and 5 real ones (6-10). These coefficients are defined as follows:

icoef 1 Not yet used, must be zero.

icoef 2 Not yet used, must be zero.

icoef 3 Type of integration of the mass flux integral.
This parameter is only used if the penalty mass flux method with the large line element is applied.
Possible values:

- **1** Linear integration, i.e. Trapezoid rule. This must be used in combination with linear internal elements.
- **2** Quadratic integration, i.e. Simpsons rule. This must be used in combination with quadratic internal elements.

icoef 4 Type of coordinate system.
Possible values:

- **0** Cartesian coordinates.
- **1** Axi-symmetric coordinates.

icoef 5 Sequence number of degree of freedom that corresponds to the normal component. Possible values:

- **1** The normal component is the first component. This is usually the u-component, but it may also be the result of a local transformation.
- **2** The normal component is the second component. This is usually the v-component.
- **3** The normal component is the third component. This is usually the w-component.

coef 6 Amount of mass through the boundary, i.e. $\int_{\Gamma} \mathbf{u} \cdot \mathbf{n} d\Gamma$

coef 7 Penalty parameter.
This parameter is only used if the penalty mass flux method with the large line element is applied.
This parameter is necessary to force the constraint. Usually a value of 10^6 suffices.
Mark that this is a real penalty parameter, which means that it must have a large value. The penalty parameter in the continuity equation, however, is in fact the inverse of a penalty parameter and therefore must have a small value.

coef 8 Not yet used, must be zero.

coef 9 Not yet used, must be zero.

coef 10 Not yet used, must be zero.

7.1.6 Type numbers to be used in the problem input block

The following type numbers indicate that the (Navier-) Stokes equations must be solved.

NAVSTOKES_PENALTY (900) Navier-Stokes equations using Crouzeix Raviart elements with discontinuous pressure. The penalty function method is applied hence velocity and pressure are segregated.

NAVSTOKES_REDUCED (901) Navier-Stokes equations using Crouzeix Raviart elements with discontinuous pressure. Coupled approach. The gradient of the pressure and the velocity in the center are eliminated.

NAVSTOKES_CR (902) Navier-Stokes equations using Crouzeix Raviart elements with discontinuous pressure. Coupled approach.

NAVSTOKES_TH (903) Navier-Stokes equations using Taylor-Hood elements with continuous pressure. Coupled approach.

The natural boundary conditions do not have to be indicated by a type number, but internally type 901 is used.

7.1.7 Derivatives

Depending on the parameter ICHELD in the input block "DERIVATIVES" the following types of derivatives are computed:

1 The derivative $\frac{\partial v_{JDEGFD}}{\partial x_{IX}}$, with JDEGFD and IX parameters in the input block "DERIVATIVES".

2 $\nabla \mathbf{v}$ is computed in the nodes of the elements.

3 $-\nabla \mathbf{v}$ is computed in the nodes of the elements.

4 div $\mathbf{v}$ is computed in the nodes of the elements.

5 curl $\mathbf{v}$ is computed in the nodes of the elements.

6 $\mathbf{t}$ is computed in the nodes of the elements.

7 The pressure is computed in the nodes of the elements.
This value may only be applied in case of Crouzeix-Raviart type elements. Taylor-Hood elements contain the pressure as unknown in the equations.

8 The rate of elongation $\dot{\epsilon}$ is computed in the nodes of the elements.
$\dot{\epsilon}$ is the rate of strain in the direction of the velocity: $\dot{\epsilon} = \frac{1}{2}\mathbf{v} \cdot \mathbf{A}_1 \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$

9 The shear rate $\dot{\gamma}$ is computed in the nodes of the elements, according to the definition:
$\dot{\gamma} = \mathbf{n}_1 \cdot \mathbf{A}_1 \mathbf{n}_2$, $\mathbf{n}_1 \cdot \mathbf{n}_2 = 0$, $\|\mathbf{n}_1\| = \|\mathbf{n}_2\| = 1$
For problems with two velocity components: $\|\mathbf{n}_1\| = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, $\|\mathbf{n}_2\| = \frac{(-v_2, v_1)}{\|\mathbf{v}\|}$
For problems with three velocity components:
$\dot{\gamma}_{12} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_2$, $\dot{\gamma}_{13} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$, $\dot{\gamma}_{23} = \mathbf{e}_2 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$

10 $II^{\frac{1}{2}}$ is computed in the nodes of the elements.

11 The viscous dissipation $\frac{1}{2}\mathbf{t} : \mathbf{A}_1$ is computed in the nodes of the elements.

12-20 All quantities are computed per element (vector of type 116).

12-16 See 2-6.

17 See 11.

18-19 See 8-9.

20 The pressure in the centroid is stored in a vector of special structure defined per element. (vector of type 116).

21-31 See 1-11, however, now defined in the vertices instead of the nodal points.

32-42 See 1-11, however, all quantities computed per element per node. (vector of type 126)

43-53 See 1-11, however, all quantities computed per element per integration point. (vector of type 129)

The stress $\mathbf{t}$ is defined in exactly the same way as in Section (5.1). Hence this tensor has always 6 components independent of the dimension of the space.
In Cartesian co-ordinates these components are $\mathbf{t} = [t_{xx}, t_{yy}, t_{zz}, t_{xy}, t_{yz}, t_{zx}]^T$.
In axi-symmetric co-ordinates: $\mathbf{t} = [t_{rr}, t_{zz}, t_{\phi\phi}, t_{rz}, t_{z\phi}, t_{\phi r}]^T$.

The gradient vector $\nabla\mathbf{v}$ is defined as follows:
2D Cartesian co-ordinates: $\frac{\partial v_1}{\partial x_1}$, $\frac{\partial v_2}{\partial x_1}$, $\frac{\partial v_1}{\partial x_2}$, $\frac{\partial v_2}{\partial x_2}$
3D Cartesian co-ordinates: $\frac{\partial v_1}{\partial x_1}$, $\frac{\partial v_2}{\partial x_1}$, $\frac{\partial v_3}{\partial x_1}$, $\frac{\partial v_1}{\partial x_2}$, $\frac{\partial v_2}{\partial x_2}$, $\frac{\partial v_3}{\partial x_2}$, $\frac{\partial v_1}{\partial x_3}$, $\frac{\partial v_2}{\partial x_3}$, $\frac{\partial v_3}{\partial x_3}$,
Axi-symmetric co-ordinates without swirl: $\frac{\partial v_r}{\partial r}$, $\frac{\partial v_z}{\partial r}$, $\frac{\partial v_r}{\partial z}$, $\frac{\partial v_z}{\partial z}$
3D Cylinder co-ordinates: $\frac{\partial v_r}{\partial r}\frac{\partial v_\phi}{\partial r}\frac{\partial v_z}{\partial r}\frac{\partial v_r}{r\partial\phi} - \frac{v_\phi}{r}\frac{\partial v_\phi}{r\partial\phi} + \frac{v_r}{r}\frac{\partial v_z}{r\partial\phi}\frac{\partial v_r}{\partial z}\frac{\partial v_\phi}{\partial z}\frac{\partial v_z}{\partial z}$
Polar co-ordinates: $\frac{\partial v_r}{\partial r}\frac{\partial v_\phi}{\partial r}\frac{\partial v_r}{r\partial\phi} - \frac{v_\phi}{r}\frac{\partial v_\phi}{r\partial\phi} + \frac{v_r}{r}$
The storage of the solution vector depends on the type of element.
The storage of the vectors of special structure is as follows:

IVEC = 1 1 unknown per point

IVEC = 2 2 unknowns per point

IVEC = 3 3 unknowns per point

IVEC = 4 6 unknowns per point

IVEC = 5 $NDIM$ unknowns per point

IVEC = 6 1 unknown per vertex

IVEC = 7 2 unknowns per vertex

IVEC = 8 3 unknowns per vertex

IVEC = 9 6 unknowns per vertex

IVEC = 10 $NDIM$ unknowns per vertex

IVEC = 11 3 (R^2) or 6 (R^3) unknowns per point

IVEC = 12 1 (R^2) or 3 (R^3) unknowns per point

IVEC = 13 4 (R^2) or 9 (R^3) unknowns per point

IVEC = 14 3 (R^2) or 6 (R^3) unknowns per vertex

IVEC = 15 1 (R^2) or 3 (R^3) unknowns per vertex

IVEC = 16 4 (R^2) or 9 (R^3) unknowns per vertex

The output vector is defined as follows:

ICHELD=1,4,7,8,10,11 IVEC=1

ICHELD=2,3 IVEC=13

ICHELD=5,9 IVEC=12

ICHELD=6 IVEC=4

ICHELD=20 Vector of special structure defined per element, with one unknown per element

ICHELD=21,24,27,28,30,31 IVEC=6

ICHELD=22,23 IVEC=16

ICHELD=25,29 IVEC=15

ICHELD=26 IVEC=9

7.1.8 Integrals

If the user wants to compute integrals over the solution, he may use the option INTEGRAL in the input block "STRUCTURE"

The parameter ICHELI in the input block "INTEGRALS" is used to distinguish the various possibilities:

ICHELI=1 $\int_\Omega f(x)d\Omega$

ICHELI=2 $\int_\Omega f(x)v_{JDEGFD}(x)d\Omega$

ICHELI=2+i $\int_\Omega f(x)\frac{\partial v_{JDEGFD}}{\partial x_i}d\Omega \qquad (i= 1,2,3)$

In this case $\mathbf{v}(\mathbf{x})$ is the vector Vj as indicated by the command INTEGRAL in the input block "STRUCTURE:. The user must define the function $f(x)$ as first coefficient by one of the methods described in 2.2.
For each element group 10 parameters and coefficients must be given. The first 3 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 7 are real coefficients.

These parameters and coefficients are defined as follows:

1	type of numerical integration, see the coefficients for the equation
2	type of co-ordinate system, see the coefficients for the equation
3	not yet used
4	f
5-10	not yet used

7.1.9 Other input issues for the Navier-Stokes equations

In this subsection we treat other issues that are important when solving the Navier-Stokes equations

- **Definition of the storage scheme**:
  The resulting finite element matrices are in general non-symmetric. Only in the case of a Newtonian Stokes flow in combination with the penalty function method the matrices are symmetric and positive definite.

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input block "MATRIX" of program SEPCOMP.

  In the case of a Stokes problem with Newtonian constitutive equation the system of equations produced by the penalty method is symmetric and positive definite. In that case METHOD = 1 may be chosen. In all other cases METHOD should be equal to 2 (direct

method) or 6 (iterative solution method).

- **Parameters with respect to the linear solver**:

  In the case of the Stokes problem with Newtonian viscosity model the matrix is not only symmetric but also positive definite if the penalty method is applied. In all other cases the matrix is non-symmetric.

  For small problems the direct solver is probably the best, however, for large problems it is advised to use an iterative solver.
  Note that in case of the penalty function method, iterative solvers do not converge, hence such solvers are restricted to the integrated approach. In case of an iterative solver, in general one has to renumber the unknowns, in order to avoid zeros on the main diagonal, except in the special case of linear Taylor-Hood elements. Furthermore, one must always use an ILU preconditioner.
  Also for an iterative solver, Picard linearization (icoef5=1) in general is more robust than Newton linearization (icoef5=2).
  An alternative is to use the simple methods to solve the Navier-Stokes equations iteratively. Until now the standard iterative approach has proven to converge better (in case it converges) than SIMPLE.

7.1.10 Time-dependent solution of the Navier-Stokes equations

The discretized Navier-Stokes equations read:

$$\mathbf{M}\frac{\partial \mathbf{u}}{\partial t} + \mathbf{S}(\mathbf{u})\mathbf{u} + \mathbf{N}(\mathbf{u})\mathbf{u} - \mathbf{L}^T\mathbf{p} = \mathbf{F} \qquad \text{(momentum equations).} \tag{7.1.1}$$

$$\mathbf{Lu} = \mathbf{0} \qquad \text{(continuity equation).} \tag{7.1.2}$$

If you want to solve the time-dependent Navier-Stokes equations, you can use the standard program SEPCOMP. Although not necessary, it is recommended to use the input block STRUCTURE (Users Manual 3.2.3) in combination with the keyword Navier_Stokes. This refers to the input block NAVIER_STOKES described in the Users Manual Section 3.2.22.
With this block you can choose between the methods described in this section.

There are several ways to solve the time-dependent equations:

Straight-forward solution of the time-dependent Navier-Stokes equations

The most simple way to solve the time dependent equations is to use program SEPCOMP in combination with the input block `time_integration`.
In this case the same methods as for the stationary Navier-Stokes equations can be applied. See for example the manual EXAMPLES, sections 7.1.5, 7.1.12.3 and 7.1.12.4.
The alternative is to use the STRUCTURE BLOCK, with a time loop and in the time loop a call to NAVIER_STOKES. This refers to the keyword NAVIER_STOKES in which you activate the subkeyword `method = standard`
In case of Navier-Stokes one must start with a given initial velocity field. The the most stable linearization of the convective terms is either using Picard (icoef5=1) or Newton (icoef5=2). If the time integration is only first order accurate, like for example in case of implicit Euler, Picard is accurate enough. Besides that the matrix is more suited for iterative linear solvers. Only in case of second accurate time-dependent schemes, like for example Crank-Nicolson, Newton is recommended. Although all methods for the stationary equation can also be applied for the instationary equations there are some things one has to take into account.

- In case of the penalty function method it is not possible to use an explicit time integration, since this method will never be stable.
  In all other cases an implicit method is highly recommended.
- If elements of type 900 are used (i.e. Crouzeix-Raviart elements in combination with the penalty function method) the velocity is always computed correctly. However, if the centroid of an element is eliminated, like for the quadratic triangle with 6 points and the 9-point bi-quadratic quadrilateral, the pressure will not be computed correctly.
  In these two cases one should use elements of type 902 (integrated method), if the pressure is important. The reason is that the mass matrix is not taken into account in the static condensation.
  This problem only appears for 2D elements.
- In case of the mini-element (Linear Taylor-Hood) also the pressure is incorrect for the same reason.

Splitting method

Another option is to use a **splitting** method, where the non-linear terms are treated explicitly and all other terms implicitly.
In order to avoid stability problems, the explicit part is solved with a smaller time-step than the implicit part.

The method works as follows.
Consider the discretized momentum equations (7.1.1) and continuity equation (7.1.2).
Let $\mathbf{u}^n$ be the velocity at the previous time level.
Let `mstep` be the number of substeps for the convection integration.
Define $\Delta t^* = \frac{\Delta t}{\text{mstep}}$.
Then the operator splitting can be described in the following scheme.

First we start with the explicit convection update:
For `istep` := 1 (1) `mstep` do

$$\mathbf{M}\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t^*} + \mathbf{N}(\mathbf{u}^n)\mathbf{u}^n = 0 \tag{7.1.3}$$

$$\mathbf{u}^n := \mathbf{u}^* \tag{7.1.4}$$

In the next step we apply Stokes using the newly computed $\mathbf{u}^n$

$$\mathbf{M}\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \mathbf{L}^T\mathbf{p}^{n+1} = \mathbf{f}^{n+1} \tag{7.1.5}$$

$$\mathbf{L}\mathbf{u}^{n+1} = 0 \tag{7.1.6}$$

to get the new velocity and pressure.

If we apply the splitting method we need two sets of coefficients, one for solving the Stokes equations and for solving the convective part.
The set of coefficients to solve the Stokes equations are identical to the ones described in Section (7.1.12). Since there is no convection the parameter `MCONTV` (`icoef`5) must have the value 0.
The set of coefficients for the convective part is identical to that of the Stokes part, except that we have to give `MCONTV` the value 2. This corresponds to Newton linearization and is necessary to compute the convection at the previous time level.
Parameters with respect to viscosity, right-hand side and continuity are not used in the convection part and may therefore be skipped.

Pressure Correction

From Finite Difference Methods and Finite Volume Methods it is known, that in case of time-dependent problems the pressure correction method is very attractive. The reason is that the computation of pressure and velocity is automatically segregated, without the need to perform extra iterations to satisfy the continuity equation.
In fact there are two possible ways to discretize the Navier-Stokes equations by a pressure correction approach. The first one is the *continuous* approach, in which the pressure correction method is applied to the continuous Navier-Stokes equations. These equations are discretized by the FEM and solved. The alternative is to apply pressure correction immediately to the discretized equations *discrete approach*. Although the latter method has the advantage of not having to prescribe boundary conditions for the pressure, this method has not been implemented yet. The reason is that it requires the inverse mass matrix, which makes the implementation quite complicated. So at the moment we restrict ourselves to the continuous approach.
The method works as follows:
Consider the time dependent Navier-Stokes equations (7.1.3) and continuity equation (7.1.1). For simplicity we skip some terms and write the equations in the more simple form:

$$\rho[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}] + \nabla P - \text{div } \mathbf{t} = \rho\mathbf{f} \tag{7.1.7}$$

$$\text{div } \mathbf{v} = 0 \tag{7.1.8}$$

We explain the method for the case of Euler implicit, although it can be applied for all types of time integration. If we apply Euler implicit to the equations (7.1.7) and (7.1.8) we would get:

$$\rho[\frac{\mathbf{v^{n+1}} - \mathbf{v^n}}{\Delta t} + (\mathbf{v^{n+1}}\cdot\nabla)\mathbf{v^{n+1}}] + \nabla P^{n+1} - \text{div } \mathbf{t^{n+1}} = \rho\mathbf{f^{n+1}} \tag{7.1.9}$$

$$\text{div } \mathbf{v}^{n+1} = 0 \tag{7.1.10}$$

with $\mathbf{v}^{n+1}$ the velocity at time level $n+1$ and $\mathbf{v}^n$ at time level n. Δt denotes the time step. In general the convective terms are linearized with one of the methods treated in Section (7.1.3). Now the trick is that we start by computing a predictor $\mathbf{v}^*$, by replacing $\mathbf{v}^{n+1}$ in Equation (7.1.9) by $\mathbf{v}^*$ and P^{n+1} by P^n. Hence the pressure at the previous time level is used. In combination with Picard linearization this gives:

$$\rho[\frac{\mathbf{v^*} - \mathbf{v^n}}{\Delta t} + (\mathbf{v^n}\cdot\nabla)\mathbf{v^*}] + \nabla P^n - \text{div } \mathbf{t^{n+1}} = \rho\mathbf{f^{n+1}} \tag{7.1.11}$$

To satisfy the incompressibility condition, Equation (7.1.11) is subtracted from Equation (7.1.9) and all higher order terms are neglected as described in van Kan (1993).
This results in:

$$\rho\frac{\mathbf{v^{n+1}} - \mathbf{v^*}}{\Delta t} = -\nabla(P^{n+1} - P^n) \tag{7.1.12}$$

Application of the divergence operator and using (7.1.10), results in:

$$-\Delta\delta p = -\rho\frac{\text{div}\mathbf{v}^*}{\Delta t} \tag{7.1.13}$$

with $\delta p = P^{n+1} - P^n$.
From Equation (7.1.13) we can compute δp provided we prescribe boundary conditions for the pressure. Mathematically speaking there is no need to give these boundary conditions for the original Navier-Stokes equations, but due to our approximations this is necessary. In the literature one prescribes the pressure at outflow, i.e. where v_n is free and sets $\frac{\partial \delta p}{\partial n} = 0$ for all boundaries where v_n is prescribed.
Once equation (7.1.13) is solved, P^{n+1} is also known. $\mathbf{v^{n+1}}$ can be computed from (7.1.12). This is called the correction step.
So the solution is split into the solution of the momentum equations to get a predictor and the solution of Laplacian equation to solve the pressure and by some algebraic manipulations

the final velocity.

Since this continuous method explicitly requires boundary conditions for the pressure, it is necessary to have the pressure in the boundaries of the elements. Therefore the continuous pressure correction method can only be applied in the case of Taylor-Hood elements.
Using the pressure correction method requires the solution of 2 problems, hence we need to define 2 problems, one for the velocity and one for the pressure. Also we need at least two solution vectors.

In the predictor step we are also allowed to perform operator splitting just as for the standard method. In that case we need three input blocks for the coefficients, one for the Stokes part, one for the convective part and one for the pressure equations.
Besides the time integration input block also an input block for pressure correction is needed. See the Users manual Section 3.2.22. This must be part of a time loop in the structure block.

Type numbers to be used for pressure correction are 905 for the velocity and 906 for the pressure.
The coefficients for type 905 are defined in exactly the same way as for type 903. The only difference is coefficient 6 which is an integer coefficient which must contain the sequence number of the pressure in the set of solution equations. Hence in general:

```
icoef6 = %pressure
```

Type number 906 does not require any coefficients.
Besides that, in case of natural boundary conditions for the pressure, element 907 can be used. This element has exactly the same definition and input as elements of type 801 for natural boundary conditions with respect to second order elliptic equations as described in Section (3.1). The only difference is that unknowns are only present in the vertices and not in the midside points.
For an example of the use of the pressure correction method, the reader is referred to the manual Examples Section 7.1.18.

Solving the time-dependent problem as part of the elements

The Navier-Stokes equations are discretized by the following modified θ-method:

$$\mathbf{M}\frac{\mathbf{u}^{n+\theta}-\mathbf{u}^{n}}{\Delta t} + \mathbf{S}'(\mathbf{u}^{n})\mathbf{u}^{n+\theta} + \mathbf{N}'(\mathbf{u}^{n})\mathbf{u}^{n+\theta} - \mathbf{L}^{T}\mathbf{p}^{n+\theta} = \mathbf{F}'(\mathbf{u}^{n+\theta}) \tag{7.1.14}$$

$$\mathbf{L}\mathbf{u}^{n+\theta} = \mathbf{0} \tag{7.1.15}$$

$$\mathbf{u}^{n+1} = \frac{1}{\theta}\mathbf{u}^{n+\theta} - \frac{1-\theta}{\theta}\mathbf{u}^{n} \qquad 0 \le \theta \le 1$$

$$t^{n+\theta} = t^{n} + \theta\Delta t$$

where $\mathbf{S}'(\mathbf{u}^n)$, $\mathbf{N}'(u^n)$ and $\mathbf{F}'(u^n)$ are the result of the Newton or Picard linearization. See non-linearity.
The θ-method is conditionally stable for $0 \le \theta \le 0.5$. When the penalty function method is used, a time-step of $O(\epsilon)$ must be used in order to get a stable scheme for $\theta < 0.5$. One is therefore recommended to restrict the values of θ to the interval $0.5 \le \theta \le 1$. For $\theta = 0.5$ the θ-scheme reduces to a modified Crank-Nicolson scheme, which is the most accurate one for the class of θ-methods. However, a Crank-Nicolson scheme is very sensitive for transients

especially in the computation of the pressure with the penalty function method. This problem can be solved by starting with θ=1, and after one or two steps change this value to θ=0.5. The damping properties of this scheme for θ=1 are excellent, even too much information may be damped.
A better approach is to use the fractional step method or generalized theta method as described in the Users Manual Section 3.2.15.
If one wants to construct only one complete time-dependent matrix, instead of a separate Stiffness and Mass matrix, it is necessary to set the parameter `ITIME` (icoef1) equal to 1 or 2. In that case it is necessary to define θ and Δt. This may be done by common block CTIMEN as defined in subroutine SOLTIM.

```
double precision t, tout, dt, tend, t0, theta, rtime
integer itime
common /ctimen/ t, tout, dt, tend, t0, theta, rtime(4), itime(10)
```

The parameter dt is the time step Δt and theta is the parameter θ in the θ method. All other parameters are not used by the element subroutines.
If one uses the time integration as described for program SEPCOMP, these parameters are automatically set.
Remark: this last method is not recommended.

7.1.11 Representation of the Navier-Stokes equations for various coordinate systems

3-dimensional Cartesian co-ordinates (x_1, x_2, x_3)

Continuity equation

$$\text{div } \mathbf{v} \;=\; \sum_{j=1}^{3} \frac{\partial v_j}{\partial x_j} \tag{7.1.1}$$

Momentum equations

$$\rho[\frac{\partial v_i}{\partial t} \;+\; \sum_{j=1}^{3}(v_j \frac{\partial v_i}{\partial x_j}) \;+\; 2\omega C_i] \;+\; \frac{\partial P}{\partial x_i} \;-\; \sum_{j=1}^{3} \frac{\partial t_{ji}}{\partial x_j} \;=\; \rho f_i \qquad\qquad i = 1,2,3 \tag{7.1.2}$$

with $C_1 = -v_1$, $C_2 = v_2$, $C_3 = 0$ the Coriolis acceleration, and

$$t_{ji} \;=\; \eta(\frac{\partial v_j}{\partial x_i} \;+\; \frac{\partial v_i}{\partial x_j})$$

2-dimensional Cartesian co-ordinates (x_1, x_2)

See 3-dimensional case, however, with $v_3 = 0$, $\frac{\partial}{\partial x_3} = 0$.
Stream function ψ: $v_1 \;=\; \frac{\partial \psi}{\partial x_2}$, $v_2 \;=\; -\frac{\partial \psi}{\partial x_1}$

3-dimensional cylindrical co-ordinates (r, ϕ, z)

Continuity equation

$$\text{div } \mathbf{v} \;=\; \frac{\partial r v_r}{r \partial r} \;+\; \frac{\partial v_\phi}{r \partial \phi} \;+\; \frac{\partial v_z}{\partial z} \tag{7.1.3}$$

Momentum equations

$$\rho[\frac{\partial \mathbf{v}}{\partial t} \;+\; (\mathbf{v} \cdot \nabla)\mathbf{v} \;+\; 2\boldsymbol{\Omega} \times \mathbf{v}] \;+\; \nabla P \;-\; \text{div } \mathbf{t} \;=\; \rho \mathbf{f} \tag{7.1.4}$$

with

$\mathbf{v} \;=\; (v_1 \;,\; v_2 \;,\; v_3) \;=\; (v_r \;,\; v_\phi \;,\; v_z),$

$[(\mathbf{v} \;\cdot\; \nabla)\mathbf{v}]_i \;=\; v_r \frac{\partial v_i}{\partial r} \;+\; v_\phi \frac{\partial v_i}{r\partial \phi} \;+\; v_z \frac{\partial v_i}{\partial z} \;-\; \frac{v_\phi^2}{r}\delta_{1i} \;+\; \frac{v_r v_\phi}{r}\delta_{2i}, \qquad i = 1,2,3$

$\mathbf{\Omega} \times \mathbf{v} \;=\; \omega(-v_\phi \;,\; v_r \;,\; 0)$

$\nabla P \;=\; (\frac{\partial P}{\partial r} \;,\; \frac{\partial P}{r\partial \phi} \;,\; \frac{\partial P}{\partial z})$

$[\text{div } \mathbf{t}]_i \;=\; \frac{\partial r t_{ri}}{r\partial r} \;+\; \frac{\partial t_{\phi i}}{r\partial \phi} \;+\; \frac{\partial t_{zi}}{\partial z} \;-\; \frac{t_{\phi\phi}}{r}\delta_{1i} \;+\; \frac{t_{\phi r}}{r}\delta_{2i}, \qquad i = 1,2,3$

$\mathbf{t} \;=\; \eta(\nabla \mathbf{v} \;+\; \nabla \mathbf{v}^T)$

$$\nabla \mathbf{v} \;=\; \begin{pmatrix} \frac{\partial v_r}{\partial r} & \frac{\partial v_\phi}{\partial r} & \frac{\partial v_z}{\partial r} \\ \frac{\partial v_r}{r\partial \phi} - \frac{v_\phi}{r} & \frac{\partial v_\phi}{r\partial \phi} + \frac{v_r}{r} & \frac{\partial v_z}{r\partial \phi} \\ \frac{\partial v_r}{\partial z} & \frac{\partial v_\phi}{\partial z} & \frac{\partial v_z}{\partial z} \end{pmatrix}$$

Axisymmetric co-ordinates with swirl. (x_1, x_2)

See cylindrical co-ordinates, however, with $\frac{\partial}{\partial \phi} = 0$.
Only two co-ordinates (r, z) remain. For the mesh generator the problem is two-dimensional in (r, z).
In case of constant viscosity and incompressible velocity, the Stokes equations reduce to

$$\eta(\Delta v_r - \frac{v_r^2}{r}) \;=\; f_r \tag{7.1.5}$$

$$\eta \Delta v_z \;=\; f_z \tag{7.1.6}$$

with $\Delta u = \frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + \frac{1}{r^2}\frac{\partial u}{\partial r}$.
Stream function ψ: $v_r \;=\; \frac{\partial \psi}{2\pi r \partial z} \;,\; v_z \;=\; -\frac{\partial \psi}{2\pi r \partial r}$
The factor 2π ensures that ψ measures the volume flux.

Axisymmetric co-ordinates without swirl. (x_1, x_2)

See cylindrical co-ordinates, however, with $v_\phi \;=\; 0$ and $\frac{\partial}{\partial \phi} = 0$.
The second momentum balance equation disappears; ω must be equal to 0. Only two co-ordinates (r, z) remain. For the mesh generator the problem is two-dimensional in (r, z).
Stream function ψ: $v_r \;=\; \frac{\partial \psi}{2\pi r \partial z} \;,\; v_z \;=\; -\frac{\partial \psi}{2\pi r \partial r}$
The factor 2π ensures that ψ measures the volume flux.

2-dimensional polar co-ordinates. (x_1, ϕ)

See cylindrical co-ordinates, however, with $v_z \;=\; 0$ and $\frac{\partial}{\partial z} = 0$.
The third momentum balance equation disappears.
Only two co-ordinates (r, ϕ) remain. For the mesh generator the problem is two-dimensional in (r, ϕ).
Stream function ψ: $v_r \;=\; \frac{\partial \psi}{r\partial \phi} \;,\; v_\phi \;=\; -\frac{\partial \psi}{\partial r}$

7.1.12 Old way to describe the coefficients for the differential equation

The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
For each element group 20 parameters and coefficients must be given. The first 5 parameters are

of integer type which means that they must be defined by ICOEFi in the input, the last 15 are real coefficients.

These parameters and coefficients are defined as follows:

1 Parameter *itime* with respect to the time integration.

2 Type of viscosity model *modelv*.

3 Type of numerical integration.

4 Parameter ICOOR, defining the type of co-ordinate system.

5 Parameter MCONTV, defining how to treat the convection and the (in)compressibility term.

6 Parameter ε for the penalty function method.
If this parameter is given in a Taylor-Hood element it is neglected.
If this parameter is given in Crouzeix-Raviart element in combination with the integrated method it is used as a kind of artificial compressibility parameter. It might improve the performance in case an iterative solver is used for the system of linear equations.
Mark that this parameter is in fact the inverse of a penalty parameter and hence must have a small value.

7 Density ρ

8 Angular velocity ω

9 First component of body force $\mathbf{f}_1$

10 Second component of body force $\mathbf{f}_2$

11 Third component of body force $\mathbf{f}_3$

12 Viscosity parameter η

13 Viscosity model parameter n

14 Viscosity model parameter λ

15 Special integer parameter (to be defined by icoef15) that defines the sequence number of the mesh velocity vector in the vector of old solutions.
The mesh velocity is subtracted from the real velocity when the convective terms are computed, even if there are no convective terms given. This parameter is especially meant for free surface flows, where the convective terms must be adapted due to the movement of the mesh. The mesh velocity must have at least, `ndim`,degrees of freedom per point. The first `ndim` are considered to be the velocity.
The pressure computation has not yet been adapted.

16 G_L, parameter for the Görtler equations. These equations will be available in the future.

17 κ, parameter for the Görtler equations.

18 f_{div}, right-hand side for the continuity equation.

19 Given stress tensor used in the right-hand side vector.
At this moment it is supposed that the stress is defined per element and that in each element all components are constant. This stress tensor can only be defined as an old solution.

20 $cdiv_1$ for special scaled form of continuity equation

21 $cdiv_2$ for special scaled form of continuity equation

22 not yet in use, must be zero

23 *cgrad* for special scaled form of momentum equation

24-27 $cconv_i$ for special scaled form of convection term in momentum equation

28-32 not yet in use, must be zero

33-36 $cvisc_i$ for special scaled form of viscous term in momentum equation

37-41 not yet in use, must be zero

The parameters that are not used must be set equal to zero, since this is the default value in case of future extensions.

With respect to the parameters 1-5 the following choices are available:

1 Parameter *itime* with respect to the time integration.
This parameter consists of 2 parts: *itime_old* and *iseqprev* according to $itime = itime\_old$ $10 \times iseqprev$.

The parameter *itime_old* must contain some information about the time integration. Possible values are:

0 If the parameter imas in subroutine BUILD is equal to 0, a time-independent flow is assumed and the mass matrix is not built.
If imas>0, a mass matrix is built according to imas, i.e.
imas=1: diagonal mass matrix
imas=2: consistent mass matrix.
The parameter imas is set by the block `time_integration` in the input file. It corresponds to the input corresponding to the mass matrix. Usually it is sufficient to set *itime_old* equal to 0.

1 The θ method is applied directly in the following way:
the mass matrix divided by $\theta\Delta t$ is added to the stiffness matrix.
The mass matrix divided by $\theta\Delta t$ and multiplied by u^n is added to the right-hand-side vector.

Default value: 0.

The parameter *iseqprev* refers to the velocity vector in the previous iteration. This parameter makes only sense if during a time step the velocity is computed iteratively. This means that the old velocity in the convective terms for example is taken equal to the velocity at the previous iteration. The velocity used in the discretization of the time derivative, however, is the old velocity of the previous time level. If no iteration per time step is used both are equal and iseqprev may be chosen equal to 0.
iseqprev defines the sequence number of the iteration velocity vector in the general array of solution vectors.

2 Type of constitutive equation (MODELV)
Possible values:

1 Newtonian liquid. In this case η is the dynamic viscosity.

2 Power-law liquid. In this case η is the parameter η_n and n the parameter n in the power of the model.

3 Carreau liquid. In this case η is the parameter η_c and n the parameter n in the power of the model. λ is the parameter λ the viscosity model.

4 Plastico-viscous liquid. In this case η is the parameter η_{pv} and n the parameter n in the power of the model. λ is the parameter s the viscosity model.

5 Use the viscous term as defined in Equations (7.1.7) and (7.1.8). In this case the number parameters must be at least 41.

10 Newtonian liquid. In this case η is the dynamic viscosity. The continuity equation is substituted in the stress tensor in order to get a more simple expression, in which velocity components are decoupled. This option is only correct in case of constant viscosity and incompressible flow.
In this case the term div $\mathbf{t}$ is replaced by $\eta\Delta\mathbf{v}$, where Δ represents the Laplace operator. The advantage of this approach is, that the reduced storage scheme as described in Section 3.2.4 of the Users Manual, can be used. This means that one can use the keyword `DECOUPLED_DEGFD = (1, 2)` inR^2 or `DECOUPLED_DEGFD = (1, 2, 3)` inR^3 in the input block `MATRIX`.

100 user provided model, depending on $\nabla\mathbf{v}$.

101 user provided model, depending on II.

102 user provided model, depending on $\mathbf{x}, \mathbf{v}$ and II.

102 user provided model, depending on $\mathbf{x}, \mathbf{v}$, II and also on extra vectors stored in array uold.

Default value: 1.

3 Type of numerical integration.
The parameter icoef3 consists of two parts irule and interpol according to: icoef3 = irule + 100 × interpol, with
irule

0: the rule is chosen by the element itself (Default)

> 0: the integration rule is defined by the user. The types of integration rules available depend on the type of elements applied.

interpol

0: Coefficients are computed by using the interpolation applied for the basis functions.

1: Coefficients that are defined by an old solution vector, are computed by linear interpolation. If the element is a quadrilateral or a hexahedron, or if the element is quadratic, subelements are used to define a local linear interpolation.
This option is useful if the coefficients are rapidly changing and standard interpolation may produce an overshoot.

4 Type of co-ordinate system.
This parameter consists of two parts ICOOR_ORIG and IMETHOD according to ICOOR = ICOOR_ORIG + 100 × IMETHOD. Possible values for ICOOR_ORIG:

0 Cartesian co-ordinates

1 Axi-symmetric co-ordinates

2 Polar co-ordinates

Possible values for IMETHOD:

0 Standard method

1 The system of equations is solved by an AUGMENTED LAGRANGE approach. The diagonal of the pressure mass matrix is also computed.

Default value: 0.

5 Parameter MCONTV, defining how to treat the convection and also the continuity equation, according to MCONTV=MCONV+10×MCONT+100×MCOEFCONV.
Possible values for MCONV:

- 0 Stokes flow, the convective terms $(\mathbf{v} \cdot \mathbf{v})$ are neglected.
- 1 Linearization by Picard's method.
- 2 Linearization by Newton's method.
- 3 Linearization by the "incorrect" Picard method.

Possible values for MCONT:

- 0 Standard method: the continuity equation is div $\mathbf{v} = 0$
- 1 A special continuity equation div $\rho\mathbf{v} = 0$, with ρ variable is solved. This is not the standard continuity equation for incompressible flow. In fact we have a time-independent compressible flow description.
- 2 The Görtler equations are solved.
- 3 The continuity equation is treated as in Equation (7.1.6). In this case the number parameters must be at least 41.
  Also the pressure term is treated as in Equations (7.1.7) and (7.1.8).

Possible values for MCOEFCONV:

- 0 the convection term is treated in the standard way.
- 1 the convection term is treated as defined in Equations (7.1.7) and (7.1.8). In this case the number parameters must be at least 41.

If MODELV $\geq$ 100 the user must provide a user written function subroutine, which computes η as function of the parameters. The following choices for user subroutines are available:

MODELV = 100: function subroutine FNV000 is used.

MODELV = 101: function subroutine FNV001 is used.

MODELV = 102: function subroutine FNV002 is used.

MODELV = 103: function subroutine FNV003 is used.

These function subroutines have the following shape:

```
FUNCTION FNV000 ( GRADV )
IMPLICIT NONE
DOUBLE PRECISION FNV000, GRADV(*)

    statements to compute the viscosity as function of GRADV
    FNV000 = computed viscosity

END



FUNCTION FNV001 ( SECINV )
IMPLICIT NONE
DOUBLE PRECISION FNV001, SECINV
```

```
      statements to compute the viscosity as function of SECINV
      FNV001 = computed viscosity

  END



  FUNCTION FNV002 ( X1, X2, X3, V1, V2, V3, SECINV )
  IMPLICIT NONE
  DOUBLE PRECISION FNV002, X1, X2, X3, V1, V2, V3, SECINV

      statements to compute the viscosity as function of SECINV, X and V
      FNV002 = computed viscosity

  END



  FUNCTION FNV003 ( X1, X2, X3, V1, V2, V3, SECINV, NUMOLD, MAXUNK,
 +                  UOLD )
  IMPLICIT NONE
  INTEGER NUMOLD, MAXUNK
  DOUBLE PRECISION FNV002, X1, X2, X3, V1, V2, V3, SECINV,
 +                 UOLD(NUMOLD,MAXUNK)

      statements to compute the viscosity as function of the parameters
      FNV003 = computed viscosity

  END
```

In these subroutines array GRADV contains the gradient of the velocity in one point, SECINV the second inverse, X1,X2,X3 the co-ordinates of the point and V1,V2,V3 the velocity components.
NUMOLD and MAXUNK define dimension parameters for array UOLD.
NUMOLD indicates the number of old solutions stored in UOLD.
MAXUNK indicates the maximum number of degrees of freedom stored in the old solution vectors stored in UOLD.
UOLD defines the array of old solutions. It is related to the array ISLOLD in subroutine BUILD, or defines the old vectors already computed in program SEPCOMP.
UOLD(1,1) contains the first unknown of the first vector in the node, UOLD(1,2) the second unknown and so on.

The storage of the gradient depends on the type of co-ordinate system. The following possibilities are available:

two-dimensional Cartesian co-ordinates

$$\frac{\partial v_1}{\partial x_1},\ \frac{\partial v_2}{\partial x_1},\ \frac{\partial v_1}{\partial x_2},\ \frac{\partial v_2}{\partial x_2} \tag{7.1.1}$$

three-dimensional Cartesian co-ordinates

$$\frac{\partial v_1}{\partial x_1},\ \frac{\partial v_2}{\partial x_1},\ \frac{\partial v_3}{\partial x_1},\ \frac{\partial v_1}{\partial x_2},\ \frac{\partial v_2}{\partial x_2},\ \frac{\partial v_3}{\partial x_2},\ \frac{\partial v_1}{\partial x_3},\ \frac{\partial v_2}{\partial x_3},\ \frac{\partial v_3}{\partial x_3} \tag{7.1.2}$$

three-dimensional cylindrical co-ordinates

$$\frac{\partial v_r}{\partial r},\ \frac{\partial v_\phi}{\partial r},\ \frac{\partial v_z}{\partial r},\ \frac{\partial v_r}{r\partial\phi}-\frac{v_\phi}{r},\ \frac{\partial v_\phi}{r\partial\phi}+\frac{v_r}{r},\frac{\partial v_r}{\partial z},\ \frac{\partial v_\phi}{\partial z},\ \frac{\partial v_z}{\partial z} \tag{7.1.3}$$

two-dimensional axi-symmetric co-ordinates without swirl (r,z)

$$\frac{\partial v_r}{\partial r},\ \frac{\partial v_z}{\partial r},\ \frac{\partial v_r}{\partial z},\ \frac{\partial v_z}{\partial z},\ \frac{v_r}{r} \tag{7.1.4}$$

two-dimensional axi-symmetric co-ordinates with swirl (r,z)

$$\frac{\partial v_r}{\partial r},\ \frac{\partial v_\phi}{\partial r},\ \frac{\partial v_z}{\partial r},-\frac{v_\phi}{r},\ \frac{v_r}{r},0,\ \frac{\partial v_r}{\partial z},\ \frac{\partial v_\phi}{\partial z},\ \frac{\partial v_z}{\partial z}, \tag{7.1.5}$$

two-dimensional polar co-ordinates (r,ϕ)

$$\frac{\partial v_r}{\partial r},\ \frac{\partial v_\phi}{\partial r},\ \frac{\partial v_r}{r\partial\phi}-\frac{v_\phi}{r},\ \frac{\partial v_\phi}{r\partial\phi}+\frac{v_r}{r} \tag{7.1.6}$$

7.1.13 Old type numbers available for the Navier-Stokes equations

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the Navier-Stokes equation in this section the following type numbers are available:

900 General type number for the internal Navier-Stokes elements based on Crouzeix-Raviart elements, hence with discontinuous pressure.
This number is restricted to the penalty function formulation only.
For this type of element, swirl is not allowed. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 4 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. Internally the velocity is treated as a quadratic polynomial with extra third order term. The velocity in the centroid is internally used, but not available to the user. The internal pressure is defined in the centroid, but also not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains two velocity components in each node.
The computation of the pressure is only accurate for stationary flows. For instationary flows one should use shape 7.

shape = 5 Linear quadrilateral. This is a so-called Crouzeix-Raviart element that does not satisfy the Brezzi-Babuska condition. The velocity is defined in the vertices.
Since the Brezzi-Babuska condition is not satisfied, the pressure may contain unrealistic wiggles. However, if at some part of the boundary the normal stress is given (or equivalently the normal component of the velocity is not prescribed), there is good chance that the wiggles are not visible. The internal pressure is defined in the centroid, but not available to the user.
The solution vector contains two velocity components in each node.

shape = 6 Bi-quadratic quadrilateral. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid, but not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains two velocity components in each node.

shape = 7 extended quadratic triangle. This is exactly the same as shape 4, but the centroid is not eliminated. In case of a pressure computation in an instationary flow, this shape should be used instead of shape 4.

shape = 14 Tri-quadratic hexahedron. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid, but not available to the user. Only the averaged pressure produced by "DERIV" is available.
The solution vector contains three velocity components in each node.

902 General type number for the internal Navier-Stokes elements, using Crouzeix-Raviart type elements with discontinuous pressure.
This number is restricted to the integrated solution method only.
For this type of element, swirl is not allowed.
Elements of this type may be used in combination with direct methods and iterative methods.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 6 Bi-quadratic quadrilateral. This is a so-called Crouzeix-Raviart element. The internal pressure and its gradient are defined in the centroid and available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two velocity components in each node.
Furthermore in the centroid (Point 9), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: p 4: $\frac{\partial p}{\partial x}$ 5: $\frac{\partial p}{\partial y}$

shape = 7 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. The velocity is treated as a quadratic polynomial with extra third order term. The velocity in the centroid is available to the user. The internal pressure is defined in the centroid and available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two velocity components in each node.
Furthermore in the centroid (Point 7), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: p 4: $\frac{\partial p}{\partial x}$ 5: $\frac{\partial p}{\partial y}$

shape = 9 Linear quadrilateral. This is a so-called Crouzeix-Raviart element that does not satisfy the Brezzi-Babuska condition. The velocity is defined in the vertices.
Since the Brezzi-Babuska condition is not satisfied, the pressure may contain unrealistic wiggles. However, if at some part of the boundary the normal stress is given (or equivalently the normal component of the velocity is not prescribed), there is good chance that the wiggles are not visible. The internal pressure is defined in the centroid, but not available to the user.
The solution vector contains two velocity components in each vertex. In the centroid only the pressure is defined.

shape = 14 Tri-quadratic hexahedron. This is a so-called Crouzeix-Raviart element. The internal pressure is defined in the centroid and available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains three velocity components in each node.
Furthermore in the centroid (Point 14), the pressure and the gradient of the pressure are available. In this point we have:
1: v_1 2: v_2 3: v_3 4: p 4: $\frac{\partial p}{\partial x}$ 6: $\frac{\partial p}{\partial y}$ 7: $\frac{\partial p}{\partial z}$

901 General type number for the internal Navier-Stokes elements, based on Crouzeix-Raviart elements with discontinuous pressure.
This number is restricted to the integrated solution method only.
For this type of element, swirl is not allowed.
The difference with type number 902 is that the gradient of the pressure and the velocity in the centroid are eliminated internally. As a consequence the number of unknowns is reduced considerably.

Elements with this type number may be used in combination with direct and iterative linear solvers. Experiments have shown that sometimes the iterative solvers do not converge. In that case it is advised to use type number 902 instead.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 7 extended quadratic triangle. This is a so-called Crouzeix-Raviart element. The velocity is treated as a quadratic polynomial with extra third order term. The velocity in the centroid is available to the user. This pressure is discontinuous over the element boundaries. The averaged pressure produced by "DERIV" is available.
The solution vector contains two velocity components in each node except the centroid. In the centroid (Point 7), the pressure is available, but not the pressure gradient.
In this point there is only one unknown (p), but this pressure is the physical degree of freedom with sequence number 3.

903 General type number for the internal Navier-Stokes elements, based on Taylor-Hood elements with continuous pressure.
This number is restricted to the integrated solution method only.
For this type of element, swirl is not allowed.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 3 linear triangle. This so-called mini-element is exactly the same as the mini-element with shape=10. However, in this case the velocities in the centroid are eliminated at element level, thus reducing the number of unknowns. Both the velocity and the pressure are continuous.
The solution vector contains two velocity components and one pressure unknown in each node, in the sequence v_1, v_2, p.

shape = 4 Quadratic triangle.
In fact this is the classical Taylor-Hood element. The velocity is defined as a quadratic field, the pressure as a linear field per element.
The solution vector contains two velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, p.

shape = 5 Bi-linear quadrilateral. This element is exactly the same as the element with shape=9. However, in this case the velocities in the centroid are eliminated at element level, thus reducing the number of unknowns. Both the velocity and the pressure are continuous.
The solution vector contains two velocity components and one pressure unknown in each node, in the sequence v_1, v_2, p.

This element has not yet been implemented.

shape = 6 Bi-quadratic quadrilateral.
This is the natural extension of the classical Taylor-Hood element to quadrilaterals. The velocity is defined as a bi-quadratic field, the pressure as a bi-linear field per element.
The solution vector contains two velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, p.

shape = 9 Bi-linear quadrilateral with bubble function.
This is the natural extension of the triangular mini-element to quadrilaterals. The velocity is defined as a bi-linear field, extended with a bubble function which is zero at the boundaries of the element and 1 in the centroid. This bubble function is necessary in order to satisfy the so-called B-B condition and may be considered as a stabilization parameter. The pressure is bi-linear per element.
The solution vector contains two velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, p.

This element has not yet been implemented.

shape = 10 Linear triangle with bubble function.
This is the classical mini element. The velocity is defined as a linear field, extended with a bubble function which is zero at the boundaries of the element and 1 in the centroid. This bubble function is necessary in order to satisfy the so-called B-B condition and may be considered as a stabilization parameter. The pressure is linear per element.
The solution vector contains two velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, p.

shape = 11 linear tetrahedron. This so-called mini-element is exactly the same as the mini-element with shape=18. However, in this case the velocities in the centroid are eliminated at element level, thus reducing the number of unknowns. Both the velocity and the pressure are continuous.
The solution vector contains three velocity components and one pressure unknown in each node, in the sequence v_1, v_2, v_3, p.

shape = 12 Quadratic tetrahedron.
In fact this is 3D-extension of the classical Taylor-Hood element. The velocity is defined as a quadratic field, the pressure as a linear field per element.
The solution vector contains three velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, v_3, p.

shape = 13 Tri-linear hexahedron. This element is exactly the same as the element with shape=17. However, in this case the velocities in the centroid are eliminated at element level, thus reducing the number of unknowns. Both the velocity and the pressure are continuous.
The solution vector contains three velocity components and one pressure unknown in each node, in the sequence v_1, v_2, v_3, p.

shape = 14 Tri-quadratic hexahedron.
This is the natural extension of the classical Taylor-Hood element to hexahedrons. The velocity is defined as a tri-quadratic field, the pressure as a tri-linear field per element.
The solution vector contains three velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, v_3, p.

shape = 17 Tri-linear hexahedron with bubble function.
This is the natural extension of the triangular mini-element to hexahedrons. The velocity is defined as a tri-linear field, extended with a bubble function which is zero at the boundaries of the element and 1 in the centroid. This bubble function is necessary in order to satisfy the so-called B-B condition and may be considered as a stabilization parameter. The pressure is tri-linear per element.
The solution vector contains three velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, v_3, p.

shape = 18 Linear tetrahedron with bubble function.
This is the extension of the classical mini element to 3D. The velocity is defined as a linear field, extended with a bubble function which is zero at the boundaries of the element and 1 in the centroid. This bubble function is necessary in order to satisfy the so-called B-B condition and may be considered as a stabilization parameter. The pressure is linear per element.
The solution vector contains three velocity components in all nodes. In the vertices there are three unknowns: v_1, v_2, v_3, p.

905 Element that can only be used for pressure correction. It is based on Taylor-Hood elements. This element is used to solve the momentum equations with out the pressure. The pressure at a previous level is used. Only quadratic elements for the velocity are allowed, the corresponding pressure approximation is linear.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 4 quadratic triangle.

shape = 6 bi-quadratic quadrilateral.

shape = 12 quadratic tetrahedron.

shape = 14 tri-quadratic hexahedron.

906 Element that can only be used for pressure correction. It corresponds to type 905 and is used to solve the pressure equation. Although the pressure approximation is linear, exactly the same elements as for type 906 must be used. 905 and 906 always are used as a couple.

910 boundary conditions of type 2 to be applied for the Crouzeix-Raviart type elements.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 6 bi-quadratic surface element (quadrilateral).

911 boundary conditions of type 2 to be applied for the Taylor-Hood type elements.
This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 6 bi-quadratic surface element (quadrilateral).

912 boundary conditions of type 3 to be applied for the Crouzeix Raviart elements in combination with a direct linear solver. This element is used to prescribe the mass flux if the penalty function approach for the mass flux is used.
The only shape number allowed in -1, which means all nodes along a curve. Such elements must be created as line elements in SEPMESH.

913 boundary conditions of type 3 to be applied for the Crouzeix Raviart elements.
This element is used to prescribe the mass flux if the approach with global unknowns for the mass flux is used.
This method may be used both for curves as for surfaces.
The shape number of the elements is already defined in the problem definition.

914 boundary conditions of type 4 to be applied for the Crouzeix Raviart elements. This type number is available for the following element shape numbers: (see the Users Manual, Section 2.2, Table 2.2.1)

shape = 1 linear line element.

shape = 2 quadratic line element.

shape = 6 bi-quadratic surface element (quadrilateral).

Since connection elements must be used the actual elements consists of pairs of the elements indicated above and the actual shape numbers are equal to -10001-shape, where shape is the value given in the table.

7.1.14 Previous versions of SEPRAN dealing with the Navier-Stokes equations

In previous versions of SEPRAN equation (7.1.1) has been solved with different type numbers. These type numbers may still be used, however, it is recommended to use the new type numbers described earlier when creating new input or new programs. In this section we will point out the differences of the previous type numbers and the present one.

- In the old version only the penalty function formulation may be used. The integrated method is not available.
- Definition of the coefficients for the differential equation:

  For each element group 11 parameters and coefficients must be given.

  These parameters and coefficients are defined as follows:

  1 Parameter ϵ for the penalty method.

  2 Density ρ.

  3 Parameter MCONV, defining the type of linearization.

  4 Angular velocity ω

  5 First component of body force $\mathbf{f}_1$

  6 Second component of body force $\mathbf{f}_2$

  The next parameters depend on the type of co-ordinate system and the viscosity model used.

  7 Third component of body force $\mathbf{f}_3$
  This parameter is only used in the three-dimensional case or if an axi-symmetric model with swirl is used.
  If $\mathbf{f}_3$ is not used, the next parameters start at position 7, otherwise at position 8.

  7 or 8 Parameter MODELV, defining the type of viscosity model

  Depending on the value of MODELV this parameter must be followed by the parameters η, n and λ.

  The following values for the parameter MCONV are available:

  0 Stokes flow, the convective terms $(\mathbf{v} \cdot \mathbf{v})$ are neglected.

  1 Linearization by Picard's method.

  2 Linearization by Newton's method.

  10-12 See MCONV-10, in this case it is assumed that the instationary Navier-Stokes equations are solved and the linear combination of Mass matrix and Stiffness matrix is computed as indicated with itime=1.

  30 Only the mass matrix $\mathbf{M}$ is computed and stored.

- Definition of coefficients for the boundary conditions:

  For each element group 2 (R^2 or 3 R^3) coefficients must be given.

  These coefficients are defined as follows:

  1 σ_n

  2 First component of $\boldsymbol{\sigma}_t$

  3 Second component of $\boldsymbol{\sigma}_t$

- Computation of derivatives:

  Depending on the parameter ICHELD in the input block "DERIVATIVES" the following types of derivatives are computed:

  1 The pressure is computed in the vertices of the elements.

2 The derivative $\frac{\partial v_{JDEGFD}}{\partial x_{IX}}$, with JDEGFD and IX parameters in the input block "DERIVATIVES".

3 div $\mathbf{v}$ is computed in the vertices of the elements.

4 curl$\mathbf{v}$ is computed in the vertices of the elements.

5 $\mathbf{t}$ is computed in the vertices of the elements.

6 $\nabla\mathbf{v}$ is computed in the vertices of the elements.

7 The rate of elongation $\dot{\epsilon}$ is computed in the vertices of the elements.
$\dot{\epsilon}$ is the rate of strain in the direction of the velocity: $\dot{\epsilon} = \frac{1}{2}\mathbf{v} \cdot \mathbf{A}_1 \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$

8 The shear rate $\dot{\gamma}$ is computed in the vertices of the elements, according to the definition:
$\dot{\gamma} = \mathbf{n}_1 \cdot \mathbf{A}_1\mathbf{n}_2$, $\mathbf{n}_1 \cdot \mathbf{n}_2 = 0$, $\|\mathbf{n}_1\| = \|\mathbf{n}_2\| = 1$
For problems with two velocity components: $\|\mathbf{n}_1\| = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, $\|\mathbf{n}_2\| = \frac{(-v_2,v_1)}{\|\mathbf{v}\|}$
For problems with three velocity components:
$\dot{\gamma}_{12} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_2$, $\dot{\gamma}_{13} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$, $\dot{\gamma}_{23} = \mathbf{e}_2 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$

9 $II^{\frac{1}{2}}$ is computed in the vertices of the elements.

The output vector is defined as follows:

ICHELD=1,2,3,7,9 a vector of special structure (IVEC=1) with one unknown in each vertex.

ICHELD=4 In the two-dimensional case a vector of special structure (IVEC=1) with one unknown in each vertex.
In the three-dimensional case a vector of special structure (IVEC=5).
This is also the case for two-dimensional axi-symmetric flow with swirl.

ICHELD=5 a vector of special structure (IVEC=3).

ICHELD=6 a vector of special structure (IVEC=4).

ICHELD=8 In the two-dimensional case a vector of special structure (IVEC=1) with one unknown in each vertex.
In the three-dimensional case a vector of special structure (IVEC=5).

- Types of integrals that may be computed:

  If the user wants to compute integrals over the solution, he may use the option INTEGRAL in the input block "STRUCTURE"

  The parameter ICHELI in the input block "INTEGRALS" is used to distinguish the various possibilities:

  ICHELI=1 $\int\limits_{\Omega} f(x)d\Omega$

  The user must define the function $f(x)$ as first coefficient by one of the methods described in 2.2.

- Input with respect to the time-dependence.

  In case of a time-dependent equation it is necessary to define θ and Δt. This may be done by common block CTIMEN. Unfortunately in the old version of SEPRAN the common block CTIMEN used is not consistent with the common block CTIMEN used in SOLTIM. This means that the old version can not be combined with SOLTIM.
  In the old version CTIMEN is defined as follows:

```
double precision t, dt, theta, rtime
integer itime
common /ctimen/ theta, dt, t, rtime(7), itime(10)
```

The parameter dt is the time step Δt and theta is the parameter θ in the θ method. All other parameters are not used by the element subroutines.

Furthermore in the previous version of SEPRAN it is not possible to compute stiffness matrix and mass matrix explicitly in the time-dependent case (except if MCONV=10). Hence only $itime = 1$ is used in the time-dependent case. This case is recognized by the value of MCONV.

Definition of type numbers

The type numbers for the old version are defined as follows:

400 Quadratic isoparametric triangle in R^2, Cartesian co-ordinates.

401 Boundary conditions of type 2 corresponding to type 400.
The element must be a quadratic boundary element in R^2.

402 Quadratic isoparametric triangle in R^2, Polar co-ordinates.

403 Boundary conditions of type 2 corresponding to type 402.
The element must be a quadratic boundary element in R^2.

404 Quadratic isoparametric triangle in R^2, Axisymmetric co-ordinates without swirl.

405 Boundary conditions of type 2 corresponding to type 404.
The element must be a quadratic boundary element in R^2.

406 Quadratic isoparametric triangle in R^2, Axisymmetric co-ordinates with swirl.

407 Boundary conditions of type 2 corresponding to type 406.
The element must be a quadratic boundary element in R^2.

410 Quadratic isoparametric hexahedron in R^3, Cartesian co-ordinates.

7.2 The temperature dependent laminar flow of incompressible liquids (Boussinesq approximation)

In this section we consider the laminar flow of incompressible liquids described by the Navier-Stokes equations, where the temperature is a driving force. It is assumed that a Boussinesq approximation is applied. Furthermore the temperature satisfies a convection diffusion equation. These equations consist of four parts:

- The conservation of mass.
- The conservation of momentum.
- The constitutive equation.
- The temperature equation.

Equations

The conservation of mass
In case of incompressible flow:

$$\text{div } \mathbf{v} = 0 \tag{7.2.1}$$

Conservation of momentum (Euler-Cauchy equations):

$$\rho[\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} + 2\boldsymbol{\Omega}\times\mathbf{v}] + \nabla P - \text{div } \mathbf{t} - \rho\mathbf{g}\beta(T-T_0) = \rho\mathbf{f} \tag{7.2.2}$$

Temperature equation

$$\rho c_p(\frac{\partial T}{\partial t} + \mathbf{v}\cdot\nabla T) - \text{div}(\kappa\nabla T) = \rho f_T + \frac{\alpha}{2}\mathbf{t} : \mathbf{A}_1 \tag{7.2.3}$$

The parameters in the Equations 7.2.1, 7.2.2 and 7.2.3 have the following meaning:

- ρ density of the liquid.
- $\mathbf{v}$ velocity.
- $\boldsymbol{\Omega}$ angular velocity $\boldsymbol{\Omega} = (0, 0, \omega)$ of rotating co-ordinate system with respect to an inertial system $\mathbf{x}$.
- $\mathbf{f}$ external force field (body force).
- $\mathbf{t}$ deviatoric stress tensor.
- T Temperature.
- c_p Heat capacity at constant pressure.
- β volume expansion coefficient.
- $\mathbf{g}$ Acceleration of gravity in the z-direction: $\mathbf{g} = (0\ ,\ 9.81)$.
  If a problem with swirl is solved then $\mathbf{g} = (0\ ,0\ ,\ 9.81)$.
- T_0 Reference temperature.
- κ Thermal conductivity.
- f_T External field for the temperature equation (e.g. heat source).

The stress tensor $\boldsymbol{\sigma}$ consists of two parts: the pressure part and the deviatoric stress tensor according to:

$$\boldsymbol{\sigma} \;=\; -P\mathbf{1} \;+\; \mathbf{t}. \tag{7.2.4}$$

Remark: The Boussinesq approximation may only be applied for small values of $\beta(T - T_0)$.

The generalized or reduced pressure P can be written as

$$P \;=\; p \;+\; \frac{1}{2}\rho(\boldsymbol{\Omega} \times \mathbf{x}) \cdot (\boldsymbol{\Omega} \times \mathbf{x}), \tag{7.2.5}$$

where p denotes the hydrostatic pressure. Mark that the quantity P contains the contribution of all conservative forces.

In order to model the deviatoric stress tensor $\mathbf{t}$ it is necessary to add the constitutive equations. These equations depend on the type of material used. In SEPRAN a Newtonian model can be used, but also a number of non-Newtonian models are available. In this section we restrict ourselves to generalized Newtonian fluids.
The following models are available:

- Newtonian model
- Power-law model
- Carreau model
- Plastico-viscous model
- Three user type models

These models are characterized by their constitutive relations.

Constitutive equations:

The constitutive equations have all the form:

$\mathbf{t} \;=\; \eta(\nabla\mathbf{v})(\nabla\mathbf{v} + \nabla\mathbf{v}^T)e^{-c_t(T-T_1)} \qquad (\eta, c_T > 0$

The choice of $\eta(\nabla\mathbf{v})$ defines the model

- Newtonian fluid: η is constant.
- Power-law liquid: $\eta \;=\; \eta_n e^{-c_t(T-T_1)} II^{\frac{n-1}{2}}$
- Carreau liquid: $\eta \;=\; \eta_c e^{-c_t(T-T_1)}(1 + \lambda II)^{\frac{n-1}{2}}$
- Plastico-viscous liquid: $\eta \;=\; \eta_{pv} e^{-c_t(T-T_1)}(1 + [\frac{s}{\eta_{pv}} II^{-\frac{1}{2}}]^{\frac{1}{n}})^n$
  $\lim_{II\to 0} \frac{1}{2}\mathbf{t} : \mathbf{t} = \lim_{II\to 0} \eta^2 II = s^2$
- user model 1: $\eta \;=$ FNVT00$(\nabla\mathbf{v}, T)$
- user model 2: $\eta \;=$ FNVT01(II, T)
- user model 3: $\eta \;=$ FNVT02$(x_1, x_2, x_3, v_1, v_2, v_3, II, T)$

η denotes the dynamic viscosity (> 0) and s the yield stress. The parameters c_T, T_1, η_n, η_c, η_{pv} and n must be all positive. These parameters including s must be provided by the user.

II denotes the second invariant of the velocity deformation tensor, defined by
$II = \frac{1}{2}\mathbf{A}_1 : \mathbf{A}_1$; $\mathbf{A}_1 = \nabla\mathbf{v} + \nabla\mathbf{v}^T$.

α denotes a switch indicating if the viscous dissipation $\frac{1}{2}\mathbf{t} : \mathbf{A}_1$ must be taken into account ($\alpha = 1$) or not ($\alpha = 0$).

Boundary and initial conditions

In the instationary case it is necessary to give an initial condition for the velocity and temperature at $t = 0$.

The following types of boundary conditions are available:

(i) Boundary conditions for the velocity

Type V_1 Components of the velocity $\mathbf{v}(\mathbf{x})$ given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type V_2 Stress tensor components given on a part of the boundary. This is a so-called natural boundary condition.
Define σ_n as the stress component in the normal direction of a surface, and $\boldsymbol{\sigma}_t$ as the tangential component. Define v_n as the velocity component in the normal direction of a surface and $\mathbf{v}_t$ as the tangential component. In R^2 $\boldsymbol{\sigma}_t$ and $\mathbf{v}_t$ are scalar quantities.

Consult Section 7.1 with respect to the combinations that may be used.

(ii) Boundary conditions for the temperature

Type T_1 Temperature $T(x)$ given on some part of the boundary. This is an essential boundary condition, i.e. no boundary elements are required for this type.

Type T_2 $\kappa\frac{\partial T}{\partial n} + \sigma T = h$ on some part of the boundary. This is a so-called natural boundary condition.
In general boundary elements are necessary, except in the case that $\sigma(x) = 0$ **and** $h(x) = 0$, when there is no need to give any condition on this part of the boundary.

Solution method

The finite element equations for the temperature dependent incompressible fluid can be written as:

$$\mathbf{M}_u\frac{\partial \mathbf{u}}{\partial t} + \mathbf{S}_{uu}(\mathbf{u},\mathbf{T})\mathbf{u} + \mathbf{S}_{uT}(\mathbf{T}) - \mathbf{L}^T\mathbf{p} = \mathbf{F}_u \qquad \text{(momentum equations).} \tag{7.2.6}$$

$$\mathbf{M}_T\frac{\partial \mathbf{T}}{\partial t} + \mathbf{S}_{TT}(\mathbf{u},\mathbf{T})\mathbf{T} = \mathbf{F}_T \qquad \text{(temperature equation).} \tag{7.2.7}$$

$$\mathbf{L}\mathbf{u} = \mathbf{0} \qquad \text{(continuity equation).} \tag{7.2.8}$$

with

$\mathbf{u}$ the discretized velocity.

$\mathbf{M}_u$ the velocity mass matrix.

$\mathbf{S}_{uu}$ the matrix corresponding to stresses and the convective terms in the momentum equations.

$\mathbf{S}_{uT}$ the discretization of the Boussinesq term.

$-\mathbf{L}^T\mathbf{p}$ represents the ∇P term.

$\mathbf{F}_u$ the momentum source term.

$\mathbf{T}$ the discretized temperature.

$\mathbf{M}_T$ the temperature mass matrix.

$\mathbf{S}_{TT}$ is the matrix due to the discretization of the space part of the temperature equation.

$\mathbf{F}_T$ the temperature source term.

For the solution of the Navier-Stokes equations the following items are distinguished:

- incompressibility condition.
- non-linearity (due to convective terms and stress tensor in the case of non-Newtonian fluids)
- coupling between temperature equation and momentum equations.
- time dependence.

Continuity equation

See Section 7.1. At this moment only the penalty function method has been implemented.

Non-linearity

See Section 7.1.
The temperature in the viscosity is always taken at a preceding time level or iteration level in the stationary case. Hence for the temperature dependence always a Picard approach is used.

Time dependence

See Section 7.1. The Temperature equation is treated in exactly the same way as the momentum equations.

Coupling between temperature equation and momentum equations

The non-linear equations must be solved iteratively (stationary case) or by a time-discretization method (time-dependent case). In either situation the momentum equations and the temperature equation may be solved simultaneously (with a large number of degrees of freedom; velocities as well as temperatures) or uncoupled. In the latter case the momentum equations and the temperature equation are solved separately in an alternating sequence. This results in two smaller systems of equations to be solved, however, the number of iterations necessary for convergence may increase considerably.

In the coupled situation the iterative procedure is exactly the same as in Section 7.1, where the vector $\mathbf{u}$ is replaced by the larger vector $(\mathbf{u}, \mathbf{T}$. For an example see the manual SEPRAN EXAMPLES Section 7.2.1
For an example of the uncoupled approach we refer to the manual SEPRAN EXAMPLES Section 7.2.2.

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrices. This storage scheme is defined by the keyword METHOD = i in the input block "MATRIX" of program SEPCOMP.

  In Nearly all cases METHOD should be equal to 2 (direct method) or 6 (iterative solution method).
  The only exception is the uncoupled case in combination with an absence of convective terms in the momentum equations. In that case the temperature part is taken at a preceding level and hence a source term. So effectively the momentum equations reduce to STOKES and the matrix is symmetrical.
  The temperature equation is always unsymmetrical.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in 2.2, where, in general, the method by SEPCOMP is recommended.
  For each element group 17 parameters and coefficients must be given.

  These parameters and coefficients are defined as follows:

  1 Real parameter ε for the penalty method.

  2 Density ρ (real).

  3 Integer parameter MCONV, defining the type of linearization.

  4 Angular velocity ω (real)

  5 Volume expansion coefficient β (real)

  6 Reference temperature T_0 (real)

  7 Heat capacity at constant temperature c_p (real)

  8 Thermal conductivity κ (real)

  9 First component of body force $\mathbf{f}_1$ (real)

  10 Second component of body force $\mathbf{f}_2$ (real)

  11 Third component of body force $\mathbf{f}_3$
  This parameter is only used in the three-dimensional case or if an axi-symmetric model with swirl is used.
  If this parameter is necessary the next parameter are all one higher than indicated.

  11 Heat source $\mathbf{f}_T$ (real)

  12 Integer parameter MODEL, defining the type of viscosity model.

  Depending on the value of MODEL this parameter must be followed by the parameters η, c_T, T_1, n and λ in that sequence. Compare with Section 7.1 (previous version of SEPRAN).

  Meaning of the integer parameters:

  MCONV defines the type of linearization of the convective terms.
  Possible values:

  0 Stokes flow, the convective terms $(\mathbf{v} \cdot \mathbf{v})$ are neglected.

  1 Linearization by Picard's method.

  2 Linearization by Newton's method.

  10-12 See MCONV-10, in this case it is assumed that the instationary Navier-Stokes equations are solved and the linear combination of Mass matrix and Stiffness matrix is computed as indicated in Section 7.1(previous version of SEPRAN).

  MODEL Type of constitutive equation. MODEL consists of two parts: MODELV and α according to:
  MODEL = MODELV + 1000 $\times$ α, with

  MODELV Type of constitutive equation. Possible values:

  1 Newtonian liquid. In this case η is the dynamic viscosity.

  2 Power-law liquid. In this case η is the parameter η_n and n the parameter n in the power of the model.

  3 Carreau liquid. In this case η is the parameter η_c and n the parameter n in the power of the model. λ is the parameter λ the viscosity model.

  4 Plastico-viscous liquid. In this case η is the parameter η_{pv} and n the parameter n in the power of the model. λ is the parameter s the viscosity model.

  100 user provided model, depending on $\nabla \mathbf{v}$.

101 user provided model, depending on II.

102 user provided model, depending on $\mathbf{x}, \mathbf{v}$ and II.

α Indication if the viscous dissipation $\frac{1}{2}\mathbf{t} : \mathbf{A}_1$ must be taken into account ($\alpha = 1$) or not ($\alpha = 0$).

If MODELV ≥ 100 the user must provide a user written function subroutine, which computes η as function of the parameters. The following choices for user subroutines are available:

MODELV = 100: function subroutine FNVT00 is used.

MODELV = 101: function subroutine FNVT01 is used.

MODELV = 102: function subroutine FNVT02 is used.

These function subroutines have the following shape:

```
FUNCTION FNVT00 ( GRADV, TEMP )
IMPLICIT NONE
DOUBLE PRECISION FNVT00, GRADV(*), TEMP

    statements to compute the viscosity as function of GRADV and TEMP
    FNVT00 = computed viscosity

END



FUNCTION FNVT01 ( SECINV, TEMP )
IMPLICIT NONE
DOUBLE PRECISION FNVT01, SECINV, TEMP

    statements to compute the viscosity as function of SECINV and TEMP
    FNVT01 = computed viscosity

END



FUNCTION FNVT02 ( X1, X2, X3, V1, V2, V3, SECINV, TEMP )
IMPLICIT NONE
DOUBLE PRECISION FNVT02, X1, X2, X3, V1, V2, V3, SECINV

    statements to compute the viscosity as function of SECINV, X, V
    and TEMP
    FNVT02 = computed viscosity

END
```

In these subroutines array GRADV contains the gradient of the velocity in one point, SECINV the second inverse, X1,X2,X3 the co-ordinates of the point V1,V2,V3 the velocity components, and TEMP the temperature.
The storage of the gradient depends on the type of co-ordinate system, see Section 7.1.

- Definition of coefficients for the boundary conditions:

  The natural boundary conditions may require input depending on the values given. Separate boundary elements for velocity and temperature are required.

Input for the velocity natural boundary conditions For each element group 2 (R^2 or 3 R^3) coefficients must be given.

These coefficients are defined as follows:

1 σ_n
2 First component of $\boldsymbol{\sigma}_t$
3 Second component of $\boldsymbol{\sigma}_t$

Input for the temperature natural boundary conditions For each element group 2 coefficients must be given.

These coefficients are defined as follows:

1 σ
2 h

- Computation of derivatives:

Depending on the parameter ICHELD in the input block "DERIVATIVES" the following types of derivatives are computed:

1 The pressure is computed in the vertices of the elements.
2 The derivative $\frac{\partial v_{JDEGFD}}{\partial x_{IX}}$, with JDEGFD and IX parameters in the input block "DERIVATIVES".
3 div **v** is computed in the vertices of the elements.
4 curl **v** is computed in the vertices of the elements.
5 **t** is computed in the vertices of the elements.
6 $\nabla \mathbf{v}$ is computed in the vertices of the elements.
7 The rate of elongation $\dot{\epsilon}$ is computed in the vertices of the elements.
$\dot{\epsilon}$ is the rate of strain in the direction of the velocity: $\dot{\epsilon} = \frac{1}{2}\mathbf{v} \cdot \mathbf{A}_1 \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}$
8 The shear rate $\dot{\gamma}$ is computed in the vertices of the elements, according to the definition:
$\dot{\gamma} = \mathbf{n}_1 \cdot \mathbf{A}_1 \mathbf{n}_2$, $\mathbf{n}_1 \cdot \mathbf{n}_2 = 0$, $\|\mathbf{n}_1\| = \|\mathbf{n}_2\| = 1$
For problems with two velocity components: $\|\mathbf{n}_1\| = \frac{\mathbf{v}}{\|\mathbf{v}\|}$, $\|\mathbf{n}_2\| = \frac{(-v_2, v_1)}{\|\mathbf{v}\|}$
For problems with three velocity components:
$\dot{\gamma}_{12} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_2$, $\dot{\gamma}_{13} = \mathbf{e}_1 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$, $\dot{\gamma}_{23} = \mathbf{e}_2 \cdot \mathbf{A}_1 \cdot \mathbf{e}_3$
9 $II^{\frac{1}{2}}$ is computed in the vertices of the elements.
10 ∇T in the vertices is computed.

The output vector is defined as follows:

ICHELD=1,2,3,7,9 a vector of special structure (IVEC=1) with one unknown in each vertex.

ICHELD=4 In the two-dimensional case a vector of special structure (IVEC=1) with one unknown in each vertex.
In the three-dimensional case a vector of special structure (IVEC=5).
This is also the case for two-dimensional axi-symmetric flow with swirl.

ICHELD=5 a vector of special structure (IVEC=3).

ICHELD=6 a vector of special structure (IVEC=4).

ICHELD=8 In the two-dimensional case a vector of special structure (IVEC=1) with one unknown in each vertex.
In the three-dimensional case a vector of special structure (IVEC=5).

ICHELD=10 a vector of special structure (IVEC=6).

- Types of integrals that may be computed:

  If the user wants to compute integrals over the solution, he may use the option INTEGRAL in the input block "STRUCTURE"

  The parameter ICHELI in the input block "INTEGRALS" is used to distinguish the various possibilities:

  ICHELI=1 $\int_\Omega f(x)d\Omega$

  The user must define the function $f(x)$ as first coefficient by one of the methods described in 2.2.

- Input with respect to the time-dependence.

  See Section 7.1 previous version of SEPRAN. MCONV=30 has not been implemented.

Definition of type numbers

The type numbers for the old version are defined as follows:

420 Quadratic isoparametric triangle in R^2, Cartesian co-ordinates.

423 Boundary conditions of type V_2 corresponding to type 420.
The element must be a quadratic boundary element in R^2.

426 Boundary conditions of type T_2 corresponding to type 420.
The element must be a quadratic boundary element in R^2.

421 Quadratic isoparametric triangle in R^2, Axisymmetric co-ordinates without swirl.

424 Boundary conditions of type 2 corresponding to type 421.
The element must be a quadratic boundary element in R^2.

427 Boundary conditions of type T_2 corresponding to type 421.
The element must be a quadratic boundary element in R^2.

422 Quadratic isoparametric triangle in R^2, Polar co-ordinates.

425 Boundary conditions of type V_2 corresponding to type 422.
The element must be a quadratic boundary element in R^2.

428 Boundary conditions of type T_2 corresponding to type 422.
The element must be a quadratic boundary element in R^2.

429 Quadratic isoparametric triangle in R^2, Axisymmetric co-ordinates with swirl.

437 Boundary conditions of type V_2 corresponding to type 437.
The element must be a quadratic boundary element in R^2.

438 Boundary conditions of type T_2 corresponding to type 437.
The element must be a quadratic boundary element in R^2.

7.3 Turbulent flow

In this Section we consider the elements that are available for turbulent flow.

7.3.1 The isothermal turbulent flow of incompressible liquids according to Boussinesq's hypothesis

Equations

For the derivation of the equations the averaging procedure of Osborne Reynolds is applied. The fluctuating instantaneous velocity $\mathbf{u}$ is decomposed into an average velocity $\mathbf{V}$, slowly varying with time and space, and the fluctuating component $\mathbf{v}$ with average $\bar{\mathbf{v}} = 0$. In a general reference system the resulting equations are:

Conservation of mass (incompressibility constraint):

$$\operatorname{div} \mathbf{V} = 0 \ .$$

Conservation of momentum (Reynolds's form of the Euler-Cauchy equations):

$$\rho \left[\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \operatorname{grad}) \mathbf{V} + 2\boldsymbol{\Omega} \times \mathbf{V} \right] + \operatorname{grad} P - \operatorname{div} (\mathbf{t}_{visc} - \rho \mathbf{v}\mathbf{v}) = \rho \mathbf{f}$$

with generalized pressure (or reduced pressure) P:

$$P = p - \frac{1}{2} \rho \, (\boldsymbol{\Omega} \times \mathbf{x}) \cdot (\boldsymbol{\Omega} \times \mathbf{x})$$

with unit tensor $\mathbf{1}$. The turbulent stress is defined as:

$$\mathbf{t}_{turb} = -\rho \, \overline{\mathbf{v}\mathbf{v}} + \frac{\rho}{3} \overline{v^2} \mathbf{1} \ , \quad P_m = P + \frac{\rho}{3} \overline{v^2} \ , \quad \boldsymbol{\Sigma} = -P_m \mathbf{1} + \mathbf{t} \ , \quad \mathbf{t} = \mathbf{t}_{visc} + \mathbf{t}_{turb},$$

which defines a new generalized pressure P_m and a total stress $\boldsymbol{\Sigma}$. The constitutive equations for purely-viscous liquids and for the Reynolds stresses according to Boussinesq's hypothesis are:

$$\mathbf{t}_{visc} = \eta \, \mathbf{A}_1 \ , \ \mathbf{t}_{turb} = \eta_m \, \mathbf{A}_1 \ , \ \mathbf{A}_1 = \operatorname{grad}\mathbf{V} + (\operatorname{grad}\mathbf{V})^T \ .$$

with $(\ )^T$ denoting the transpose of a tensor. The tensor $\mathbf{A}_1$ is two times Euler's rate of deformation tensor.

The equations are rewritten with help of the new generalized pressure P_m and extra stress $\mathbf{t}$

$$\rho \left[\frac{\partial \mathbf{V}}{\partial t} + (\mathbf{V} \cdot \operatorname{grad}) \mathbf{V} + 2\boldsymbol{\Omega} \times \mathbf{V} \right] + \operatorname{grad} P_m - \operatorname{div} \mathbf{t} = \rho \mathbf{f}$$

The governing equations have the same structure as the equations for isothermal laminar flow for a generalized newtonian liquid. This means that for fully-developed flow there is only one component of the velocity $\mathbf{V} = (V_1(x_2), V_2 = 0, V_3 = 0)$ different from zero: the component in the direction of the mean flow and it varies only perpendicular to this direction. The pressure P_m is a *linear* function of the coordinate along the mean flow only!

Meaning of the parameters:

ρ	density of the liquid
$\mathbf{V}$	mean velocity of the liquid
$\mathbf{v}$	turbulent fluctuating part of the velocity
$\boldsymbol{\Omega}$	angular velocity $(0, 0, \omega)$ of the co-ordinate system with respect to an inertial system.
p	hydrostatic pressure
P_m	generalized (turbulent) pressure
$\boldsymbol{\Sigma}$	total stress tensor
η	dynamic viscosity
η_m	turbulence or eddy viscosity (Boussinesq's hypothesis)
$\mathbf{f}$	external force field (body force)

Criticism

At the axis of symmetry of a channel or a pipe the elements of the tensor $\mathbf{A}_1$ are equal to zero for fully developed flow conditions, and therefore, according to Boussinesq's hypothesis:

$$\mathbf{t}_{turb} = \mathbf{0} \ , \ \overline{v_1^2} = \overline{v_2^2} = \overline{v_3^2} = \overline{v^2}/3$$

which differs from experiments. For comments and criticism see Hinze (1975) and Tennekes and Lumley (1974). However, for flows with no more than one characteristic length scale and one characteristic velocity scale Boussinesq's hypothesis is useful. These flows are approximately-parallel flows.

The velocity profile close to a wall

Much of the reliability of the computation of a turbulent flow depends on the accurate modeling of η_m near a solid wall. The viscosity η is small with respect to η_m *except for a thin layer close to the wall.* At the wall $\overline{\mathbf{v}\mathbf{v}}$ must be zero because $\mathbf{v} = 0$ there. Close to the wall are —in order of

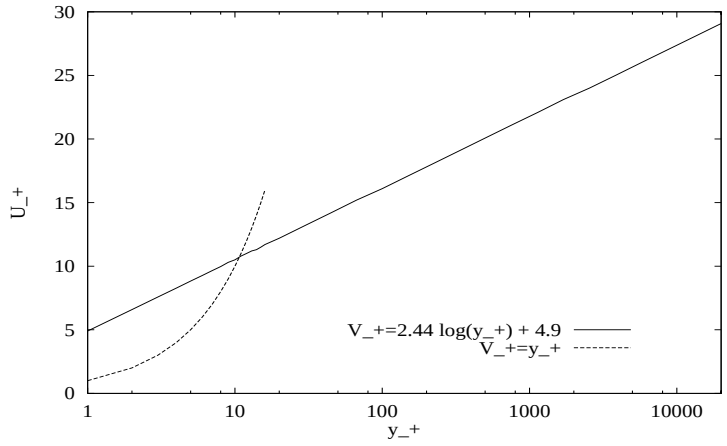


Figure 7.3.1.1: Velocity profile near a fixed wall

increasing distance from the wall— the viscous sublayer, the buffer layer, the inertial sublayer, and the core region of an internal flow (or the outer layer of an external flow).

The friction velocity u_* is a useful concept. It is defined with help of the shear stress along the wall τ_w. Then also a dimensionless distance y_+ from the wall can be defined:

$$u_*^2 = \frac{\tau_w}{\rho} \ , \quad \tau_w = \mathbf{n} \cdot \mathbf{\Sigma} \cdot \mathbf{t} \ , \ y_+ = \frac{y u_* \rho}{\eta}$$

with distance from the wall y, normal and tangential vectors $\mathbf{n}$ and $\mathbf{t}$, and stress tensor $\mathbf{\Sigma}$. An initial estimate of u_* is therefore required. It depends on the type of flow and the smoothness of the wall how u_* depends on the global flow characteristics.

It has been given by Hussain and Reynolds (1975) on empirical grounds for channel flow, pipe flow, and flow along a flat plate with negligible pressure gradient:

$$Re_* \simeq 0.1058\, Re^{0.911} \ , \ 10^4 \leq Re \leq 6.10^4 \ , \ Re = \frac{\rho V_m L}{\eta} \ , \ Re_* = \frac{\rho u_* L}{\eta}$$

with characteristic velocity V_m and length L of the mean flow. Re is the Reynolds number. V_m can be based of the average flow rate, and L can be the diameter of a tube. Then we have also

$$y_+ = \frac{y}{L} Re_* \ , \ V_+ = \frac{V}{u_*} = \frac{V}{V_m} \frac{Re}{Re_*}$$

For illustrative purposes a picture of the velocity profile near the wall has been given in Figure 7.3.1.1. In the viscous sublayer ($y_+ \leq 5$) $V_+ = y_+$ and in the inertial sublayer ($y_+ > 55$) $V_+ = \frac{1}{\kappa}\log(y_+) + B$, $\kappa \approx 0.33 - 0.41$, $B \approx 4 - 6$. The constants κ (von Karmann's constant) and B have been determined from experiments. For the velocity profile in the buffer layer is no analytical expression.

Close to the wall in the sublayers the flow is parallel to the wall. With coordinates x and y parallel and perpendicular to the wall respectively, the equation for conservation of momentum yields

$$\frac{\partial (t_{visc} + t_{turb})_{yx}}{\partial y} = \frac{\partial P_m}{\partial x} - \rho f_x \quad , \quad \left(\frac{\partial}{\partial x} \equiv 0 \text{ for } \mathbf{V} \text{ and } \mathbf{\Sigma} \right)$$

Integration of this equation over y from 0 to δ in the inertial subrange yields $t_{yx}(y = \delta) = t_{yx}(y = 0) + O(\delta)$. For large values of Re we have therefore in the inertial sublayer

$$t_{yx}(y = \delta) = (\eta_m + \eta) \left(\frac{dV_x}{dy} \right)_{y=\delta} = \eta \left(\frac{dV_x}{dy} \right)_{y=0} = \tau_w$$

$$-\rho \overline{u_x u_y} \approx \tau_w = \rho u_*^2$$

when η is neglected with respect to η_m which is allowed when δ is in the midst of the inertial sublayer.

Boundary conditions.

Application of boundary conditions requires extreme care, as they determine the solution completely. The differential equations are of elliptic type, so we need boundary conditions everywhere at the boundary.

Permitted are the combinations

(i) $\mathbf{V}$ prescribed

(ii) Σ_n and $\mathbf{V}_t$ prescribed

(iii) $\mathbf{\Sigma}_t$ and V_n prescribed

(iv) Σ_n and $\mathbf{\Sigma}_t$ prescribed

with unit normal and tangential vectors $\mathbf{n}, \mathbf{t}$ on/along a surface and with the definitions:

$$\mathbf{\Sigma} = -P_m \mathbf{1} + (\eta + \eta_m)\mathbf{A}_1 \quad , \quad \Sigma_n = \mathbf{n} \cdot \mathbf{\Sigma} \cdot \mathbf{n} \quad , \quad \mathbf{\Sigma}_t = \mathbf{n} \cdot \mathbf{\Sigma} - \mathbf{n}\Sigma_n$$

$$\Longrightarrow \Sigma_n = -P_m + 2(\eta + \eta_m)\frac{\partial V_n}{\partial n} \ .$$

The following combinations of the boundary conditions are frequently used

(i) no-slip condition $\mathbf{V} = \mathbf{0}$ at a fixed wall

(ii) velocity prescribed in inertial sublayer at $y_+ \approx 100$, $V_n = 0$, $V_t = u_*/\kappa \log(y_+) + B$, which requires estimation of u_*. Along a rough wall we can use exactly the same formula with slightly different value of the constant B.

(iii) free slip condition or symmetry condition $V_n = 0$, $\mathbf{\Sigma}_t = \mathbf{0}$

(iv) instream or outstream condition Σ_n and $\mathbf{V}_t$ given. For fully developed flow $\Sigma_n = -P_m$, $\mathbf{V}_t = \mathbf{0}$

Prandtl's mixing length hypothesis

Prandtl's mixing length hypothesis specifies η_m in terms of a length scale l_m. Implicitly is assumed that the turbulent problem can be described in terms of one characteristic length scale only.

$$\eta_m = \rho l_m^n II^{\frac{1}{2}} , \quad II = \frac{1}{2}\mathbf{A}_1 : \mathbf{A}_1 , \quad n > 0 .$$

The quantity $II = \sum_i \sum_j A_{1(ij)} A_{1(ji)}/2$ is the second invariant of the tensor $\mathbf{A}_1$; for parallel flow with $\mathbf{V} = (V_1(x_2), 0, 0)$ we have $II = |dV_1/dx_2|$. Strictly speaking the above model has been derived by Prandtl for channel flow and pipe flow with $n = 2$. For other values of the parameter n the physical dimensions of the equation are not correct. The formula is then of empirical value only. The user may either choose between standard models from the package or may submit his own model for the mixing length l_m. The choice of this model depends strongly of the type of flow. For the channel and pipe flow the power $n = 2$ must be applied. A short discussion will be presented to this case below. This is of special importance for the application of the boundary conditions for fully developed flow.

For the turbulent stress in the inertial sublayer can be written:

$$-(\overline{u_x u_y})_{y=\delta} = l_m^2 II^{1/2} \frac{dV_x}{dy} \Longrightarrow = l_m^2 \left|\frac{dV_x}{dy}\right| \frac{dV_x}{dy}$$

Therefore, using the definitions for y_+, V_+, we can derive $l_{m+} dV_+/dy_+ = 1$. Using the expression for the velocity profile in the inertial sublayer the mixing length becomes

$$l_{m+} = \kappa y_+ ,$$

which is also valid for a totally rough wall. In order to make this expression valid also in the buffer layer and the viscous sublayer van Driest proposed:

$$l_{m+} = \kappa y_+ \left(1 - e^{-y_+/A}\right) , \; A \approx 26$$

In the core region of a channel or pipe flow the mixing length according to Nikuradse can be used

$$l_{m,k}/h = 0.14 - 0.08(1 - y/h)^2 - 0.06(1 - y/h)^4 , \; \frac{y}{h} = \frac{L}{h\,Re} y_+$$

with distance h from the wall to the axis of symmetry. For $y \to \delta$, $l_{m,k} \to 0.4y$ which is equal to the equation $l_{m+} = \kappa y_+$ for $\kappa = 0.4$. Now for a complete generalization of the mixing length for both kernel area and wall area the Nikuradse form is corrected with van Driest's proposal:

$$l_m/h = l_{m,k}/h \left(1 - e^{-y_+/A}\right)$$

It will be clear from the above discussion that u_* or τ_w must be known for the determination of y_+ and l_m. The formula of Hussain and Reynolds can be used for that purpose. In a more advanced version τ_w can be computed by the package (not yet available). The mixing length must be specified in dimensional units.

Practical computations

The velocity gradient close to the wall is very large. In order to compute the velocity field $\mathbf{V}$ accurately in the whole field including the wall region, at least 4 very thin elements along the wall are required to cover $0 < y_+ < 100$. They must have increasing width ! It depends on u_* what $y_+ = 100$ is in real units. For the kernel region of a channel or pipe flow another 6 elements are required for a reasonable approximation. For higher accuracy more elements are required in both regions.
It is strongly advised to gain experience with channel flow or pipe flow and compare the numerical and experimental results.

In many cases one is interested only in the solution away from the wall. In this region viscosity does not play an important role and van Driest's correction can be left out. Nikuradse's formula can de used without correction. The boundary of the region of the computations must be within the inertial sublayer (for example at $y_+ \approx 100$), u_* must be estimated and a slip velocity at the wall must be prescribed.

The resulting equations are nonlinear due to convective terms and the nonlinear η_m. The effective viscosity of the mixing-length model resembles that of the power-law model of generalized-newtonian liquids (laminar flow) with power $n = 2$ and $\eta_p = \eta + \eta_m$. This model also requires an initial estimate of the solution with help of another model: choose the newtonian flow with $\eta \approx \eta_p$ and without convection. As second step perform a few iterations with convection. If the Reynolds number based on $\max \eta_m$ for this flow is to large, then a lower value of Reynolds must be chosen for the initial estimate. Then, finally the computations with the mixing-length model can be started. The values of η_m for this initial estimates can be gained from experiments. For fully developed flow and $Re \approx 3.10^4$ the maximum of η_m/η is about 110, so $\eta_p = 110\eta$ can be taken, i.e. the flow at a Reynolds number about a factor 100 lower: $Re \approx 300$.
It will be clear from the discussion in this section that a user should be more or less familiar with the literature on turbulence in order to use the package effectively.

For some details about the solution method and the treatment of the nonlinear terms, Section 7.1 may be consulted.

Input for the various subroutines

- Definition of the storage scheme:

  The first thing to be chosen is the type of storage scheme for the matrix. This storage scheme is defined by the keyword METHOD $= i$ in the input block "MATRIX" of program SEPCOMP. The matrix is not symmetrical due to convection, so `METHOD=2` must be used.

- Definition of the coefficients for the differential equation:

  The coefficients for the differential equation may be defined by one of the methods described in Section 10.1, where, in general, the method by SEPCOMP is recommended.
  The input parameters are similar to the laminar case and, additionally, information about the wall(s) with respect to which the mixing length must be computed. The walls are specified by curve numbers of the mesh. Suppose that the wall is composed of c1,c2,c3 and c5,c6,c7. Then we have 2 sets of curves c1,c3 and c5,c7 if the last point of c_n coincides with the first point of c_{n+1} for $n = 1, 2; 3, 4$. Otherwise you should, for this example, specify `NSETS=6` for the sets of curves c1,c1; c2,c2; c3,c3; c5,c5; c6,c6; and c7,c7. In the parameter list (see below) the *numbers* of the curves must be used. These are denoted by `IC1, IC2`, etcetera.

  For the input of the parameters subroutine `FIL100` can be used, which means that the arrays `IWORK` and `WORK` must be filled as sketched in the table. The total number of parameters is also indicated.

meaning of parm	sequence number	`integer`	`real`
penalty	1	0	ϵ
density	2	0	ρ
linearization	3	`MCONV`	-
rotation	4	0	ω
body force x_1	5	`ICHF1` or 0	`F1`
body force x_2	6	`ICHF2` or 0	`F2`
body force x_{ncomp}	4+ncomp	`ICHF.` or 0	`Fncomp`
viscosity model	5+ncomp	`MODELV`	-
viscosity	6+ncomp	`ICHETA` or 0	η
power	7+ncomp (modelv=1)	0	`n`
friction velocity	8+ncomp (modelv=1)	`ICH`u_* or 0	u_*
length scale	9+ncomp (modelv=1)	`ICHh` or 0	`h`
n sets of curves	7+ncomp or 10+ncomp	`NSETS`	-
first curve of 1	$\cdots$	`ICS11`	-
second curve of 1	$\cdots$	`ICS12`	-
	$\vdots$	$\vdots$	$\vdots$
first curve of n	$\cdots$	`ICSN1`	-
second curve od n	$\cdots$	`ICSN2`	-

`MODELV`	`NPARM`
1	12+`NSETS`$*$2
100	9+`NSETS`$*$2

Meaning of the parameters:

ϵ penalty function parameter (see section 7.1).
ϵ must be chosen such that $\epsilon P_m = O(10^{-6})$.

ρ density of the liquid.

`MCONV` indicates type of linearization of the convective terms. Possible:
1 the convective terms $(\mathbf{V} \cdot \text{grad})\mathbf{V}$ are linearized according to Picard's method.
2 linearization by Newton's gradient method.

ω angular velocity of the rotating system.

f_i external forces (body force).

`MODELV` indicates the type of model. Possibilities:
1 Prandtl's mixing length defined by the package.
100 User-defined model depending on $\mathbf{x}$ and distance from the wall curves.

η the dynamic viscosity of the liquid.

n the power of the length scale (`MODELV=1` only).

u_* friction velocity (`MODELV=1` only).

h local length scale from wall to symmetry axis for Nikuradse's formula (`MODELV=1` only).

`NSETS` number of sets of curves that form the fixed wall.

`ICS11` curve number of the first curve of the first set.

`ICS12` curve number of the last curve of the first set.

$\vdots$

`ICSN2` curve number of the last curve of the `NSETS`th set.

When `MODELV` is equal to 100 the user must provide a function subprogram `FML100` that computes the mixing length as a function of $\mathbf{x}$ and the distance from wall curves. This distance is computed by the package for any nodal point (or integration point). This function is called by the matrix building subroutines in the following form:

```
      FUNCTION FML100 (X1,X2,X3,DISTWL)
      IMPLICIT NONE
      DOUBLE PRECISION FML000, X1, X2, X3, DISTWL
      ...
C
C       statements to give PRANDTL a value as function of x1,x2,distwl
```

```
C
      FML100=PRANDTL
      END
```

- Definition of the coefficients for the boundary conditions:

  The coefficients for the boundary conditions may be defined by one of the methods described in Section 10.1, where, in general, the method by SEPCOMP is recommended.

  When stresses must be prescribed that are different from zero, then boundary elements must be created and Σ_n and Σ_t must be specified. These are the only two coefficients required and are numbered 1 and 2 respectively.

- Computation of derivatives:

  The parameter ICHELD in the input block "DERIVATIVES" is used to distinguish the various possibilities:

  ICHELD = 1 the generalized pressure is computed in the vertices of the elements (IVEC=1).

  ICHELD = 2 the derivative $\frac{\partial V_j}{\partial x_i}$ is computed with JDEGFD=j,IX=i (IVEC=1).

  ICHELD = 3 div$\mathbf{V}$ is computed in the vertices of the elements (IVEC=1).

  ICHELD = 4 curl$\mathbf{V}$ is computed in the vertices of the elements (IVEC=1).

  ICHELD = 5 $\mathbf{t} = \mathbf{t}_{visc} + \mathbf{t}_{turb}$ is computed in the vertices of the elements (IVEC=3).

  ICHELD = 6 grad $\mathbf{V}$ is computed in the vertices of the elements (IVEC=4).

  ICHELD = 7 the elongation rate $\dot{\epsilon}$ is computed in the vertices of the elements (IVEC=1). For the definition see Section 7.1.

  ICHELD = 8 the shear rate $\dot{\gamma}$ is computed in the vertices of the elements (IVEC=1). For the definition see Section 7.1.

  ICHELD = 9 the quantity $II^{1/2}$ is computed in the vertices of the elements (IVEC=1).

  The output vector is defined as follows:

  ICHELD= 1 the generalized pressure P_m is computed in the vertices of the elements.
  2 the derivative $\frac{\partial V_j}{\partial x_i}$ is computed with JDEGFD=j, IX=i.
  3 div$\mathbf{V}$ is computed in the vertices of the elements.
  4 curl$\mathbf{V}$ is computed in the vertices of the elements.
  5 $\mathbf{t} = \mathbf{t}_{visc} + \mathbf{t}_{turb}$ is computed in the vertices of the elements.
  Cartesian coordinates: t_{11}, t_{12}, t_{22}, axi-symmetrical coordinates $t_{rr}, t_{rz}, t_{zz}, t_{\phi\phi}$.
  6 grad$\mathbf{V}$ is computed in the vertices of the elements.
  See section 7.1.1 for details.
  7 the elongation rate $\dot{\epsilon}$ is computed in the vertices of the elements.
  For the definition see section 7.1.1.
  8 the shear rate $\dot{\gamma}$ is computed in the vertices of of the elements.
  For the definition see section 7.1.1.
  9 the quantity $II^{1/2}$ is computed in the vertices of the elements.

- Types of integrals that may be computed:

  If the user wants to compute integrals over the solution, he may use the option INTEGRAL in the input block "STRUCTURE"

  The parameter ICHELI in the input block "INTEGRALS" is used to distinguish the various possibilities:

ICHELI=1 $\int_\Omega f(\mathbf{x}) d\mathbf{\Omega}$ is computed. The function $f(\mathbf{x})$ must be specified as first coefficient.

- Computation of stream function

  When `ICHELS`=1 in STREAM then the stream function is computed in the nodal points (`IVEC`=2).

Definition of type numbers

The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the second order elliptic equation in this section the following type numbers are available:

430 Quadratic isoparametric triangle in R^2 (Cartesian co-ordinates).
element shape number for mesh generation: 4

401 Boundary conditions of stress type:
(Cartesian co-ordinates).
element shape number for mesh generation: 2

431 Quadratic isoparametric triangle in R^3 (Axi-symmetrical co-ordinates, no swirl: 2 space co-ordinates).
element shape number for mesh generation: 4

405 Boundary conditions of stress type:
(Axi-symmetrical co-ordinates, no swirl: 2 space co-oordinates).
element shape number for mesh generation: 4 element shape number for mesh generation: 2

433 Quadratic isoparametric triangle in R^3 (Axi-symmetrical co-ordinates, with swirl: 2 space co-oordinates, 3 velocity components).
element shape number for mesh generation: 4

407 Boundary conditions of stress type:
(Axi-symmetrical co-ordinates, with swirl: 2 space co-oordinates, 3 velocity components).
element shape number for mesh generation: 2

7.4 Methods to compute solid-fluid interaction

In this section we consider some methods available to tackle the problem of a solid-fluid interaction. At this moment the following specific methods are available:

Fictitious domain method This method is especially meant for a moving solid in a fluid domain.

Below we give a short description of the various methods; examples are treated in the next sections.

Fictitious domain method The problem we consider is a rigid body that moves with some speed in a fluid domain. The speed of the solid may be given. Due to the movement of the solid, the velocity field in the fluid changes.
The idea behind the fictitious domain method is that it is difficult to mesh the fluid region since the position of the solid changes in each time step. For that reason we use a fixed mesh for the fluid, which includes the solid part and a fixed mesh for the solid. Both meshes are considered as stand-alone, hence no points of the two meshes are coupled.
In order to couple the solid velocity and the fluid velocity we introduce the constraint that the velocity at part of the boundary of the solid that is inside the fluid is equal to the velocity in the same position in the fluid. Since, in general the solid boundary does not coincide with fluid nodes, it is necessary to apply this constraint in a weak sense.
For that purpose we define so-called Lagrangian multipliers at the boundary of the solid elements as far as this boundary is in the fluid. So we define boundary elements for that boundary and on these boundary elements we define Lagrangian multipliers. The constraint is translated into an integral relation.
For details of the method, the user is referred to Glowinski et al (1994a, 1994b, 1999) and Bertrand et al (1997).
The Lagrange multipliers act as extra unknowns, necessary to satisfy the constraint. From the literature it is known that the basis functions (shape functions) corresponding to the Lagrange multipliers must be one degree lower than the approximation of the displacement in the solid element.
Hence to apply the fictitious domain method we have to apply the following steps:

- Create a mesh for the fluid without taking the solid into account.
- Create a mesh for the solid without taking the fluid into account.
- Define fictitious elements along the part of the solid that is in the fluid.
- solve the problem

Hence not much extra steps are necessary to apply the fictitious domain method. Besides the fact that we need to independent meshes, all we have to do is to provide input stating where the fictitious unknowns must be positioned.
This step is done in the input block `problem`.
It consists of two parts of input:

- Part `fictitious_unknowns`.
  In this parts the various groups of fictitious elements are defined and the corresponding element types as described in this section.
- Part `fictitious_elements`.
  In this part we define where the fictitious elements must be positioned.
  Furthermore it is described which fluid element groups correspond to these elements as well which structural (solid) element groups.
  Also the shape of the basis functions for the Lagrange multipliers is defined.
  At this moment the shape must always be of one degree less than that of the structural elements. Hence if we use linear two-dimensional elements for the structural elements, we need constant basis functions along the boundary. This is indicated by `multiplier_shape=1`.

In the manual SEPRAN EXAMPLES Section 7.4.1 a very simple example is given to show what kind of input is expected.

Warning

If you define fictitious unknowns along a boundary where the fluid part satisfies a Dirichlet boundary condition, for example no-slip, the result will be a singular matrix. In that case the fictitious unknowns are not coupled to any fluid unknown and this causes the problem.

Definition of type numbers

At this moment only one general type number for the fictitious elements is available.

921 Standard type number for fictitious elements.

923 Standard type number for fictitious elements with only one degree of freedom per point. Hence this element is meant to be used for scalar equations.

7.5 Methods to compute fluid flow in the presence of an obstacle

In this section we consider some methods available to tackle the problem of a stiff obstacle in the fluid. The obstacle may be either stationary or moving in time. At this moment the following specific methods are available:

Adapted Mesh method In this case the mesh is adapted to the obstacle.
This method is the most accurate since only the flow region is covered by elements.
It is less suited for moving obstacles.

Fictitious domain method This method is especially meant for a moving solid in a fluid domain.

Approximate Adapted Fixed Mesh method In this method the fixed mesh is locally adapted to the obstacle in some approximate sense.
This method can be used both for fixed as well moving objects.

Pseudo concentration method This fixed mesh method is very similar to the level set method.
A description is not yet available.

All methods we describe (except the adapted mesh method) have in common that the fixed mesh is intersected by the obstacle. All these methods therefore start by computing the intersection of the edges of the fixed mesh with the obstacle in a very efficient way.
Once this has been done, nodes and elements are labeled.
We distinguish between the following types of elements:

- Elements that are completely outside the obstacle. These elements have no special name and correspond to the fluid mesh only.
- Elements that are completely inside the obstacle. These elements are defined in the Users Manual by `inner_obstacle`. See for example Section 3.2.2 (part SKIP_ELEMENTS).
- Elements that are partly inside the obstacle and partly outside the obstacle. So in fact they are intersected by the obstacle. These elements are defined in the Users Manual by `on_obstacle`. See for example the Users Manual Section 3.2.2 (part SKIP_ELEMENTS).
- Of this last category there is a class of elements that has all nodes either inside the obstacle or on the boundary of the domain. These elements may be special with respect to the solution, since it is possible that all nodes have prescribed velocities.
  Such elements are indicated by the combination `on_obstacle and curves ( ci to cj)`. See also the Users Manual Section 3.2.2 (part SKIP_ELEMENTS)

Also the nodal points can be distinguished into several groups:

- Nodes that are in elements that are completely outside the obstacle. These nodes have no special name and correspond to the fluid mesh only.
- Nodes that are in elements that are completely inside the obstacle. These nodes are referred to as `in_inner_obstacle` in the Users Manual Section 3.2.2 (part ESSENTIAL BOUNDARY CONDITIONS).
- Nodes that are in elements that are intersected by the obstacle, i.e. these elements are partly inside and partly outside the obstacle. These nodes are called `in_boun_obstacle` in the Users Manual Section 3.2.2 (part ESSENTIAL BOUNDARY CONDITIONS).
- Nodes of the fixed mesh that are on the boundary of the obstacle. These nodes are called `on_boun_obstacle` in the Users Manual Section 3.2.2 (part ESSENTIAL BOUNDARY CONDITIONS).

- Finally there is the set of nodes that are completely inside the obstacle (boundary of the obstacle included). These points are not directly coupled to a special class of elements.
  These nodes are denoted by `in_all_obstacle` in the Users Manual Section 3.2.2 (part ESSENTIAL BOUNDARY CONDITIONS).

Below we give a short description of the various methods; examples are treated in the next sections.

Adapted Mesh method The adapted mesh method is the most accurate method of all methods that treat the flow around an obstacle. However, this method requires that the mesh is adapted to the obstacle. For a two-dimensional fixed obstacle this is no problem at all and also for a three-dimensional obstacle mesh generation can be carried out.
However, as soon as the obstacle starts to move the generation of the mesh may be become a problem.
Examples of this method can be found in the manual SEPRAN EXAMPLES Section 7.5.1

Fictitious domain method This method has already been described in Section 7.4. It can also be used for a free moving object in a fluid, as will be shown in the manual SEPRAN EXAMPLES Section 7.5.1

Approximate Adapted Fixed Mesh method The idea of the Approximate Adapted Fixed Mesh method (AAFM) is the following. We start with a fixed mesh in the fluid domain that like all other fixed mesh methods, does not take the obstacle into account.
After that we compute the intersection of the obstacle with all the edges of the fixed mesh. At this moment the method is restricted to linear triangles (2D) and tetrahedrons (3D) only. The extension to other types of elements is very difficult, but quadratic elements do not form a bottle neck.
The intersection points on the intersected edges are all marked as new points for the mesh. However, we make some important limitations that make the method feasible. First of all, if the intersection is close to the end point of an edge, we mark that end point as an intersection point. o the corresponding end point is said to be on the boundary of the obstacle, even if it is actually not on the boundary. At this moment close is defined as a distance to the end point which is less than 30% of the length of the edge. This construction is necessary in order to avoid ill-shaped elements.
The other restriction we impose is that each edge may only once be intersected by the obstacle (outside the end points of course). If more than one intersection point is found, which are not close to the end point, the intersection is supposed to be positioned in the middle of both intersection points.
Once the new intersection points are defined, new subelements of the original fixed fluid mesh are made based on the extra points. For example in a triangle the number of extra points is at most 3 and the triangle may be subdivided into at most 4 subtriangles.
After the new subdivision we have only elements that are completely inside or completely outside the obstacle. In fact the obstacle is approximated by a new obstacle defined by the intersection points. This new obstacle may be close to the original one, but in general it will not be exactly the same. The influence of this approximation is usually only local, i.e. in the neighbourhood of the obstacle itself. If such an approximation is not good enough, the only solution is to use a finer mesh in the neighbourhood of the obstacle.
After the new mesh is created we have a mesh consisting of two parts, one inside the obstacle and one outside. By prescribing the velocities inside the obstacle we return back to the same method as used for the adapted mesh method.
Once the solution on the new mesh has been computed completely the adapted mesh is removed and the solution is mapped back onto the old mesh. This last step is trivial since all nodes of the old mesh are also part of the new mesh.
If the problem is time-dependent and the obstacle moves through the fixed mesh, this process of creation a new mesh and removing it is repeated in each time step.
In the structure block described in the Users Manual Section 3.2.3, the following keywords are available for the AAFM:

`MAKE_OBSTACLE_MESH`: Creates an adapted mesh.
`REMOVE_OBSTACLE_MESH`: Removes an adapted mesh.
`MOVE_OBSTACLE`: Moves an obstacle.
If an adapted mesh is made, also all previous solutions are interpolated to that mesh. After removing the mesh, all new solutions are copied back to the old mesh.

Pseudo concentration method A description is not yet available.

Definition of type numbers

At this moment the following general type number for the fictitious elements are available.

921 Standard type number for fictitious elements.

923 Standard type number for fictitious elements with only one degree of freedom per point. Hence this element is meant to be used for scalar equations.

7.6 Stationary free surface flows

Free surface problems are those problems where not only the flow problem must be solved, but also the position of some of the boundaries is unknown. This means that such boundary must be detected during the computations. Usually the number of boundary conditions on the free surface is one more than required to solve the differential equations for the flow problem. In that case the free boundary is defined implicit;y by this extra boundary condition.
In this section we shall restrict ourselves to stationary free boundary problems only. Instationary free boundary problems are the subject of Section 7.7.
In general one may say that the solution of a stationary free surface problem consists of the following steps:

Estimate the free boundary. This defines the initial region.
while not converged **do**
Solve the flow problem on the present region using all but one of the boundary conditions on the free boundary.
Adapt the free boundary using the boundary condition that is left.
Adapt the mesh by adapting the coordinates or if necessary by remeshing.
end while

At this moment the following methods to adapt the free boundary are available

1. Iterative adaptation of the free boundary using one of the boundary conditions
   The extra boundary condition is immediately applied to compute the free surface.

2. Implicit update of the free boundary
   The position of the free boundary is used as an extra unknown. In this way the solution of the free boundary problem is made implicit.

3. Approximating the free boundary by a convection problem
   The free boundary is considered to be a streamline. The position of the stream line is solved by solving a simple convection equation.

Below we give a short description of the various methods; examples are treated in the next sections.

Iterative adaptation of the free boundary using one of the boundary conditions In a 2d case we need two velocity related boundary conditions to solve the incompressible (Navier-)Stokes equations. So on the free surface we need in that case three boundary conditions. A common set of boundary conditions for a stationary free boundary problem in R^2 is:

$$\mathbf{u} \cdot \mathbf{n} = 0; \quad \sigma^{nn} = -p_0; \quad \sigma^{nt} = 0 \tag{7.6.1}$$

The first one expresses that there is no flow through the free boundary, the second one that the normal stress is equal to the pressure outside the flow (usually 0) and the last one that the shear stress is equal to 0.
In order to solve the problem one usually takes 2 of the 3 boundary conditions to solve the flow problem and the third one to update the free boundary. Intuitively the most simple combination is to solve the Navier-Stokes equations using the stress boundary conditions and to update the free boundary using the condition $u_n = 0$. However, depending on the surface tension on the free boundary, a better convergence might be achieved by prescribing the shear stress and normal velocity and using the normal stress to update the free boundary.
Methods that use the no-flow boundary condition are called kinematic conditions, methods that utilize the normal stress conditions to update the boundary will be referred to as normal stress balance.
In order to explain these method we assume that we have a free boundary in R^2, which can be written as $y = h(x)$, and that the velocity has two components u and v.
Examples of kinematic methods are

- the streamline variant.
  Since the surface is a stream line we have $\frac{dh}{dx} = \frac{v}{u}$. This first order differential equation is solved along the actual free boundary, starting with a fixed point. In fact this method can be generalized to more dimensions using the convection variant treated under part 3.
- the film variant.
  This method, introduced by Caswell and Viriyayuthakorn (1983), is based on conservation of mass. The difference between old and new free surface is given by a thin film with uniform velocity. Formulas are given in the Users Manual Section 3.4.4.
- other methods like the method described by Silliman and Scriven (1980) or the optimal control method (See for example Cuvelier (1980)) are not yet implemented in SEPRAN.

Also normal stress balance methods have not been implemented in SEPRAN.

Implicit update of the free boundary In this case the height $h(x)$ is introduced as an extra unknown on the free surface. So on the free boundary we have three unknowns (u, v and h). The Navier-Stokes equations together with the boundary conditions on the free surface and the position of the free surface are linearized by a Newton linearization. This results in a larger system of equations due to the extra unknowns but in a faster convergence. In our case the total linearization method of Kruyt et al (1988) has been implemented. It requires an extra boundary element along the free boundary.

Approximating the free boundary by a convection problem This method is based on replacing the equation for the stream line by a convection equation of the shape:

$$u\frac{dh}{dx} = v \tag{7.6.2}$$

This equation is integrated along the free boundary. So in each iteration step we start by solving the Navier-Stokes equations with the stress boundary conditions along the free boundary. Then we solve the convection equation and finally we update the free boundary. It is clear that this method can be applied both in R^2 as well as in R^3.

Definition of type numbers

At this moment the following general type numbers for the total linearization method are available.

915 This element may only be used in combination with quadratic Crouzeix Raviart extended triangles for the Navier-Stokes equations.
It requires the following input for the 10 coefficients (the first 5 are integer the second 5 are real):

1. not yet in use, must be 0
2. not yet in use, must be 0
3. not yet in use, must be 0
4. not yet in use, must be 0
5. not yet in use, must be 0
6. density ρ
7. viscosity η
8. surface tension coefficient γ
9. in case of line elements: the pressure at the previous iteration.
   In case of point elements: first component of tangential vector t_1
10. only for point elements: second component of tangential vector t_2

Type 915 may be used both for the line elements as well as for point elements in the end of the free surface boundary to prescribe the tangential vector.

7.7 Instationary free surface flows

This section is under construction

8 Second order elliptic and parabolic equations using spectral elements

Spectral elements
The spectral element method combines the rapid convergence rate of the plain spectral methods with the generality of the finite element techniques. Spectral methods provide highly accurate solutions to partial differential equations governing complex physical phenomena, but they have been limited to idealized research problems due to their lack of geometric flexibility. The spectral element method was developed to increase the geometrical flexibility of high order spectral methods. More information on the spectral element method can be found in Canuto et al (1988) and in van de Vosse and Minev (1996).

In this chapter we consider elliptic and parabolic equations of second order that are solved by spectral elements.
The following Sections are available:

8.1 Second order real linear elliptic and parabolic equations with one degree of freedom

Equation

In this section we consider equations of the following form :

$$\rho c_p(\frac{\partial c}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{c}) - \ \mathrm{div}(\alpha\nabla\mathbf{c}) + \beta\mathbf{c} = \mathbf{f} \tag{8.1.1}$$

i.e.

$$\rho c_p \frac{\partial c}{\partial t} + \rho c_p \sum_{i=1}^{n} u_i \frac{\partial c}{\partial x_i} - \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial}{\partial x_i}\left(\alpha_{ij}\frac{\partial c}{\partial x_j}\right) + \beta c = f, \tag{8.1.2}$$

where $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}) \in \mathbf{\Omega} \subset \mathbf{R^n}$.

In stationary case ($\frac{\partial c}{\partial t} = 0$) (8.1.1) reduces to

$$\mathbf{u}\cdot\nabla\mathbf{c} - \ \mathrm{div}\ (\alpha\nabla\mathbf{c}) + \beta\mathbf{c} = \mathbf{f}. \tag{8.1.3}$$

In this case the equation is elliptic, otherwise it is parabolic.
The coefficients $\rho c_p, \mathbf{u}, \alpha, \beta$ and $\mathbf{f}$ may depend on space and time and also of solutions of other problems. The following restrictions are required :

- $\rho c_p > 0$.
- The matrix α with coefficients α_{ij} must be positive definite symmetric matrix. In the extreme case α may be equal to zero in which case the equation is of hyperbolic type.

Remark
If the matrix α_{ij} is non-symmetric it may be split into a symmetric and and an anti-symmetric part. Using the chain rule of differentiations it is easy to see that the anti-symmetric part yields only first order derivatives of c.

Typical examples of equations of the form (8.1.1) are given in Section 3.1.

Upwinding

The combination of upwinding and spectral elements is not yet available.

Boundary and initial conditions

In the in-stationary case it is necessary to give an initial condition at $t = 0$.
The following type of boundary conditions are available :
Type 1 (Dirichlet boundary condition) $c(\mathbf{x})$ given on some part of the boundary.
Type 2 (Neumann or mixed boundary condition)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{ij}\frac{\partial c}{\partial x_j} n_i + \sigma(x)c(x) = h(x) \qquad (\sigma(x) \geq 0) \tag{8.1.4}$$

on some part of the boundary. this is a so-called natural boundary condition. In general, boundary elements are necessary , except in the case that $\sigma(x) = 0$ and $h(x) = 0$, when there is no need to give any condition on this part of the boundary. For $\sigma = 0$ this is a Neumann boundary condition, otherwise it is a mixed boundary condition.

Also other special type of natural boundary conditions are possible in combination with (8.1.1) (see Section 3.1). However they have not been implemented yet (contact SEPRA).

Input for the various subroutines

- Definition of the storage scheme :

  The first thing to be chosen is the type of storage scheme for the matrices. the storage scheme is defined by the keyword METHOD = i in the input block "MATRIX" of the program SEPCOMP.

  In general, two matrices may be created : the mass matrix and the stiffness matrix. The mass matrix is only used for time-dependent problems. This matrix, diagonal for spectral elements, pre-multiplies the discretized time-derivative. The stiffness matrix represents the discretization of the stationary terms in the left hand side of equation (8.1.1).

  In the case of a symmetrical stiffness matrix METHOD = 1 or 5 may be chosen, otherwise METHOD should be equal to 2 or 6. The choice between 1,2 or 5,6 depends on the choice of solution method. A direct method requires METHOD = 1 or 2, an iterative method, using a compact storage of the matrix, 5 or 6.

- Definition of the coefficients for the differential equation : The coefficients for the differential equation, using spectral elements (ITYPE = 600), are defined similar to the coefficients for finite elements (ITYPE = 800).
  For each elements group 20 parameters and coefficients must be given. the first 5 parameters are of integer type, which means that they must be defined by ICOEFi in the input, the last 15 are real coefficients.

  The parameters and coefficients are defined as follows:

1	not yet used
2	type of upwinding and convective terms
3	type of numerical integration
4	type of co-ordinate system
5	not yet used
6	α_{11}
7	α_{12}
8	α_{13}
9	α_{22}
10	α_{23}
11	α_{33}
12	u_1
13	u_2
14	u_3
15	β
16	f
17	ρc_p
18-20	not yet used

  Parameters that are not yet used must be set equal to zero. they are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
  The coefficients 6-20 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors. Of course, in 1D and 2D not all coefficients are used.
  The default values for the real coefficients 6 to 16 are zero, for coefficient 17 (ρc_p), however, the default is one.

  With respect to the parameters 1-5 the following choices are available:

2 Type of upwinding and convective terms.
At this moment only the default value 0 is allowed. This means no upwinding and the convective term in equation (8.1.1) has the form $(\mathbf{u} \cdot \nabla)\mathbf{c}$.

3 Type of numerical integration rules.
At this moments only the default value 0 is allowed. This means a Gauss-Lobatto-Legendre rule is used.

4 Type of co-ordinate system.
At this moment only Cartesian co-ordinates may be used (default value 0).

- Definition of the coefficients for boundary conditions:

The coefficients for the boundary conditions may be defined by one of the methods described in Section 10.1, where, in general the method by SEPCOMP is recommended. For each element group 15 parameters and coefficients must be given. The first 5 parameters are of integer type which means that they must be defined by ICOEFi in the input, the last 10 are real coefficients.

These parameters and coefficients are defined as follows:

1 Type of natural boundary condition
2 Not yet used (must be zero)
3 type of numerical integration
4 type of co-ordinate system
5 not yet used

6 σ
7 h
8 α_{11}
9 α_{12}
10 α_{13}
11 α_{22}
12 α_{23}
13 α_{33}
14-15 not yet used

Parameters that are not used must be set equal to zero. They are meant for future extensions. In the input for SEPCOMP this means that no information about these coefficients has to be given.
The coefficient 6-15 may be zero, constants or functions as described in Section 10.1. They may also depend on pre-computed vectors. Of course, in 1D and 3D not all coefficients are used. The coefficient 8-13 are for future use for boundary conditions of type 4 (see Section 3.1, element 800).

With respect to the parameters 1-5 the following choices are available:
For the type of natural boundary condition possible values are 0,2. For type of numerical integration rule only the default 0 is possible (Gauss-Lobatto-Legendre). Only Cartesian co-ordinates are available for spectral elements.

- Parameters for subroutine BUILD :
With respect to subroutine BUILD the parameter IMAS (IINBLD(4)) is of importance. Note that the combination of the spectral basis functions the Gauss-Lobatto integration rule leads to a diagonal mass matrix (see also van de Vosse and Minev (1996)).

- Parameters with respect to linear solver :
  In the case $\mathbf{u} = \mathbf{0}$ and $\beta \geq 0$ the stiffness matrix is not only symmetric but also positive definite.

- Computation of derivatives:

  The parameter ICHELD in the input block "DERIVATIVES" is used to distinguish the various possibilities:

  ICHELD = 1 $\frac{\partial c}{\partial x_i}$, where x_i is defined by the parameter ix.
  ICHELD = 2 ∇c
  ICHELD = 3 -∇c
  ICHELD = 4 not yet defined
  ICHELD = 5 $\left(\frac{\partial c}{\partial y}, -\frac{\partial c}{\partial x}\right)$ 2D only
  ICHELD = 6-10 not yet defined
  ICHELD = 11-24 not yet defined
  ICHELD = 25 Compute residual **Su**
  ICHELD = 26-29 not yet defined
  ICHELD = 30 $(\nabla c, \psi)$ computed weak,$ix = 1, 2$ (only 3D) $, 3$
  ICHELD = 31 $(\nabla \cdot c, \psi)$ computed weak
  ICHELD = 32 $(\nabla\nabla \cdot c, \psi)$ $ix = 1, 2$ (only 3D) $, 3$
  ICHELD = 33 Compute convection $\mathbf{u} \cdot \nabla \mathbf{c}$
  ICHELD = 34 Compute non-constant viscosity term $\nabla \cdot (\nu \nabla c + (\nabla c)^T)$

  ICHELD = 35-39 not yet defined
  ICHELD = 40 Interpolation routines (only for odd number of side-points)

  The output vector is defined as follows :
  1,11-40: a vector of the type solution vector with one unknown per point
  2,3,5: a vector of the type vector special structure with *ndim* unknowns per point

- Types of integrals that may be computed :
  The parameter ICHELI in the block "INTEGRALS" is used to distinguish the various possibilities:

  ICHELI=1 $\int\limits_{\Omega} f(x) d\Omega$
  ICHELI=2 $\int\limits_{\Omega} f(x) c(x) d\Omega$
  ICHELI=2+i $\int\limits_{\Omega} f(x) \frac{\partial c}{\partial x_i} d\Omega$ (i= 1,2,3)

- **Creation of spectral mesh**

  To define a spectral mesh two lines have to added to the mesh input file:

```
intermediate points
   sidepoints = 4, subdivision = legendre, midpoints = filled
```

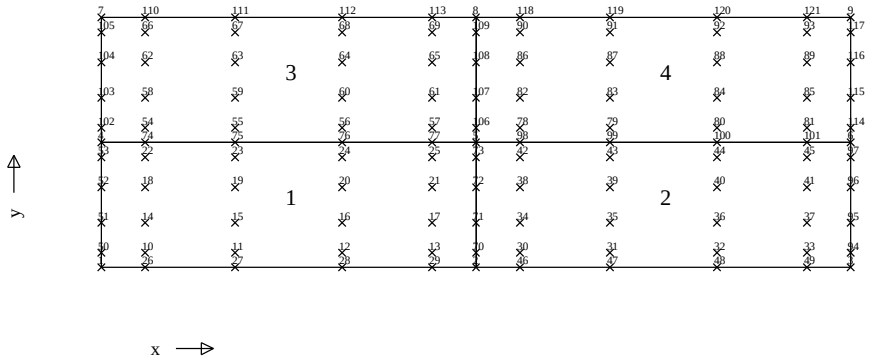
  By changing the number of sidepoints the order of the spectral method is altered. The order of the method is always one more than the number of sidepoints. At this moment the number of sidepoints can be chosen between 1 and 23.
  The option sidepoints = 0 has not been implemented yet.

Example

Consider the following mesh input-file :

```
mesh2d
  points
    p1=(-1.0d0, 0.0d0)
    p2=( 0.5d0, 0.0d0)
    p3=( 0.5d0, 0.5d0)
    p4=(-1.0d0, 0.5d0)
  curves
    c1=line1(p1,p2,nelm=2)
    c2=line1(p2,p3,nelm=2)
    c3=line1(p3,p4,nelm=2)
    c4=line1(p4,p1,nelm=2)
  surfaces
    s1=rectangle5(c1,c2,c3,c4)
  intermediate points
    sidepoints=4,subdivision=legendre,midpoints=filled
  plot(jmark=3, numsub=1)
  norenumber
end
```

Program sepmesh in combination with this input file creates the four spectral elements of 5th order that are shown in Figure 8.1



MESH

Definition of type numbers
The type numbers, which are given in the input block "PROBLEM" for SEPCOMP define the type of differential equation to be solved.

For the second order elliptic equation the following type numbers are available:

600 general type number for the internal elements. Defines the differential equation.
This type number is available for the following elements shape numbers :

shape = 1 linear line element.

shape = 5 bilinear quadrilateral.

shape = 13 trilinear hexahedron.

General Remark

In general any problem using elements 800 and 801 can be solved by spectral elements by just changing the problems numbers into 600 and 601 respectively. Further, the mesh input-file should be modified (addition of section intermediate points). Note however, that some options of elements 800 and 801 are not yet implemented in the spectral elements. Please contact SEPRA if you need additional options.

9 Fourth order elliptic and parabolic equations

In general fourth order equations require special elements that allow continuity of the first order derivatives of the basis functions. A disadvantage of such an approach is that high degree polynomials are required, which are difficult to construct.
An alternative is to split the unknown in a set of two unknowns each satisfying a second order equation. Such an option is not always possible. For example the boundary conditions must be formulated in terms of these new unknowns. This can be done only for special combinations of boundary conditions. In this chapter we consider such a case:

9.1 The Cahn-Hilliard equation

The Cahn-Hilliard equation is used to model pattern formation, or spinodal decomposition. The equation is important where phase transitions occur. Models based on Cahn-Hilliard equations are applicable to the following phenomena that occur during phase transitions within the domain of computation Ω:

- nucleation of particles or secondary phases (this is the formation of particles);
- growth and dissolution of the particles or secondary phases;
- merging of particles or secondary phases.

In Cahn-Hilliard type models, the secondary phase (or the particles) is usually identified by by the portion of the computational domain where a certain threshold is exceeded, that is

$$\Omega_p = \{\mathbf{x} \in \Omega : c(t, \mathbf{x}) \geq \hat{c}\}, \tag{9.1.1}$$

in which $c = c(t, \mathbf{x})$ denotes the solution and $\mathbf{x}$ and t, respectively denote the spatial position and time.

Equations

Let c be the volume fraction of a phase in a binary system, *i.e.* a system that consists of two species only, then, the total Ginzburg-Landau free energy of the system is given by

$$F(c) = \int_\Omega \left\{ f(c) + \frac{\kappa}{2}|\nabla c|^2 \right\} dV, \tag{9.1.2}$$

where $\kappa \geq 0$ denotes the gradient energy coefficient. Further, $f(c)$ is the bulk free energy, which can be obtained from thermodynamic databases. A typical form is the following

$$f(c) = RT \left(\frac{c \ln(c)}{N_1} + \frac{(1-c) \ln(1-c)}{N_2} \right) + \omega c(1-c). \tag{9.1.3}$$

Here ω denotes the interaction parameter. An example of a bulk free energy of the above form is shown in Figure 9.1.1. Here N_1 and N_2 are related to the molecular size. The values c_L and c_R are known as the binodal points, determined by the common tangent construction, as shown in Figure 9.1.1. The second term in equation (9.1.2) is crucial in the interfacial region, where the gradient of c is large. Therefore, the second term is also refer to as the interfacial energy. The diffusive flux, J, is postulated to be proportional to the gradient of the chemical potential, hence

$$J = -M\nabla\mu(c) = -M\nabla\frac{\delta F(c)}{\delta c}. \tag{9.1.4}$$

Here the derivative $\frac{\delta F(c)}{\delta c}$ is given by

$$\frac{\delta F}{\delta c} = f'(c) - \kappa\Delta c. \tag{9.1.5}$$

From now on, we will assume M to be a constant. Substitution of equation (9.1.5) into (9.1.4) and use of the mass-balance,

$$\frac{\partial c}{\partial t} = -\nabla \cdot J, \tag{9.1.6}$$

gives the Cahn-Hilliard equation on Ω

$$\frac{\partial c}{\partial t} = \nabla \cdot \{M\left[f''(c)\nabla c - \kappa\nabla\Delta c\right]\}. \tag{9.1.7}$$

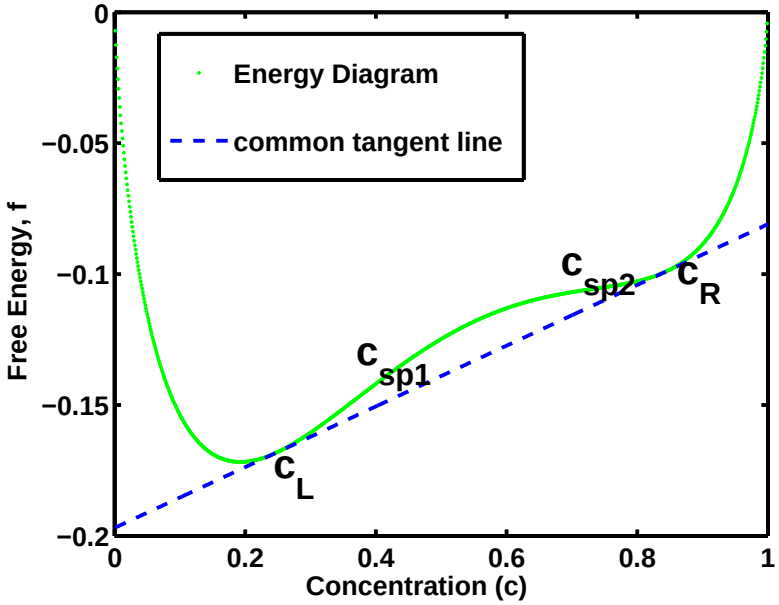


Figure 9.1.1: A plot of a Gibbs free energy function and the determination of the binodal concentrations.

In the above equation the second derivative $Mf''(c)$ acts like a diffusion coefficient. The fourth order term, with κ, in the Cahn-Hilliard equation is also considered as a stabilization term, for the case where $f''(c) < 0$. This equation has been applied to model phase segregation and spinodal decomposition in numerous studies. In the case of phase-separation and spinodal decomposition, the second derivative of f with respect to c becomes negative in the interface part of the domain. Physically, the negative values of the second derivative of f give rise to 'uphill diffusion', which is diffusion from low concentration areas to high concentration regions. As boundary conditions we use symmetry conditions, *i.e.*

$$\frac{\partial c}{\partial n} = \frac{\partial(\Delta c)}{\partial n} = 0, \text{ on } \partial\Omega. \tag{9.1.8}$$

These two boundary conditions are necessary due to the fourth order spatial partial derivative that occur in the Cahn-Hilliard equation. In many other studies, periodic boundary conditions are used instead. Further, we have an initial condition for the concentration c:

$$c = c_0, \text{ for } t = 0. \tag{9.1.9}$$

It should be realized that $\sqrt{\kappa}$ is a measure of the interface thickness. Furthermore, one can demonstrate that solutions to the Cahn-Hilliard equation satisfy the following fundamental properties:

- Solutions to the Cahn-Hilliard equation are mass conserving. that is $\frac{d}{dt}\int_\Omega c d\Omega = 0$;
- The total energy is non-increasing, that is $\frac{dF(c)}{dt} \leq 0$.

Solution method

In our approach, we follow the work due to Elliott *et al* and the more recent work due to Ceniceros & Roma in which we use reduction of order for the spatial derivatives. This approach is suitable

if one works with symmetry boundary conditions. Therewith we introduce a function u such that $u = \Delta v$, then this gives the following system for the binary case:

$$\begin{aligned} \Delta c &= u, \\ \frac{\partial c}{\partial t} &= \nabla \cdot \{M\left[f''(c)\nabla c - \kappa \nabla u\right]\}. \end{aligned} \tag{9.1.10}$$

The above system to be solved is allowable because of the nature of the boundary conditions, which turn into

$$\frac{\partial c}{\partial n} = 0 = \frac{\partial u}{\partial n}, \text{ for } \mathbf{x} \in \partial \mathbf{\Omega}. \tag{9.1.11}$$

These equations can be solved by the method described in Section 3.7.
An example can be found in the manual SEPRAN Examples.

10 coefficients in case of own programs

10.1 How to provide coefficients and parameters

In general all standard problems described in this manual require information concerning coefficients and parameters. These parameters may be parameters with respect to the equation, but also parameters concerning the solution method. Essentially, we distinguish between two types of parameters, parameters which may have only an integer value (usually parameters influencing the solution process) and parameters which may have both real and integer values (coefficients). In each standard problem parameters and coefficients get a sequence number. This sequence number is essential for the recognition of these parameters.

The user may provide parameters and coefficients in several ways. The most simple way is provided by or program SEPCOMP (which in fact calls subroutine FILCOF). A less simple way is providing coefficients and parameters through the subroutines FIL100-FIL104 or FIL150-FIL154. The advantage of these subroutines is that no input from the standard input file is required. Both the subroutines FILCOF and FIL100-FIL104 fill information of the parameters and coefficients into the arrays IUSER and USER. These arrays are used by subroutines like BUILD, DERIV and INTEGR. The most complicated but also most general way of providing coefficients is by filling the arrays IUSER and USER directly.
We shall now consider these three possibilities in more detail.

Method 1 simple input provided by the user through the standard input file.

If program SEPCOMP is used, or if the subroutines LINSOL(LINSTM) or FILCOF are used the user must provide information about the coefficients and parameters through the standard input file.
Program SEPCOMP is described in the manual SEPRAN INTRODUCTION, the description of FILCOF can be found in the SEPRAN PROGRAMMERS GUIDE.

The syntax of the input in the input file is as follows (see also SEPRAN INTRODUCTION and USERS MANUAL):

```
[COMPLEX] COEFFICIENTS
    ELGRP1 (NPARM = n_1)  (information of coefficients for element group 1)
    ELGRP2 (NPARM = n_2)  (information of coefficients for element group 2)
          .
          .
          .
    ELGRP nelgrp (NPARM = n_nelgrp)  (information of coefficients for element group nelgrp)
    BNGRP1 (NPARM = m_1)  (information of coefficients for boundary element group 1)
          .
          .
          .
END
```

n_i denotes the number of parameters (coefficients) for element group i and m_i the number of parameters for boundary element group i.

The syntax for the information of coefficients is as follows:

```
   ICOEF_1 = i
   ICOEF_2 = j
   COEF_3 = description_1
   COEF_4 = description_2
```

.
.
.

where *description* may have one of the following shapes:

```
VALUE = v
v
FUNC = f
POINTS, IREF = r_1
ELEMENTS, IREF = r_2
COEF_j
OLD_SOLUTION j [,DEGREE_OF_FREEDOM d],[COEF = desc]
```

where *desc* may have one of the following shapes:

```
VALUE = v
v
FUNC = f
```

For the meaning of these parameters the reader is referred to the Users Manual (3.2.6).

Method 2 simple input provided by the user directly in the main program.

An alternative for the filling of coefficients by FILCOF which requires input from the standard input is to fill information in the arrays IUSER and USER directly through the subroutine FIL1.. . The following subroutines are available:

FIL100 (10.2.1) Simple filling of coefficients for real differential equations.

FIL101 (10.2.2) Simple filling of coefficients for natural boundary conditions corresponding to real differential equations.

FIL103 (10.2.5) Extended filling of coefficients for real differential equations.

FIL104 (10.2.6) Extended filling of coefficients for natural boundary conditions corresponding to real differential equations.

FIL150 (10.2.3) Simple filling of coefficients for complex differential equations.

FIL151 (10.2.4) Simple filling of coefficients for natural boundary conditions corresponding to complex differential equations.

FIL153 (10.2.7) Extended filling of coefficients for complex differential equations.

FIL154 (10.2.8) Extended filling of coefficients for natural boundary conditions corresponding to complex differential equations.

For a description of these subroutines see Section 10.2.

Method 3 directly filling of the arrays IUSER and USER by the user.

This is far most the most complicate way of filling coefficients and parameters. For a description of this method see Section 10.3.

10.2 Subroutines FIL...

10.2.1 Subroutine FIL100

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. Besides with subroutine FILCOF this may also be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL100 may not be used for boundary elements.

Call

```
CALL FIL100 ( IELGRP, IUSER, USER, KPROB, IPARM, IWORK, WORK )
```

Parameters

INTEGER IELGRP, IUSER(∗), KPROB(∗), IPARM, IWORK(IPARM)

DOUBLE PRECISION USER(∗), WORK(IPARM)

IELGRP Standard element number ($1 \leq IELGRP \leq NELGRP$).
The calls of subroutines FIL... must be done in a natural sequence, hence first a call for IELGRP = 1, then for IELGRP = 2 etc.

IUSER Integer array in which information concerning the coefficients is stored.
Variable length array.

USER Real array in which information concerning the coefficients is stored.
Variable length array. In general 100 positions are sufficient.
The arrays IUSER and USER are used as input for subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

IPARM In this position the number of parameters for the differential equation must be stored.
This parameter is given for each standard problem.

IWORK,WORK User arrays of length IPARM. In these arrays information concerning the coefficients must be stored in the sequence as indicated at the description of the standard problems.
When IWORK(i) = 0 then the i^{th} parameter is a constant, whose value must be stored in WORK(i).
When IWORK(i) > 0 then the i^{th} parameter is a function.
When IWORK(i) < 1000 the user written function subroutine FUNCCF is used to compute the value of this parameter in a point. For a description of FUNCCF see 10.4. The value of IWORK(i) is used as the parameter ICHOIS in FUNCCF
When IWORK(i) > 1000 **and** IWORK(i) < 2000 then the user written function subroutine FUNCC1 is used to compute the value of this parameter. For a description of FUNCC1 see 10.4. The value of IWORK(i) - 1000 is used as the parameter ICHOIS in FUNCC1.
IWORK(i) > 2000 is not allowed.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

Input

The user must give IELGRP and IPARM a value.
When IELGRP $>$ 1 the arrays IUSER and USER must have been filled before by IELGRP - 1 preceding calls of subroutines FIL... .
The arrays KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.2 Subroutine FIL101

Description

The specification of the coefficients for natural boundary conditions is stored in the arrays IUSER and USER.
Subroutine FIL101 must be called for each standard boundary element separately. Hence at most NUMNATBND calls of FIL101 are necessary before the call of BUILD.

Call

```
CALL FIL101 ( IBNGRP, IUSER, USER, KPROB, IPARM, IWORK, WORK )
```

Parameters

INTEGER IBNGRP, IUSER(∗), KPROB(∗), IPARM, IWORK(IPARM)

DOUBLE PRECISION USER(∗), WORK(IPARM)

IBNGRP Standard boundary element number ($1 \leq IBNGRP \leq NUMNATBND$).
The calls of subroutines FIL... must be done in a natural sequence, first for all internal elements (i.e. corresponding to the differential equation) and then for the boundary element with boundary element number IBNGRP = 1, then for IBNGRP = 2 etc.

IUSER,USER See subroutine FIL100 (10.2.1).

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

IPARM In this position the number of parameters for the natural boundary conditions must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays of length IPARM. In these arrays information concerning the coefficients must be stored in the sequence as indicated at the description of the standard problems.
When IWORK(i) = 0 then the i^{th} parameter is a constant, whose value must be stored in WORK(i).
When IWORK(i) > 0 then the i^{th} parameter is a function.
When IWORK(i) < 1000 the user written function subroutine FUNCCF is used to compute the value of this parameter in a point. For a description of FUNCCF see 10.4. The value of IWORK(i) is used as the parameter ICHOIS in FUNCCF
When IWORK(i) > 1000 **and** IWORK(i) < 2000 then the user written function subroutine FUNCC1 is used to compute the value of this parameter. For a description of FUNCC1 see 10.4. The value of IWORK(i) - 1000 is used as the parameter ICHOIS in FUNCC1.
IWORK(i) > 2000 is not allowed.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

Input

The user must give IBNGRP and IPARM a value.
The arrays IUSER and USER must be filled by NELGRP + IBNGRP - 1 preceding calls of subroutines FIL...
Array KPROB must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.3 Subroutine FIL150

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. In the case of the simple input (see 10.1), this may be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL150 must be used for complex coefficients and may not be used for boundary elements.

Call

```
CALL FIL150 ( IELGRP, IUSER, USER, KPROB, IPARM, IWORK, WORK )
```

Parameters

INTEGER IELGRP, IUSER(∗), KPROB(∗), IPARM, IWORK(IPARM)

DOUBLE PRECISION USER(∗)

COMPLEX ∗16 WORK(∗)

IELGRP Standard element number ($1 \leq IELGRP \leq NELGRP$).
The calls of subroutines FIL... must be done in a natural sequence, hence first a call for IELGRP = 1, then for IELGRP = 2 etc.

IUSER Integer array in which information concerning the coefficients is stored.
Variable length array.

USER Real array in which information concerning the coefficients is stored.
Variable length array. In general 100 positions are sufficient.
The arrays IUSER and USER are used as input for subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

IPARM In this position the number of parameters for the differential equation must be stored.
This parameter is given for each standard problem.

IWORK,WORK User arrays of length IPARM. In these arrays information concerning the coefficients must be stored in the sequence as indicated at the description of the standard problems.
Array WORK must be a complex array.
When IWORK(i) = 0 then the i^{th} parameter is a constant, whose value must be stored in WORK(i).
When IWORK(i) > 0 then the i^{th} parameter is a function.
When IWORK(i) < 1000 the user written function subroutine CFUNCF is used to compute the value of this parameter in a point. For a description of CFUNCF see 10.4. The value of IWORK(i) is used as the parameter ICHOIS in CFUNCF
When IWORK(i) > 1000 **and** IWORK(i) < 2000 then the user written function subroutine CFUNC1 is used to compute the value of this parameter. For a description of CFUNC1 see 10.4. The value of IWORK(i) - 1000 is used as the parameter ICHOIS in CFUNC1.
IWORK(i) > 2000 is not allowed.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

Input

The user must give IELGRP and IPARM a value.

When IELGRP $>$ 1 the arrays IUSER and USER must be filled before by IELGRP - 1 preceding calls of subroutines FIL... .
The arrays KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.
Each complex quantity in WORK takes two positions in array USER.

10.2.4 Subroutine FIL151

Description

The specification of the coefficients for natural boundary conditions is stored in the arrays IUSER and USER.
Subroutine FIL151 must be called in the case of complex coefficients for each standard boundary element separately. Hence at most NUMNATBND calls of FIL151 are necessary before the call of BUILD.

Call

```
CALL FIL151 ( IBNGRP, IUSER, USER, KPROB, IPARM, IWORK, WORK )
```

Parameters

INTEGER IBNGRP, IUSER(∗), KPROB(∗), IPARM, IWORK(IPARM)

DOUBLE PRECISION USER(∗)

COMPLEX ∗16 WORK(∗)

IBNGRP Standard boundary element number ($1 \leq IBNGRP \leq NUMNATBND$).
The calls of subroutines FIL... must be done in a natural sequence, first for all internal elements (i.e. corresponding to the differential equation) and then for the boundary element with boundary element number IBNGRP = 1, then for IBNGRP = 2 etc.

IUSER,USER See subroutine FIL150 (10.2.3).

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

IPARM In this position the number of parameters for the natural boundary conditions must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays of length IPARM. In these arrays information concerning the coefficients must be stored in the sequence as indicated at the description of the standard problems.
Array WORK must be a complex array.
When IWORK(i) = 0 then the i^{th} parameter is a constant, whose value must be stored in WORK(i).
When IWORK(i) > 0 then the i^{th} parameter is a function.
When IWORK(i) < 1000 the user written function subroutine CFUNCF is used to compute the value of this parameter in a point. For a description of CFUNCF see 10.4. The value of IWORK(i) is used as the parameter ICHOIS in CFUNCF
When IWORK(i) > 1000 **and** IWORK(i) < 2000 then the user written function subroutine CFUNC1 is used to compute the value of this parameter. For a description of CFUNC1 see 10.4. The value of IWORK(i) - 1000 is used as the parameter ICHOIS in CFUNC1.
IWORK(i) > 2000 is not allowed.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

Input

The user must give IBNGRP and IPARM a value.
The arrays IUSER and USER must be filled by NELGRP + IBNGRP - 1 preceding calls of subroutines FIL...
Array KPROB must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.5 Subroutine FIL103

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. In the case of the simple input (see 10.1), this may be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL103 may not be used for boundary elements.
Subroutine FIL103 differs from subroutine FIL100 in the sense that it permits the reading of nodal values or element values of the coefficients from a file.

Call

```
CALL FIL103 (IPROB, IELGRP, IUSER, USER, KPROB, NPARM, IWORK, WORK, KMESH)
```

Parameters

INTEGER IPROB, IELGRP, IUSER(∗), KPROB(∗), NPARM, IWORK(1:4, 1:NPARM), KMESH(∗)

DOUBLE PRECISION USER(∗), WORK(NPARM)

IPROB Problem number. With IPROB the user indicates to which problem the arrays IUSER and USER correspond. Usually IPROB = 1, however, if different problems are to be solved on the same mesh, IPROB may have a value larger than zero. Of course that problem must have been introduced in subroutine SEPSTR or PROBDF.

IELGRP Standard element number ($1 \leq IELGRP \leq NELGRP$).
The calls of subroutines FIL... must be done in a natural sequence, hence first a call for IELGRP = 1, then for IELGRP = 2 etc.

IUSER Integer user array of variable length that is filled with integer information concerning the coefficients of the standard problems. Since IUSER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications.

USER Double precision user array of variable length that is filled with real information concerning the coefficients of the standard problems. Since USER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications. However, if the user uses the input option with iref (See input), then the length needed will be considerably longer, since information for each nodal point or each element must be stored.
The arrays IUSER and USER are used as input for the subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

NPARM In this position the number of parameters for the differential equation must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays in which the user must store some information concerning the coefficients in the sequence as given in the description of the standard problems.
IWORK is an integer two-dimensional array with dimension:
IWORK(1:4,1:NPARM)
and WORK a double precision one-dimensional array with dimension:
WORK(1:NPARM).
When IWORK(1,i) = 0, the i^{th} parameter is a constant, whose value must be stored in

WORK(i).

When IWORK(1,i) > 0, the i^{th} parameter is a function.

When IWORK(1,i) < 1000, the user written function subroutine FUNCCF is used to compute the value of this parameter in a point. For a description of FUNCCF see 10.4. The value of IWORK(1,i) is used as the parameter ICHOIS in FUNCCF

When IWORK(1,i) > 1000 **and** IWORK(1,i) < 2000, the user written function subroutine FUNCC1 is used to compute the value of this parameter. For a description of FUNCC1 see 10.4. The value of IWORK(1,i) - 1000 is used as the parameter ICHOIS in FUNCC1.

When IWORK(1,i) = -j <0: In this case the coefficient number i is identical to coefficient number j (j<i).

When IWORK(1,i) = 2001, the coefficient is given by the user in each nodal point.
Define IREF = IWORK(2,i). If IREF > 0 then the NPOINT values are read from file IREF (free format). Exactly NPOINT values must be given. All these coefficients are stored in array USER, hence this array must have sufficient length. If IREF < 0 then the user written subroutine FUNCFL (see Programmers Guide 5.11.1) must be used to fill array USER in these NPOINT points. The parameter ICHOIS in the call of FUNCFL is equal to -IREF at the call of FUNCFL.
All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2002, then the coefficient is given by the user in each element of element group IELGRP. The NELEM(IELGRP) (= number of elements in element group IELGRP) values are read from file IREF (free format), where IREF is equal to IWORK(2,i). Exactly NELEM(IELGRP) values must be given. All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2003, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i).

When IWORK(1,i) = 2004, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD multiplied by a function or a constant. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i). IWORK(4,i) must contain information of the multiplication factor. If IWORK(4,i) = 0, then the multiplication factor is a constant whose value must be stored in WORK(i), when IWORK(4,i) > 0 the user written function subroutine FUNCCF is used to compute the value of this multiplication factor in a point. For a description of FUNCCF see 10.4. The value of IWORK(4,i) is used as the parameter ICHOIS in FUNCCF. Only values of IWORK(4,i) < 1000 are permitted. When IWORK(1,i) > 10000, the user written function subroutine FUNCC3 is used to compute the value of this parameter. For a description of FUNCC3 see 10.4. The value of IWORK(1,i) - 10000 is used as the parameter ICHOIS in FUNCC3
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

KMESH Output of the mesh generator

Input

The user must give IPROB, IELGRP and NPARM a value.
When IELGRP > 1 the arrays IUSER and USER must be filled before by IELGRP - 1 preceding calls of subroutines FIL... .

The arrays KMESH, KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.6 Subroutine FIL104

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. In the case of the simple input (see 10.1), this may be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL104 must be called for each boundary element separately. Hence at most NUMNATBND calls of FIL104 are necessary. Subroutine FIL104 differs from subroutine FIL101 in the sense that it permits the reading of nodal values or element values of the coefficients from a file.

Call

```
CALL FIL104 (IPROB, IBNGRP, IUSER, USER, KPROB, NPARM, IWORK, WORK, KMESH)
```

Parameters

INTEGER IPROB, IBNGRP, IUSER(∗), KPROB(∗), NPARM, IWORK(1:4, 1:NPARM), KMESH(∗)

DOUBLE PRECISION USER(∗), WORK(NPARM)

IPROB Problem number. With IPROB the user indicates to which problem the arrays IUSER and USER correspond. Usually IPROB = 1, however, if different problems are to be solved on the same mesh, IPROB may have a value larger than zero. Of course that problem must have been introduced in subroutine SEPSTR or PROBDF.

IBNGRP Standard boundary element number ($1 \leq IBNGRP \leq NUMNATBND$).
The calls of subroutines FIL... must be done in a natural sequence, first for all internal elements (i.e. corresponding to the differential equation) and then for the boundary element with boundary element number IBNGRP = 1, then for IBNGRP = 2 etc.

IUSER Integer user array of variable length that is filled with integer information concerning the coefficients of the standard problems. Since IUSER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications.

USER Double precision user array of variable length that is filled with real information concerning the coefficients of the standard problems. Since USER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications. However, if the user uses the input option with iref (See input), then the length needed will be considerably longer, since information for each nodal point or each element must be stored.
The arrays IUSER and USER are used as input for subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

NPARM In this position the number of parameters for the differential equation must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays in which the user must store some information concerning the coefficients in the sequence as given in the description of the standard problems.
IWORK is an integer two-dimensional array with dimension:
IWORK(1:3,1:NPARM)
and WORK a double precision one-dimensional array with dimension:
WORK(1:NPARM).
When IWORK(1,i) = 0, the i^{th} parameter is a constant, whose value must be stored in

WORK(i).

When IWORK(1,i) > 0, the i^{th} parameter is a function.

When IWORK(1,i) < 1000, the user written function subroutine FUNCCF is used to compute the value of this parameter in a point. For a description of FUNCCF see 10.4. The value of IWORK(1,i) is used as the parameter ICHOIS in FUNCCF

When IWORK(1,i) > 1000 **and** IWORK(1,i) < 2000, the user written function subroutine FUNCC1 is used to compute the value of this parameter. For a description of FUNCC1 see 10.4. The value of IWORK(1,i) - 1000 is used as the parameter ICHOIS in FUNCC1.

When IWORK(1,i) = -j <0: In this case the coefficient number i is identical to coefficient number j (j<i).

When IWORK(1,i) = 2001, the coefficient is given by the user in each nodal point.
Define IREF = IWORK(2,i). If IREF > 0 the NPOINT values are read from file IREF (free format). Exactly NPOINT values must be given. All these coefficients are stored in array USER, hence this array must have sufficient length. If IREF < 0 then the user written subroutine FUNCFL (see Programmers Guide 5.11.1) must be used to fill array USER in these NPOINT points. The parameter ICHOIS in the call of FUNCFL is equal to -IREF at the call of FUNCFL.
All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2002, then the coefficient is given by the user in each element of element group IBNGRP. The NELEM(IBNGRP) (= number of elements in element group IBNGRP) values are read from file IREF (free format), where IREF is equal to IWORK(2,i). Exactly NELEM(IBNGRP) values must be given. All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2003, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i).

When IWORK(1,i) = 2004, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD multiplied by a function or a constant. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i). IWORK(4,i) must contain information of the multiplication factor. If IWORK(4,i) = 0, then the multiplication factor is a constant whose value must be stored in WORK(i), when IWORK(4,i) > 0 the user written function subroutine FUNCCF is used to compute the value of this multiplication factor in a point. For a description of FUNCCF see 10.4. The value of IWORK(4,i) is used as the parameter ICHOIS in FUNCCF. Only values of IWORK(4,i) < 1000 are permitted.

When IWORK(1,i) > 10000, the user written function subroutine FUNCC3 is used to compute the value of this parameter. For a description of FUNCC3 see 10.4. The value of IWORK(1,i) - 10000 is used as the parameter ICHOIS in FUNCC3
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

KMESH Output of the mesh generator.

Input

The user must give IPROB, IBNGRP and NPARM a value.
When IBNGRP > 1 the arrays IUSER and USER must be filled before by NELGRP + IBNGRP

- 1 preceding calls of subroutines FIL... .
The arrays KMESH, KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.7 Subroutine FIL153

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. In the case of the simple input (see 10.1), this may be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL153 may not be used for boundary elements.
Subroutine FIL153 differs from subroutine FIL150 in the sense that it permits the reading of nodal values or element values of the coefficients from a file. It is the complex equivalent of FIL103.

Call

```
CALL FIL153 (IPROB, IELGRP, IUSER, USER, KPROB, NPARM, IWORK, WORK, KMESH)
```

Parameters

INTEGER IPROB, IELGRP, IUSER(∗), KPROB(∗), NPARM, IWORK(1:4, 1:NPARM), KMESH(∗)

DOUBLE PRECISION USER(∗)

COMPLEX ∗16 WORK(∗)

IPROB Problem number. With IPROB the user indicates to which problem the arrays IUSER and USER correspond. Usually IPROB = 1, however, if different problems are to be solved on the same mesh, IPROB may have a value larger than zero. Of course that problem must have been introduced in subroutine SEPSTR or PROBDF.

IELGRP Standard element number ($1 \leq IELGRP \leq NELGRP$).
The calls of subroutines FIL... must be done in a natural sequence, hence first a call for IELGRP = 1, then for IELGRP = 2 etc.

IUSER Integer user array of variable length that is filled with integer information concerning the coefficients of the standard problems. Since IUSER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications.

USER Double precision user array of variable length that is filled with real information concerning the coefficients of the standard problems. Since USER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications. However, if the user uses the input option with iref (See input), then the length needed will be considerably longer, since information for each nodal point or each element must be stored.
The arrays IUSER and USER are used as input for the subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

NPARM In this position the number of parameters for the differential equation must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays in which the user must store some information concerning the coefficients in the sequence as given in the description of the standard problems.
IWORK is an integer two-dimensional array with dimension:
IWORK(1:4,1:NPARM)
and WORK a complex one-dimensional array with dimension:
WORK(1:NPARM).

When IWORK(1,i) = 0, the i^{th} parameter is a constant, whose value must be stored in WORK(i).

When IWORK(1,i) > 0, the i^{th} parameter is a function.

When IWORK(1,i) < 1000, the user written function subroutine CFUNCF is used to compute the value of this parameter in a point. For a description of CFUNCF see 10.4. The value of IWORK(1,i) is used as the parameter ICHOIS in CFUNCF

When IWORK(1,i) > 1000 **and** IWORK(1,i) < 2000, the user written function subroutine CFUNC1 is used to compute the value of this parameter. For a description of CFUNC1 see 10.4. The value of IWORK(1,i) - 1000 is used as the parameter ICHOIS in CFUNC1.

When IWORK(1,i) = -j <0: In this case the coefficient number i is identical to coefficient number j (j<i).

When IWORK(1,i) = 2001, the coefficient is given by the user in each nodal point.
Define IREF = IWORK(2,i). If IREF > 0 then the NPOINT values are read from file IREF (free format). Exactly NPOINT values must be given. All these coefficients are stored in array USER, hence this array must have sufficient length. If IREF < 0 then the user written subroutine FUNCFL (see Programmers Guide 5.11.1) must be used to fill array USER in these NPOINT points. The parameter ICHOIS in the call of FUNCFL is equal to -IREF at the call of FUNCFL.
All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2002, then the coefficient is given by the user in each element of element group IELGRP. The NELEM(IELGRP) (= number of elements in element group IELGRP) values are read from file IREF (free format), where IREF is equal to IWORK(2,i). Exactly NELEM(IELGRP) values must be given. All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2003, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i). When IWORK(1,i) = 2004, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD multiplied by a function or a constant. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i). IWORK(4,i) must contain information of the multiplication factor. If IWORK(4,i) = 0, then the multiplication factor is a constant whose value must be stored in WORK(i), when IWORK(4,i) > 0 the user written function subroutine CFUNCF is used to compute the value of this multiplication factor in a point. For a description of CFUNCF see 10.4. The value of IWORK(4,i) is used as the parameter ICHOIS in CFUNCF. Only values of IWORK(4,i) < 1000 are permitted.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

KMESH Output of the mesh generator

Input

The user must give IPROB, IELGRP and NPARM a value.
When IELGRP > 1 the arrays IUSER and USER must be filled before by IELGRP - 1 preceding calls of subroutines FIL... .
The arrays KMESH, KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.2.8 Subroutine FIL154

Description

The specification of the coefficients for the standard problems is stored in the arrays IUSER and USER. In the case of the simple input (see 10.1), this may be done by the subroutines FIL... .
A subroutine FIL... must be called for each standard element and for each standard boundary element separately. Hence at most NELGRP + NUMNATBND calls of subroutines FIL... are necessary, before the call of subroutine BUILD. Subroutine FIL154 must be called for each boundary element separately. Hence at most NUMNATBND calls of FIL154 are necessary. Subroutine FIL154 differs from subroutine FIL101 in the sense that it permits the reading of nodal values or element values of the coefficients from a file. It is the complex equivalent of FIL103.

Call

```
CALL FIL154 (IPROB, IBNGRP, IUSER, USER, KPROB, NPARM, IWORK, WORK, KMESH)
```

Parameters

INTEGER IPROB, IBNGRP, IUSER(∗), KPROB(∗), NPARM, IWORK(1:4, 1:NPARM), KMESH(∗)

DOUBLE PRECISION USER(∗)

COMPLEX ∗16 WORK(∗)

IPROB Problem number. With IPROB the user indicates to which problem the arrays IUSER and USER correspond. Usually IPROB = 1, however, if different problems are to be solved on the same mesh, IPROB may have a value larger than zero. Of course that problem must have been introduced in subroutine SEPSTR or PROBDF.

IBNGRP Standard boundary element number ($1 \leq IBNGRP \leq NUMNATBND$).
The calls of subroutines FIL... must be done in a natural sequence, first for all internal elements (i.e. corresponding to the differential equation) and then for the boundary element with boundary element number IBNGRP = 1, then for IBNGRP = 2 etc.

IUSER Integer user array of variable length that is filled with integer information concerning the coefficients of the standard problems. Since IUSER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications.

USER Double precision user array of variable length that is filled with real information concerning the coefficients of the standard problems. Since USER is a variable length array, the first position must be filled by the user with the declared length. In general a length of 100 will be sufficient for standard applications. However, if the user uses the input option with iref (See input), then the length needed will be considerably longer, since information for each nodal point or each element must be stored.
The arrays IUSER and USER are used as input for subroutine BUILD.

KPROB Array containing information of the problem to be solved.
Output of subroutine SEPSTR.

NPARM In this position the number of parameters for the differential equation must be stored. This parameter is given for each standard problem.

IWORK,WORK User arrays in which the user must store some information concerning the coefficients in the sequence as given in the description of the standard problems.
IWORK is an integer two-dimensional array with dimension:
IWORK(1:3,1:NPARM)
and WORK a double precision one-dimensional array with dimension:

WORK(1:NPARM).
When IWORK(1,i) = 0, the i^{th} parameter is a constant, whose value must be stored in WORK(i).

When IWORK(1,i) > 0, the i^{th} parameter is a function.

When IWORK(1,i) < 1000, the user written function subroutine CFUNCF is used to compute the value of this parameter in a point. For a description of CFUNCF see 10.4. The value of IWORK(1,i) is used as the parameter ICHOIS in CFUNCF

When IWORK(1,i) > 1000 **and** IWORK(1,i) < 2000, the user written function subroutine CFUNC1 is used to compute the value of this parameter. For a description of CFUNC1 see 10.4. The value of IWORK(1,i) - 1000 is used as the parameter ICHOIS in CFUNC1.

When IWORK(1,i) = -j <0: In this case the coefficient number i is identical to coefficient number j (j<i).

When IWORK(1,i) = 2001, the coefficient is given by the user in each nodal point.
Define IREF = IWORK(2,i). If IREF > 0 the NPOINT values are read from file IREF (free format). Exactly NPOINT values must be given. All these coefficients are stored in array USER, hence this array must have sufficient length. If IREF < 0 then the user written subroutine FUNCFL (see Programmers Guide 5.11.1) must be used to fill array USER in these NPOINT points. The parameter ICHOIS in the call of FUNCFL is equal to -IREF at the call of FUNCFL.
All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2002, then the coefficient is given by the user in each element of element group IBNGRP. The NELEM(IBNGRP) (= number of elements in element group IBNGRP) values are read from file IREF (free format), where IREF is equal to IWORK(2,i). Exactly NELEM(IBNGRP) values must be given. All these coefficients are stored in array USER, hence this array must be sufficiently long.

When IWORK(1,i) = 2003, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i).

When IWORK(1,i) = 2004, then the coefficient is equal to the d^{th} degree of freedom in each node in each j^{th} vector in array IVCOLD multiplied by a function or a constant. IVCOLD denotes the parameter ISLOLD in subroutine BUILD. j must be stored in IWORK(2,i) and d in IWORK(3,i). IWORK(4,i) must contain information of the multiplication factor. If IWORK(4,i) = 0, then the multiplication factor is a constant whose value must be stored in WORK(i), when IWORK(4,i) > 0 the user written function subroutine CFUNCF is used to compute the value of this multiplication factor in a point. For a description of CFUNCF see 10.4. The value of IWORK(4,i) is used as the parameter ICHOIS in CFUNCF. Only values of IWORK(4,i) < 1000 are permitted.
Mark that IWORK and WORK are not treated as SEPRAN variable arrays, hence the user must take care of the declaration himself.

KMESH Output of the mesh generator.

Input

The user must give IPROB, IBNGRP and NPARM a value.
When IBNGRP > 1 the arrays IUSER and USER must be filled before by NELGRP + IBNGRP - 1 preceding calls of subroutines FIL... .
The arrays KMESH, KPROB, IWORK, and WORK must have been filled.

Output

A part of the arrays IUSER and USER have been filled.

10.3 Direct filling of IUSER and USER

Direct filling of the arrays IUSER and USER requires more effort of the user, but gives some extra possibilities compared to the simple input. This possibility may be combined with the subroutines described in Section 10.2.
The way the arrays IUSER and USER must be filled is standard for each problem. We distinguish between 4 possibilities:

- the parameter is given as a function.
- the parameter is given as a constant (including zero).
- the parameter is given for each nodal point.
- the parameter is given for each element.

In order to distinguish between all possibilities the following scheme is adopted:

- (i) The user must give IUSER(5+IELGRP) a value: ISTART. IELGRP is the standard element number ($1 \leq IELGRP \leq NELGRP$); for boundary elements:
  ($NELGRP + 1 \leq IELGRP \leq NELGRP + NUMNATBND$).
  Hence ISTART = IUSER(5 + IELGRP).
  ISTART must satisfy: ISTART > 5 + NELGRP + NUMNATBND.

- (ii) The actual information of the parameters must be stored in array IUSER from position ISTART. Let the parameters for the standard element be p_1 , p_2 , p_3 , ... in that sequence. Then:

  IUSER(ISTART) corresponds to parameter p_1.
  When IUSER(ISTART) = 0, $p_1 = 0$.
  When IUSER(ISTART) < -5, p_1 is a constant. The value of the constant must be stored by the user in USER(-IUSER(ISTART)).
  When 0 < IUSER(ISTART) < 1000, p_1 is a function. The function values are computed by the user written function subroutine FUNCCF (see 10.4). The parameter ICHOIS in FUNCCF gets the value IUSER(ISTART).
  When 1000 < IUSER(ISTART) < 2000, p_1 is a function. The function values are computed by the user written function subroutine FUNCC1 (see 10.4). The parameter ICHOIS in FUNCC1 gets the value IUSER(ISTART)-1000.
  When IUSER(ISTART) = 2001, p_1 must be stored by the user in array USER for each nodal point. In this case an extra position in array IUSER is required. Hence $ISTART_{new} = ISTART_{old} + 1$.
  In IUSER($ISTART_{new}$) the starting position of the information in array USER must be stored by the user. The values of p_1 in each nodal point must be stored by the user in the following way:

  - Fill IADRES in IUSER($ISTART_{new}$).
  - Fill $p_1(x^i)$ in USER(IADRES+i-1) (i = 1 (1) NPOINT).

  In the following $ISTART_{new}$ is used instead of ISTART.
  When IUSER(ISTART) = 2002, p_1 is supposed to be constant in each element. The user must fill the values of p_1 for each element, with standard element IELGRP in USER. In this case an extra position in array IUSER is required. Therefore ISTART must be raised by one, i.e. $ISTART_{new} = ISTART_{old} + 1$. In IUSER($ISTART_{new}$) the starting position of the information in array USER must be stored by the user.
  Hence the arrays IUSER and USER must be filled by the user in the following way:

  IUSER(ISTART) = 2002
  ISTART = ISTART + 1
  IUSER(ISTART) = IADRES

USER(IADRES+i-1) = $p_1(element^i{}_{IELGRP})$ (i = 1(1) $NELEM_{IELGRP}$)
where $NELEM_{IELGRP}$ is the number of elements with standard element number IELGRP and $element^i{}_{IELGRP}$ is the i^{th} element with standard element IELGRP.
Furthermore IUSER(ISTART+1) corresponds to the parameter p_2 exactly in the same way as IUSER(ISTART) corresponds to p_1. The value of ISTART must be the corrected value of ISTART when p_1 is given in nodal points or elements. In the same way IUSER(ISTART+2) corresponds to the parameter p_3 etc.

Example

Consider the differential equation:

$$\frac{\partial}{\partial x}\alpha_{11}\frac{\partial \phi}{\partial x} - \frac{\partial}{\partial x}\alpha_{12}\frac{\partial \phi}{\partial y} - \frac{\partial}{\partial y}\alpha_{12}\frac{\partial \phi}{\partial x} - \frac{\partial}{\partial y}\alpha_{22}\frac{\partial \phi}{\partial y} + u_1\frac{\partial \phi}{\partial x} + u_2\frac{\partial \phi}{\partial y} + \beta\phi = f$$

Let there be one standard element, hence NELGRP = 1.
Let NUMNATBND be zero.
Let furthermore $\alpha_{12} = 0,\ \alpha_{11} = \alpha_{22} = 1$; u_1 , u_2 be given point-wise, and β be given element-wise and f be given by subroutine FUNCCF with ICHOIS = 10. The sequence of the parameters is $\alpha_{11},\ \alpha_{12}$, α_{22} , u_1 , u_2 , β , f. Then IUSER and USER may be filled in the following way:

ISTART = 7
IUSER(6) = ISTART

IUSER(ISTART) = -6 α_{11}
USER(6) = 1

IUSER(ISTART+1) = 0 α_{12}
IUSER(ISTART+2) = -6 α_{22}

IUSER(ISTART+3) = 2001 u_1
ISTART = ISTART + 1
IUSER(ISTART+3) = 7
statements to fill USER(7), . . . ,USER(6+NPOINT)
with $u_1(x^1), ...,\ u_1(x^{NPOINT})$

IUSER(ISTART+4) = 2001 u_2
ISTART = ISTART + 1
IUSER(ISTART+4) = 7 + NPOINT
statements to fill USER(7+NPOINT), . . . ,USER(6+2 × NPOINT)
with $u_2(x^1), ...,\ u_2(x^{NPOINT})$

IUSER(ISTART+5) = 2002 β
ISTART = ISTART + 1
IUSER(ISTART+5) = 7 + 2 × NPOINT
statements to fill USER(7+ 2 × NPOINT), . . . ,USER(6+2 × NPOINT + NELEM)
with $\beta(elem_1), ...,\ \beta(elem_{NELEM})$

IUSER(ISTART+6) = 10 f

Combination of simple and extended input

The user may couple the simple input for one standard element, with the extended input for another standard element. In that case he must use the following positions of IUSER and USER.

In IUSER(2) the last standard element number IELGRP filled, must be stored.
$1 \leq IELGRP \leq NELGRP + NUMNATBND$
In IUSER(4) the last position used in IUSER must be stored.
In USER(4) the last position used in USER must be stored.

These positions are used and also filled by the subroutines FIL... (See 10.2)

10.4 Subroutines FUNCCF, FUNCC1, FUNCC3, CFUNCF and CFUNC1

10.4.1 Function subroutine FUNCCF

Description

With this function subroutine a function may be defined.
FUNCCF must be written by the user.

Heading

```
function funccf ( ichoice, x, y, z )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION FUNCCF, X, Y, Z

ICHOICE Choice parameter. This parameter enables the user to distinguish between several cases. ICHOICE must have been given a value before in a subroutine, for example BUILD.

X,Y,Z X, y, respectively z co-ordinates of the nodal point. For each nodal point this subroutine is called.

FUNCCF At output FUNCCF should get the computed value of the function in the nodal point.

Input

ICHOICE, X, Y, Z have got a value, depending on the dimension of the space.

Output

FUNCCF must have got a value.

Remark

Function subroutine FUNCCF must be programmed by the user and satisfy the following requirements (see also the user manual):

```
FUNCTION FUNCCF ( ICHOICE, X, Y, Z )
IMPLICIT NONE
INTEGER ICHOICE
DOUBLE PRECISION  FUNCCF, X, Y, Z
statements to compute a value
FUNCCF = value
END
```

10.4.2 Function subroutine FUNCC1

Description

Function subroutine FUNCC1 has the same task as FUNCCF (10.4.1), however, it contains an extra parameter.

Heading

```
function funcc1 ( ichoice, x, y, z, uold )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION FUNCC1, X, Y, Z, UOLD(∗)

ICHOICE,X,Y,Z See function FUNCCF (10.4.1).

UOLD In this array of length NUNKP positions, the values of UOLD* in the nodal point are stored, where UOLD* is the array in the call of subroutine BUILD. Hence UOLD(1) contains the first unknown in the nodal point, UOLD(2) the second one etc.
UOLD is filled by subroutine BUILD.

Input

ICHOICE, X, Y, Z, UOLD have got a value, depending on the dimension of the space.

Output

FUNCC1 has got a value.

10.4.3 *Function subroutine FUNCC3*

Description

Function subroutine FUNCC3 has the same task as FUNCC1 (10.4.2), however, it contains some extra parameters.

Heading

```
VALUE = FUNCC3 ( ICHOICE, X, Y, Z, NUMOLD, MAXUNK, UOLD )
```

Parameters

INTEGER ICHOICE, NUMOLD, MAXUNK

DOUBLE PRECISION FUNCC3, X, Y, Z, UOLD(NUMOLD, MAXUNK)

ICHOICE,X,Y,Z See function FUNCCF (10.4.1).

NUMOLD Dimension parameter for array UOLD. Indicates the number of old solution vectors stored in UOLD.

MAXUNK Dimension parameter for array UOLD. Indicates the maximum number of degrees of freedom stored in the old solution vectors stored in UOLD.

UOLD In this two-dimensional array of length NUMOLD $\times$ MAXUNK positions, the values of UOLD* in the nodal point are stored, where UOLD* is the array in the call of subroutine BUILD. Hence UOLD(1,1) contains the first unknown of the first vector in the nodal point, UOLD(1,2) the second one etc.
UOLD(2,$*$) corresponds to the second vector and so on.
UOLD is filled by subroutine BUILD.

Input

ICHOICE, X, Y, Z, UOLD have got a value, depending on the dimension of the space.

Output

FUNCC3 has got a value.

10.4.4 Subroutine CFUNCF

Description

Subroutine CFUNCF has the same task as FUNCCF (10.4.1), however, it is used for complex problems.

Heading

```
subroutine cfuncf ( ichoice, x, y, z, comval )
```

Parameters

INTEGER ICHOICE
DOUBLE PRECISION X, Y, Z
COMPLEX ∗**16** COMVAL
ICHOICE,X,Y,Z See function FUNCCF (10.4.1).
COMVAL Complex variable in which the function value must be stored.

Input

ICHOICE, X, Y, Z have got a value, depending on the dimension of the space.

Output

COMVAL must have been filled by the user.

10.4.5 Subroutine CFUNC1

Description

Subroutine CFUNC1 has the same task as CFUNCF (10.4.4), however, it contains an extra parameter.

Heading

```
subroutine cfunc1 ( ichoice, x, y, z, comval, uold )
```

Parameters

INTEGER ICHOICE

DOUBLE PRECISION X, Y, Z

COMPLEX ∗**16** COMVAL, UOLD(∗)

ICHOICE,X,Y,Z See function FUNCCF (10.4.1).

COMVAL See subroutine CFUNCF (10.4.4).

UOLD See function FUNCC1 (10.4.2).
The only difference is that uold is complex in this case.

Input

ICHOICE, X, Y, Z, UOLD have got a value, depending on the dimension of the space.

Output

COMVAL must have been filled by the user.

References

Bath Klaus-Jürgen (1982) *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Inc, Englewood Cliffs, New Jersey 07632, 1982.

Bertrand F., P.A. Tanguy and F. Thibault (1997) *A three-dimensional fictitious domain method for incompressible fluid flow problems*, Int. J. for Num. Methods in Fluids, Vol. 25, pp. 719-736, 1997

Brooks A.N. and T.J.R. Hughes (1982) *Stream-line upwind/Petrov-Galerkin formulation for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 32, pp 199-259, 1982.

Brusche John (2007) *Mark formation model for optical rewritable recording* Thesis, Delft, The Netherlands.

Canuto C., M.Y. Hussaini, A. Quarteroni, and T.A. Zang (1988) *Spectral methods in fluid dynamics.* Springer−Verlag, New York, Berlin.

Caswell B. and M. Viriyayuthakorn (1983) *Finite element simulation of die swell for a Maxwell fluid*, J. of Non-Newt. Fluid Mech. Vol 12, 13-29.

Cuvelier C. (1980) *On the numerical solution of a capillary free boundary problem governed by the Navier-Stokes equations*, Lecture Notes in Physics, Vol 141, pp 373-384, (1980). Editor: M. Jean.

Cuvelier C., A. Segal and A.A. van Steenhoven (1986) *Finite element methods and Navier-Stokes equations.* Mathematics and its applications, Reidel publishing company, Dordrecht.

Dowson D. (1962) *A generalized Reynolds equation for fluid-film lubrication*, Int, J, Mech. Sci. Vol. 4, p. 159-170.

Van Duijn, C.J., L.A. Peletier, R.J. Schotting (1993) *On the Analysis of Brine Transport in Porous Media*, Eur. J. Appl. Math. Vol. 4, p. 271-302.

Duijn, C.J. van, R.A. Wooding, A. van der Ploeg (2000) *Stability Criteria for the Boundary Layer Formed by Throughflow at a Horizontal Surface of a Porous Medium, to appear*

Fachinotti, V.D. A. Cardano and A.E. Huespe (1999) *A fast convergent and accurate temperature model for phase-change heat conduction*, Numer. Met. Engng, 44 (12), pp 1863-1884.

Gielen A. W. J. (1998) *A continuum approach to the mechanics of contracting skeletal muscle.* PhD thesis, Eindhoven University of Technology, 1998.

Girault V. and P.A. Raviart (1979) *Finite element approximation of the Navier-Stokes equations.* Lecture notes in mathematics, 749, Springer Verlag, Berlin.

Glowinski R. and A. Marrocco (1974) *Analyse numerique du champ magnetique d'un alternateur par element finis et sur-relaxation ponctuelle non lineaire.* Computer Methods in applied mechanics and engineering (1974), 3, 55-85

Glowinski Roland, Tsorng-Way Pan and Jacques Periaux (1994a) *A fictitious domain method for Dirichlet problem and applications* , Computer Methods in Applied Mechanics and Engineering, 111, pp 283-303, 1994.

Glowinski Roland, Tsorng-Way Pan and Jacques Periaux (1994b) *A fictitious domain method for external incompressible viscous flow modeled by Navier-Stokes equations*, Computer Methods in Applied Mechanics and Engineering, 112, pp 133-148, 1994.

Glowinski Roland, Tsorng-Way Pan, Todd I Hesla, Daniel D. Joseph and Jacques Periaux (1999) *A distributed Lagrange multiplier/fictitious domain method for flows around moving rigid bodies: application to particulate flow*, Int. J. for Num. Methods in Fluids, Vol. 30, pp. 1043-1066, 1999

Harrison W.J. (1913) *The hydrodynamical theory of lubrication with special reference to air as a lubricant,* Trans. Cambridge Philos. Soc., (1913), 22, , pp 39-54

Heijningen G.G.J. van and C.G.M. Kassels (1987) *Elastohydrodynamic lubrication of an Oil Pumping Ring Seal.*

Hinze J. O. (1975) *Turbulence,* New York, McGraw-Hill, 2^{nd} edition.

Hirasaki, G.J., J.D. Hellums (1968) *A General Formulation of the Boundary Condition on the Vector Potential in Three-Dimensional Hydrodynamics*, Q. Appl. Math., Vol. 26, p. 331

Hughes Thomas J.R. (1987) *The Finite Element Method. Linear, static and dynamic Finite Element Analysis*, Prentice-Hall, Inc, Englewood Cliffs, New Jersey 07632, 1987.

Hussain A.K.M.F. and W.C. Reynolds (1975) *Measurements in fully developed channel flow*, Journal of Fluids Engineering, pp 568-578.

Kruyt N.P., C. Cuvelier and A. Segal (1988) *A total linearization method for solving viscous free boundary flow problems by the finite element method,* Int. J. for Num. Methods in Fluids, Vol. 8, pp. 351-363, 1988

Kumar A. and Booker J.F. (1994) *A mass and energy conserving finite element lubrication algorithm* Journal of Tribology, Vol 116, Issue 4, 1994.

Mizukami A. and T.J.R. Hughes (1985) *A Petrov-Galerkin finite element method for convection-dominated flows: an accurate upwinding technique for satisfying the maximum principle*, Computer Methods in Applied Mechanics and Engineering, 50, pp. 181-193, 1985

Mohr G.A. (1992) *Finite elements for solids, fluids and optimization*, Oxford University Press, Oxford, 1992.

Morgan K.J., J. Périaux and F. Thomasset (1984) *Analysis of Laminar Flow over a Backward Facing Step*, Proceedings of the GAMM Workshop held at Bièvres (Fr.), Vieweg Verlag Braunschweig, 1984.

Ogden R.W. (1984) *Non-linear elastic deformations*, Mathematics and its applications. Ellis Horwood Limited, 1984.

Peng S. H. and W. V. Chang. (1997) *A compressible approach in finite element analysis of rubber-elastic materials.* , Computers & Structures, 62(3):573–593, 1997.

Pieters, G.J.M. (2000) *Natural Convection Drive By Groundwater Flow in a Porous Medium*, Master thesis, Delft University of Technology, Faculty of Mathematics.

Rodi W. (1980) *Turbulence models and their application in hydraulics*, Delft, Int. Ass. for Hydraulic Res., 1980.

Segal Guus, Kees Vuik, Kees Kassels (1994) *On the implementation of symmetric and anti-symmetric periodic boundary conditions for incompressible flow*, Int. J. for Num. Methods in Fluids, Vol. 18, pp. 1153-1165, 1994

Souza Neto E.A. de, D.Perić, M.Dutko, and D.R.J. Owen (1996) *Design of simple low order finite elements for large strain analysis of nearly incompressible solids.*, Int. J. Solids Structures, Vol. 33, pp.3277-3296, 1996.

Silliman W.J. and L.E. Scriven (1980) *Separating flow near a static contact line: Slip at Wall and Shape of a Free Surface,* Journal of Computational Physics, Vol. 34, pp. 287-313, 1980

Tanner, R.I, R.E. Nickel and R.W. Bilger (1975) *Finite element methods for the solution of some incompressible non-Newtonian fluid mechanics problems with free surfaces.* Comput. Methods Appl. Mech. Eng., 6, p. 155-174.

Tabata, Masahisa and Kazuhiro Itakura (1995) *Precise computation of drag coefficients of the sphere.* INSAM report no 12 (95-07), Department of Mathematics, Hiroshima University, Higashi-Hiroshima, 739, Japan

Tennekes H. and J. L. Lumley (1974) *An introduction to turbulence,* Cambridge (Mass.), The MIT Press, 3^{rd} printing.

Vahl Davis, G. de (1982) *Natural convection of air in a square cavity: A bench mark numerical solution,* Report 1892/FMT/2, School of Mechanical and Industrial Engineering, University of South Wales.

van de Vosse F.N. (1987) *Numerical analysis of carotid artery flow.* Thesis Eindhoven University of Technology, 1987.

van de Vosse F.N. and P.D. Minev (1996) *Spectral elements methods : Theory and applications.* EUT Report 96-W-001 ISBN 90-236-0318-5, Eindhoven University of Technology, June 1996.

C. Vuik, A. Segal, J.A. Meijerink (1998) *An efficient preconditioned CG method for the solution of layered problems with extreme contrasts in the coefficients.* Report 98-20. Reports of the Faculty of Math. and Inf., Delft University of Technology. ISSN 0922-5641

Wassenaar R.H. (1994) *Simultaneous heat and mass transfer in a horizontal tube film absorber; numerical tools for present and future absorber designs*, Ph.D. thesis Delft University of Technology, p. 73-75

Wekken B.J.C. van der , R.H. Wassenaar (1988) *Simultaneous heat and mass transfer accompanying absorption in laminar flow over a cooled wall*, Int. J. Refrig. 11, 70-77

Wekken B.J.C. van der, R.H. Wassenaar, A. Segal (1988) *Finite element method solution of simultaneous two-dimensional heat and mass transfer in laminar film flow*, Wärme- und Stoffübertragung (1988) 22, 347-354.

Wooding, R.A. (1960) *Rayleigh Instability of a Thermal Boundary Layer in Flow Through a Porous Medium*, J. Fluid Mech., Vol. 9 p.183-192

Yih S.M. (1986) *Modeling heat and mass transfer in falling liquid films*, in N.P. Cheremisinoff (ed.), Handbook of heat and mass transfer 2, Gulf Publ. Corp., Houston, ch. 5.

Zienkiewicz O.C. and R.L. Taylor (1989) *The Finite Element Method*, Volume 1, Fourth Edition, McGraw-Hill Book Company, London

Zienkiewicz O.C. and R.L. Taylor (1989) *The Finite Element Method*, Volume 2, Fourth Edition, McGraw-Hill Book Company, London

Index