Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

Executive Summary

Full Wave Analysis of the Contribution to the Radar Cross Section of the Jet Engine Air Intake of a Fighter Aircraft

Master Thesis for the Degree of Master of Science in Applied Mathematics: Literature Report

Problem area
Radar cross section prediction methods are used to analyse the radar signature of military platforms when the radar signature can not be determined experimentally because:

- The platform is in the design, development or procurement phase.
- The platform belongs to a hostile party.

For jet powered fighter aircraft, the radar signature is dominated by the contribution of the jet engine air intake for a large range of forward observation angles. The intake can be regarded as a one-side open large and deep forward facing cavity. Although the contribution of the outer mould shape of the platform can be efficiently and accurately computed using simple scattering models, these can not be used to accurately compute the contribution of the jet engine air intake. Previously, an algorithm was developed to enable so-called full wave analysis of cavity scattering, but the computational work involved prohibits the application for analysis of jet engine air intakes at the relevant excitation frequency band.

Description of work
A thorough analysis of the mathematical model of full wave cavity scattering has been made. The key elements of the numerical method

Report no.
NLR-TR-2007-310

Author(s)
P.B. Hooghiemstra

Classification report
Unclassified

Date
May 2007

Knowledge area(s)
Numerical Mathematics

Descriptor(s)
Radar
RCS
Large Sparse Systems
Iterative Methods
Preconditioning

have been assayed and potential alternatives have been identified.

Results and conclusions
The properties of the mathematical model indicate a significant reduction in computational work can be achieved when technology, that has been developed recently to solve large acoustic and seismic problems will be incorporated in the current solution algorithm. This will enable the application of the current method for the analysis of jet engine air intake scattering for X-band radar.

Applicability
The developed technology will be applied for the analysis and optimization of jet engine air intake geometries of current intermediate observable and future low observable fighter aircraft.

Nationaal Lucht- en Ruimtevaartlaboratorium, National Aerospace Laboratory NLR

Anthony Fokkerweg 2, 1059 CM Amsterdam,
P.O. Box 90502, 1006 BM Amsterdam, The Netherlands
Telephone +31 20 511 31 13, Fax +31 20 511 32 10, Web site: www.nlr.nl

Nationaal Lucht- en Ruimtevaartlaboratorium

National Aerospace Laboratory NLR

NLR-TR-2007-310

Full Wave Analysis of the Contribution to the Radar Cross Section of the Jet Engine Air Intake of a Fighter Aircraft

Master Thesis for the Degree of Master of Science in Applied Mathematics: Literature Report

P.B. Hooghiemstra

Customer	National Aerospace Laboratory NLR
Contract number	—-
Owner	National Aerospace Laboratory NLR
Division	Aerospace Vehicles
Distribution	Limited
Classification of title	Unclassified
	May 2007

Approved by:

Author	Reviewer	Managing department
03-05-'07	03-05-'07	03/05/07

Summary

Radar (Radio Detection and Ranging) is commonly known as a device to detect aircraft and ships by means of electromagnetic waves. These waves impinge on an object and the scattered electric field is received. If the *signal-to-noise* ratio exceeds a minimum hardware specific value, the platform will be detected. Therefore, during the development of a platform a study to determine its detectability is performed and the so-called *radar cross section* (RCS) a measure for the detectability is analysed. The RCS is a well known indicator of the platform, but in computing the RCS it is assumed that electromagnetic waves scatter isotropically which is not the case in practise. The RCS may be calculated from the scattered electric field and the incident field, normalized with the distance R between the radar installation and the object.

It is well known from practice that for frontal observation angles, the jet engine air intake of a modern fighter aircraft accounts for the greatest part of the RCS for a large angular region. This jet engine air intake can be seen as a forward facing cavity. The scattered electric field at the aperture of the cavity is computed by solving the *vector wave equation* which stems from the Maxwell's equations.

Inside the cavity the finite element method (FEM) is used to determine the electric field. Tetrahedral elements are used since they easily conform to the shape of the cavity. Furthermore, vector based basis functions are used to prevent spurious solutions to occur (which is the case if scalar basis functions are used). The order of the basis functions (also called the element order) is very important for the accuracy of the solution as well as for the size of the coefficient matrix. From the discretization of the vector wave equation inside the cavity in combination with an integral equation for the field outside the cavity, a large linear system follows which has to be solved.

In order to solve the large linear system, iterative methods were studied in combination with an effective preconditioner. At the time, only standard preconditioning techniques were used, which have no significant effect on the rate of convergence of the iterative methods. Therefore, the present implementation uses a direct linear solver which is able to compute the solution, but this approach is very time consuming.

Recently a new class of preconditioners is developed, with the so-called *shifted Laplace operator* as an example. It has been showed by Erlangga et al. in reference 5 that this preconditioner is robust and efficient for the Helmholtz equation. Since the vector wave equation is the logical extension of the Helmholtz equation from scalar to vector form, this preconditioner is expected to be an efficient preconditioner for the vector wave equation combined with an iterative method.

Samenvatting

Radar (Radio Detection and Ranging) is een algemeen bekend middel om vliegtuigen of schepen te detecteren door middel van elektromagnetische golven. Wanneer deze golven een object raken, wordt het verstrooide elektrische veld opgevangen door de radarontvanger. Als het signaal sterk genoeg is om de drempelwaarde te overschrijden, dan wordt het platform gedetecteerd. Daarom wordt er tijdens de ontwikkeling van een platform een studie gedaan naar de mate van detecteerbaarheid, de zogenaamde *radar cross section* RCS, van het platform. De RCS is een algemeen bekend kengetal van het platform, maar wordt bepaald door aan te nemen dat de gereflecteerde golven in all richtingen even sterk worden teruggekaatst (isotropisch) wat niet het geval is in de praktijk. De RCS is het quotiënt van het reflecterende- en het inkomende elektrische veld, genormaliseerd met de afstand R tussen de radar installatie en het platform.

Uit de praktijk is bekend dat de luchtinlaat van het vliegtuig voor het grootste deel van de RCS zorgdraagt wanneer het platform van de voorkant wordt aangestraald, voor een groot hoekbereik. Deze luchtinlaat kan gezien worden als een cilindervormige holte. Het verstrooide elektrische veld aan de opening van de holte wordt berekend door het oplossen van de *vector golf vergelijking* die volgt uit de vergelijkingen van Maxwell.

Binnenin de holte wordt de eindige elementen methode gebruikt om het elektrische veld te bepalen. Tetraëder elementen worden gebruikt omdat deze de vorm van de holte moeiteloos volgen. In plaats van scalaire basisfuncties worden er vector basisfuncties geintroduceerd om het ontstaan van zogenaamde *spurious solutions* te voorkomen. De orde van de basisfuncties (ook wel de element orde genoemd) is van belang voor de nauwkeurigheid van de berekening alsmede voor de grootte van de coëfficiënt matrix. Het discretizeren van de vector golf vergelijking binnenin de holte in combinatie met een integraal vergelijking voor het veld buiten de holte, resulteert in een groot lineair stelsel dat opgelost dient te worden.

Om het grote stelsel op te lossen zijn verschillende iteratieve methoden en preconditioneringen bestudeerd. Tijdens de ontwikkeling van het huidige oplosalgoritme zijn alleen standaard preconditioneringen geanalyseerd die niet robuust bleken te zijn. Daarom is in de huidige implementatie gekozen voor het gebruik van een directe methode voor het oplossen van het stelsel. Deze methode geeft gegarandeerd een oplossing, maar is niet efficient.

Recent is een nieuwe klasse van preconditioneringen ontwikkeld waarvan de zogenaamde *verschoven Laplace operator* een voorbeeld is. Erlangga et al. heeft in reference 5 laten zien dat

zo'n preconditionering effectief en robuust is voor de Helmholtz vergelijking. Omdat de vector golf vergelijking de vectorvorm van deze vergelijking is, wordt verwacht dat dit type preconditionering ook voor de vector golf vergelijking gebruikt kan worden.

Acknowledgments

This work could not have been finished without the support of a number of people.

I would like to thank the NLR to let me perform my master's thesis there and in particular the manager of the division Flight Physics and Loads, ir. Koen de Cock. I also would like to thank my direct supervisor dr. ir. Duncan van der Heul for leading me in the field of radar technologie and pointing out the direction of my survey.

Also I would like to thank prof. dr. ir. Kees Vuik for his guidance from Delft and his expertise in numerical methods. His comments on the report were of great value to me.

Contents

(112 pages in total)

List of tables

List of figures

Preface

This is the literature report which is part one of the Master thesis for the degree of Master of Science in Applied Mathematics, faculty of Electrical Engineering, Mathematics and Computer Science of Delft University of Technology. The graduation is performed in the unit of Numerical Analysis and has a duration of nine months.

The Master thesis is carried out at the National Aerospace Laboratory (NLR) in Amsterdam. This institute is the key center of knowledge and experience for aerospace technology in the Netherlands. The main subject is the optimization of the solution procedure of a very large system of complex valued linear equations, which result from the discretization of the vector wave equation by finite elements.

This thesis will be supervised on location by dr. Duncan van der Heul and, in Delft by dr. C. Vuik.

Amsterdam, April 2007 Pim Hooghiemstra

This page is intentionally left blank.

Part I

Full Wave Analysis of Cavity Scattering

This page is intentionally left blank.

1 Introduction

RADAR (Radio Detection and Ranging) is technology to detect aircraft and ships by means of electromagnetic waves. Since the early days of radar, a large effort has been put into improving radar to increase the likelihood of detection of airborne platforms. At the same time platform developers try to avoid detection of their aircraft. Reduction of the probability of detection of a platform can be achieved by shape optimization and application of special radar absorbing coatings.

In the early stages of the development of an aircraft it is very important to be able to predict the detectability of the aircraft by radar. A measure for this is the so called *radar cross section* (RCS). In the definition of the RCS it is assumed that the object reflects the electromagnetic wave isotropically which is not the case physically. Theoretically, the RCS is the ratio of the scattered electric field obtained from the target and the incident field send out by the radar installation, normalized with the distance R between the radar installation and the target. Apart from simple geometrical shapes it is in general not possible to compute the RCS exactly and the electric field on the aperture is determined numerically in practice. The far-field components of the scattered electric field are computed from the induced current distributions whereafter the RCS, proportional to the norm of the far-field squared, is computed. The unit of RCS is area $([m^2])$, but since the RCS varies heavily for different incident angles, it is normalized with the object size yielding decibels $([dB])$.

It is known that the jet engine air intake of a typical figther aircraft, a forward facing cavity, accounts for the greatest part of the radar cross section for a large angular region, if excited from the front side. The electric field scattered by the jet engine air intake can be computed by solving the *vector wave equation* obtained from the Maxwell's equations with the appropriate boundary conditions inside this cavity. By using the finite element method, this equation is discretized, which gives rise to a large system of linear equations. This matrix is complex valued with a sparsely and a fully populated part. In the present implementation a direct method based on Gaussian elimination is used to solve the system. This approach is accurate, but is very computer time and memory consuming. The purpose of this thesis is to investigate alternative linear solvers to speed up the solution process of this system. Therefore, iterative methods will be introduced. To increase the convergence rate of these iterative methods preconditioners are used.

The first part of this report (chapters 2-4) is divided into three subjects: the governing equations, the finite element discretization and the linear solvers. The choice of elements and basis

functions is explained and several direct and iterative methods are discussed including preconditioners. After this, in the second part the actual problem is analysed. The domain of interest will be given with appropriate boundary conditions. Then the problem is discretized yielding the large complex valued matrix. This matrix will be investigated and properties of this matrix will be given. The choice of linear solver for this case is the next item and this choice is justified by some numerical experiments. At the end an outline for the most promising direction of research is given. A number of suitable test problems is identified to test the quality of the (preconditioned) iterative method, before the target problem is considered.

2 Cavity Scattering

2.1 Introduction

In this chapter a general description of cavity scattering for a given domain Ω is given. It is explained why full wave analysis has to be applied whereafter the Maxwell equations are introduced. Additional constitutive relations are introduced and the *vector wave equation* is derived. To formulate a well-posed problem, boundary conditions are imposed. The vector wave equation is made dimensionless in order to apply dimensional analysis. Finally, the weak formulation of the vector wave equation is given.

As already mentioned in the introduction, it is known that the jet engine air intake of a modern fighter aircraft accounts for the main part of the electric field for an *electromagnetic* wave excited from the frontside for a large angular region. This air intake is a deep open cavity which is characterized by a large length/diameter ratio ($L/d > 3$). Therefore, it is not possible to analyse the cavity scattering by using high frequency approximate methods and a full wave method is used.

2.2 Maxwell's Equations

The governing equations of electromagnetism were found by a number of scientists in the nineteenth century and coupled by James Clerk Maxwell in 1873. Below they are given for a general domain Ω in differential form (see Jin reference 9, Chapter 1).

$$\nabla \times \mathcal{E} = -\frac{\partial \mathcal{B}}{\partial t}, \tag{2.1}$$

$$\nabla \times \mathcal{H} = \frac{\partial \mathcal{D}}{\partial t} + \mathcal{J}, \tag{2.2}$$

$$\nabla \cdot \mathcal{D} = \mathcal{Q}, \tag{2.3}$$

$$\nabla \cdot \mathcal{B} = 0, \tag{2.4}$$

$$\nabla \cdot \mathcal{J} = -\frac{\partial \mathcal{Q}}{\partial t}. \tag{2.5}$$

Here the following variables are used:

$\mathcal{E}$ = electric field intensity $[V/m]$
$\mathcal{D}$ = electric flux density $[C/m^2]$
$\mathcal{H}$ = magnetic field intensity $[A/m]$
$\mathcal{B}$ = magnetic flux density $[Wb/m^2]$
$\mathcal{J}$ = electric current density $[A/m^2]$
$\mathcal{Q}$ = electric charge density $[C/m^3]$.

2.2.1 Time-Harmonic Fields

When field quantities in Maxwell's equations are harmonic oscillating functions with a single frequency ω, the field is referred to as *time-harmonic*. If, for example, $\mathcal{F}(\mathbf{x}, t)$ is such a field quantity, it may be written as

$$\mathcal{F}(\mathbf{x}, t) = \mathbf{F}(\mathbf{x}) e^{j\omega t}, \tag{2.6}$$

where $j^2 = -1$. The derivative of $\mathcal{F}$ with respect to time t is then simply

$$\frac{\partial \mathcal{F}}{\partial t} = j\omega \mathcal{F}(\mathbf{x}, t). \tag{2.7}$$

Since the cavity scattering analysis is performed for X-band frequency only, time-harmonic fields are used in Maxwell's equation for the electric and magnetic fields. Hence, (2.1), (2.2) and (2.5) turn into

$$\begin{aligned} \nabla \times \mathbf{E} &= -j\omega \mathbf{B}, & (2.8)\\ \nabla \times \mathbf{H} &= j\omega \mathbf{D} + \mathbf{J}, & (2.9)\\ \nabla \cdot \mathbf{J} &= -j\omega q, & (2.10) \end{aligned}$$

where the quantities $\mathbf{E}, \mathbf{D}, \mathbf{H}, \mathbf{B}, \mathbf{J}$ and q are the *phasor* quantities corresponding to the variables defined in the preceeding section.

2.2.2 Constitutive Relations

The Maxwell equations need some extra relations to get a closed problem: the constitutive relations, which describe the macroscopic properties of the medium. These relations are given by

$$\begin{aligned} \mathbf{D} &= \varepsilon(\mathbf{x})\mathbf{E}, & (2.11)\\ \mathbf{B} &= \mu(\mathbf{x})\mathbf{H}, & (2.12)\\ \mathbf{J} &= \sigma(\mathbf{x})\mathbf{E}, & (2.13) \end{aligned}$$

where $\varepsilon,\ \mu,\ \sigma$ are the permittivity [farads/meter], the permeability [henrys/meter] and the conductivity [siemens/meter] respectively. The parameters are written as the product of the vacuum value ε_0 and a relativity constant ε_r. Hence, $\varepsilon = \varepsilon_0\, \varepsilon_r$ and $\mu = \mu_0\, \mu_r$. For simple mediums, these parameters are just real constants. However, if radar absorbing materials are used, like radar absorbing foam, the permittivity will be complex valued, $\varepsilon \in \mathbb{C}$. In appendix A these units are related to the standard SI-units and the vacuum values are given.

2.2.3 Vector Wave Equation

The constitutive relation (2.12) linking the magnetic flux density $\mathbf{B}$ to the magnetic field intensity $\mathbf{H}$, is used in equation (2.8) and yields

$$\nabla \times \mathbf{E} = -j\omega\mu\mathbf{H}. \tag{2.14}$$

After dividing by the permeability μ on both sides, $\mathbf{H}$ is eliminated in this equation by taking the curl on both sides, assuming a sufficiently smooth electric field $\mathbf{E}$

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right) = -j\omega\left(\nabla \times \mathbf{H}\right). \tag{2.15}$$

Using (2.9) in the last term yields

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right) = -j\omega\left(j\omega\mathbf{D} + \mathbf{J}\right) = \omega^2\mathbf{D} - j\omega\mathbf{J}. \tag{2.16}$$

Rewriting this and using (2.11) for $\mathbf{D}$ gives the *vector wave equation* for the electric field in the presence of a source

$$\nabla \times \left(\frac{1}{\mu}\nabla \times \mathbf{E}\right) - \omega^2\varepsilon\mathbf{E} = -j\omega\mathbf{J}. \tag{2.17}$$

The permeability and permittivity are written as a product yielding

$$\nabla \times \left(\frac{1}{\mu_r\,\mu_0}\nabla \times \mathbf{E}\right) - \omega^2\varepsilon_r\,\varepsilon_0\mathbf{E} = -j\omega\mathbf{J}. \tag{2.18}$$

Introducing the free-space wavenumber $k_0 = \omega\sqrt{\varepsilon_0\mu_0}$ and the free-space impedance $Z_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}}$ equation (2.18) can be written as

$$\nabla \times \left(\frac{1}{\mu_r}\nabla \times \mathbf{E}\right) - k_0^2\varepsilon_r\mathbf{E} = -jk_0Z_0\mathbf{J}. \tag{2.19}$$

When there is no source ($\mathbf{J} = 0$), the right hand side vanishes as is the case inside the cavity. The vector wave equation is then homogeneous:

$$\nabla \times \left(\frac{1}{\mu_r}\nabla \times \mathbf{E}\right) - k_0^2\varepsilon_r\mathbf{E} = 0. \tag{2.20}$$

2.3 Boundary Conditions

To have a well-posed problem, boundary conditions have to be specified on the boundary. The boundary of a cavity consists of the aperture (S_a) joint with the mantle (S_s) of the cavity. A boundary condition for each of the two is described below.

2.3.1 Boundary Condition on the Cavity Mantle

On the mantle of the cavity a homogeneous Dirichlet condition for the tangential electric field is prescribed since it is a perfect conductor:

$$(\hat{n} \times \mathbf{E})_{S_s} = 0. \tag{2.21}$$

2.3.2 Boundary Integral for the Aperture

For the aperture an integral equation is derived below. In the derivation the quantity $\mathbf{K} = \hat{n} \times \mathbf{E}$ is introduced (i.e. the fictitious magnetic current). The integral equation involved has $\mathbf{K}$ as a variable. This equation is given by

$$\hat{n} \times \mathbf{H}^{\text{inc}} = 4\hat{n} \times \left\{ \frac{\nabla\nabla \cdot \mathbf{N} + k_0^2 \mathbf{N}}{j\omega\mu_o} \right\}, \tag{2.22}$$

where

$$\mathbf{N} = \mathbf{K} * G = \iint_{S_a} \mathbf{K}(\mathbf{r}')G(\mathbf{r}, \mathbf{r}')\, d\mathbf{r}' = \iint_{S_a} \left[\hat{n} \times \mathbf{E}(\mathbf{r}')\right]\, G(\mathbf{r}, \mathbf{r}')\, d\mathbf{r}', \tag{2.23}$$

and G is the three dimensional Green's function.

The derivation of the integral equation follows the analysis of Peterson et al. (Ref. 13). The geometry is given in Figure 2.1, hence the source is in region 1.

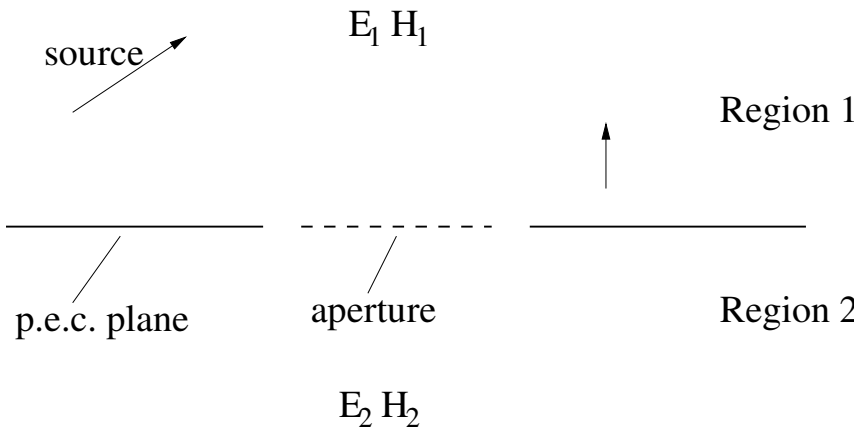

Fig. 2.1 Sideview of the geometry of the scattering problem. The aperture lies in the infinite perfect conducting plane.

The idea is to give two equivalent formulations. The first one for the electric and magnetic fields in region 1, the second one for region 2.

Region 1. First, a mathematical surface S is placed on the region 1 side of the conducting plane. Equivalent electric current and magnetic sources are introduced following

$$\begin{aligned} \mathbf{J}_1 &= \hat{n} \times \mathbf{H}_1 \\ \mathbf{K}_1 &= \mathbf{E}_1 \times \hat{n} \end{aligned} \tag{2.24}$$

These sources $(\mathbf{J}_1, \mathbf{K}_1)$ radiating together with the original source will replicate the original fields $\mathbf{E}_1$ and $\mathbf{H}_1$ in region 1 and at the same time create null fields in region 2. On the aperture, the electric current source $\mathbf{J}_1$ is nonzero, but the tangential electric field $\mathbf{K}_1$ vanishes on a perfect electric conductor (p.e.c.) hence the magnetic source $\mathbf{K}_1$ is nonzero on the aperture only. Since according to reference 13, the fields in region 2 vanish, it is possible to modify the material present without changing the fields in region 1. Therefore, the aperture is closed off with an additional piece of perfect conducting material. Now the situation as illustrated in Figure 2.2 is obtained where the original source and the new introduced sources $\mathbf{J}_1, \mathbf{K}_1$ are radiating in front of the infinite perfect conducting plane.

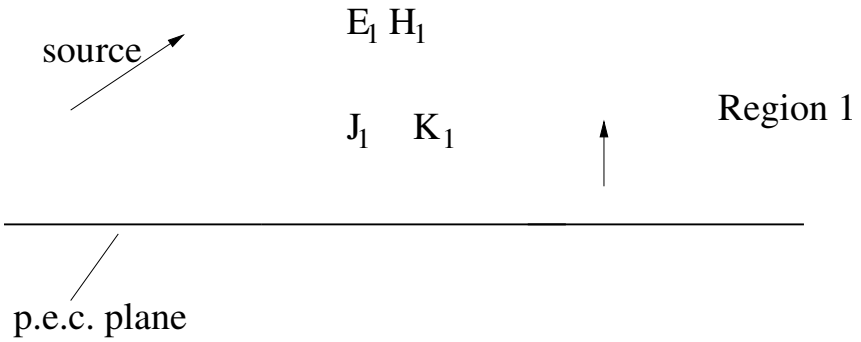

Fig. 2.2 The three sources (original one, $\mathbf{J}_1$ and $\mathbf{K}_1$) radiating in front of the infinte p.e.c. plane.

Now, the method of images, as described in reference 1 (p.317) is applied to remove the p.e.c. plane. The following happens to the sources

- The image of the magnetic current = the mirror image
- The image of the electric source = the negative mirror image
- The image of the tangential electric source cancels
- The image of the tangential magnetic source adds to the original.

Application of the image theory eliminates the p.e.c. plane and the electric source $\mathbf{J}_1$, leaving only an equivalent magnetic source $2\mathbf{K}_1$ located in the original aperture as depicted in Figure 2.3.

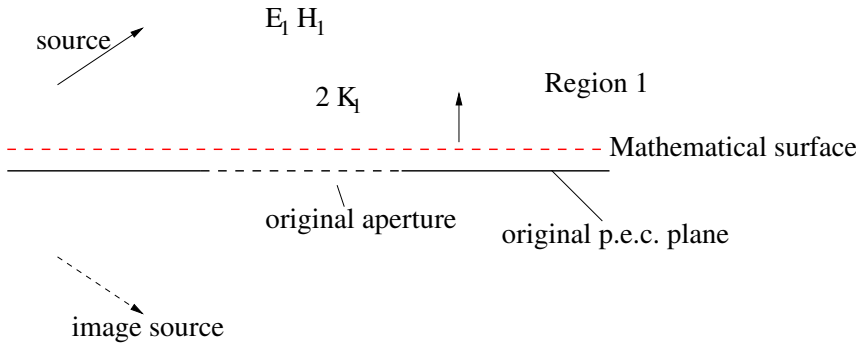

Fig. 2.3 Equivalent problem for region 1 associated with Figure 2.2.

The original source, its image and the doubled magnetic source superpositioned replicate the original field in region 1. These sources also produce nonzero fields in region 2, but these differ from the original fields in region 2. Therefore, the equivalence only applies to region 1.

The same kind of argument can be given for region 2. However, the normal vector used for region 1 is also used here. This means that the new introduced sources $\mathbf{J}_2$ and $\mathbf{K}_2$ satisfy

$$\begin{aligned} \mathbf{J}_2 &= (-\hat{n}) \times \mathbf{H}_2 \\ \mathbf{K}_2 &= \mathbf{E}_2 \times (-\hat{n}) = \hat{n} \times \mathbf{E}_2 \end{aligned} \tag{2.25}$$

Again, the aperture is closed off by a infinite p.e.c. plane and a magnetic source $2\mathbf{K}_2$ is found. Because of the continuity of the original tangential electric field through the aperture, $\mathbf{K}_2 = -\mathbf{K}_1$, hence the only variable needed is $\mathbf{K}_2$. According to reference 13, the integral equation can be written as

$$\hat{n} \times \bar{H}^{\text{inc}} = -4\hat{n} \times \left\{ \frac{\nabla\nabla \cdot \mathbf{N} + k^2\mathbf{N}}{j\omega\mu_0} \right\}_{S_a} \tag{2.26}$$

where $\mathbf{N}$ is the vector potential produced by $\mathbf{K}_2$ radiating in free space and $\mathbf{H}^{\text{inc}}$ is the field produced by the original source and its image. From reference 13 it follows that $\mathbf{N}$ is equal to

$$\mathbf{N} = \mathbf{K}_2 * G, \tag{2.27}$$

where the scalar function G is the three dimensional Green's function

$$G = \frac{e^{-jk|\mathbf{r}|}}{4\pi|\mathbf{r}|}. \tag{2.28}$$

The aterisk (*) denotes three-dimensional convolution, hence

$$\mathbf{N}(\mathbf{r}) = \iint_{S_a} \mathbf{K}_2(\mathbf{r}') \frac{e^{-jk|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}|}\, d\mathbf{r}'. \tag{2.29}$$

2.4 Dimensional Analysis

In order to reduce the number of parameters in the vector wave equation and to find the important parameters in this equation, it is made dimensionless. There are 4 scale factors for the quantities of: length, mass, time and electric current. These are: R, M, T and S. Next, the quantities, variables, parameters and operator have to be made dimensionless.

Quantities

$\mathbf{E}$: electric field intensity,

$$\text{dimension } \frac{V}{m} = \frac{m\,kg}{s^3\,A} \quad \Rightarrow \quad \text{define } \mathbf{E}^* = \mathbf{E}\,\frac{T^3\,S}{R\,M}. \tag{2.30}$$

$\mathbf{J}$: electric current density,

$$\text{dimension } \frac{A}{m^2} \quad \Rightarrow \quad \text{define } \mathbf{J}^* = \mathbf{J}\,\frac{R^2}{S}. \tag{2.31}$$

Position Variables

$$\text{Define } x^*,\, y^*,\, z^* = \frac{x}{R},\, \frac{y}{R},\, \frac{z}{R} \quad \text{respectively.} \tag{2.32}$$

Parameters

k_0 : free space wavenumber,

$$\text{dimension } \frac{1}{m} \quad \Rightarrow \quad \text{define } k_0^* = k_0\,R. \tag{2.33}$$

Z_0 : intrinsic free space impedance,

$$\text{dimension } \frac{kg\,m^2}{A^2\,s^3} \quad \Rightarrow \quad \text{define } Z_0^* = Z\,\frac{S^2\,T^3}{M\,R^2}. \tag{2.34}$$

Operator

∇ : gradient,

$$\nabla = \mathbf{i}\frac{\partial}{\partial x} + \mathbf{j}\frac{\partial}{\partial y} + \mathbf{k}\frac{\partial}{\partial z} = \mathbf{i}\frac{\partial}{\partial (x^*\,R)} + \mathbf{j}\frac{\partial}{\partial (y^*\,R)} + \mathbf{k}\frac{\partial}{\partial (z^*\,R)} = \frac{1}{R}\left(\mathbf{i}\frac{\partial}{\partial x^*} + \mathbf{j}\frac{\partial}{\partial y^*} + \mathbf{k}\frac{\partial}{\partial z^*}\right) = \frac{1}{R}\nabla^*. \tag{2.35}$$

Substituting (2.30) - (2.35) in (2.18) yields

$$\frac{1}{R}\nabla^* \times \frac{1}{R}\nabla^* \times \mathbf{E}^* \, \frac{R\,M}{T^3\,S} - (k_0^*)^2 \frac{1}{R^2}\mathbf{E}^* \, \frac{R\,M}{T^3\,S} = -j\frac{1}{R}k_0^* \frac{M\,R^2}{S^2\,T^3} Z_0^* \frac{S}{R^2}\mathbf{J}^* \Rightarrow$$
$$\left(\frac{M}{R\,T^3\,S}\right)\left[\nabla^* \times \nabla^* \times \mathbf{E}^* - (k_0^*)^2\mathbf{E}^*\right] = -j\left(\frac{M}{R\,T^3\,S}\right) k_0^* Z_0^* \mathbf{J}^*. \quad (2.36)$$

After division by $M/(R\,T^3\,S)$ the dimensionless vector wave equation is given by

$$\nabla^* \times \nabla^* \times \mathbf{E}^* - (k_0^*)^2\mathbf{E}^* = -jk_0^* Z_0^* \mathbf{J}^*. \quad (2.37)$$

From (2.37) it is immediately observed that k_0^* is the indicator of this equation. The choice of the scale variable R depends on physical aspects of the problem. In this problem the diameter of the geometrical cross section of the cavity is used as characteristic length, hence $k_0^* = k_0 d$. This will be explained in more detail in the next section.

2.4.1 Dependance of RCS on Wavenumber

According to Knott et al. (Ref. 12) the RCS is characterised by the ratio of the object size and the wavelength (d/λ) which is proportional to the wavenumber times the object size $(k_0 d = k_0^*)$. There are three regions of different behavior for the RCS depending on the value of the dimensionless wavenumber k_0^* :

(1) Rayleigh region: $0.1 < k_0^* < 1$.
(2) Resonance region: $1 < k_0^* < 10$.
(3) Optics region: $10 < k_0^* < 100$.

The dependance of the RCS upon the wavenumber is illustrated in Figure 2.4 for a metallic sphere with radius a. The RCS σ is normalized with the (geometrical) cross section of the sphere $\pi\, a^2$. This dependance of the RCS on k_0^* is discussed further in Section 8.

2.5 Weak Formulation

As can be seen in equation (2.20), the electric field is required to be continuously differentiable if the strong form of the vector wave equation is used. To make this equation more amenable to numerical solution the weak formulation is derived by multiplying (2.20) by a test function $\mathbf{T}$ and integrating over the domain Ω

$$\int_\Omega \mathbf{T} \cdot \left\{\nabla \times \left(\frac{1}{\mu_r}\nabla \times \mathbf{E}\right) - k_0^2 \varepsilon_r \mathbf{E}\right\} d\Omega = 0. \quad (2.38)$$

Invoking the second vector Green's identity which states

$$\int_\Omega \mathbf{P} \cdot (\nabla \times \nabla \times \mathbf{Q})\, d\Omega = \int_\Omega (\nabla \times \mathbf{P}) \cdot (\nabla \times \mathbf{Q})\, d\Omega - \oint_S (\mathbf{P} \times \nabla \times \mathbf{Q}) \cdot \hat{n}\, dS \quad (2.39)$$

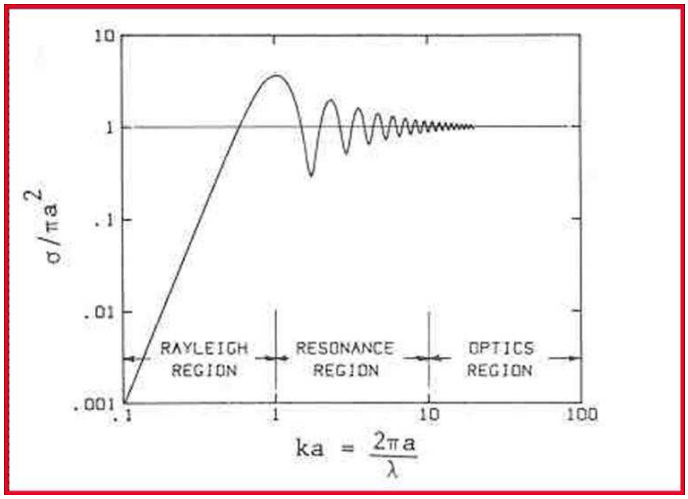

Fig. 2.4 The RCS σ of a metallic sphere with radius a illustrates the three scattering regions.

equation (2.38) may be written in weak form as

$$\int_\Omega (\nabla\times\mathbf{T})\cdot\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\,d\Omega-\oint_S\left\{\mathbf{T}\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\right\}\cdot\hat{n}\,dS-\int_\Omega k_0^2\varepsilon_r\,\mathbf{T}\cdot\mathbf{E}\,d\Omega=0. \quad (2.40)$$

The surface term is split up in two terms, one concerning the mantle of the cavity, the other concerns the aperture. This yields

$$\int_\Omega (\nabla\times\mathbf{T})\cdot\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\,d\Omega-\oint_{S_s}\left\{\mathbf{T}\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\right\}\cdot\hat{n}\,dS- \oint_{S_a}\left\{\mathbf{T}\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\right\}\cdot\hat{n}\,dS-k_0^2\varepsilon_r\int_\Omega\mathbf{T}\cdot\mathbf{E}\,d\Omega=0. \quad (2.41)$$

Now the integrand of the surface integrals is rewritten as

$$\hat{n}\cdot\left\{\mathbf{T}\times\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\right\}=-\hat{n}\cdot\left\{\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\times\mathbf{T}\right\}= -\left\{\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)\times\mathbf{T}\right\}\cdot\hat{n}=-(\mathbf{T}\times\hat{n})\cdot\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right). \quad (2.42)$$

Choosing the testfunction equal to the electric field one obtains $\mathbf{T}=\mathbf{E}$. The integrand then becomes

$$-(\mathbf{E}\times\hat{n})\cdot\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)=(\hat{n}\times\mathbf{E})\cdot\left(\frac{1}{\mu_r}\nabla\times\mathbf{E}\right)=\frac{1}{\mu_r}(\hat{n}\times\mathbf{E})\cdot(\nabla\times\mathbf{E}). \quad (2.43)$$

Since the tangential electric field vanishes on the mantle of the cavity, the first surface integral vanishes. The other surface integral is combined with the integral equation derived in Section

2.3.2 to obtain the functional F to be minimized which is given by Jin (Ref. 9) as

$$F(\mathbf{E}) = \frac{1}{2}\iiint_V \left[\frac{1}{\mu_r}(\nabla\times\mathbf{E})\cdot(\nabla\times\mathbf{E}) - k_0^2\varepsilon_r\,\mathbf{E}\cdot\mathbf{E}\right] dV$$
$$- k_0^2 \iint_{S_a} [\hat{z}\times\mathbf{E}(\mathbf{r})]\cdot\left\{\iint_{S_a}[\hat{z}\times\mathbf{E}(\mathbf{r}')]G_0(\mathbf{r},\mathbf{r}')\,dS'\right\} dS$$
$$+ \iint_{S_a}\Big[\nabla\cdot[\hat{z}\times\mathbf{E}(\mathbf{r})]\Big]\left\{\iint_{S_a} G_0(\mathbf{r},\mathbf{r}')\nabla'\cdot[\hat{z}\times\mathbf{E}(\mathbf{r}')]\,dS'\right\} dS$$
$$+ 2jk_0Z_0\iint_{S_a}[\hat{z}\times\mathbf{E}(\mathbf{r})]\cdot\mathbf{H}^{inc}(\mathbf{r})\,dS. \quad (2.44)$$

3 Finite Element Method Discretization

3.1 Introduction

In this chapter the finite element method is introduced for the numerical approximation of the electric field. This includes the discretization of the domain Ω by appropriate elements and the introduction of appropriate basis functions. First linear elements and basis functions will be derived, then vector based basis functions are considered, both linear and of higher order.

3.2 Domain Discretization

The domain Ω is divided in rectilinear tetrahedral elements. As a result of this the boundary Γ of the domain is divided in rectilinear triangles. These elements are chosen (tetrahedra and triangles) because they easily conform to the shape of the domain and boundary. Note that the elements approximate the domain and boundary so an error is introduced here.

3.3 Node Based Basis Functions

First, node based (scalar) linear basis functions are introduced for triangles (2-D), which are easily extended to tetrahedra (3-D). Sophisticated vector based basis functions and higher order elements are introduced in the next sections.

3.3.1 Triangular Elements

Let a triangular element be given and assume the unknown is a function $\phi(\mathbf{x})$. Within an element, say element e, the unknown function ϕ is approximated by

$$\phi^e(\mathbf{x}) = \sum_{j=1}^{3} L_j^e(\mathbf{x})\phi_j^e, \tag{3.1}$$

where ϕ_j^e is the nodal value of ϕ^e in node j. Write $\mathbf{x}_j = (x_j^1, x_j^2)$ for the coordinates of the jth node. Furthermore, the basis functions (L_j^e) corresponding to the jth node satisfy

(1) $L_i^e(\mathbf{x}_j) = \delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$

(2) L_j^e is linear.

Hence, these basis functions are given by

$$L_j^e(x^1, x^2) = \frac{1}{2\Delta^e}(a_j^e + b_j^e\, x^1 + c_j^e\, x^2), \tag{3.2}$$

where Δ^e is the area of the element and the coefficients $a_j^e,\ b_j^e,\ c_j^e$ satisfy, for $j = 1, 2, 3$

$$\begin{bmatrix} 1 & x_1^1 & x_1^2 \\ 1 & x_2^1 & x_2^2 \\ 1 & x_3^1 & x_3^2 \end{bmatrix} \begin{bmatrix} a_1^e & a_2^e & a_3^e \\ b_1^e & b_2^e & b_3^e \\ c_1^e & c_2^e & c_3^e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

3.3.2 Tetrahedral Elements

The extension from triangular elements to tetrahedra is straightforward. Start again with the approximation of the unknown function $\phi(\mathbf{x})$ within the element by

$$\phi^e(\mathbf{x}) = \sum_{j=1}^{4} L_j^e(\mathbf{x})\phi_j^e. \tag{3.4}$$

In the same way as described above it is assumed to know the nodal values of ϕ^e. Again the basis function for node j vanishes in the other nodes and is linear. Hence the linear basis functions for this case are given by

$$L_j^e(\mathbf{x}) = \frac{1}{6V^e}(a_j^e + b_j^e\, x^1 + c_j^e\, x^2 + d_j^e\, x^3), \tag{3.5}$$

where V^e is the volume of the tetrahedron, and the coefficients $a_j^e, \ldots, d_j^e$ are computed in a similar way to the computation for triangles.

3.4 Vector Based Basis Functions

Application of the basis functions discussed in the preceding sections to approximate vector electric or magnetic fields, can lead to the occurence of nonphysical or spurious solutions. This is caused by the following:

The basis functions do not satisfy the divergence condition (Ref. 9). This can be solved by either using basis functions that do have a continuous derivative, or to add a penalty to the functional to enforce the divergence condition. But a common solution to this problem is the use of *vector based basis functions*. The elements in this case are called *edge* or *vector based* elements.

In the rest of this section the linear or zeroth order vector basis functions for the triangular and tetrahedral elements will be derived and after this, these functions will be extended to higher order.

3.4.1 Triangular Elements

The vector basis function $\mathbf{N}_i^e(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^2$ defined along the ith edge should satisfy the following conditions

(1) $\nabla \cdot \mathbf{N}_i^e = 0$ (Since $\nabla \cdot \mathbf{E} = 0$.)

(2) $\mathbf{N}_i^e$ has a constant tangential component along the ith edge only. (Necessary condition for inter-elemental continuity of the expansion of the electric field.)

Consider the vector function for edge (i_1, i_2) given by

$$\mathbf{W}_{i_1 i_2} = L^e_{i_1} \nabla L^e_{i_2} - L^e_{i_2} \nabla L^e_{i_1}, \tag{3.6}$$

Here the linear basis functions from Section 3.3 are used again and the first observation is that the vector function $\mathbf{W}_{i_1 i_2}$ is indeed divergence free, since

$$\begin{aligned} \nabla \cdot \mathbf{W}_{i_1 i_2} = \nabla \cdot (L^e_{i_1} \nabla L^e_{i_2} - L^e_{i_2} \nabla L^e_{i_1}) = \\ L^e_{i_1} \nabla \cdot (\nabla L^e_{i_2}) + \nabla L^e_{i_1} \cdot \nabla L^e_{i_2} - \left(L^e_{i_2} \nabla \cdot (\nabla L^e_{i_2}) + \nabla L^e_{i_1} \cdot \nabla L^e_{i_2} \right) = 0. \end{aligned} \tag{3.7}$$

The second observation is that $\mathbf{W}_{i_1 i_2}$ has a constant tangential field along the edge (i_1, i_2). This is shown[1] by

$$\nabla \times \mathbf{W}_{i_1 i_2} = \nabla \times (L^e_{i_1} \nabla L^e_{i_2}) - \nabla \times (L^e_{i_2} \nabla L^e_{i_1}) = 2 \nabla L^e_{i_1} \times \nabla L^e_{i_2}, \tag{3.8}$$

since the gradient of L_j is a constant for $j = 1, 2, 3$. The tangential field vanish for the other two edges. If for $i_1 = 1, i_2 = 2$ edge $(1, 2)$ is defined to be edge 1 then the first vector basis function is defined by

$$\mathbf{N}^e_1 = \mathbf{W}_{12}\, l^e_1 = l^e_1 \,(L^e_1 \nabla L^e_2 - L^e_2 \nabla L^e_1), \tag{3.9}$$

where l^e_1 is the length of edge 1. In the same way the basis functions for the edges $(2, 3), (3, 1)$ respectively edge 2 and 3 are defined by

$$\begin{aligned} \mathbf{N}^e_2 &= \mathbf{W}_{23}\, l^e_2 = l^e_2 \,(L^e_2 \nabla L^e_3 - L^e_3 \nabla L^e_2), \\ \mathbf{N}^e_3 &= \mathbf{W}_{31}\, l^e_3 = l^e_3 \,(L^e_3 \nabla L^e_1 - L^e_1 \nabla L^e_3). \end{aligned}$$

3.4.2 Tetrahedral Elements

Since the tetrahedron is the simplex extension of the triangle from two to three dimensions, the vector basis functions are easily extended. Indeed, using the linear basis functions $L_1, \ldots, L_4$ from Section 3.3 yields the vector basis function

$$\mathbf{W}_{i_1 i_2} = L^e_{i_1} \nabla L^e_{i_2} - L^e_{i_2} \nabla L^e_{i_1}. \tag{3.10}$$

The properties of divergence and curl given in the last section are extended to the tetrahedral element in a straightforward manner, so these are not given here. Since a tetrahedron has six edges, define the vector basis functions $\mathbf{N}^e_i$ for $i = 1, \ldots, 6$, by

$$\mathbf{N}^e_i = \mathbf{W}_{i_1\, i_2}\, l^e_i = (L^e_{i_1} \nabla L^e_{i_2} - L^e_{i_2} \nabla L^e_{i_1})\, l^e_i, \tag{3.11}$$

[1] The last equality in (3.8) is true indeed, by using the vector identities $\nabla \times (a\mathbf{b}) = a \nabla \text{ x } \mathbf{b} - \mathbf{b} \times \nabla a$ and $\nabla \text{ x } (\nabla \mathbf{b}) = 0$.

where i is the edge connecting nodes i_1 and i_2.

The next section is concerned with higher order elements, but it has to be mentioned that the vector based basis depends on the ordering and the direction of the edges in a triangle or tetrahedral. To this end Table 3.1 with edge numbers and corresponding nodes is included here. This is also illustrated in Figure 3.1

Table 3.1 *Edge definition for a tetrahedral element.*

Edge i	Node i_1	Node i_2
1	1	2
2	1	3
3	1	4
4	2	3
5	4	2
6	3	4

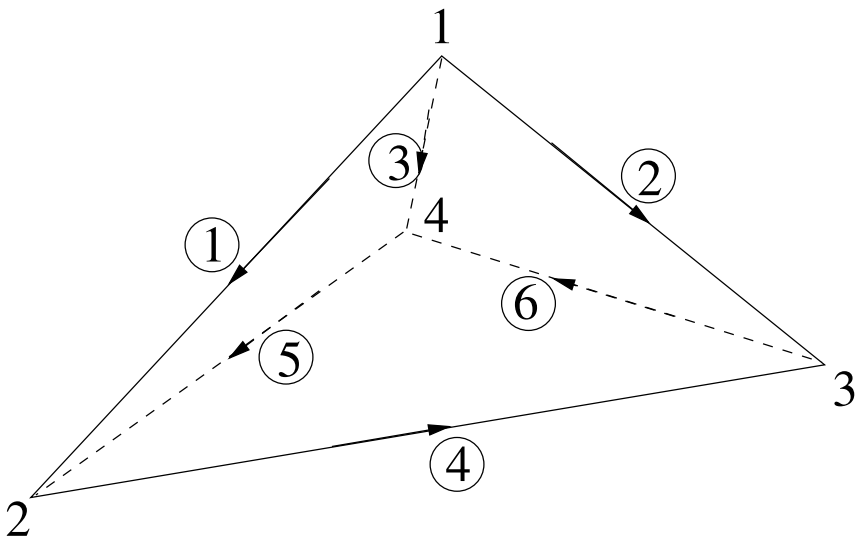

Fig. 3.1 The ordering of Table 3.1 is used here to number the edges of the tetrahedron.

3.4.3 Higher Order Vector Elements

Since the basis functions defined sofar, are of first order only, the convergence of the numerical solution will be slow. Therefore, basis functions of arbitrary order, say order p, will be introduced in this section. There are two types of bases: hierarchic and interpolatory bases. The hierarchic type builds new basis functions using the old ones. This is done by adding new functions to the lower-order basis functions. The second type uses the previously defined linear basis

functions and pre-multiplies these with a polynomial of order p. This second method will be discussed here, this idea was first proposed by Graglia (Ref. 6).

Remember that the zeroth order vector basis function on the edge connecting node i_1 and i_2 is $\mathbf{W}_{i_1\,i_2}$. Now this function is pre-multiplied by a complete polynomial of order p to obtain a vector basis of degree p. This complete polynomial consists of al terms $x_1^k\,x_2^l\,x_3^m$ such that $k+l+m\,=\,p$. Here a higher order vector based element for tetrahedra is considered. The complete polynomial is constructed using *Lagrange interpolatory polynomials* in combination with the *Silvester* and *shifted Silvester polynomials*. The description can be found in detail in references 9, 15. Here only the result is given. The new, pth order basis functions are given by (Ref. 9)

$$\mathbf{N}_{ijkl}^{i_1\,i_2} = \alpha_{ijkl}^{i_1\,i_2}\frac{(p+2)^2}{\gamma\,\beta}\xi_{i_3}\xi_{i_4}\hat{P}_i^{p+2}(\xi_1)\hat{P}_j^{p+2}(\xi_2)\hat{P}_k^{p+2}(\xi_3)\hat{P}_l^{p+2}(\xi_4)\mathbf{W}_{i_1\,i_2}, \tag{3.12}$$

where i_3 and i_4 are integers among $(1,2,3,4)$ other than i_1 and i_2 and $\gamma(\beta)$ is taken to be i,j,k or l for $i_3(i_4)=1,2,3$ or 4 respectively.

To construct the higher order vector basis functions for an edge, interpolation points on this edge and on the faces of the tetrahedron that coincide at this edge are used (see Figure 3.2). For each interpolation point on an edge there is one basis function, but for interpolation points on a face three basis functions are defined (since the face has three edges). The tangential electric field on the face is spanned by two independent basis functions, so for a point on the face where three basis functions are defined, one of them must be discarded since it is dependent. For an interior point six basis functions for each interpolation point in the interior are defined. In the same way as for a face there are only three independent basis functions for such a point, so the other three are discarded. This must be done with care however. To be independent, the three chosen basis functions should not have all the zeroth order basis functions associated with edges bounding the same face. Therefore, the three basis functions associated with edges $1,2,6$ in Figure 3.1 must be discarded for example. In reference 16 the choice of basis functions that are discarded is made as follows: Each edge has an individual number. Now the basis functions on a face correspond to the three edge numbers. The two basis functions with the highest edge number are maintained, the other is discarded. In this way the same basis functions are maintained for two adjacent faces, hence interelemental continuity of these basis function is assured.

Finally, 45 basis functions are obtained for the second order tetrahedral element. In Figure 3.2 the used interpolation points for a second order basis function on edge (1,3) are marked.

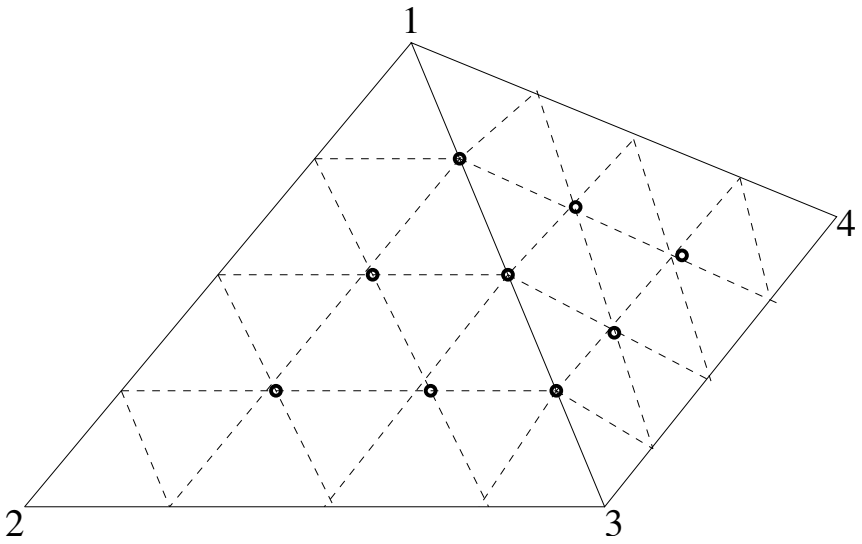

Fig. 3.2 The interpolation points on the second order tetrahedral element for vector basis functions associated with edge (1,3).

4 Linear Solvers

4.1 Introduction

In this chapter a description of some fundamental linear solvers for solving the (sparse) linear system $Ax = b$ will be given. The first section considers direct methods. Although the memory requirements using a direct method for three dimensional problems are usely high, there are some considerations that make a direct method a working alternative: Direct methods compute solutions for multiple right-hand sides at low extra cost.

The second section of this chapter is devoted to iterative solving techniques. As mentioned above, the memory and work requirements involved in direct methods are high, so iterative methods are considered. These methods are only of interest if the number of iterations is low. Therefore, if the convergence rate of a method is slow, a preconditioner is introduced. A trade-off has to be made when the preconditioner is analysed, between the extra cost of using the preconditioner versus the faster convergence rate. Preconditioning is the subject of the last section of this chapter.

4.2 Direct Methods

Since it is well known that the major problem for a direct method is its huge memory requirement, two direct methods which take the nonzero structure of the matrix in consideration will be discussed. The first one uses reordering to reduce the bandwidth of the matrix. The second one, called the *frontal solution method* (Ref. 8), reduces the number of active unknowns. Although the usual starting point in discussing direct methods is Gaussian elimination, this is only treated in the appendix for completeness.

4.2.1 Sparse Matrices and Ordering

In order to reduce the work involved in solving the system, full advantage must be taken from the sparsity structure of the matrix. Several ordering methods will be discussed, including the band method, the variable band (or profile) method and the block method.

4.2.1.1 Band Method

A *band* matrix is a sparse matrix, whose non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side. For an unsymmetric matrix define the upper bandwidth $q \geq 0$ by the smallest number such that $a_{ij} = 0$ for $j > i + q$. In the same way the lower bandwidth $p \geq 0$ is defined by the smallest number such that $a_{ij} = 0$ for $j < i - p$. Note that for a symmetric matrix $p = q$. A banded matrix may be stored in many different ways. One obvious way is to store the nonzero band-diagonals in vectors $v_1, \ldots, v_m$

and storing the place of the diagonals in a vector of length m, where m is the number of nonzero diagonals.

The amount of work for a symmetric banded matrix with bandwidth $p \ll n$ is greatly reduced in comparison with a full matrix. The amount of work for the Gaussian elimination is reduced to $2p^2\, n$ flops.

4.2.1.2 Profile Method

The *profile* method is preferable if a matrix has varying bandwidths throughout the rows. Such a matrix is defined as follows: In the lower triangular part the elements of each row from the first nonzero to the diagonal belong to the profile. For the upper part the elements of each column from the first nonzero to the diagonal also belong to the profile. This matrix may be stored in three vectors for the rows, columns and diagonal and one additional vector with entry positions.

4.2.1.3 Block Method

The next method to be considered is the *block* method. If a matrix is reduced to block triangular form, the corresponding set of linear equations can be solved as a sequence of subproblems. A matrix A reduced to block triangular form is written as

$$A = \begin{pmatrix} A_{11} & & \varnothing \\ A_{21} & A_{22} & \\ A_{31} & A_{32} & A_{33} \end{pmatrix}. \tag{4.1}$$

The solution vector and right-hand side are also decomposed into blocks and the equation $Ax = b$ is solved by

$$A_{ii}x_i = b_i - \sum_{j=1}^{i-1} A_{ij}x_j, \quad i = 1, 2, 3. \tag{4.2}$$

In this way all fill in is confined to the diagonal blocks.

4.2.2 Frontal Solution Method

In a finite element method, as described in this report, the discretization matrix A is constructed as a sum of element-matrices

$$A = \sum_k A^{[k]}. \tag{4.3}$$

The element matrices $A^{[k]}$ have entries only in the submatrix corresponding to the variables in the element. The summing of these element matrices is called assemblage and is performed by

$$a_{ij} := a_{ij} + a_{ij}^{[k]}. \tag{4.4}$$

A variable is *fully summed* as soon as all element matrices containing this variable are assembled. In that case the variable may be eliminated directly using Gaussian elimination.

The frontal solution method applies the idea mentioned above to matrices originating from the finite element method. It keeps track of a number of variables (the *active variables*) and by continuously adding new elements (and hence of new variables), other variables are fully summed and will be eliminated. In general there are only a few variables in action at the same time. These active variables form a *front*, which explains the name of the method. As soon as a variable is eliminated, the row and column of that variable are written out of core, since they do not change anymore. In this way only the *frontal matrix*, the matrix consisting of the equations corresponding to the front, needs to be stored in core.

This method is explained in more detail in Appendix B using a small example to visualize the ideas.

4.3 Iterative Methods

In this section iterative methods for solving the system

$$Ax = b \tag{4.5}$$

will be introduced. Here, $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. An iterative solver seeks an approximate solution x^k and tries to improve this solution every iteration until a certain tolerance is reached, which means that the residual r^k defined as $r^k = b - Ax^k$ is small enough in some norm.

Starting point will be a basic iterative method, called Jacobi's method to explain the idea of iterative solving. Thereafter this is extended to the more sophisticated Krylov subspace methods.

The Krylov subspace methods are described in three sections. The first section considers the *conjugate gradient* method which applies to symmetric positive definite (SPD) matrices only. After this, Krylov subspace methods for general matrices are discussed. Then a special method for symmetric complex matrices will be discussed. Finally, the multigrid method is treated which may be used in combination with preconditioners (see Chapter 4.4).

4.3.1 Basic Iterative Method

The system (4.5) has to be solved iteratively. Note that this equation can also be written as

$$\begin{aligned}
a_{11}\,x_1 + \ldots + a_{1n}\,x_n &= b_1 \\
a_{21}\,x_1 + \ldots + a_{2n}\,x_n &= b_2 \\
&\vdots \\
a_{n1}\,x_1 + \ldots + a_{nn}\,x_n &= b_n
\end{aligned}$$

These equations can be rewritten as

$$\begin{aligned} x_1 &= (b_1 - (a_{12}\, x_2 + \ldots + a_{1n}\, x_n))/a_{11} \\ x_2 &= (b_2 - (a_{21}\, x_1 + a_{23}\, x_3 + \ldots + a_{2n}\, x_n))/a_{22} \\ &\vdots \\ x_n &= (b_n - (a_{n1}\, x_1 + \ldots + a_{nn-1}\, x_{n-1}))/a_{nn}. \end{aligned}$$

4.3.1.1 Jacobi's Method

Note that the components $x_2, \ldots, x_n$ in the first equation above are unknown. To overcome this, choose a starting vector x^0 and compute the components of x^1 by setting

$$x_i^1 = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij}\, x_i^0 \right) / a_{ii}. \tag{4.6}$$

Going on this way the iterates $x^2, x^3, \ldots$ are computed. In matrix vector notation this becomes for iteration $k+1$

$$Dx^{k+1} = b - Cx^k \;\Rightarrow\; x^{k+1} = D^{-1}(b - Cx^k), \tag{4.7}$$

where $D = \text{diag}(A)$ and $C = A - D$. This method is known as Jacobi's method: From the matrix A a splitting is constructed by $A = B + (A - B)$, such that B is easy to invert. The iterative method then reads

$$Bx^{k+1} + (A - B)x^k = b. \tag{4.8}$$

Rewriting this equation one obtains

$$x^{k+1} = B^{-1}b - B^{-1}(A - B)x^k = B^{-1}b - B^{-1}Ax^k + x^k = B^{-1}b + (I - B^{-1}A)x^k. \tag{4.9}$$

Hence in general

$$x^{k+1} = B^{-1}b + (I - B^{-1}A)x^k. \tag{4.10}$$

It is obvious that for the Jacobi method $B = D$ and $B - A = C$.

Finally, note that the last equation can also be written by using the residual $r^k = b - Ax^k$. Equation (4.10) then becomes

$$x^{k+1} = x^k + B^{-1}(b - Ax^k) = x^k + B^{-1}r^k. \tag{4.11}$$

This formulation is advantageous when considering Krylov subspace methods which will be discussed in the next sections.

4.3.2 Krylov Subspace Methods for Symmetric Positive Definite Matrices

In the last section it was observed that the iterates x^k are computed by

$$x^{k+1} = x^k + B^{-1}r^k. \tag{4.12}$$

Hence starting with a given x^0 yields

$$\begin{aligned} x^1 &= x^0 + B^{-1}r^0 \\ x^2 &= x^1 + B^{-1}r^1 = x^0 + B^{-1}r^0 + B^{-1}(b - Ax^1) \\ &\quad x^0 + B^{-1}r^0 + B^{-1}b - B^{-1}A(x^0 + B^{-1}r^0) = x^0 + 2B^{-1}r^0 - B^{-1}A(B^{-1}r^0) \\ &\quad \vdots \end{aligned}$$

By induction it can be proven that

$$x^k \in x^0 + \text{Span}\left\{B^{-1}r^0,\ B^{-1}A(B^{-1}r^0), \ldots, (B^{-1}A)^{k-1}(B^{-1}r^0)\right\} \tag{4.13}$$

Definition 4.1 *The subspace*

$$K^k(A; r^0) = \textit{Span}\left\{r^0,\ A(r^0), \ldots, A^{k-1}(r^0)\right\} \tag{4.14}$$

is called the Krylov-*space of dimension k corresponding to the matrix A and the initial residual r^0.*

Hence, the k^{th} iterate of the Jacobi method is an element of

$$x^k \in x^0 + K^k(B^{-1}A;\ B^{-1}r^0). \tag{4.15}$$

In the next section the conjugate gradient method will be introduced.

4.3.2.1 The Conjugate Gradient Method

The conjugate gradient (CG) method starts with an initial guess x^0 and computes the first iterate x^1, which is an element of the Krylov subspace $K^1(A; r^0) = \text{Span}\{r^0\} = \alpha_0 r^0$ by minimizing

$$\|x - x^1\|_A. \tag{4.16}$$

The A-norm is induced by the inner product

$$< x, y >_A = x^\top A\, y. \tag{4.17}$$

In order to define the inner product and norm correctly it is necessary that the matrix A is symmetric positive definite.

Definition 4.2 *A matrix is called symmetric positive definite (SPD) if it satisfies*

(i) $A = A^\top$,

(ii) $x^\top Ax > 0, \qquad \forall x \neq 0.$

Now consider the square of the norm in (4.16).

$$\|x - x^1\|_A^2 = \|x - \alpha_0\, r^0\|_A^2 = (x - \alpha_0\, r^0)^\top A\,(x - \alpha_0\, r^0) = \\ x^\top A\, x - 2\alpha_0\,(r^0)^\top \overbrace{A\, x}^{=b} + \alpha_0^2 (r^0)^\top A\, r^0. \quad (4.18)$$

Note that the squared norm is a quadratic function in α_0 :

$$f(\alpha_0) = \Big((r^0)^\top A\, r^0\Big)\, \alpha_0^2 - \Big(2(r^0)^\top\, b\Big)\, \alpha_0 + x^\top A\, x. \quad (4.19)$$

To find the minimum set $f'(\alpha_0) = 0$, hence the minimum is attained for

$$\alpha_0 = \frac{(r^0, b)}{(r^0, r^0)_A}. \quad (4.20)$$

In subsequent iterates will be computed by minimizing $\|x - x^k\|_A$ over the Krylov subspace $K^k(A; r^0)$

$$\|x - x^k\|_A = \min_{y \in K^k(A;r^0)} \|x - y\|_A. \quad (4.21)$$

The method can be written in template form as (see reference 2)

Conjugate Gradient Method

Compute $r^0 = b - Ax^0$ for some initial guess x^0

for $k = 1, 2, \ldots$

  if $k = 1$

    $p^1 = r^0$

  else

    $\beta_{k-1} = (r^{k-1})^\top r^{k-1} / (r^{k-2})^\top r^{k-2}$

    $p^k = r^{k-1} + \beta_{k-1} p^{k-1}$

  end

  $q^k = Ap^k$

  $\alpha_k = (r^{k-1})^\top r^{k-1} / (p^k)^\top\, q^k$

  $x^k = x^{k-1} + \alpha_k\, p^k$

  $r^k = r^{k-1} - \alpha_k\, q^k$

  Check convergence; continue if necessary

end

Vector Properties in CG The properties of the vectors used in the method are listed below. These properties are very important and follow (among other things) from the fact that the matrix A is SPD.

(i) $(r^j)^\top r^i = 0$ for $i = 1, \ldots, j-1$ and $j = 1, \ldots, k$.

(ii) $(r^j)^\top p^i = 0$ for $i = 1, \ldots, j-1$ and $j = 1, \ldots, k$.

(iii) $(p^j)^\top A\, p^i = 0$ for $i = 1, \ldots, j-1$ and $j = 2, \ldots, k$.

The first property states that the residuals are mutually orthogonal. the second one states that the residuals and search directions are also orthogonal. The last property is the A-conjugate orthogonality of the search directions, and hence that is where the name of the method originates from.

Convergence If the iterates are defined as in equation (4.21), the question rises how fast the method converges. To answer this question the following theorem from reference 18 is invoked.

Theorem 4.1 *The iterates x^k obtained from the CG algorithm satisfy the inequality*

$$\|x - x^k\|_A \leq 2 \left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k \|x - x^0\|_A$$

Theorem 4.1 shows that the CG method is linear convergent. But in general the convergence is superlinear. This means that the upper estimate for convergence only applies for the first few iterates. During these first iterates the convergence depends on the condition number of the matrix A. After a number of iterates, this condition number is replaced by the *effective* condition number which is smaller, resulting in faster convergence.

Note that after $n(= \dim(A))$ iterates, the Krylov subspace has become identical to $\mathbb{R}^n$. Hence $x^n = x$ and the exact solution is obtained. In practise this is not guaranteed due to the presence of rounding errors. It is also not of practical interest, since n is often very large.

Work Formally, the amount of work involving an iterative method is computed by the amount of work during one iteration times the total number of iterations. Therefore, it is advantageous to have as little work as possible in every iteration. The extra work excluding a matrix vector product is neglectable. Therefore, the work per iteration is defined as the number of matrix vector products.

For the CG method one matrix vector product is done to compute q^k ($= 2n^2$ flops) if A is a fully populated matrix. In practise, the matrix A is sparse and hence the work is reduced due to a lot of zero entries in the matrix. For a sparse matrix of size n the amount of work to perform a matrix vector product is limited by $2np$ for p the maximum number of nonzeros in the rows of A.

Example To illustrate the method an example is given. Consider the system

$$Ax = b, \tag{4.22}$$

where A is the matrix obtained from the standard second order finite difference discretization of the Poisson operator in two dimensions on a square. $n = 625$ nodes are used, hence $A \in \mathbb{R}^{625\times625}$. The nodes are regularly spaced over the square. The resulting matrix has a sparse structure which can be written as follows

$$A = ((1/h^2)[-e, -e, 4e, -e, -e], [-\sqrt{n} - 1, 0, 1, \sqrt{n}], n, n) \tag{4.23}$$

where e is an all ones vector and

$$[-e, -e, 4e, -e, -e]$$

represent the diagonals.

$$[-\sqrt{n}, -1, 0, 1, \sqrt{n}]$$

represent the place of these diagonals, with 0 being the main diagonal and 1 the first upper diagonal.

This matrix is SPD according to reference 18, hence we are allowed to apply the CG method, after specifying the right-hand side vector b. In this example $b = e$.

Figure 4.1 depicts the convergence behavior of the method for this particular example. Horizontal is the number of matrix vector products (MAT-VEC-OP), and vertical the logarithm of the residual $10 \log \|r^k\|_2$. The tolerance level is $tol = 10^{-3}$, and the stopping criterium is $\|r^k\|_2/\|b\|_2 \leq tol$.

4.3.3 Krylov Subspace Methods for General Matrices

As has been seen in the derivation of the CG method, the method has three favorable properties

(a) The iterate x^k is an element of the Krylov subspace $K^k(A; r^0)$.

(b) The error $x - x^k$ is minimized with respect to the A-norm.

(c) The method uses only short recurrences.

But to obtain these nice properties the matrix A has to be SPD.

For a general matrix, there is no method for which these three properties (a)-(c) are satisfied, so a choice has to be made for the problem under consideration, which property to drop and

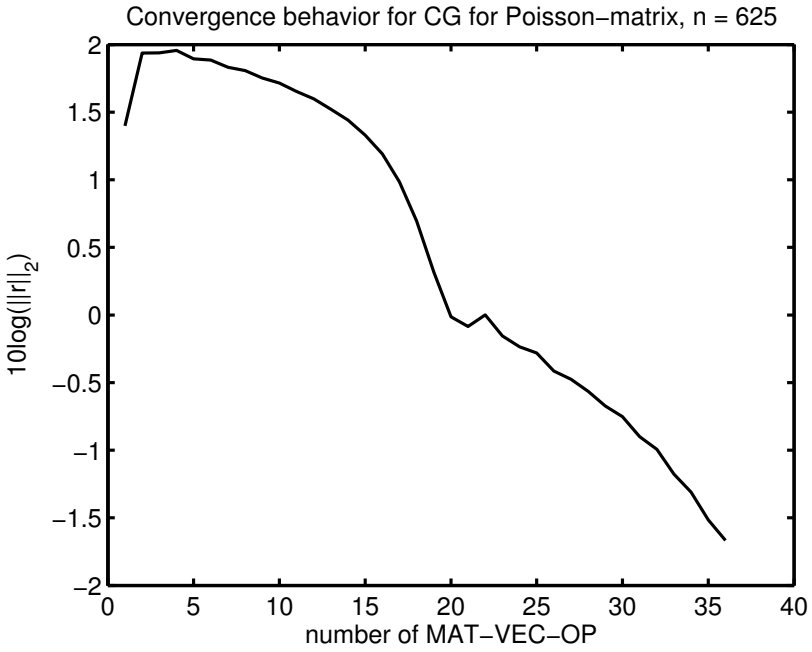

Fig. 4.1 The convergence of the CG method for the Poisson equation in two dimensions with $n = 625$. Note that the residuals are not monotonically decreasing which is consistent with the theory.

which not. Nowadays, there are three ways to solve a system for a general (non-singular) matrix (Ref. 18).

(1) Solve the normal equations $A^\top A\, x \;=\; A^\top\, b$ with the CG method. Now the iterate is no longer an element of $K^k(A; r^0)$, but an element of $K^k(A^\top A; A^\top r^0)$.
(2) Construct a basis for the Krylov subspace by a three-term bi-orthogonality relation, loosing the minimization property.
(3) Make all the residuals explicitly orthogonal in order to have an orthogonal basis for the Krylov subspace, which implies long recurrences.

Each of these type of methods will be discussed in the next three subsections.

4.3.3.1 Normal Equations and CG

In this section a general non singular matrix A is considered. Using CG is not permitted in this case since the matrix is in general not SPD. A SPD matrix B is constructed from A by

$$B = A^\top A. \tag{4.24}$$

Then B is symmetric and positive definite. Since

$$B_{ij} = (A^\top A)_{ij} = \sum_{k=1}^{n} A_{ik}^\top A_{kj} = \sum_{k=1}^{n} A_{ki} A_{jk}^\top = \sum_{k=1}^{n} A_{jk}^\top A_{ki} = (A^\top A)_{ji} = B_{ji}, \tag{4.25}$$

and

$$y^\top By = y^\top A^\top Ay = (Ay)^\top Ay = \|Ay\|_2^2 > 0, \tag{4.26}$$

for $y \neq 0$ and A invertible.

Hence B is SPD. The CG method may now be used to solve the system

$$A^\top Ax = A^\top b. \tag{4.27}$$

This method is known as CGNR, *the Conjugate Gradient on the Normal equations to minimize the Residual.* This name is explained by the following observation (Ref. 11). Let the exact solution be given by x

$$\|x - x^k\|_{A^\top A}^2 = (x - x^k)^\top A^\top A(x - x^k) = \\ (Ax - Ax^k)^\top (Ax - Ax^k) = (b - Ax^k)^\top (b - Ax^k) = \|r^k\|_2^2. \tag{4.28}$$

Hence, the method minimizes the residual and this implies a monotone decreasing sequence of residuals. Note that r^k is the residual of the original system

$$r^k = b - Ax^k. \tag{4.29}$$

Convergence All the convergence properties discussed previously for the CG method are applicable here, but one should remember that the method is applied to another matrix this time. The rate of convergence of CG depends on the condition number of the matrix, hence it depends on $\kappa_2(B) = \kappa_2(A)^2$. Therefore slow convergence may be expected. Thereby, the convergence itself depends on the eigenvalues of the matrix, in this case the convergence depends on the square of the singular values of A. Finally, the rounding errors also depend on $\kappa_2(A^\top A)$.

Work Per iteration a multiplication with A and $A^\top$ is performed. The amount of work done per iteration is therefore approximately twice the work involved in CG. The multiplication by $A^\top$ may cause problems especially when the multiplications are row wize vectorized implemented.

4.3.3.2 BiCG and Bi-CGSTAB

BiCG constructs a basis for the Krylov subspace $K^k(A; r^0)$ using three-term bi-orthogonality relations. In comparison with the CG method the minimization property is lost. To construct the Krylov subspace basis two sequences of residuals are used, one depending on the matrix A and the other depends on $A^\top$.

The two residual sequences $\{r^k\}$ and $\{s^k\}$ are updated following

$$r^k = r^{k-1} - \alpha_k A\, p^k,$$
$$s^k = s^{k-1} - \alpha_k A^\top q^k.$$

In this update two search directions p^k, q^k are used which in turn are updated beforehand by

$$p^k = r^{k-1} + \beta_{k-1} p^{k-1},$$
$$q^k = s^{k-1} + \beta_{k-1} q^{k-1}.$$

The scalars α_k and β_{k-1} are chosen in such a way that the bi-orthogonality between the sequences $\{r^k\}, \{s^k\}$ is ensured. It follows that they satisfy

$$\alpha_k = \frac{(r^{k-1})^\top s^{k-1}}{(q^k)^\top A p^k}, \qquad \beta_{k-1} = \frac{(r^{k-1})^\top s^{k-1}}{(r^{k-2})^\top s^{k-2}} \tag{4.30}$$

The bi-orthogonality then reads

$$(s^k)^\top r^l = 0, \qquad (q^k)^\top A p^l = 0, \quad \text{for } k \neq l. \tag{4.31}$$

This method can be written in template form as (see reference 2)

BiCG Method

Compute $r^0 = b - Ax^0$for some initial guess x^0.
Choose s^0 such that $(s^0, r^0) \neq 0$ (e.g. $s^0 = r^0$).
for $k = 1, 2, \ldots$
  $\rho_{k-1} = (r^{k-1})^\top s^{k-1}$
  if $\rho_{k-1} = 0$ failure: division by zero.
  if $k = 1$
    $p^k = r^k$
    $q^k = s^k$
  else
    $\beta_{k-1} = \rho_{k-1}/\rho_{k-2}$
    $p^k = r^k + \beta_{k-1} p^{k-1}$
    $q^k = s^k + \beta_{k-1} q^{k-1}$
  end
  $u^k = Ap^k$
  $v^k = A^\top q^k$
  $\alpha_k = \rho_{k-1}/(q^k)^\top u^k$
  $x^k = x^{k-1} + \alpha_k p^k$
  $r^k = r^{k-1} - \alpha_k u^k$
  $s^k = s^{k-1} - \alpha_k v^k$
  Check convergence; continue if necessary
end

Convergence There are only a few theoretical results known for the BiCG method. For a symmetric positive definite method, BICG delivers the same iterates as CG (Ref. 18). For nonsymmetric matrices the convergence is comparable to full GMRES (to be discussed briefly in the next section) when the norm of the residuals reduces significantly. On the other hand it is observed that the convergence behavior can be quite irregular. The breakdown due to nearly zero $\rho_{k-1} = (r^{k-1})^\top s^{k-1}$ may be catched by restarting the method after the iteration right before break down.

Work Per iteration two matrix vector products have to be done (one with A and one with $A^\top$), hence the work is proportional to CGNR and twice as much as for CG.

Note that it has been observed that the residual r^k of the BiCG method can be seen as a product of r^0 and a polynomial of degree k in A, hence

$$r^k = P_k(A)r^0. \tag{4.32}$$

This polynomial also satisfies

$$s^k = P_k(A^\top)s^0 \tag{4.33}$$

hence

$$(s^k, r^k) = (P_k(A^\top)s^0,\ P_k(A)r^0) = (s^0,\ P_k^2(A)r^0). \tag{4.34}$$

This observation suggests that if one application of the operator $P_k(A)$ reduces r^0 to a smaller residual r^k, it might be worthwile to consider application twice and compute $P_k^2(A)r^0$. This yields the method *Conjugate Gradient Squared* or shortly CGS.

But the method considered here does not apply the same polynomial $P_k(A)$ twice, but constructs another polynomial $Q_k(A)$ and applies this one to r^0 after $P_k(A)$. This new polynomial Q is formed by

$$Q_k(x) = (1 - \omega_1\, x) \ldots (1 - \omega_k\, x), \tag{4.35}$$

for suitable constants ω_j. This method is called Bi-CGSTAB, where STAB stands for stabilized. The template reads (Ref. 2)

Bi-CGSTAB Method

Compute $r^0 = b - Ax^0$ for some initial guess x^0.
Choose $\tilde{r}$ (e.g. $\tilde{r} = r^0$).
for $k = 1, 2, \ldots$
  $\rho_{k-1} = \tilde{r}^\top r^{k-1}$
  if $\rho_{k-1} = 0$ failure: division by zero.
  if $k = 1$
    $p^1 = r^0$
  else
    $\beta_{k-1} = (\rho_{k-1}/\rho_{k-2})(\alpha_{k-1}/\omega_{k-1})$
    $p^k = r^{k-1} + \beta_{k-1}(p^{k-1} - \omega_{k-1}v^{k-1})$
  end
  $\hat{p} = p^k$
  $v^k = A\hat{p}$
  $\alpha_k = \rho_{k-1}/\hat{r}^\top v^k$
  $s = r^{k-1} - \alpha_k\, v^k$
  Check norm of s; if small enough: set $x^k = x^{k-1} + \alpha_k\hat{p}$ and stop.
  $\hat{s} = s$
  $t = A\hat{s}$
  $\omega_k = (t^\top s)/(t^\top t)$
  $x^k = x^{k-1} + \alpha_k\hat{p} + \omega_k\hat{s}$
  $r^k = s - \omega_k\, t$
  Check convergence; continue if necessary
  For continuation it is necessary that $\omega_k \neq 0$
end

Convergence Bi-CGSTAB does not have the optimality property. Therëfore, no theoretical results with respect to convergence can be found. In practice it has been observed that the convergence behavior is on average twice as fast as BiCG for a large class of problems, but the influence of rounding errors can be more significant.

Work Just like BiCG, Bi-CGSTAB needs to compute two matrix vector products every iteration, but only multiplication by the matrix A itself is necessary. Hence the amount of work is roughly the same as for BiCG but Bi-CGSTAB is preferable if $A^\top$ is not available, which depends on the specific implementation.

4.3.3.3 GMRES and GCR

In the last section the methods BiCG and Bi-CGSTAB were discussed. These methods do have short recurrences, but do not minimize the residual. Therefore, little is known about the convergence behavior of these methods. Alternatively, algorithms can be derived that construct an orthonormal basis for the Krylov subspace explicitly. Examples of these methods are GMRES and GCR. GMRES is in practice the most robust method, but the memory requirement increases linearly with the number of iterations. Therefore, the method may be restarted after a number (m) of iterations. After a short introduction of GMRES, GCR will be discussed. Although GCR takes twice the memory of GMRES, it is admitted here nonetheless because it is easy to implement, flexible and can be truncated.

For the template of the GMRES method the reader is refered to (Ref. 2) (p. 20). The difficult part in using GMRES properly is of course the value of m, the number of iterations after the method is restarted. Choosing m too small may not lead to a converging algorithm, but choosing m too large results in an enormous storage of orthonormal residual vectors.

The template for GCR can be found in reference 18 (p. 85) and reads

GCR Method

Compute $r^0 = b - Ax^0$ for some initial guess x^0.
for $k = 1, 2, \ldots$
  $s^k = r^{k-1}$
  $v^k = As^k$
  for $j = 1, \ldots, k-1$
    $\alpha = (v^j, v^k)$
    $v^k = v^k - \alpha v^j$
    $s^k = s^k - \alpha s^j$
  end
  $s^k = s^k / \|v^k\|_2$
  $v^k = v^k / \|v^k\|_2$
  $x^k = x^{k-1} + (v^k, r^{k-1}) s^k$
  $r^k = r^{k-1} - (v^k, r^{k-1}) v^k$
end

Note that the order of the inner product is changed in this template in comparison with reference 18 (p.85). This is done in order to be able to apply this method for complex matrices.

Convergence The rates of convergence for GCR and GMRES are comparable. But as mentioned before, GMRES is a very robust method. Indeed there are occasions where GCR may break down, where GMRES will not.

Work The work per iteration is one matrix vector product. So this is comparable with the CG method. But in this algorithm the orthogonal basis for the Krylov subspace is constructed explicitly which is very memory expensive if a large number of iterations are needed. Therefore, the method may be restarted after a number of iterations, loosing the constructed orthogonal basis. A nice last property is that truncated GCR performs better than restarted GMRES.

4.3.4 A Krylov Subspace Method for a Symmetric Complex Matrix

In this section a method that may be particularly useful for complex symmetric matrices is analysed. The method was first proposed in (Ref. 17).

4.3.4.1 COCG

The method to be considered here is called *conjugate orthogonal conjugate gradients* (COCG). This method is developed especially for symmetric complex systems. Since the final problem deals with an almost symmetric complex matrix this method might be very useful. The basic idea of this method is to replace the standard Hermitian inner product, given by

$$(x, y) = \sum_j \bar{x}_j \, y_j \tag{4.36}$$

by a conjugate orthogonality relation

$$< u, v >= \sum_j u_j \, v_j. \tag{4.37}$$

The residuals are made mutually orthogonal using the new inner product and a basis for the Krylov subspace is generated. The idea of the method is roughly the same as CG.

The COCG template reads (Ref. 17)

COCG Method

x^0 given; $v^0 = b - Ax^0$
$p^{-1} = 0; \quad \beta_{-1} = 0;$
$w^0 = v^0$
$\rho_0 = (\bar{v}^0, w^0)$
for $k = 0, 1, 2, \ldots$
  $p^k = w^k + \beta_{k-1} p^{k-1}$
  $u^k = Ap^k$
  $\mu_k = (\bar{u}^k, p^k)$; **if** $\mu_k = 0$ then quit (failure)
  $\alpha_k = \rho_k / \mu_k$
  $x^{k+1} = x^k + \alpha_k p^k$
  $v^{k+1} = v^k - \alpha_k u^k$
  if x^{k+1} is accurate enough then quit (convergence)
  $w^{k+1} = v^{k+1}$
  $\rho_{k+1} = (\bar{v}^{k+1}, w^{k+1})$; **if** $|\rho_{k+1}|$ small then quit (failure)
  $\beta_k = \rho_{k+1} / \rho_k$
end

For a real valued matrix the COCG reduces to the CG method, which can be seen by comparing the two templates of both methods.

Convergence It is written in reference 17 that the method has a convergence behavior similar to BiCG, but this is not proven there.

Work The COCG method only requires one matrix vector product per iteration. In comparison with the other methods (not GCR) that have been discussed this is only half the work per iteration. But the multiplication of two complex numbers is 4 times as expensive as the product of two real numbers. In the discretization matrix, the upper left corner is completely real valued[1] as it corresponds to the finite elements inside the cavity. The complexity in the matrix is due to the presence of the Green's function. Therefore, a significant reduction of work is possible if the real and complex valued parts of the matrix are treated separately.

[1] If radar absorbing materials are used this is no longer the case: The permittivity $\varepsilon \in \mathbb{C}$ and parts of the matrix become complex valued.

4.3.5 Multigrid Method

The multigrid method, as the name indicates performs iterations of an iterative method on different grids. These grids differ in gridsize and for regular grids the grids are generated by subsequently doubling the gridsize until the problem can be solved directly. This is the case if the number of nodes is of $O(100)$. Multigrid is driven by two basic principles

(1) Basic iterative methods have a strong *smoothing effect* on the error of any approximation, that is, on $\varepsilon = u_h - u_h^i$. Where u_h is the solution of the discretized system and u_h^i is an approximation of u_h after i iterations.

(2) *Coarse grid correction* (CGC): Any smooth quantity on a fine grid can be well approximated on a coarse grid.

To explain the multigrid method, *twogrid* is considered first. This method uses only two grids, a fine and a coarse grid. Consider a discretized differential operator L^h. The corresponding system with Dirichlet boundary conditions is written as

$$\begin{cases} (L^h u^h)_i = f_i, & \text{for } i = 1, \ldots, N-1 \\ (u^h)_0 = u_l, \text{ and } (u^h)_N = u_r, \end{cases} \tag{4.38}$$

and assume that N is a power of 2. The fine grid is now defined as the grid with meshwidth $h = 1/N$ and nodes $1, \ldots, N-1$. the coarse grid is defined as the grid with meshwidth $2h$ and nodes $1, \ldots, \frac{N}{2} - 1$ (see Figure 4.2).

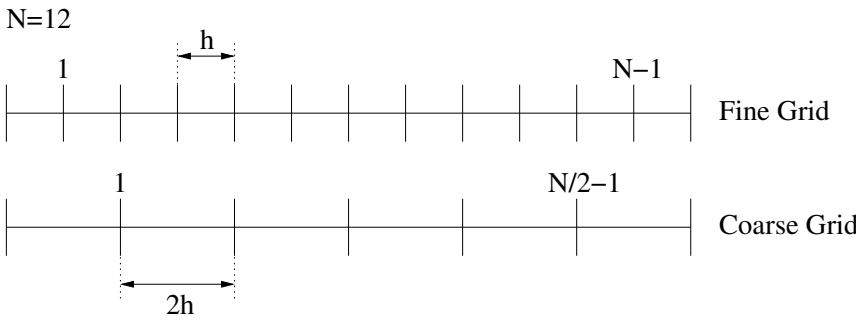

Fig. 4.2 The fine grid (left) and the coarse grid (right).

Assume also that $\tilde{u}$ is the approximation of u^h after a number of basic iterative steps performed on the fine grid. The basic iterative method that may be used is damped Jacobi, since according to reference 19 this method smooths the *algebraic error*. This error is defined by

$$e^h = u^h - \tilde{u}. \tag{4.39}$$

It is possible (Ref. 19) to find a system for the algebraic error by subtracting a trivial system from the system in (4.38). This trivial system is given by

$$\begin{cases} (L^h\tilde{u}^h)_i = (L^h\tilde{u}^h)_i, \quad \text{for } i = 1, \ldots, N-1 \\ (\tilde{u}^h)_0 = u_l, \text{ and } (\tilde{u}^h)_N = u_r, \end{cases} \tag{4.40}$$

Subtracting system (4.40) from (4.38) yields

$$\begin{cases} (L^h e^h)_i = f_i^h - (L^h\tilde{u}^h)_i, \quad \text{for } i = 1, \ldots, N-1 \\ (e^h)_0 = 0, \text{ and } (e^h)_N = 0, \end{cases} \tag{4.41}$$

where r^h is the residual vector. Then this system is solved on the coarse grid using a direct method and a solution e^h is found. This vector is then prolongated to the fine grid and the iterate $\tilde{u}$ is improved following

$$\tilde{u} := \tilde{u} + e^h. \tag{4.42}$$

In order to be able to *transfer information* between the fine and coarse grids, two operators are defined: the *prolongation* operator I_{2h}^h and the *restriction* operator I_h^{2h}. The restriction operator is used for transfering from *fine-to-coarse* and the prolongation operator is used for transfering from *coarse-to-fine* (Ref. 19).

Suppose now that multiple grids are introduced and the same idea as described above is performed on these grids subsequently. The solution process can be given schematically by Figure 4.3. The template for the multigrid routine is given by (Ref. 19)

Multigrid algorithm

If the coarsest grid is reached, solve $L^h u^h = f^h$ and return u^h, otherwise
- Apply relaxation v_1 times yielding u^h
- Compute the residual vector $r^h = f^h - L^h u^h$
- Restrict the residual $f^{2h} = I_h^{2h} r^h$
- Set $u^{2h} = 0$
  - Treat coarse grid problem recursively until coarsest grid is reached
  - Interpolate and add correction $u^h := u^h + I_{2h}^h u^{2h}$
- Apply relaxation v_2 times yielding u^h.

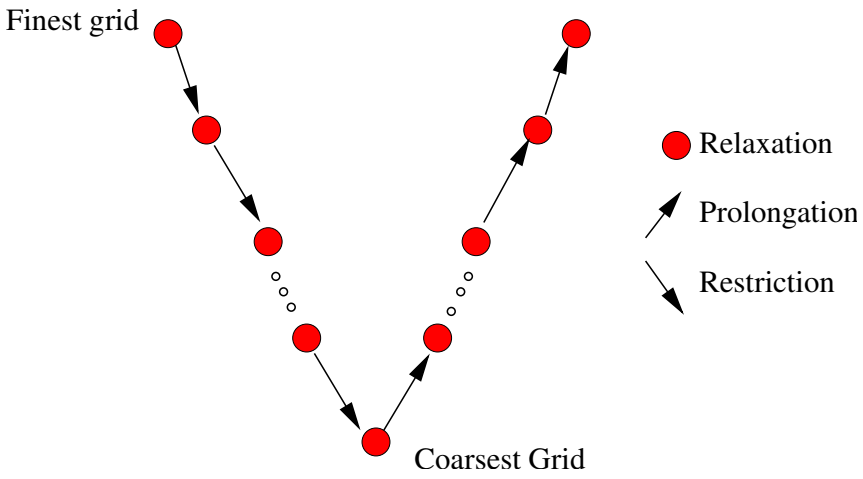

Fig. 4.3 A schematic description of the multigrid method. The red circles are relaxations*, which stands for the application of a number of steps with damped Jacobi. The downward arrows are applications of the restriction operator, the upward arrows indicate the prolongation operator.*

4.4 Preconditioning

In the last section it was observed that for a SPD matrix the convergence behavior of the Krylov subspace based methods depend on the eigenvalue distribution of the coefficient matrix A. For a general matrix this property holds for a large class of matrices, but there are exceptions. Loosely speaking, if the eigenvalue spectrum is favorable convergence is fast and if not, convergence may be very slow. A *preconditioner* can be used to speed up the rate of convergence of a system. Before this is discussed in detail, we introduce the definition of a preconditioner (Ref. 18).

Definition 4.3 *A preconditioner is a matrix M that transforms the linear system $Ax = b$ such that the transformed system $M^{-1}Ax = M^{-1}b$ has the same solution, but the transformed matrix $M^{-1}A$ has a more favorable eigenvalue distribution.*

In general the spectrum for a coefficient matrix is favorable, if the eigenvalues are clustered around 1. So if a preconditioner is used, it is desirable that the transformed system, with coefficient matrix $M^{-1}A$, has its eigenvalues around 1. Furthermore, the solution y of the system $My = c$ has to be computed every iteration, so this must be cheap and easy to perform.

4.4.1 Preconditioners, choice of M

In this section several preconditioners will be introduced and it is shown how they are constructed. Roughly speaking, there are two approaches to construct a preconditioner. The first approach is the construction of a preconditioner that is optimal in some way, but only applicable to a small range of problems. This is called the *application-specific* approach. The user of this type of

methods is required to have a great knowledge of the underlying physical problem. For a general problem this approach may not lead to a satisfactory preconditioner. To avoid this a preconditioning technique is needed which is widely applicable. These techniques are not optimal, but perform reasonably well on a wide range of problems. This approach is called the *algebraic approach*.

In general the algebraic approach is used initially. But as the expertise in the structure of the matrix increases, a more optimal and sophisticated preconditioner can be constructed. In the following the description of some algebraic methods as incomplete factorizations, block factorization, polynomial preconditioning, approximate inverses and the *shifted Laplace operator* are discussed.

4.4.1.1 Incomplete Factorization

Assume the matrix A in the system $Ax = b$ is sparse and the LU-decomposition is used to solve the system by subsequently solving

$$Ly = b, \quad \text{and} \quad Ux = y. \tag{4.43}$$

Since the matrices L and U are triangular, the systems in (4.43) are solved by backward and forward substitution respectively. But in general the matrices L and U are not as sparse as the matrix A, due to *fill-in* during the decomposition.

Definition 4.4 *The nonzero elements in the factored matrices L and U that were zero in the original matrix A are called* fill-in.

Hence the sparseness in A is lost and that deteriorates the efficiency of solving the system. The idea of incomplete factorization is to discard the fill-in partly or fully. This discarding is done by choosing a set $\mathcal{S}$ of matrix positions. $\mathcal{S}$ is used to set all positions outside $\mathcal{S}$ to zero during the factorization. An incomplete factorization step is then described by reference 3

$$a_{ij} = \begin{cases} a_{ij} - a_{ik}a_{kk}^{-1}a_{kj}, & \text{if } (i,j) \in \mathcal{S}, \\ a_{ij} & \text{otherwise}, \end{cases} \tag{4.44}$$

for each k and for $k < i, j$.

The most straightforward way to define $\mathcal{S}$ is

$$\mathcal{S} = \{(i,j) \mid a_{ij} \neq 0\}, \tag{4.45}$$

hence $\mathcal{S}$ is the sparsity pattern of A itself and the factorization is called ILU(0) or no-fill ILU factorization. This factorization is effective for M-matrices and diagonally dominant matrices. In our case the matrix[2] is neither diagonally dominant nor is it an M-matrix, hence more sophisticated preconditioners are needed for which $\mathcal{S} \supset \{(i,j) \mid a_{ij} \neq 0\}$. This gives rise to the development of strategies for fill-in.

In general there are two types of fill-in strategies. The first type is based on the so-called *levels of fill-in*. The second type is based on a numerical *drop tolerance*. These two methods are described below.

Levels of Fill-in During the incomplete factorization process the adjusted matrix elements are numbered: their *fill-in level*. If an element is modified this level changes. After the factorization all elements with fill-in level higher then a tolerance level l are discarded. Formally, an initial level for each matrix entry is defined by

$$lev_{ij} = \begin{cases} 0, & \text{if } a_{ij} \neq 0 \quad \text{or } i = j, \\ \infty, & \text{otherwise.} \end{cases} \tag{4.46}$$

If a matrix entry is modified during the factorization process its level is adjusted according to

$$lev_{ij} = \min\{lev_{ij},\ lev_{ik} + lev_{kj} + 1\}. \tag{4.47}$$

Roughly speaking, a zero in the original matrix has a higher level if it is made nonzero in a later iteration of the factorization process.

For $l = 0$ the no-fill ILU(0) is obtained. In many applications the ILU(1) performs much better than ILU(0) while this is not very expensive to use. For very difficult problems it might pay to use a high value of l, but the tradeoff between efficiency in the method and the computer storage costs has to be made.

Drop Tolerance For matrices that are badly scaled or far from diagonally dominant, the ILU(l) might use a lot of memory for storing matrix entries that are very small in absolute value. Since they are that small they do not contribute that much and hence, it is inefficient to use them. So these fill-ins are thrown away if they are smaller than a certain value. A dropping criterion is then obtained.

There are two types of drop tolerance:

- *absolutely* drop tolerance accepts new fill-ins only if they are greater then τ.

[2] Properties of this matrix are discussed in Chapter 6

- *relatively* drop tolerance accepts the new fill-ins if they are greater than $\tau\|a_i\|_2$, where a_i is the vector with the elements of the ith row when the ith row is treated.

A difficulty in this approach is the choice of the parameter τ and the resulting amount of storage that will be required for the incomplete LU-factors.

Modified Incomplete Factorization If incomplete factorizations are used it can be shown that the eigenvalues are clustered around 1 and there are some small eigenvalues of order $O(h^2)$. If modified incomplete factorization is used, the preconditioner M is required to have the same rowsums as the original matrix A. For this preconditioner it can be proven that nearly all the eigenvalues are clustered around 1 and the largest eigenvalues are of order $O(1/h)$. Hence the condition number is also $O(1/h)$.

A combination of the two methods described above yields the relaxed incomplete factorization. A relaxation factor $0 \leq \alpha \leq 1$ is used to cluster the eigenvalues better around 1.

4.4.1.2 Block Factorizations and Block Preconditioners

The block variant of the LU-factorizations is a first start to constructing a block preconditioner. In order to construct such a preconditioner, the concept of the *Schur* complement of a matrix is introduced.

Definition 4.5 *Suppose $A_{11}, A_{12}, A_{21}, A_{22}$ are respectively n×n, m×n, n×m, m×m matrices and A_{11} is invertible. Let*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}. \tag{4.48}$$

Then the Schur complement *of block A_{11} is defined as the m×m matrix*

$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}. \tag{4.49}$$

Schur complements originate from the Gaussian elimination procedure applied to block matrices. Indeed, if the matrix A is defined as above, the block LU-decomposition of A may be written as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I_n & 0 \\ A_{21}A_{11}^{-1} & I_m \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix}, \tag{4.50}$$

where S is the Schur complement of A_{11}. The idea is to use the matrix

$$P_T = \begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix} \tag{4.51}$$

as a preconditioner. According to reference 4 the spectrum of the matrix AP_T^{-1} is equal to

$$\sigma(AP_T^{-1}) = \sigma(P_T^{-1}A) = \{1\}. \tag{4.52}$$

Hence the only eigenvalue is $\lambda = 1$ with multiplicity $m + n$.

The same kind of derivation can be performed for an SPD block matrix and the associated Cholesky factorization. Since our matrix is not SPD this is not treated here.

4.4.1.3 Polynomial Preconditioning

Polynomial preconditioners can be obtained in several ways. The first k iterates of a basic stationary iterative method can be seen as a preconditioner, but another perspective is using the Neumann series. Below each of these methods will be explained.

Polynomial Preconditioning and Jacobi In Section 4.3.1 it was observed that the $k + 1$th iterate is computed by (comp. (4.7))

$$x^{k+1} = D^{-1}b - D^{-1}Cx^k. \tag{4.53}$$

If $G = D^{-1}C$ and $x^0 = 0$ is chosen, the iterate after $k + 1$ iterates reads

$$x^{k+1} = (I - G - G^2 - \ldots - G^k)D^{-1}b. \tag{4.54}$$

If $M^{-1} = (I - G - G^2 - \ldots - G^k)D^{-1}$ is chosen then $Mx = b$ hence, the matrix M can be seen as a preconditioner.

Polynomial Preconditioning and Neumann series The second method described considers the following. Suppose the matrix A can be written as $A = I - B$ and suppose as well that $\sigma(B) < 1$. Then the Neumann series can be used to write the inverse of A as

$$A^{-1} = \sum_{k=0}^{\infty} B^k. \tag{4.55}$$

Hence, A^{-1} may be approximated by truncating this infinite series.

4.4.1.4 Approximate Inverse

If it is possible to find a matrix M that is close to A^{-1}, the eigenvalue distribution of MA is clustered around 1. In the *approximate inverse* technique, the idea is to find a sparse matrix M that is in some sense close to A^{-1}. This might be difficult since the inverse matrix of a sparse one is normally dense. But in general there are very much entries in the inverse matrix very small in comparison to the main diagonal entries, hence a possibility is to discard all these values and

obtain a sparse matrix that approximates A^{-1}. This is not considered here, but two commonly used methods are discussed: *Frobenius norm minimization* and *incomplete (bi)-conjugation* (see e.g. reference 3).

Frobenius norm minimization In the Frobenius norm minimization method a matrix M that is close to A^{-1} is constructed according to

$$\min_{M \in \mathcal{B}} \|I - AM\|_F \tag{4.56}$$

where $\mathcal{B}$ is a class of sparse matrices. It is observed that for $M \approx A^{-1}$ the norm $\|I - AM\|_F$ is small, since $AM \approx I$. Using the identity

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_j\|_2^2, \tag{4.57}$$

it follows that solving the constraint minimization problem (4.56) is equivalent to solving n independent linear minimization problems, with a sparsity constraint.

Note that in the above description a right preconditioner M is computed. It is also possible to compute a left preconditioner by considering

$$\|I - MA\|_F = \|(I - MA)^\top\|_F = \|I - A^\top M^\top\|_F. \tag{4.58}$$

Hence the left preconditioner is computed as a right preconditioner for the transposed matrix $A^\top$ and then the transpose of $M^\top$ is taken.

In the rest of this section it is assumed that the class of nonzero patterns $\mathcal{B}$ is given and fixed. Then it is straightforward to compute the preconditioning matrix M and this is explained below (Ref. 3). The nonzero pattern is a subset $\mathcal{G} \subseteq \{(i,j) \,|\, 1 \leq i,j \leq n\}$, hence $m_{ij} = 0$ for $(i,j) \notin \mathcal{G}$. The constraint set $\mathcal{B}$ is then simply the set of all real valued matrices with nonzero pattern in $\mathcal{G}$

$$\mathcal{B} = \{M \in \mathbb{R}^{n \times n} \,:\, m_{ij} = 0 \Leftrightarrow (i,j) \in \mathcal{G}\} \tag{4.59}$$

Therefore, let m_j be the jth column of M and define the nonzero pattern of this vector by

$$\mathcal{J} = \{i \,|\, (i,j) \in \mathcal{G}\}. \tag{4.60}$$

The following step is to simplify the multiplication of A with m_j, since advantage must be taken of the sparseness of both. For the multiplication of Am_j the columns of A are needed with indices from $\mathcal{J}$. Hence the matrix with these columns is defined by $A(:, \mathcal{J}.)$ Note that this matrix

has several zero rows so the nonzero pattern of the rows of $A(:, \mathcal{J})$ is defined by $\mathcal{I}$. For the product Am_j the only worry is about the entries in the matrix $A(\mathcal{I}, \mathcal{J}) = \hat{A}$. The constraint linear minimization problem $\|e_j - AM\|_2^2$ can now be solved by minimizing the much smaller problem

$$\min \|\hat{e}_j - \hat{A}\hat{m}_j\|_2, \tag{4.61}$$

where $\hat{e}_j = e_j(\mathcal{I})$ and $\hat{m}_j = m_j(\mathcal{J})$.

Incomplete (bi)-conjugation In the Frobenius norm minimization method it is assumed that the sparsity pattern of $\mathcal{B}$ is given. If this is not the case another method called (incomplete) biconjugation could be used. This method uses the matrix A to approximate the inverse A^{-1}. If two complete sets of A-biconjugate vectors are generated the approach is as follows.

Assume the matrix A to have an LDU decomposition. Hence, write $A^{-1} = U^{-1}D^{-1}L^{-1} = ZD^{-1}W^\top$, where $Z = U^{-1}$ and $W = L^{-\top}$. A *sparse* approximation to A^{-1} is wanted but in general Z and W are dense. So the idea is to approximate the matrices Z, W by sparse ones: $\bar{Z} \approx Z$ and $\bar{W} \approx W$. Finally, A^{-1} is approximated by

$$A^{-1} = \overbrace{\bar{Z}\bar{D}^{-1}\bar{W}^\top}^{M}, \tag{4.62}$$

and $\bar{D}^{-1} \approx D^{-1}$.

Assume for now that two complete sets of vectors $\{w_i\}_{i=1}^n$ and $\{z_i\}_{i=1}^n$ are given which are A-biconjugate, that is,

$$w_i^\top A z_j = 0, \quad \Leftrightarrow i \neq j. \tag{4.63}$$

Furthermore, let $Z = [z_1, \ldots, z_n]$ and $W = [w_1, \ldots, w_n]$. Since the vectors are biconjugate

$$W^\top AZ = D = \begin{pmatrix} p_1 & & \varnothing \\ & \ddots & \\ \varnothing & & p_n \end{pmatrix}, \tag{4.64}$$

where $p_i = w_i^\top A z_i$. Hence $A = W^{-\top}DZ^{-1}$ is obtained and

$$A^{-1} = ZD^{-1}W^\top = \sum_{i=1}^{n} \frac{z_i w_i^\top}{p_i}. \tag{4.65}$$

Until now it was assumed that the vectors w_i, z_i are known. The algorithm that computes these A-biconjugate vectors is described in reference 3 (p.454) and starts with two initial matrices Z^0, W^0. It is common to start with $Z^0 = W^0 = I$.

If a closer look upon the A-biconjugate algorithm is given it is observed that the starting vectors are sparse. However during the algorithm they are filled rapidly. To ensure the vectors to stay sparse the biconjugation process is done incompletely. This incompleteness can either be enforced by means of discarding values on prescribed positions or by using a drop-tolerance level as described in Section 4.4.1.1.

Although the idea of approximate inverse seems to be very good, in practise it is not that optimal. This is the case because for a given accuracy quite full matrices are obtained. Therefore, after a second look, this method will not be our first choice, since the storage for this preconditioner would be unacceptably high.

4.4.1.5 Operator Preconditioning

The next technique for preconditioning is called *operator preconditioning*. This technique uses the discretized operator or a part of it as a preconditioner. The generalized eigenvalue problem can then be used to analyse the spectrum of the preconditioned system. A detailed example of this can be found in reference 5. The operator to speed up the solution process here is the shifted Laplace operator for the Helmholtz equation. In this case the class of operators used can be written as

$$\mathcal{M}_{\beta_1,\beta_2} = -\Delta - (\beta_1 - j\beta_2)k^2, \tag{4.66}$$

where k is the wave number and j is the imaginary unit. A shift in the real or imaginary direction, defined by choosing an appropriate value of β_1 and β_2, is applied to cluster the eigenvalues around 1. Thereby, the values of the shift-parameters are defined in such a way that the condition number (κ_2) of the preconditioned system is minimal.

This method will be discussed in some more detail by giving a simple one dimensional example (Ref. 5) where only a real shift is defined, that is, $\beta_2 = 0$ and β_1 is chosen to minimize κ_2. Consider the one dimensional Helmholtz equation on the domain $\Omega = (0, 1)$

$$-\frac{d^2u}{dx^2} - k^2u = 0, \tag{4.67}$$

with Dirichlet boundary conditions $u(0) = u(1) = 0$. For the one dimensional problem the preconditioning operator (with real shift) (4.66) looks like

$$\mathcal{M}_{\beta_1} = -\frac{d^2}{dx^2} - \beta_1\, k^2, \quad \text{with } \beta_1 \leq 0. \tag{4.68}$$

This last restriction is chosen such that the preconditioner (after discretizing (4.68)) will be SPD. For this simple problem a value of β_1 can be found ($\beta_1 = -1$) such that the condition number of the preconditioned Helmholtz operator is minimal.

4.4.2 Preconditioning for Krylov Subspace Methods

In the following sections the templates for the several methods treated before will be updated. Due to the presence of the preconditioner M, the templates will change slightly.

4.4.2.1 The Preconditioned Conjugate Gradient Method

Assume the preconditioner M to be SPD. Then the new coefficient matrix $M^{-1}A$ may not be SPD. if this is the case it is not allowed to use the CG method for this new system. To overcome this, the preconditioner M is build up from a general nonsingular matrix P,

$$M = PP^{\top}. \tag{4.69}$$

The system $Ax = b$ is transformed to

$$\tilde{A}\tilde{u} = \tilde{b}, \tag{4.70}$$

where $\tilde{A} = P^{-1}AP^{-\top}$, $\tilde{u} = P^{\top}u$ and finally, $\tilde{b} = P^{-1}b$. Now the conjugate gradient may be applied on the transformed system (4.70). It is possible to write the template without the $\sim$ quantities appearing. Only the new template is given here, the derivation of the preconditioned conjugate gradient (PCG) method is given in appendix D.

Preconditioned Conjugate Gradient Method

Compute $r^0 = b - Ax^0$ for some initial guess x^0
for $k = 1, 2, \ldots$
solve z^{k-1} from $Mz^{k-1} = r^{k-1}$
  if $k = 1$
    $p^1 = z^0$
  else
    $\beta_{k-1} = (r^{k-1})^{\top} z^{k-1} / (r^{k-2})^{\top} z^{k-2}$
    $p^k = z^{k-1} + \beta_{k-1} p^{k-1}$
  end
  $q^k = Ap^k$
  $\alpha_k = (r^{k-1})^{\top} z^{k-1} / (p^k)^{\top} q^k$
  $x^k = x^{k-1} + \alpha_k p^k$
  $r^k = r^{k-1} - \alpha_k q^k$
  Check convergence; continue if necessary
end

Convergence Since the CG method is applied to the transformed system, the iterates x^k are in the space spanned by

$$x^k \in x^0 + K^k(M^{-1}A;\, M^{-1}r^0). \tag{4.71}$$

So theorem 4.1 is applied to the transformed system and the following result is obtained

$$\|x - x^k\|_A \leq 2\left(\frac{\sqrt{\kappa_2(P^{-1}AP^{-\top})} - 1}{\sqrt{\kappa_2(P^{-1}AP^{-\top})} + 1}\right)^k \|x - x^0\|_A. \tag{4.72}$$

Hence, a small condition number of $P^{-1}AP^{-\top}$ leads to a fast converging algorithm.

Work The idea of preconditioning is to increase the convergence rate by introducing the transformed linear system. But in every iteration z^{k-1} has to be solved from the linear system $Mz^{k-1} = r^{k-1}$. If this new system is hard to solve, nothing is gained by using the preconditioner since every iteration is very expensive. Therefore, the solution of the system $Mz^{k-1} = r^{k-1}$ is required to be cheap compared to a matrix vector product using the original coefficient matrix A. If this is the case, the additional operation is negligible compared to the matrix vector product.

4.4.2.2 Preconditioned Bi-CGSTAB

The Bi-CGSTAB method in combination with a preconditioner is described in reference 2 (p.27). The template will not be copied here.

4.4.2.3 Preconditioned GCR

The template for GCR will slightly change due to the presence of the preconditioner M. Since the search direction vectors and the residuals are explicitly orthogonalized in the algorithm, the matrix M only appears in line 3 of the template which is given below.

Preconditioned GCR Method

Compute $r^0 = b - Ax^0$ for some initial guess x^0.
for $k = 1, 2, \ldots$
  Solve s^k from $Ms^k = r^{k-1}$
  $v^k = As^k$
  for $j = 1, \ldots, k-1$
    $\alpha = (v^j, v^k)$
    $v^k = v^k - \alpha v^j$
    $s^k = s^k - \alpha s^j$
  end
  $s^k = s^k / \|v^k\|_2$
  $v^k = v^k / \|v^k\|_2$
  $x^k = x^{k-1} + (v^k, r^{k-1}) s^k$
  $r^k = r^{k-1} - (v^k, r^{k-1}) v^k$
end

In comparison with the unpreconditioned GCR method there, only one extra system involving the preconditioner M has to be solved.

4.4.2.4 Preconditioned COCG

If a SPD preconditioner M for the COCG method is to be included, this is as straightforward as for PCG. Therefore this is left out. Only the template is given (Ref. 17).

Preconditioned COCG Method

x^0 given; $r^0 = b - Ax^0$

$p^{-1} = 0; \quad \beta_{-1} = 0;$

Solve w^0 from $Mw^0 = r^0$

$\rho_0 = (\bar{r}^0, w^0)$

for $k = 0, 1, 2, \ldots$

$p^k = w^k + \beta_{k-1} p^{k-1}$

$u^k = Ap^k$

$\mu_k = (\bar{u}^k, p^k)$; **if** $\mu_k = 0$ then quit (failure)

$\alpha_k = \rho_k / \mu_k$

$x^{k+1} = x^k + \alpha_k p^k$

$r^{k+1} = r^k - \alpha_k u^k$

if x^{k+1} is accurate enough then quit (convergence)

Solve w^{k+1} from $Mw^{k+1} = r^{k+1}$

$\rho_{k+1} = (\bar{r}^{k+1}, w^{k+1})$; **if** $|\rho_{k+1}|$ small then quit (failure)

$\beta_k = \rho_{k+1} / \rho_k$

end

This page is intentionally left blank.

Part II

Scattering Analysis for the Jet Engine Air Intake of a Modern Fighter Aircraft

This page is intentionally left blank.

5 Problem Description

5.1 Introduction

In this chapter the actual problem is described: The electromagnetic wave entering the deep cavity. The domain of interest for the jet engine air intake and the appropriate boundary conditions for a well-posed problem are defined. The finite element method is used to discretize the vector wave equation which results in a large linear system of equations for the electric field inside the cavity.

5.2 Domain Description

The wave equation that was derived in Section 2.2.3 is now applied to analyse the scattering behavior of an electromagnetic wave entering a deep cavity: The jet engine air intake of a typical fighter aircraft. The surrounding of the cavity is the entire halfspace $x > 0$. The cavity is depicted in Figure 5.1.

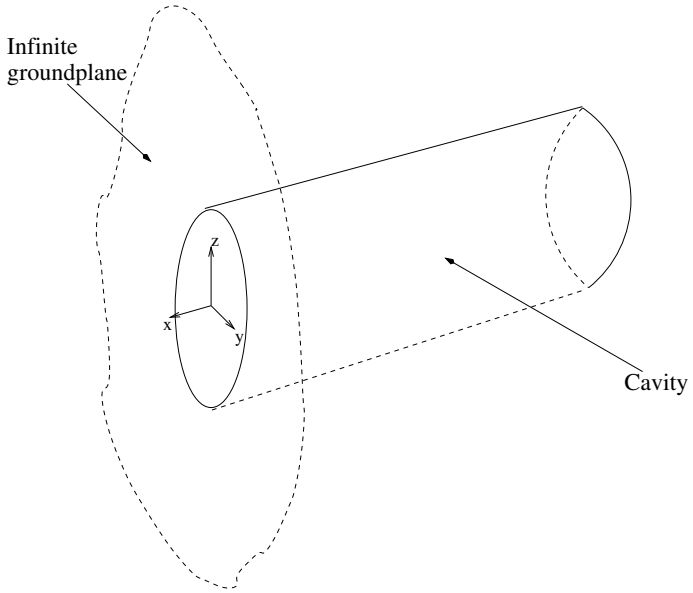

Fig. 5.1 The sideward cavity in an infinite groundplane. The plane $x = 0$ is defined on the aperture.

5.3 Boundary Conditions

The boundary conditions that are used for this problem combined with the vector wave equation were given in Section 2.3, but are repeated here for clarity. On the mantle of the cavity a zero tangential field is required, hence

$$\hat{n} \times \mathbf{E} = 0 \quad \text{on the cavity mantle.} \tag{5.1}$$

The second condition was derived in Section 2.3 and is given by

$$\hat{n} \times \mathbf{H}^{\text{inc}} = 4\hat{n} \times \left\{ \frac{\nabla\nabla \cdot \mathbf{N} + k_0^2 \mathbf{N}}{j\omega\mu_o} \right\}, \tag{5.2}$$

where

$$\mathbf{N} = \mathbf{K} * G = \iint_{S_a} \mathbf{K}(\mathbf{r}')G(\mathbf{r},\mathbf{r}')\, d\mathbf{r}' = \iint_{S_a} \left[\hat{n} \times \mathbf{E}(\mathbf{r}')\right]\, G(\mathbf{r},\mathbf{r}')\, d\mathbf{r}', \tag{5.3}$$

and G is the three dimensional Green's function. The incident magnetic field $\mathbf{H}^{\text{inc}}$ is determined by the incidence angle ϕ. For a fixed ϕ a linear system is obtained with the incident magnetic field in the right-hand side. Since the RCS is computed for a 120^o angular range, the system has to be solved for multiple right-hand sides.

5.4 Finite Element Formulation

The cavity volume is now divided into small volume elements, in this case tetrahedra. Within such an element the field is approximated, using the vector basis functions, by

$$\mathbf{E}^e(\mathbf{x}) = \sum_{i=1}^{n} E_i^e \mathbf{N}_i^e(\mathbf{x}) = \{E^e\}^\top \{\mathbf{N}^e(\mathbf{x})\}. \tag{5.4}$$

Here n is the number of basis functions in an element and $\{\mathbf{N}^e\}$ is used for a vector with the vector basis functions as its elements[1]. Hence for zeroth order elements $n = 6$ and for second order elements $n = 45$. By subdividing the volume of the cavity into M small volume elements, the aperture is divided into M_s small surface elements. The surface field expansion is given by

$$\hat{z} \times \mathbf{E}^s(\mathbf{x}) = \sum_{i=1}^{n_s} E_i^e \mathbf{S}_i^s(\mathbf{x}) = \{E^e\}^\top \{\mathbf{S}^s(\mathbf{x})\}, \tag{5.5}$$

where n_s is the number of basis functions for the triangular elements. For zeroth order elements $n_s = 3$, for second order elements $n_s = 15$. For a compatible expansion $\mathbf{S}_i^s = \hat{z} \times \mathbf{E}_i^s$. Substituting both the volume and surface expansions into the functional yields

$$F = \frac{1}{2}\sum_{e=1}^{M} \{E^e\}^\top [K^e] E^e + \frac{1}{2}\sum_{s=1}^{M_s}\sum_{t=1}^{M_s} \{E^s\}[P^{st}]\{E^t\} - \sum_{s=1}^{M_s} \{E^s\}^\top \{b^s\}. \tag{5.6}$$

In equation (5.6) the following matrices have been defined:

$$\begin{aligned} [K^e] &= \iiint_{V^e} \left[\frac{1}{\mu_r} \{\nabla \times \mathbf{N^e}\} \cdot \{\nabla \times \mathbf{N^e}\} - k_0^2 \varepsilon_r \{\mathbf{N}^e\} \cdot \{\mathbf{N}^e\} \right] dV, \\ \{b^s\} &= -2jk_0 Z_0 \iint_{S^s} \{\mathbf{S^s} \cdot \mathbf{H}^{inc}\}\, dS. \end{aligned}$$

[1] Note that this vector is actually a vector with elements that are vector itself.

The matrix $[P^{st}]$ is obtained from the boundary integral and has the form

$$[P^{st}] = 2 \iint\limits_{S^s} \{\nabla \cdot \mathbf{S^s}\} \left\{ \iint\limits_{S^t} \{\nabla' \cdot \mathbf{S^t}\}^\top G_0 \, dS' \right\} dS - 2k_0^2 \iint\limits_{S^s} \{\mathbf{S^s}\} \cdot \left\{ \iint\limits_{S^t} \{\mathbf{S^t}\}^\top G_0 \, dS' \right\} dS, \quad (5.7)$$

The first two integrals $([K^e], \{b^e\})$ are computed numerically using Gauss' Quadrature formulas. For the evaluation of the integral in the matrix $[P^{st}]$ a method described by reference 14 is employed and uses Duffy's method (see reference 9 appendix D) to get rid of the singularity in the Green's function.

5.5 Computing the RCS

In the end, when the system is solved, the electric field $\mathbf{E}$ and the magnetic current $\hat{n} \times \mathbf{E}$ are known on the aperture. These two quantities are used to determine the far-field approximation $\mathbf{E}^s$ that is used to determine the radar cross section. According to Sommerfeld's radiation condition, the far-field $\mathbf{E}^s$ has no radial component and the angular components E^s_θ and E^s_ϕ are determined by the following two equations

$$\begin{aligned} E^s_\theta &= (-jk_0 Z_0)(A_\theta + \tfrac{F_\phi}{Z_0}), \\ E^s_\phi &= (-jk_0 Z_0)(A_\phi - \tfrac{F_\theta}{Z_0}). \end{aligned} \quad (5.8)$$

The functions A and F used here are potential functions for the electric current and the magentic current respectively. Since the electric current is zero, A vanishes and F is given by (Ref. 7)

$$F(\mathbf{r}) = \frac{e^{-jk_0\mathbf{r}}}{4\pi\,\mathbf{r}} \int_{S_a} [\hat{n} \times \mathbf{E}(\mathbf{r'})] e^{jk_0\mathbf{r'}cos(\psi)} dS'. \quad (5.9)$$

Using the fact that A is zero, the far-field components are written as

$$\begin{aligned} E^s_\theta &= -jk_0 Z_0 \, F_\phi, \\ E^s_\phi &= jk_0 Z_0 \, F_\theta. \end{aligned} \quad (5.10)$$

6 The System of Equations

6.1 Introduction

In this chapter the dispersion error, introduced during the discretization is analysed. Herewith, an estimate of the total number of degrees of freedom (DoF) is given and the memory requirements to store this matrix are discussed. It will be shown that both the number of DoF and the memory requirements depend on the element order p and the dimensionless wavenumber k_0^*. At the end, recommendations about the choice of element order and maximum error are given. The last section of this chapter contains matrix properties for a small problem. This problem is chosen to be discussed here, although it is not completely equivalent to the target problem, but it is manageable in Matlab.

6.2 Dispersion Analysis

In this section the dispersion error is introduced. First some physical explanation about cavity scattering is given, and the importance of accurate preservation of the phase difference is discussed.

Assume an electromagnetic wavefront enters a cavity. Then this front can be thought of as a large number of individual rays. Each ray has a *direction of propagation* and a *strength*. This is illustrated in Figure 6.1 where the direction of two waves and their strength is shown[1].

If two (incident) rays with the same incident angle enter a cavity, the exact phase difference after reflection through the cavity is denoted by ψ_{out}. Due to the discretization, the computed phase difference $\tilde{\psi}_{out}$ differs slightly from the exact phase difference, so an error is introduced. This error is called the *dispersion error* and denoted by ε. Hence the computed phase difference can be written as

$$\tilde{\psi} = \psi + \varepsilon. \tag{6.1}$$

Figure 6.1 follows two rays from a wavefront through the cavity. Starting with phase difference $\psi_{in} = \lambda/4$, after reflection through the cavity the rays have a certain phase difference ψ_{out} as can be seen in Figure 6.2 (left). However, the computed phase difference $\tilde{\psi}_{out}$ differs from ψ_{out} due to the dispersion error (right).

It is the (exact) phase difference ψ that prescribes the strength of the scattered field $\mathbf{E}^s$. If the waves are in phase, they fortify each other while two waves in counter phase are subdued. Since

[1] Note, that a model assumption is made here, to explain the occurence of dispersion: Imagine that the electromagnetic wave consists of a large bundle of rays.

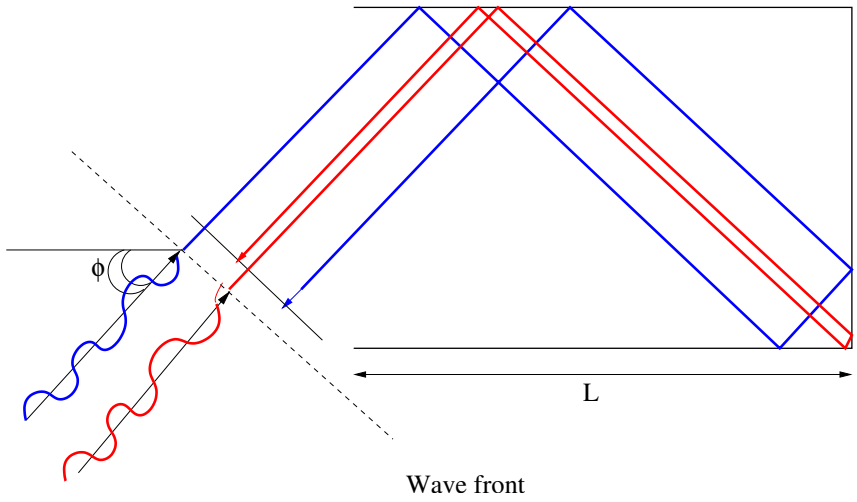

Fig. 6.1 The wave front enters the cavity with incidence angle ϕ. Two waves with initial phase difference $\psi_{in} = \lambda/4$ are followed. After reflection through the cavity there is an accumulated phase error ε.

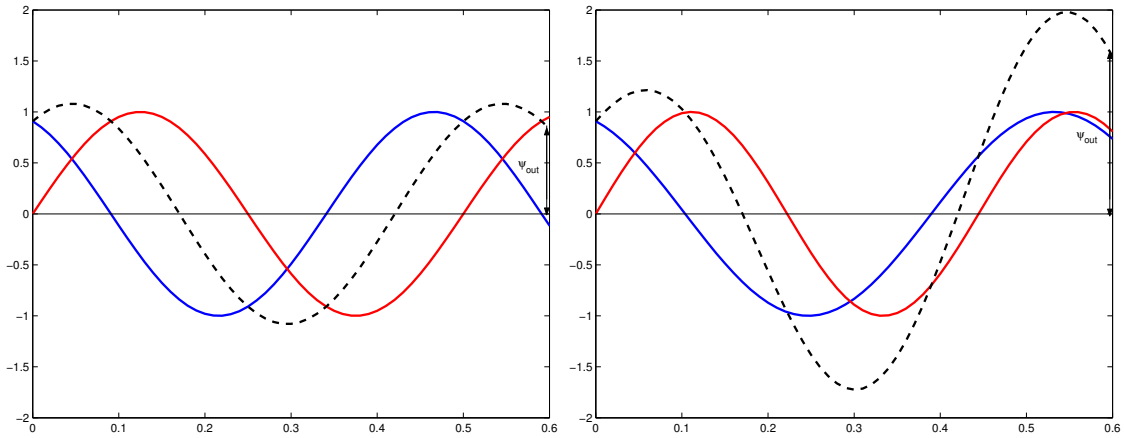

Fig. 6.2 The exact phase difference ψ_{out} (left) and the computed phase difference $\tilde{\psi}_{out}$ (right).

the Sommerfeld's radiation condition is applied, the radial component E_r^s of $\mathbf{E}^s$ equals zero. Hence, the other two components (transversal and axial), written as E_ϕ^s and E_θ^s respectively determine $\mathbf{E}^s$. After these components are calculated the RCS is proportional to the norm of the scattered electric field squared, hence

$$\sigma \propto ||\mathbf{E}^s||^2. \tag{6.2}$$

In order to be able to predict the strength of the scattered field it is very important to minimize the dispersion error ε. Accurate computation of the phase difference $\tilde{\psi}$ enables one to compute an approximated RCS $\tilde{\sigma}$ that resembles the exact RCS σ.

According to Jin et al. (Ref. 10), the dispersion error is a fundamental error in the finite element simulation of wave propagation. Because of the (finite element) discretization, the exact wavenumber k is approximated by $\tilde{k}$, introducing a dispersion error in the solution. This error accumulates as the wave propagates and becomes significant after propagating over a large distance like a deep cavity. In such a cavity, the electrical path of wave propagation is much longer than the length of the cavity, especially for oblique incidence, the dispersion error accumulates rapidly and completely controls the accuracy of the final FE solution.

There are several ways to decrease the dispersion error. The first one is to use smaller zeroth order elements. In practice this is not efficient since the error decreases slowly as the element size decreases, hence a huge system would be encountered for a given (small) accuracy. The second way is to use higher order elements. In the following, the number of unknowns needed to decrease the dispersion error ε to a sufficient level, is calculated as a function of the element order p and the dimensionless wavenumber k_0^*.

6.2.1 Meshwidth and the Number of Degrees of Freedom

As already mentioned in Section 2.4, the important number in analysing electromagnetic waves is the dimensionless wavenumber k_0^*. As soon as the scattering object size and the radar frequency are known, the wavenumber is calculated by

$$k_0^* = \frac{2\pi}{\lambda^*} = \frac{2d\pi}{\lambda} \;\Rightarrow\; \lambda = \frac{2d\pi}{k_0^*}, \tag{6.3}$$

where d is diameter of the geometrical cross section of the cavity. The derivation below shows how the number of elements and unknowns in the system relates to k_0^* and the element order p. At first, the dimensionfull wavelength λ is used in the derivation, but after equation (6.7) the dimensionless wavenumber k_0^* is introduced in the equations. According to reference 10, the maximum phase error *per wavelength* δ_p, when using vector-based tetrahedral elements, is given by

$$\delta_p = \left(\frac{\lambda}{h^*}\right)^{-2(p+\alpha)}, \tag{6.4}$$

where λ/h^* is the number of unknowns per wavelength, $\alpha \in [1, 2]$ is the *structuredness* of the grid ($\alpha \approx 1$ for a structured mesh) and p is the order of the element (a linear element is of zeroth order). A relation between the mesh width h and h^* is given by

$$h^* = \frac{h}{(p+2)}. \tag{6.5}$$

This relation is justified by considering Figure 3.2.

A wave incident at an angle ϕ from the normal of the cavity's aperture, travels a length $2L/\cos(\phi)$ before it exits the cavity (see Figure 6.1). The total accumulated dispersion error ε is

$$\varepsilon = \delta_p \cdot \sharp \text{ wavelengths in } \frac{2L}{\cos(\phi)} = \delta_p \frac{2L}{\lambda \cos\phi} = \left(\frac{\lambda}{h^*}\right)^{-2(p+\alpha)} \left(\frac{2L}{\lambda \cos\phi}\right) \tag{6.6}$$

The number of elements per wavelength required to achieve an accumulated dispersion error ε, $D = \lambda/h$ is then equal to

$$D(p) = \frac{\lambda}{h} = \left(\frac{2L}{\varepsilon \lambda \cos\phi}\right)^{\frac{1}{2(p+\alpha)}} \frac{1}{(p+2)}. \tag{6.7}$$

D as a function of p also depends on ε, α. Since the wavelength as a function of the wavenumber is known (equation (6.3)), the meshwidth h is expressed as a function of k_0^* and p by

$$h(k_0^*, p) = C_1 \left(\frac{1}{k_0^*}\right)^{1+\frac{1}{2(p+\alpha)}}, \tag{6.8}$$

where C_1 is a constant given by

$$C_1 = (p+2) 2\pi\, d \left(\frac{\varepsilon\, 2\pi\, d \cos\phi}{2L}\right)^{\frac{1}{2(p+\alpha)}}. \tag{6.9}$$

For a fixed value of k_0^*, corresponding to the X-band radar frequency of 10GHz, the meshwidth as a function of p is illustrated by figure 6.3 for different values of α and ε. Note that for an RCS in the optical region $10 < k_0^* < 100$. For this problem $k_0^* \approx 140$. This case is studied in more detail in the next section.

Next the total number of elements will be estimated and hence the total number of unknowns for the cavity. This is done by computing the volume of a typical tetrahedron V_t. Thereafter, the volume of the cavity V_c is determined. The total number of tetrahedral elements is equal to

$$N_{elements} = \frac{V_c}{V_t}. \tag{6.10}$$

Since V_c is a fixed number, the only thing that changes with the meshsize is the volume V_t. Assuming regular tetrahedra are used, the volume V_t is given by[2]

$$V_t = \frac{\sqrt{2}}{12} h^3. \tag{6.11}$$

The final step is to determine the number of unknowns in the system using the total number of elements. But then it is required to know how much *unique* DoF a typical element contains. This

[2]Note that equation (6.11) only applies for a typical tetrahedron away from the boundary. Close to the boundary the length h of the tetrahedron is adapted to conform to the shape of the cavity boundary.

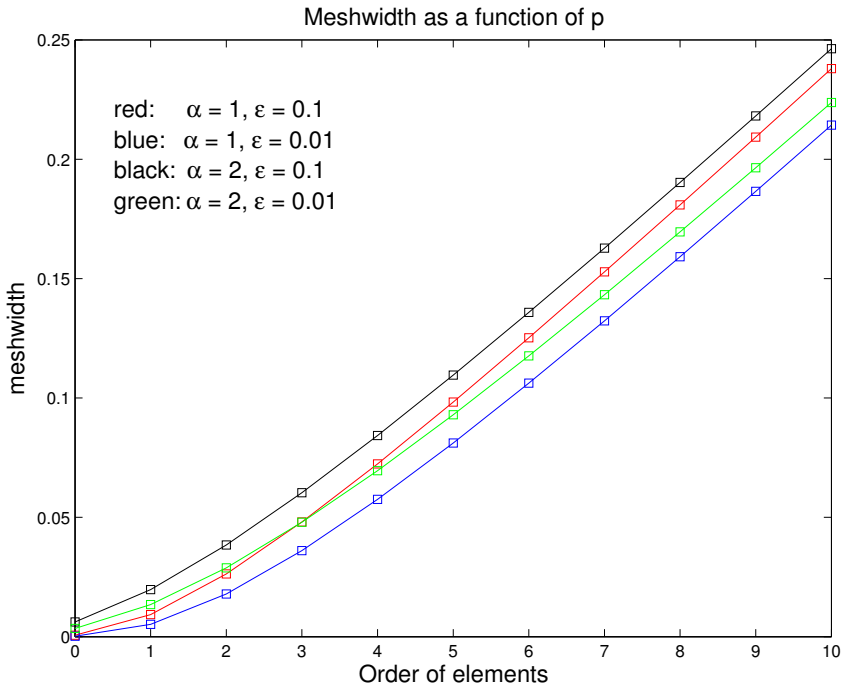

Fig. 6.3 The dimensionless wavenumber is fixed $k_0^* \approx 140$. The meshwidth h is given as a function of the element order p. The dependance on the parameters α, ε is minimal.

is determined by the following observation: Let each edge be used by six tetrahedra (because away from the boundary of the domain the mesh has a regular structure as depicted in Figure 6.4). For an element of order p there are $p+1$ DoF on every edge, hence, the DoF for the edges are

$$DoF_{edge} = (p+1) \cdot 6/6 = (p+1). \tag{6.12}$$

In the same way the DoF for a face are determined. There are $\sum_{j=1}^{p} j$ DoF on a face, but keep in mind that a face is only used by two distinct elements, and their are four faces in a tetrahedron. Furthermore, a point on a face is defined by two DoF, hence, the total DoF for the faces are

$$DoF_{face} = 4 \cdot \sum_{j=1}^{p} j = 2p(p+1). \tag{6.13}$$

Since the DoF in the interior of the element do not affect each other between elements, these can be eliminated (fully summed, see Section 4.2.2) from the element matrix beforehand and are not unknown. The total number of unique DoF for a tetrahedral element of order p is

$$(p+1) + 2p(p+1) = (2p+1)(p+1). \tag{6.14}$$

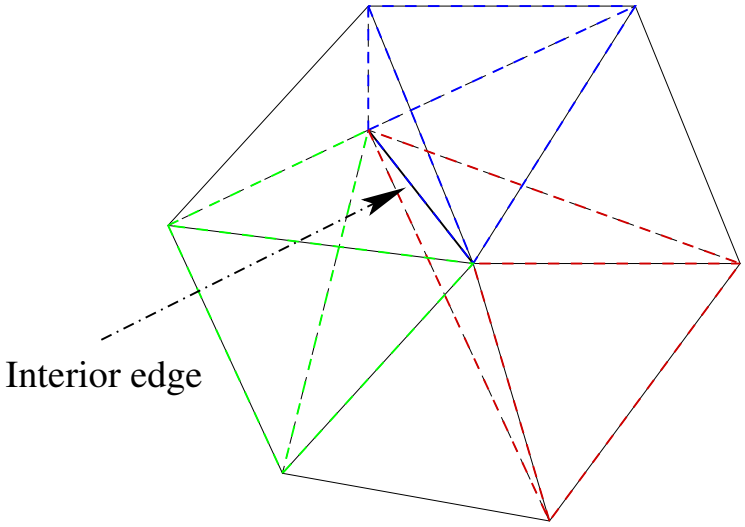

Fig. 6.4 Frontview of six connected tetrahedra. An interior edge is coupled to 19 other edges. To distinguish between the different elements, three are colored.

From this the total number of DoF for the cavity is computed by

$$DoF_{cavity} = (2p+1)(p+1) \cdot N_{elements}. \tag{6.15}$$

The dependance of the total numer of DoF is depicted in Figure 6.5. In the next section some results for different values of the dispersion error ε and α are given for the deep cavity $L/d = 10$ excited with X-band radar.

6.2.2 Application: The total number of DoF for a deep cavity

The X-band radar has a frequency of $f = 10\,\text{GHz}$. The corresponding wavelength λ is computed by

$$\lambda = \frac{c}{f}, \tag{6.16}$$

where $c = 3.0 \cdot 10^8$m/s is the speed of light. Hence the wavelength is $0.03\,\text{m}$. According to v.d. Heul et al. (Ref. 16) the (geometrical) cross section of a typical jet engine air intake is of the order of $400\lambda^2$, whereas the depth to diameter ratio is approximately 10. With this information the diameter, assuming a circular cross section, is

$$d = 2r = 2\frac{20}{\sqrt{\pi}}\lambda \approx 0.677\,\text{m}. \tag{6.17}$$

The depth L is then $6.77\,\text{m}$. For this cavity the meshwidth h, the number of elements N_{el} and the total number of DoF are compared for an incidence angle of $\phi = \pi/3$. This value is used since it defines a $120°$ cone of observation angles centered around the boresight direction, which corresponds to the common range of interest in cavity scattering. For this scattering problem, the dimensionless wavenumber $k_0^* \approx 140$. The results are in Table 6.1.

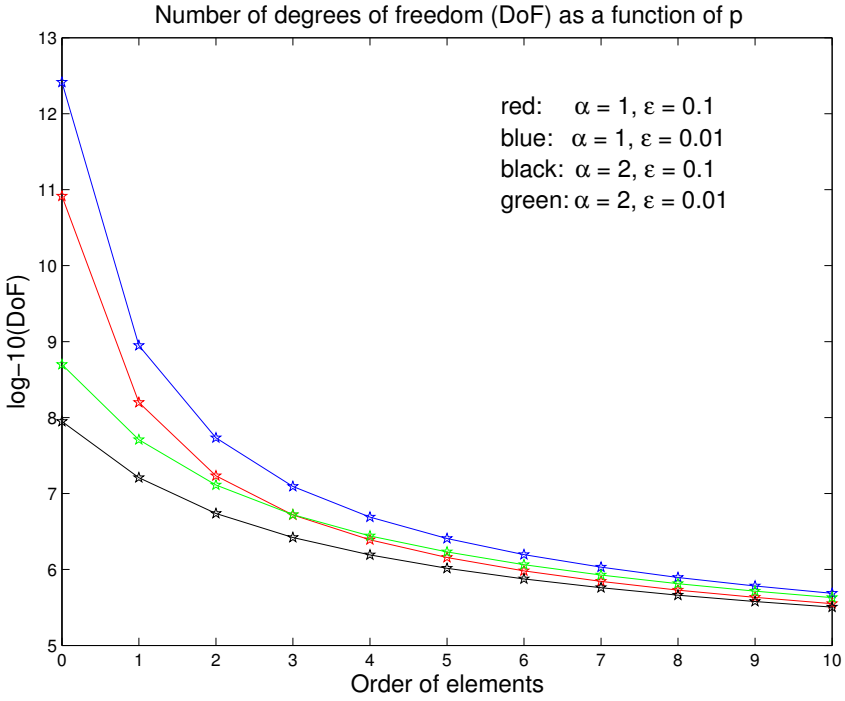

Fig. 6.5 The number of DoF for different values of the parameters ε, α. The dependance is on p for a fixed value of $k_0^* \approx 140$. The matrix size decreases monotonically as a function of p. For small p the number of DoF changes rapidly as α is altered between $\alpha = 1$ and $\alpha = 2$.

Table 6.1 *Comparison of the complexity for different values of ε and α. The dimensionless wavenumber has a value of $k_0^* \approx 140$.*

values α, ε	h	$N_{el} \cdot 10^6$	$DoF_{cavity} \cdot 10^6$
$\alpha = 1$, $\varepsilon = 0.1$	0.024	1.99	29.9
$\alpha = 1$, $\varepsilon = 0.01$	0.016	6.32	94.8
$\alpha = 2$, $\varepsilon = 0.1$	0.037	0.61	9.1
$\alpha = 2$, $\varepsilon = 0.01$	0.027	1.44	21.7

6.2.3 The value of k_0^*

In the last section the meshwidth, the total number of elements and the total number of DoF were calculated for a deep cavity with $k_0^* \approx 140$. As illustrated in Section 2.4.1, equivalence of different problems depends on the value of k_0^*. For $k_0^* \approx 140$ the problem is in the optics region. But for $k_0^* \geq 10$ this is also the case, but it is expected to obtain a much lower number of DoF. Therefore, Table 6.1 is repeated here for another value of the wavenumber: $k_0^* = 10$.

Table 6.2 *Comparison of the complexity for different values of ε and α. The dimensionless wavenumber has a value of $k_0^* = 10$.*

values α, ε	h	$N_{el} \cdot 10^3$	$DoF_{cavity} \cdot 10^3$
$\alpha = 1$, $\varepsilon = 0.1$	0.25	1.19	18
$\alpha = 1$, $\varepsilon = 0.01$	0.17	3.78	56.9
$\alpha = 2$, $\varepsilon = 0.1$	0.34	0.49	7.37
$\alpha = 2$, $\varepsilon = 0.01$	0.26	1.16	17.5

6.2.4 Memory Requirements

As calculated in Section 6.2 assume to have 21 million unknowns in the system (it is assumed that $\alpha = 2$ and $\varepsilon = 0.01$ and second order elements are used as is the case in the present implementation (Ref. 16)). With such a big system, the question rises whether it is possible to store this matrix in *core*, i.e. the memory the computer uses to store temporal variables. Even if it is possible to store the complete matrix, it might not be possible to store the additional matrices and vectors used in the iterative solvers, not to mention the preconditioner that will be used to speed up convergence.

First the number of nonzero elements in the matrix is calculated. This is done in two steps by first determining the number of nonzeros in the sparse part and then, adding the number of nonzeros in the fully populated part of the matrix. For the sparse part of the matrix it is possible to explicitly compute the number of nonzeros on every row. Assume that there are six tetrahedra connected to each other (known from practise see also Figure 6.4). Then an edge is coupled to 19 other edges. On each of these edges $p + 1$ basis functions are active and nonvanishing. Hence, for the edges the number of nonzeros per row equals $19(p + 1)$. An interior face is coupled to 18 faces. On every face $2\sum_{j=1}^{p} j$ basis functions are active so in total this yields $18p(p + 1)$ nonzeros. The total number of nonzeros on a row is then given by $(19 + 18p)(p + 1)$. For a second order element this means 165 nonzeros on every row for the sparse part. The matrix consists of 21 million rows hence, the sparse part consists of $165 * 21 \cdot 10^6$ nonzeros $= (3.5 \cdot 10^9)$.

For the fully populated part it is calculated that the number of unknowns[3] in this case is equal to 3000. Hence, the full part consists of $3000^2 = 9 \cdot 10^6$ nonzeros. The total number of nonzeros in the matrix is calculated by adding the number of nonzeros from the sparse and fully populated

[3]This is done in almost the same way as the computation of the degrees of freedom for the cavity and not treated here.

parts of the matrix yielding

$$\text{Nonzeros in A: } 3.5 \cdot 10^9 + 9 \cdot 10^6 \approx 3.5 \cdot 10^9 \text{ nonzeros.} \tag{6.18}$$

The storage for a real number is 8 bytes, for a complex number it is 16 bytes. Hence the total storage for the matrix is approximately 25 Gb (see Figure 6.6).

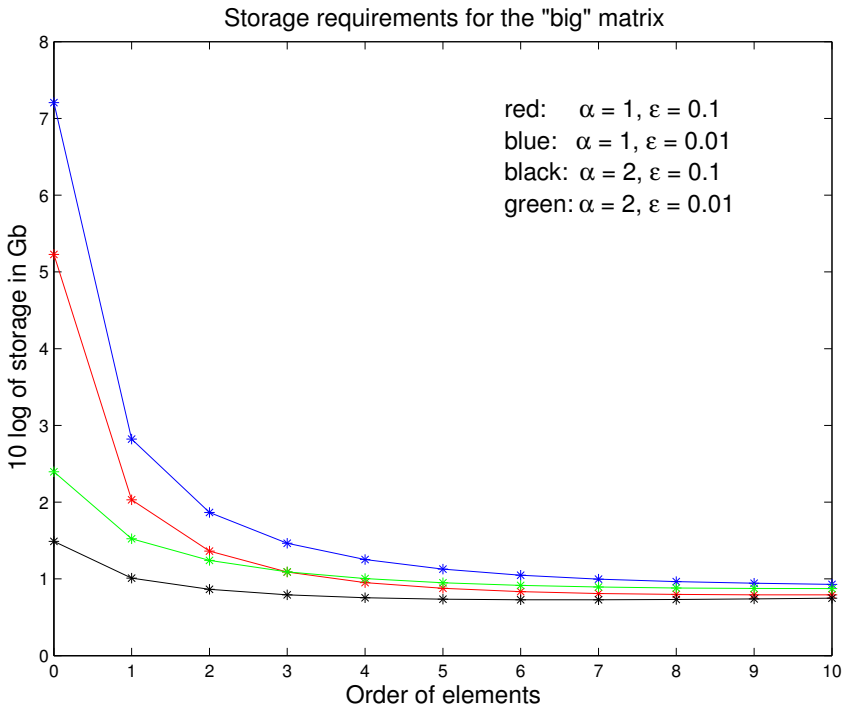

Fig. 6.6 The storage requirements as a function of the element order p and the parameters ε and α for a fixed value of $k_0^* \approx 140$. The element order p is most important. For higher order elements the matrix becomes more dense and the storage costs converge.

To store an ILU(0) preconditioner requires another 25 Gb, whereas for ILU(1) the storage is unacceptibly high. (see Chapter 4.4). From this point of view it is not desirable to store the matrix A and other used matrices and vectors in core.

A first solution to this problem is to do matrix vector products *matrix free*. Matrix-free can be seen in different perspectives. The first one is to store the entire assembled matrix, but not in core. The second one is to store the element matrices on a fixed location and loading them in if needed. The final one is to compute the element matrices every time they are needed.

6.2.5 Remarks

(1) Although it seems in Figure 6.5 that it is a good idea to use the highest order possible to decrease the size of the matrix this is not the best option for the storage of the matrix, since

the sparsity of the matrix decreases and hence, this may increase the total storage. An illustration of the storage for the matrix, which will be computed in Section 6.2.4 in detail is given by Figure 6.6.

(2) Thereby it has to be mentioned that the polynomials that have to be computed for higher order elements become more and more complicated as can be seen in equation (3.12). Finally, for higher order elements the number of basis functions per element increases very rapidly. A formula for this number is given by (Ref. 6)

$$\#\text{ of basisfunctions}(p) = (p+1)(6+4p+0.5p(p-1)) \tag{6.19}$$

Zeroth order means 6 basis functions but for second order, 45 basis functions have to be computed. As can be seen in the figures in this section it seems reasonable to use elements of fifth order, but note that many (216) basis functions have to be computed.

6.2.6 Recommendations

In the last sections several properties were found and figures presented. In this section a suggestion is made for the choice of parameters. Which choice of parameters is reasonable with respect to storage requirements, computing difficulty and matrix size?

According to Figure 6.7 and 6.8 it is recommended to take the following values for the parameters. The structuredness parameter $\alpha = 1$ and the maximum accumulated dispersion error $\varepsilon = 0.01$. The choice of ε yields a better accuracy and the value of α is recommended since the mesh that is used is quite structured in the interior. Furthermore it is suggested that no order higher than 4 is used, since the number of basisfunctions to be computed grows very fast as p increases.

The memory requirements for this choice are approximately 15 Gb for the storage of the matrix. The number of degrees of freedom is $5 \cdot 10^6$. Finally, the number of basis functions to be computed for $p = 4$ equals 140.

6.3 Matrix Properties

In this section a discretization matrix is introduced. The matrix is obtained for a rectangular cavity of moderate depth with dimensions $1.5\lambda \times 1.5\lambda \times 0.6\lambda$. The meshwidth is set to $h = 0.25$. Zeroth order basis functions are used for simplicity. The scattering problem considered here is *not equivalent* to the target problem, since the depth-to-diameter ratio L/d is not even close to 10 and the dimensionless wavenumber $k_0^* \approx 10$ instead of $k_0^* \approx 140$. However, since it is not possible to load a matrix for an equivalent problem in Matlab, a smaller problem is chosen in order to

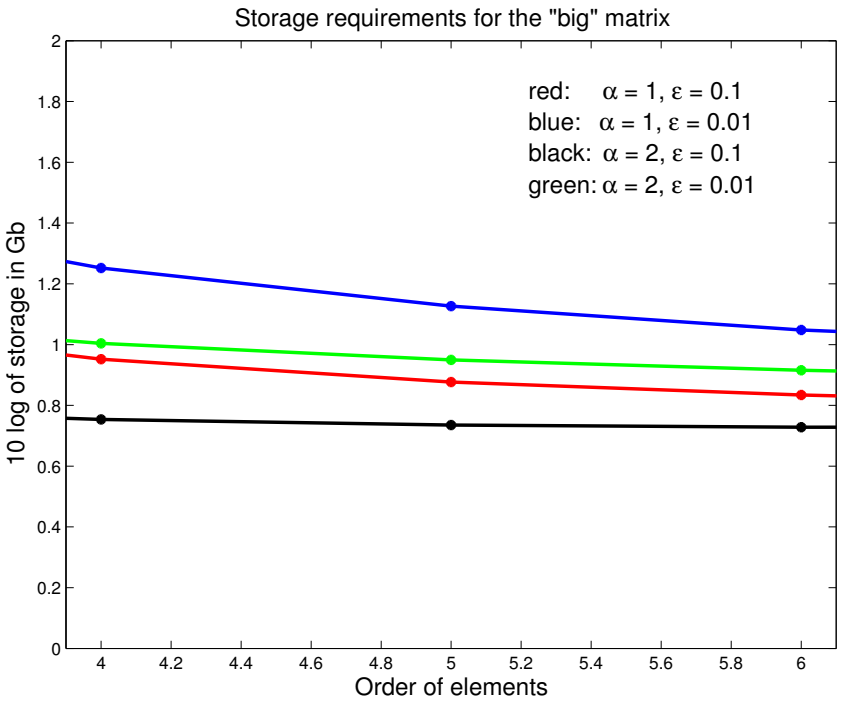

Fig. 6.7 Close up of Figure 6.6 showing details of the relation between the element order p and the required storage.

find some properties of the discretization matrix using Matlab. It should be mentioned that conclusions, drawn from this matrix properties do not have to hold for the target problem. The cavity is illustrated in Figure 6.9 (left). The right part of the figure gives a little look inside the box.

6.3.1 Sparsity Pattern

As illustrated in Figure 6.10 the matrix has a sparse and a fully populated part. The latter stems from the boundary integral equation on the aperture and is complex valued due to the presence of the Green's function. If the permittivity $\varepsilon \in \mathbb{R}$ this is the only complex valued part of the matrix. The number of nonzeros for this matrix equals 19189. The fully populated part consists of 9801 nonzeros which is nearly half of the total number of nonzeros in the matrix. Hence, the discretization of the boundary integral largely determines the properties of the system since the cavity depth-to-diameter ratio is $L/d \approx 0.4$.

6.3.2 Eigenvalue Spectrum

The eigenvalue distribution of the matrix is depicted in Figure 6.11, (left) where the horizontal axis is the real part and the vertical axis is the imaginary part of the eigenvalues.

It is observed that the major part of the eigenvalues is close to zero, but also some bigger neg-

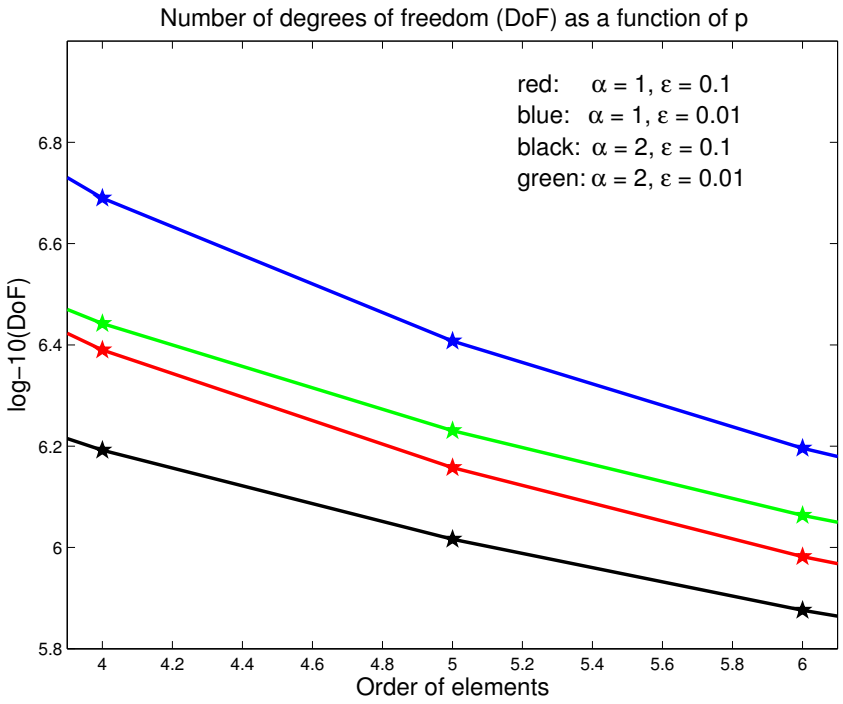

Fig. 6.8 Close up of Figure 6.5 showing details of the relation between element order p and the total number of DoF.

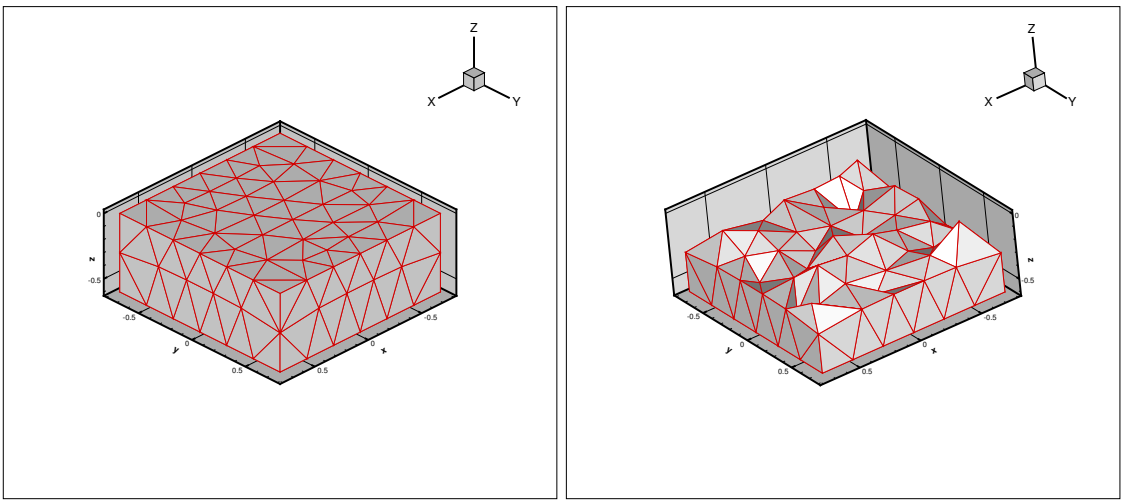

Fig. 6.9 The small, undeep box discretized (left). The placement of tetrahedral elements is shown in the right part.

ative and positive eigenvalues are present. Hence, the matrix is indefinite. The real-part of the most negative eigenvalues are in Table 6.3

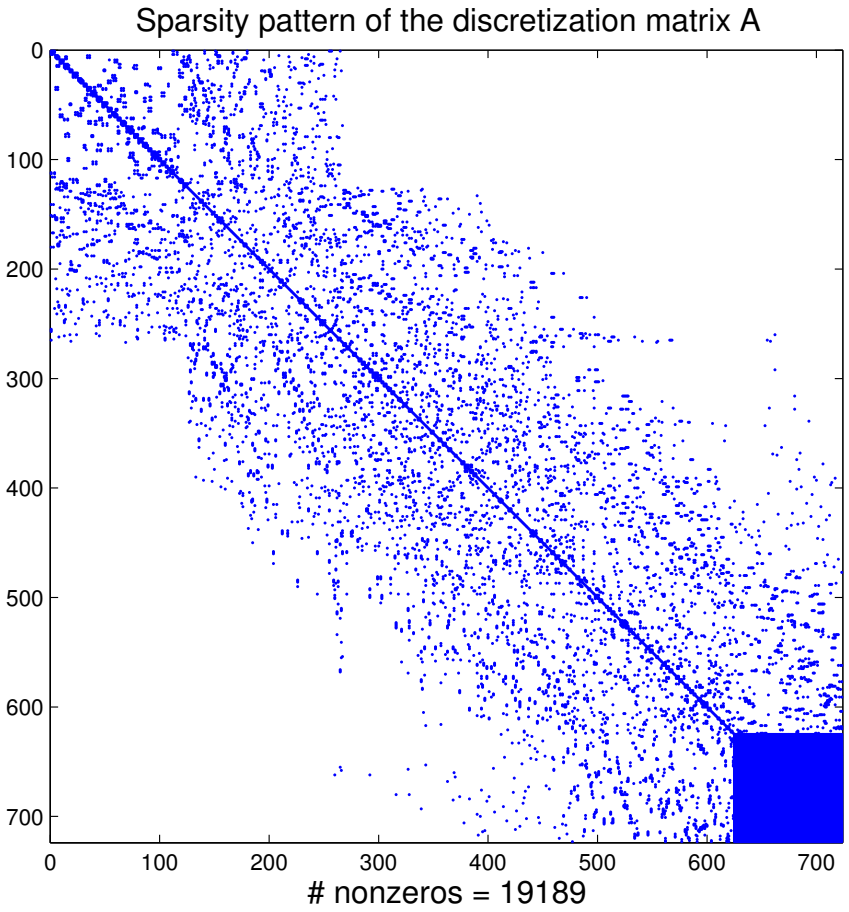

Fig. 6.10 The sparsity structure of the matrix A for $n = 723$. The number of nonzeros is 19189. The complex valued part of the matrix consists of the unknowns on the aperture only. This fully populated block consists of 9801 nonzeros.

Table 6.3 *List of 6 smallest eigenvalues.*

$-0.24 \cdot 10^{-3}$
$-0.23 \cdot 10^{-3}$
$-0.21 \cdot 10^{-3}$
$-0.18 \cdot 10^{-3}$
$-0.17 \cdot 10^{-3}$
$-0.17 \cdot 10^{-3}$

6.3.3 Condition Number

With the Matlab command *cond(full(A))*, the condition number $\kappa_2(A)$ of the matrix is calculated and given by

$$\kappa_2(A) = 880. \tag{6.20}$$

This implies that the matris is ill-conditioned, which may strongly affect the convergence of iterative methods.

6.3.4 Complex valued vs Real

Since a part of the matrix is complex valued, multiplying the matrix with a vector is expensive compared to a completely real valued matrix, since a complex/real multiplication is twice as expensive as a real/real multiplication. Therefore, it is desirable to know which part of the matrix complex valued. As long as the permittivity $\varepsilon \in \mathbb{R}$, the lower right part is the only complex valued part. But as soon as radar absorbing coatings are placed inside the cavity this is no longer the case and it becomes difficult to distinguish between real valued and complex valued parts of the matrix.

6.3.5 Summary

The main properties of the matrix are written below

(1) The matrix is *nearly* symmetric and complex valued.
(2) The matrix has a condition number $\kappa_2(A) = 880$ so it is *ill-conditioned*.
(3) There are positive and negative eigenvalues, hence the matrix is indefinite.
(4) The eigenvalues are not clustered at all. There are a few (very) negative eigenvalues.

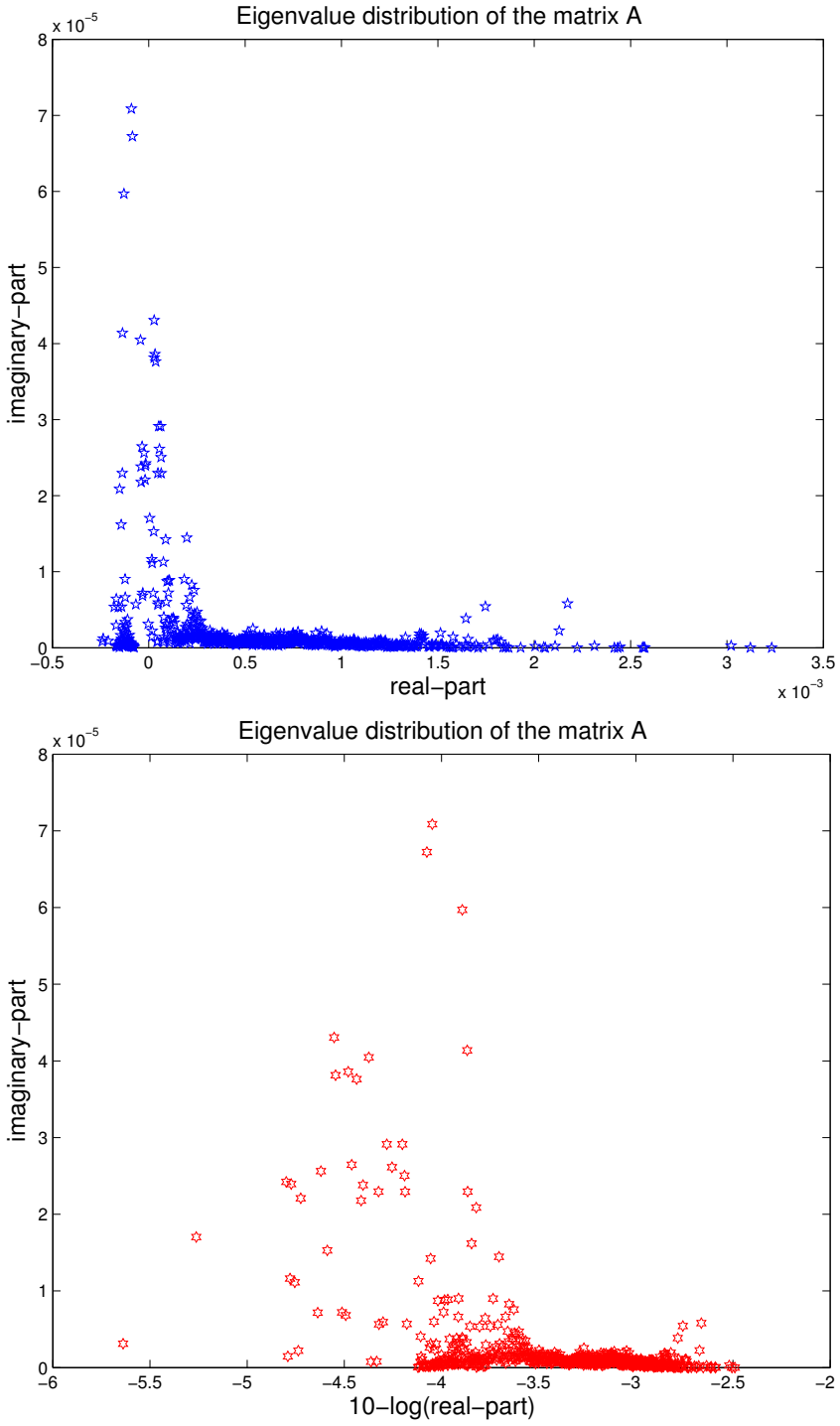

Fig. 6.11 The eigenvalue distribution for the matrix A. A number of negative eigenvalues (up) imply a indefinite matrix. A whole cluster of eigenvalues around zero (low) is present. Note that horizontally, the 10-log is taken for the absolute value of the real part of the eigenvalues.

7 Numerical Experiments

7.1 Introduction

In this chapter the iterative methods described in Section 4.3 are applied to some example systems. First, the different iterative methods are compared to each other by applying them to the matrices stemming from the standard second order finite difference discretization of the two dimensional Helmholtz equation and the finite difference discretization of the two dimensional Poisson equation with additional $+i$ on the diagonal. Second, the methods are applied to a matrix corresponding to a cavity scattering problem to gather preliminary insight in the behavior of this particular linear system. Finally, preconditioners are introduced to speed up the convergence for this last system only. The preconditioned system is solved directly to investigate the quality of the preconditioner.

7.2 Examples Without Preconditioning

7.2.1 Example 1

In this example the homogeneous Helmholtz equation with homogeneous Dirichlet boundary conditions is considered

$$\left(\nabla^2 + k^2\right) T = 0. \tag{7.1}$$

This differential equation is discretized on a regular two dimensional grid with $n = 256$ using finite differences for $k = 20$. Hence the meshwidth in both x and y direction is equal to $1/\sqrt{n} = 1/16$. The matrix is very similar to the discretization matrix from the Poisson equation (4.23), only the diagonal elements change.

$$A = \frac{1}{h^2}\begin{pmatrix} 4-k^2h^2 & -1 & 0 & -1 & 0 \\ -1 & 4-k^2h^2 & -1 & 0 & \ddots \\ 0 & \ddots & \ddots & \ddots & 0 \\ -1 & 0 & -1 & 4-k^2h^2 & -1 \\ 0 & \ddots & 0 & -1 & 4-k^2h^2 \end{pmatrix} \tag{7.2}$$

The matrix is ill-conditioned with a condition number, $\kappa_2(A) \approx 416$. The right hand side vector $b = 0$.

In Figure 7.1 the convergence behavior of the methods CGNR, Bi-CGSTAB and GCR for this matrix is illustrated for a random[1] startvector x^0. It is immediately observed that GCR is the

[1] The seed of the random number generator is kept the same for the different methods.

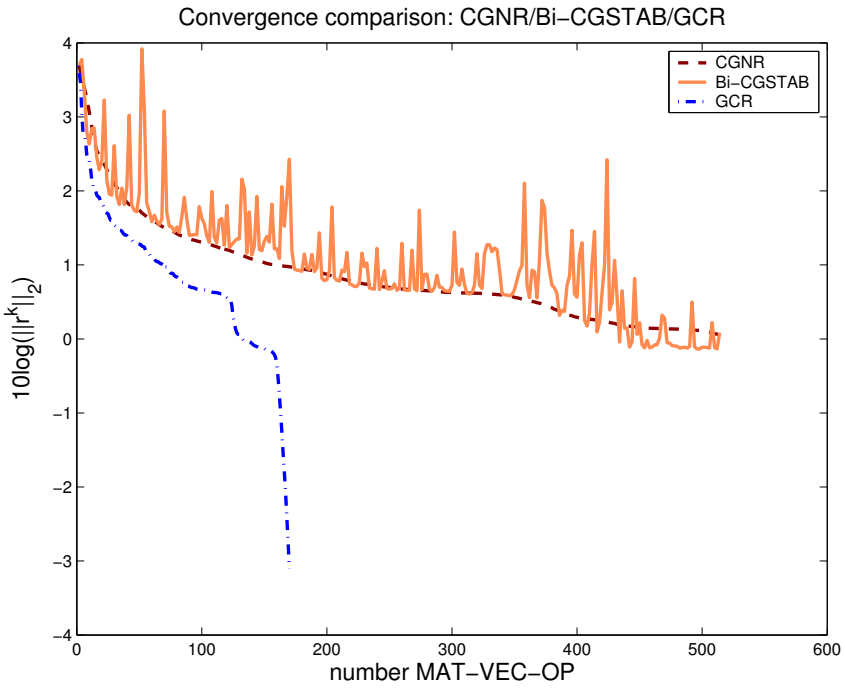

Fig. 7.1 The convergence behavior of the methods CGNR, Bi-CGSTAB and GCR for the Helmholtz discretization matrix of size $n = 256$ and $k = 20$. The matrix is badly conditioned which results in very slow or non convergence.

method with the fastest convergence and that CGNR and Bi-CGSTAB do not converge after 500 iterations. Bi-CGSTAB has a rough converging structure. It is also observed that preconditioning is necessary for faster convergence.

7.2.2 Example 2

To be able to compare the COCG method to the other ones a complex symmetric matrix is introduced: the two dimensional Poisson matrix with additional $+i$ on the diagonal. The number of nodes is $n = 256$ again. The matrix A becomes[2]

$$A = \frac{1}{h^2} \begin{pmatrix} 4+i & -1 & 0 & -1 & 0 \\ -1 & 4+i & -1 & 0 & \ddots \\ 0 & \ddots & \ddots & \ddots & 0 \\ -1 & 0 & -1 & 4+i & -1 \\ 0 & \ddots & 0 & -1 & 4+i \end{pmatrix} \tag{7.3}$$

The right-hand side $b = 0$ and a random startvector x^0 are chosen. For this matrix the convergence behavior of the methods is illustrated in Figure 7.2. COCG performs better than Bi-

[2] See example in Section 4.3.2.1 for details about this equation.

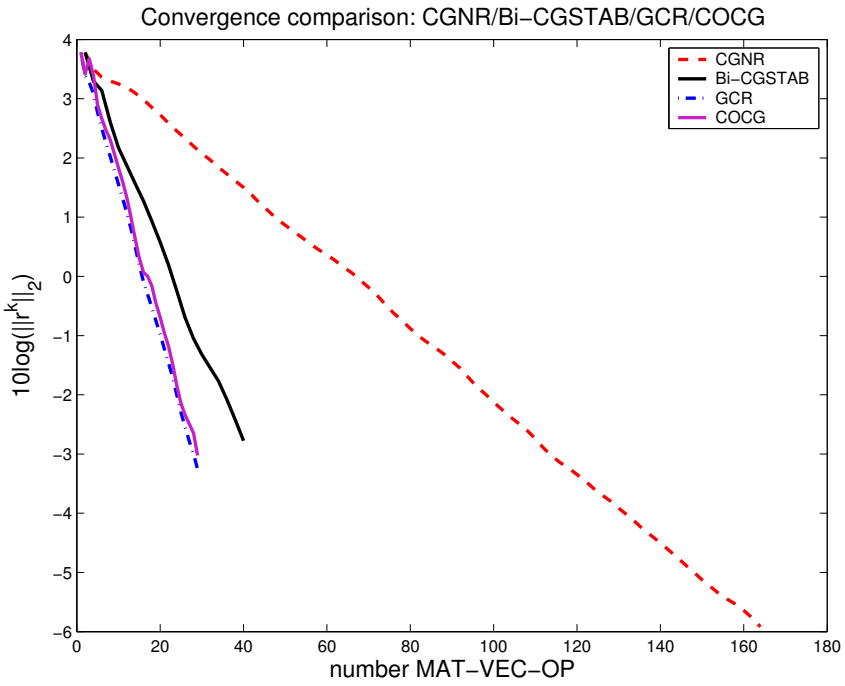

Fig. 7.2 The convergence behavior for the different methods for the complex matrix, equation (7.3). GCR has best convergence closely followed by COCG. CGNR is not very fast converging and Bi-CGSTAB is in between.

CGSTAB. GCR is the best method however although it requires more storage, since two sequences of orthogonal vectors need to be stored.

7.2.3 Example 3

The final matrix to be considered is the matrix originating from the finite element analysis performed for the rectangular undeep cavity as described in Section 6.3. This matrix is solved using CGNR, Bi-CGSTAB and GCR and the convergence is illustrated in Figure 7.3. A cavity of moderate depth is considered in order to keep the total number of degrees of freedom limited. Again it is observed that GCR is the fastest method.

7.2.4 Discussion

In the sections before, several methods for solving large linear systems were numerically tested. Each method has its own benefits and disadvantages and here they are compared based on the amount of work involved and the convergence behavior.

First, the workload of the methods is considered. The most time consuming part of these methods, are the matrix vector products. Methods that use A and A^T such as CGNR, have to do twice

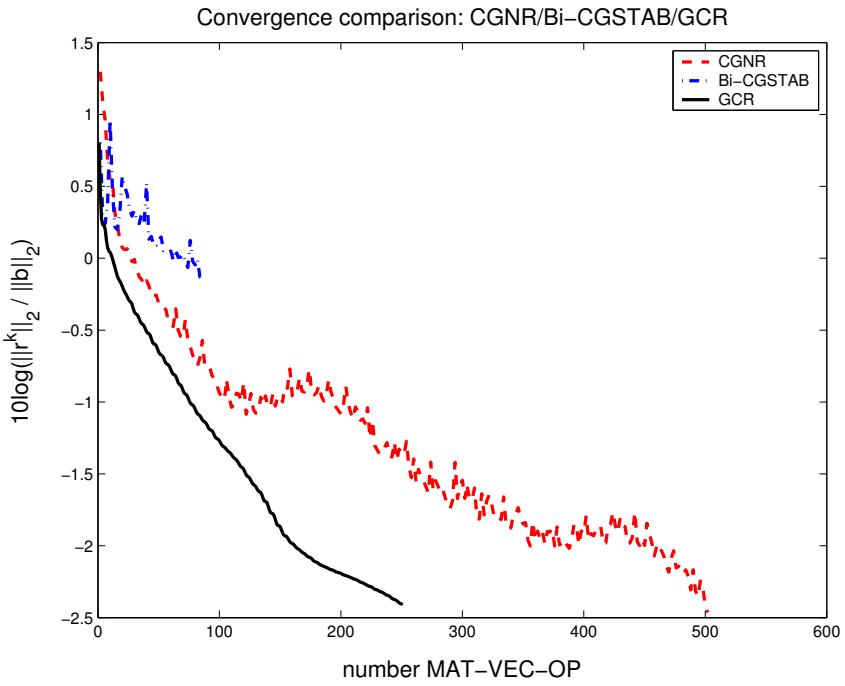

Fig. 7.3 The convergence behavior of GCR, Bi-CGSTAB and CGNR for the matrix corresponding to the undeep cavity. GCR converges very fast. Thereby it only uses one matrix vector product every iteration in comparison with CGNR and Bi-CGSTAB. Bi-CGSTAB breaks down after 43 iterations.

the work per iteration compared to methods only considering multiplication by A (COCG and GCR).

Then the convergence behavior of the methods is considered. Both GCR and CGNR have monotonically decreasing residuals but for Bi-CGSTAB this is not the case. COCG has nearly the same convergence as GCR. This is illustrated in Table 7.1. This table also include execution times. Note that from this table (column 2) it seems that CGNR is a very slow method because it takes twice as much time. However, the method needs four times as much iterations for the same convergence.

Table 7.1 *The number of matrix vector operations and CPU time for three examples. The first column corresponds to the complex Poisson matrix. The second column corresponds to the Helmholts discretization matrix. The third column corresponds to the cavity scattering matrix.*

Method	**Number of Matrix Vector Products / CPU(s)**		
Example	*Complex Poisson*	*Helmholtz*	*Cavity*
CGNR	162 / 0.16	512 / 0.16	702 / 8.2
Bi-CGSTAB	40 / 0.08	512 / 0.23	43 / 1.27
COCG	28 / 0.06	-	-
GCR	28 / 0.12	169 / 0.47	299 / 11.6

7.3 Examples Including Preconditioning

In this section the GCR method is considered with two standard preconditioning methods for the matrix defined in Section 6.3. GCR is chosen for its promising results for unpreconditioned systems. Firstly, the ILU-preconditioner is considered which is expected to perform not very well as described by reference 5. Secondly, the approximate inverse preconditioner is used.

7.3.1 Incomplete LU as preconditioner

In Matlab, the incomplete LU preconditioner is constructed using the command *luinc(A,τ)*, where τ is the desired droptolerance. This is applied to the discretization matrix, but for small values of τ ($\tau \leq 0.3$) the constructed preconditioners are so close to singular (the conditionnumber is ∞) that it is not possible to solve the preconditioner system $Ms = r$. So this preconditioner is of use only for drop tolerances $\tau \geq 0.4$. The preconditioner is then rougly the diagonal of A as can be seen in Figure 7.4. Therefore, the quality of this preconditioner is expected to be poor.

The question rises if ILU-preconditioned GCR has a better convergence than unpreconditioned GCR. To analyse this, the two methods are compared. Note that a system has to be solved for

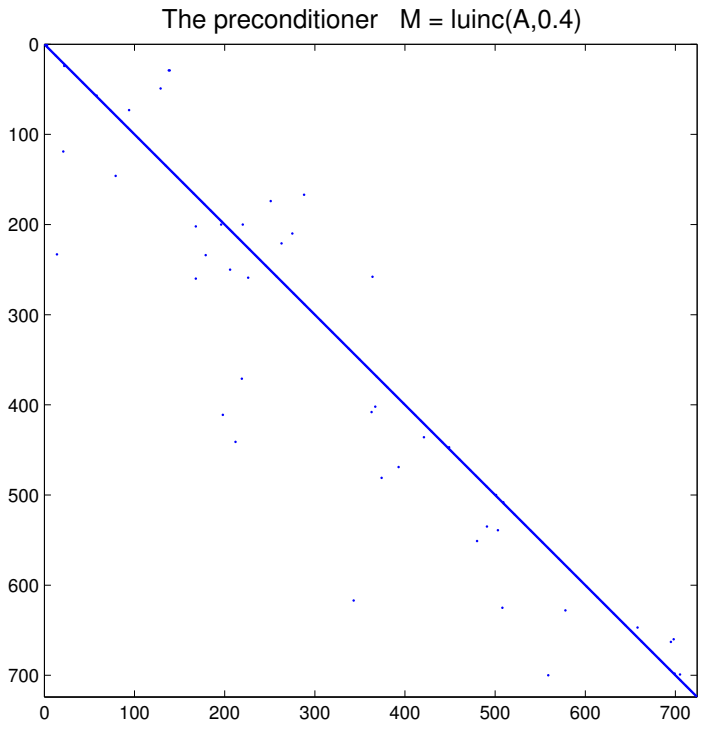

Fig. 7.4 The luinc(A,0.4) *preconditioner. Almost all elements of A except for the diagonal entries are set zero.*

the preconditioner every iteration. Since the used preconditioner is very sparse (Figure 7.4) the additional costs for solving the preconditioned system is neglected. Figure 7.5 illustrates the convergence behavior of unpreconditioned GCR versus ILU(0.4) preconditioned GCR.

7.3.2 Approximate Inverse Preconditioner

When an approximate inverse matrix is used, the question is what to choose for the sparsity pattern $\mathcal{G}$ as described in Section 4.4.1.4. In this example the sparsity pattern of the matrix A is copied for the sparsely populated part and a band pattern with bandwidth 10 is used for the fully populated part of the matrix A. In this way, the construction time for the preconditioner is limited and the performance is expected to be good. Figure 7.6 (left) illustrates the sparsity pattern of the preconditioner. The convergence of the GCR method for the preconditioned system (black) and the unpreconditioned system (blue) is illustrated in the right part. The quality of the preconditioner is low, since the tolerance level is not reached after the maximum number of iterations (250). Not to mention the additional work per iteration for solving the preconditioner system.

Since, the convergence improvement is poor another approximate inverse with more nonzeros in the pattern is considered next. Here, the sparsity pattern of the matrix A is copied completely,

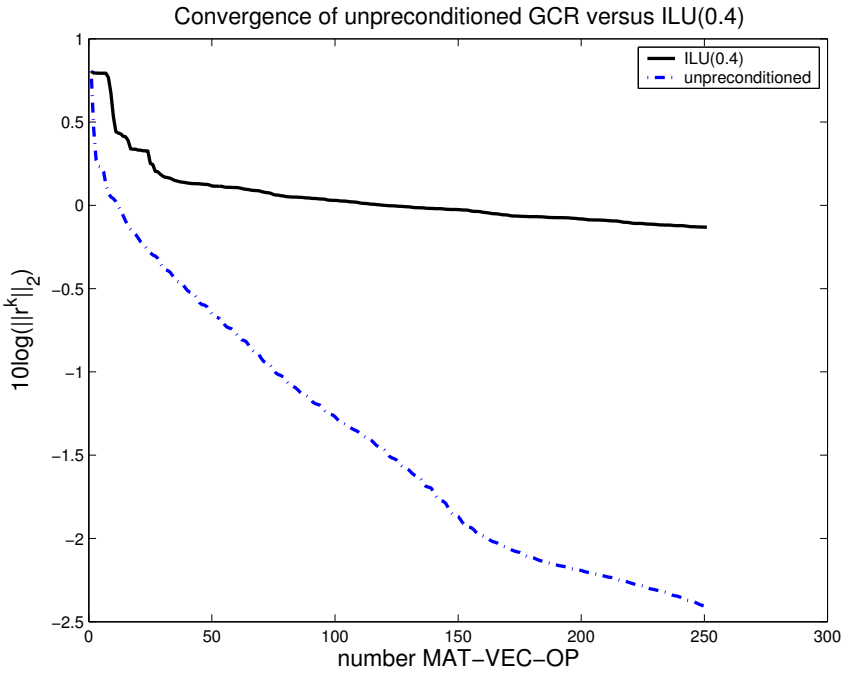

Fig. 7.5 The unpreconditioned GCR method (blue) perform much better than ILU-preconditioned GCR (black). Hence, this preconditioner is not very effective.

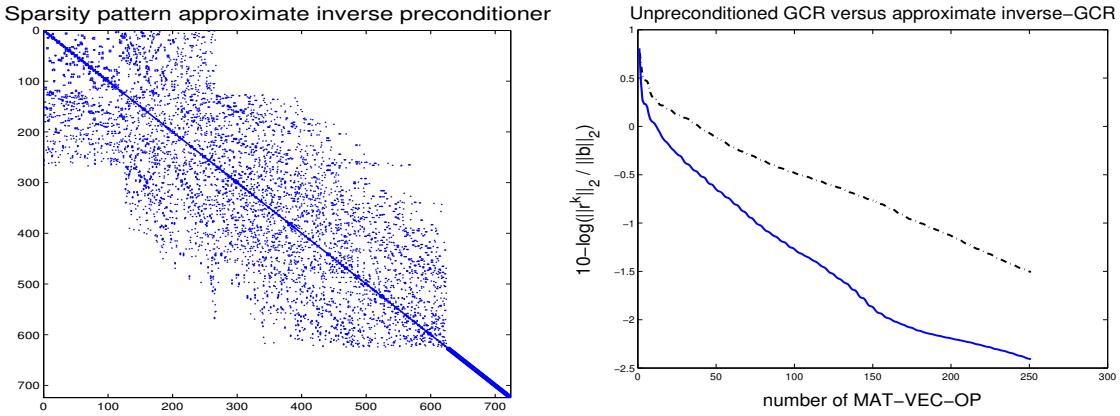

Fig. 7.6 The sparsity pattern of the approximate inverse preconditioner (left) and its quality (right). The convergence of GCR with an approximate inverse preconditioner (black) versus the unpreconditioned GCR (blue). Convergence is not improved by including the preconditioner.

hence the construction of the preconditioner itself takes a bit longer, but the performance is expected to be better. The result is illustrated in Figure 7.7. From this Figure it can be seen that the improvement is neglectable.

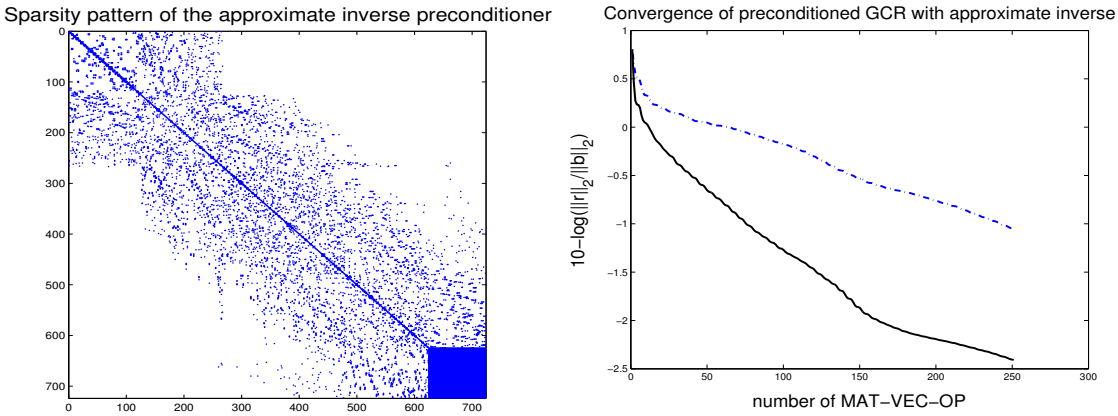

Fig. 7.7 The sparsity pattern of the approximate inverse preconditioner (left) and its quality (right). The convergence of GCR with an approximate inverse preconditioner (black) versus the unpreconditioned GCR (blue). The convergence improvement is still poor.

In the end note that for this example, preconditioning with an approximate inverse preconditioner seems not to be a good idea, since it does not exceed the convergence behavior of unpreconditioned GCR for this matrix.

8 Test Problems

According to Sections 2.4.1 and 6.2.2, two important ratio's have to be taken into account in creating testproblems that are equivalent to the target problem. The first ratio, $k_0^* = \frac{2\pi d}{\lambda}$ determines the scattering region of the problem. This ratio is fixed for the target problem $k_0^* \approx 140$, but in order to be in the same scattering region (see Figure 2.4) this value can be decreased to $k_0^* \geq 10$. If this is done, equivalent testproblems with a much smaller discretization matrix are expected.

The second ratio, L/d defines the depth of the cavity relative to the diameter of the cross section. For an undeep cavity, the aperture becomes to dominant, so for a testproblem to be equivalent to the target problem it is required to have this ratio approximately 10.

With these two considerations in mind, two testproblems are introduced. The first problem considers a rectangular cavity with $L/d = 10$ and a wavenumber $k_0^* = 10$ (Figure 8.1 left). The second testproblem considers a cylindrical cavity with the same ratios (Figure 8.1 right).

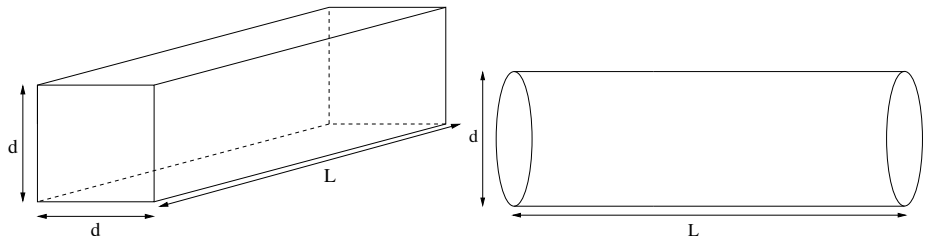

Fig. 8.1 Left: Testproblem 1, a rectangular cavity. Right: Testproblem 2, a long cylindrical cavity.

Now, a worst and a best case scenario with respect to the total number of DoF and total storage is investigated. According to the results from Section 6.2.4 the total number of DoF and storage decrease for $\alpha \approx 2$ in comparison with $\alpha = 1$. This is also the case for enlarging the accumulated dispersion error ε (for example from $\varepsilon = 0.01$ to $\varepsilon = 0.1$). Therefore, the total number of DoF and corresponding storage requirements are computed for $\alpha = 2$, $\varepsilon = 0.1$ (best scenario) and for $\alpha = 1$, $\varepsilon = 0.01$ (worst scenario). Tables 8.1 and 8.2 illustrate the requirements for storage and matrix size as a function of element order p for $\lambda = 0.03$ m and $\phi = \pi/3$. The fourth and seventh column indicate the amount of memory needed to store the fully populated block.

8.1 Discussion

It has been observed that both the memory requirements and the total number of DoF decreases rapidly as the element order increases. But for increasing element order the rows of the matrix

Table 8.1 *The total number of degrees (DoF) decreases for higher order elements and the storage requirements for* $\alpha = 1$, $\varepsilon = 0.01$ *and* $d/\lambda = 6$.

p	**Test problem 1**			**Test problem 2**		
	DoF $\cdot 10^3$	**Storage Mb**	**Storage full kb**	**DoF** $\cdot 10^3$	**Storage Mb**	**Storage full kb**
0	21733	6421	3120484	17069	4517	1924871
1	54	32.4	387	42.7	25.4	239
2	6.4	8.45	14.9	5.05	6.64	9.18
3	2.0	4.78	2.51	1.62	3.75	1.55
4	0.99	3.59	0.79	0.78	2.82	0.49
5	0.59	3.08	0.34	0.47	2.42	0.21

Table 8.2 *The total number of degrees (DoF) decreases for higher order elements and the storage requirements for* $\alpha = 2$, $\varepsilon = 0.1$ *and* $d/\lambda = 6$.

p	**Test problem 1**			**Test problem 2**		
	DoF $\cdot 10^3$	**Storage Mb**	**Storage full kb**	**DoF** $\cdot 10^3$	**Storage Mb**	**Storage full kb**
0	5.47	0.87	49.0	4.30	0.67	30.2
1	1.93	1.14	4.5	1.52	0.89	2.78
2	0.91	1.19	1.09	0.72	0.94	0.67
3	0.54	1.24	0.42	0.42	0.98	0.26
4	0.36	1.30	0.20	0.28	1.02	0.13
5	0.27	1.37	0.12	0.21	1.08	0.07

have more nonzero entries which affects the total storage requirements. Furthermore, the storage for the fully populated block is only a small fraction of the total storage requirements except when zeroth order basis functiosn are used.

9 Conclusion

In this literature study report the problem of computing the contribution of a generic jet engine air intake to the radar cross section of a modern fighter aircraft is considered. Since this air intake of the aircraft accounts for the major part of the radar cross section (RCS), it is important to be able to accurately estimate the contribution to the RCS of the air intake (a deep cavity). In order to actually compute the RCS, the electric field on the aperture of the cavity is computed by solving the vector wave equation

$$\nabla \times \left(\frac{1}{\mu_r} \nabla \times \mathbf{E} \right) - k_0^2 \varepsilon_r \mathbf{E} = -j k_0 \, Z_0 \mathbf{J}. \tag{9.1}$$

This equation is discretized by finite elements. The elements used here are tetrahedra, which conform easily to the shape of the cavity. Vector based basis functions are used since they prevent spurious solutions to occur. In the present implementation by v.d. Heul et al. (Ref. 16), second order elements are being used, but it has been observed that both memory requirements and accuracy are improved when higher order elements e.g. fourth or fifth order elements are used. However, higher order elements require many basis functions to be computed and give lead to a decrease in sparseness of the system matrix.

Discretization of equation (9.1) results in a large discretization matrix with a sparsely populated part and a fully populated part. Linear solvers, both direct and iterative have been discussed. It was observed that direct methods, although yielding a solution, are very slow and memory consuming in computing this solution, so iterative methods were considered. Theoretically, the rate of convergence of iterative methods based on Krylov subspaces can be improved by introducing a preconditioner. A trade-off has to be made between the additional work involved in the application of the preconditioner and the relative improvement of the convergence.

It has been observed that standard preconditioning techniques such as ILU-decomposition and approximate inverse do not yield satisfactory results. These preconditioners will only slightly improve the convergence speed. Therefore, another preconditioner has to be constructed. In the work of Erlangga et al. (Ref. 5), a robust and efficient preconditioner for the Helmholtz equation is given. Since the vector wave equation is the generalization from scalar to vector of the Helmholtz equation, it is expected that this kind of preconditioner, called *shifted Laplace operator*, might be very useful. It is recommended to study the application of this kind of preconditioners for the linear system resulting from the finite element discretization of the vector wave equation.

10 Future Research

In this chapter an outline for the second part of the project is given. The target problem to be solved in the end is to find the solution of the system

$$Ax = b \tag{10.1}$$

for a large matrix A in less than 7 days work. This system stems from the finite element discretization of the vector wave equation using higher order (e.g. fourth order[1]) vector based basis functions. Since this system is badly conditioned, the system is solved with an iterative method combined with an effective preconditioner. The iterative method consists of an outer-loop and an inner-loop, where the preconditioner (e.g. the shifted Laplace operator as described by reference 5) is used to speed up the convergence rate. The preconditioned system (inner-loop) is solved using the multigrid method.

In the last section, two testproblems were discussed. A first start is to solve the systems involved in these problems. As mentioned before, the systems are expected to be ill-conditioned. Therefore, preconditioners need to be constructed to speed up the rate of convergence.

If a preconditioner such as the shifted Laplace operator is considered, the problem must be cast into the framework that is desirable for this kind of problem. Since the assembled matrix consists of a sparse and a fully populated part, a first idea for the preconditioner is to scrap the fully populated part and work with the sparse part of the matrix as a preconditioner. A second idea might be to drop the entire sparse part of the matrix and construct a preconditioner from the fully populated part.

In the iterative method with preconditioning the outer loop yields iterates that approximate the exact solution x. The inner loop for the preconditioned system, written as $Mp = r$, must be solved every iteration. There are several possibilities for solving the preconditioner system.

1. Solve the preconditioned system using a direct method (e.g. Gaussian elimination.) This is only used to investigate the quality of the preconditioner.
2. If the problem size increases the direct method in the inner-loop becomes to expensive and an iterative method for solving the preconditioned system may be used (e.g. Bi-CGSTAB).
3. If convergence of the iterative method is slow a new (basic) preconditioner (e.g. ILU or approximate inverse) can be used for the solution of the preconditioned system, $Mp = r$.

[1] Note that until now only second order basis functions have been used. Fourth order basis functions that are suggested here have to be implemented.

4. Finally, if the convergence speed is still slow, a multigrid method may be used for the preconditioned system. For multigrid there are two types: Geometric multigrid and algebraic multigrid. Since it is not likely that a full multigrid implementation can be realized in the time available, a black box implementation of algebraic multigrid will be used.

The road to the target problem is depicted in Figure 10.1

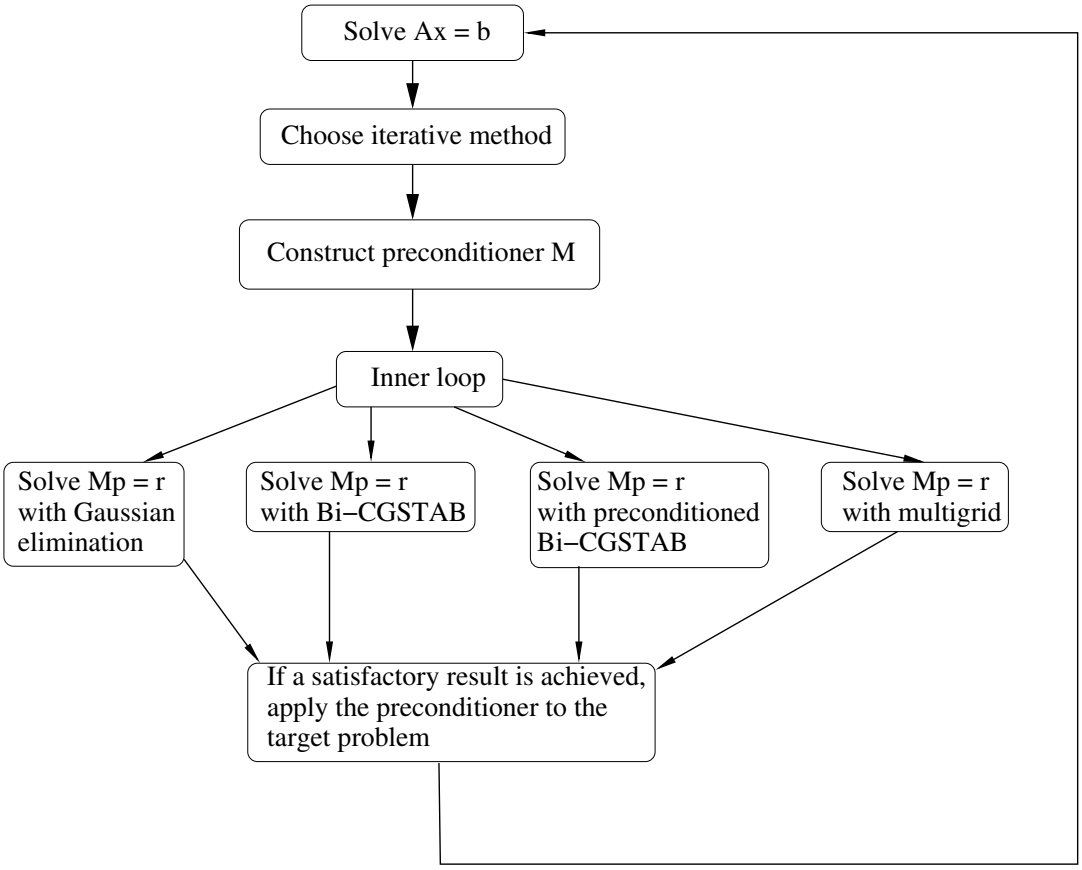

Fig. 10.1 Flow chart visualisation of the project.

References

1. C. A. Balanis. *Advanced Engineering Electromagnetics*. John Wiley and Sons, 1989.
2. R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, 1994.
3. M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182:418–477, 2002.
4. M. Benzi. Block preconditioning markov chain problems, 2006. www.csc2.ncsu.edu/conferences/nsmc/MAM2006/benzi.pdf.
5. Y. A. Erlangga. *A robust and efficient iterative method for the numerical solution of the Helmholtz equation*. PhD thesis, Delft University of Technology, 2005.
6. R. D. Graglia, D. R. Wilton, and A. F. Peterson. Higher order interpolatory vector bases for computational electromagnetics. *IEEE Trans. Magn.*, 45(3):329–342, 1997.
7. J. D. Greenwood and J. Jin. A hybrid mom/fem technique for scattering from a complex bor with appendages. *Aces Journal*, 15(1):1–12, 2000.
8. B. M. Irons. A frontal solution program for finite element analysis. *Int. J. Num. Meth. Eng.*, 2:5–32, 1970.
9. J. Jin. *The Finite Element Method in Electromagnetics*. John Wiley and Sons, second edition, 2002.
10. J. Jin, J. Liu, Z. Lou, and C. S. T. Liang. A fully high-order finite-element simulation of scattering by deep cavitities. *IEEE Trans. Magn.*, 51(9):2420–2429, september 2003.
11. C. T. Kelly. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics*. SIAM, 1995.
12. E. F. Knott, J. F. Shaeffer, and M. T. Tuley. *Radar Cross Section*. Artech House, Inc., 1985.
13. A. F. Peterson, S. L. Ray, and R. Mittra. *Computational Methods for Electromagnetics*. IEEE PRESS, first edition, 1998.
14. S. M. Rao, D. R. Wilton, and A. W. Glisson. Electromagnetic scattering by surfaces of arbitrary shape. *IEEE Trans. Magn.*, AP-30(3):409–418, 1982.
15. P. P. Silvester and R. L. Ferrari. *Finite Elements for Electrical Engineers*. Cambridge Press, university of cambridge edition, 1990.
16. D. R. van der Heul, H. van der Ven, and J. W. van der Burg. Full wave analysis of the influence of the jet engine air intake on the radar signature of modern fighter aircraft. unpublished.
17. H. A. van der Vorst and J. B. M. Melissen. A *p*etrov-*g*alerkin type method for solving $Ax = b$, where A is symmetric complex. *IEEE Trans. Magn.*, 26(2):706–708, 1990.

18. C. Vuik and C. W. Oosterlee. *Scientific Computing*. Lecture Notes. TU Delft, 2005.
19. I. Yavneh. Why multigrid methods are so efficient. *Computing in Science and Engineering*, 8(6):12–22, 2006.

This page is intentionally left blank.

Appendix A Electromagnetic Quantities

In Section some electromagnetic quantities with their units were introduced, but no translation of these units to basic SI units has been made yet. In table A.1 the quantities with their units and corresponding SI-units are stated.

Table A.1 *List of quantities with their units and base units.*

Quantity	**Name**	**Unit**	**Base Units**
ε	permittivity	[farads/m]	$[kg^{-1}\, m^{-3}\, A^2\, s^4]$
μ	permeability	[henry/m]	$[kg\, m\, s^{-2}\, A^{-2}]$
σ	conductivity	[siemens/m]	$[kg^{-1}\, m^{-3}\, s^3\, A^2]$

In the same section the vacuum values ε_0 and μ_0 were introduced. Their values are given by

$$\mu_0 = 4\pi * 10^{-7}\, \frac{F}{m}, \qquad \varepsilon_0 = \frac{1}{c^2 \mu_0} = 8.8542 * 10^{-12}\, \frac{Wb}{A\, m}, \tag{A.1}$$

where c is the speed of light with value $c = 2.9979 * 10^8\, m/s$.

Appendix B Frontal Solution Method: An Example

In Section 4.2.2, the frontal solution method was discussed briefly. Here an example of this method is given. Consider a square divided in 8 triangles of the same size, see Figure B.1

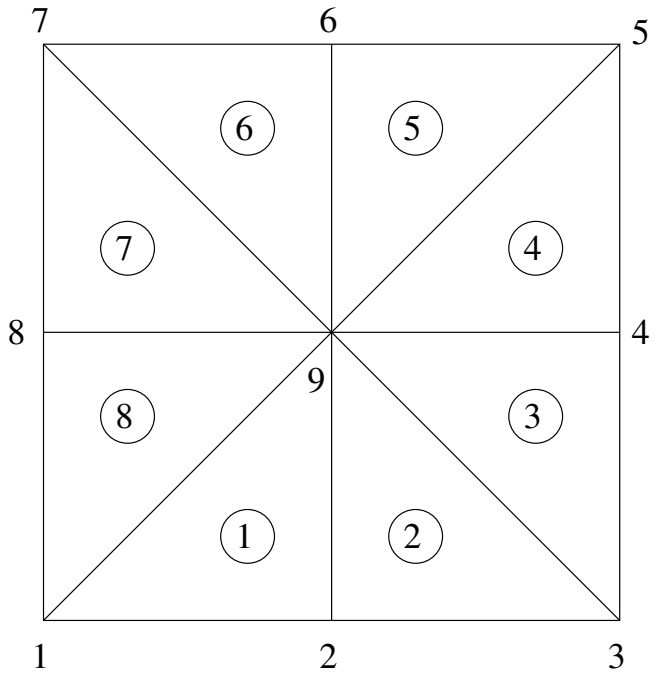

Fig. B.1 The frontal solution method applied to a small example.

The assemblage starts with the elements (1) and (2). Variable 2 is then fully summed up. The matrix then becomes after the assemblage of the first two elements

$$\left(\begin{array}{c|cccc} & 2 & 3 & 1 & 9 \\ \hline 2 & \times & \times & \times & \times \\ 3 & \times & \times & & \times \\ 1 & \times & & \times & \times \\ 9 & \times & \times & \times & \times \end{array}\right) \xrightarrow{\text{var } 2} \left(\begin{array}{c|cccc} & 2 & 3 & 1 & 9 \\ \hline 2 & u & u & u & u \\ 3 & l & \times & \blacksquare & \times \\ 1 & l & \blacksquare & \times & \times \\ 9 & l & \times & \times & \times \end{array}\right). \tag{B.1}$$

where the arrow with *var 2* represents the elimination of this variable from the system. The entries in the second matrix need some additional explanation. The entries u will not be adjusted

anymore (these are the entries of the matrix U from the LU-decomposition). The same holds for the values l (these elements belong to L in de LU-decomposition). Since these entries remain the same in the rest of the assembling-elimination procedure they are written out of core, to save computer memory. The ■ is a formal empty entry (in the left matrix) but it is filled in due to the Gaussian elimination procedure. Before the elimination of variable 2, the active variables, also called *the front*, are $(2, 3, 1, 9)$. After the elimination the front is $(3, 1, 9)$.

The following step is to assemble the next element, element (3). This means adding variable 4. A new matrix is then obtained

$$\begin{pmatrix}
 & | & 2 & 3 & 1 & 9 & 4 \\
- & + & - & - & - & - & - \\
2 & | & u & u & u & u & \\
3 & | & l & \times & \times & \times & \times \\
1 & | & l & \times & \times & \times & \\
9 & | & l & \times & \times & \times & \times \\
4 & | & & \times & & \times & \times
\end{pmatrix}
\overset{\text{var 3}}{\Longrightarrow}
\begin{pmatrix}
 & | & 2 & 3 & 1 & 9 & 4 \\
- & + & - & - & - & - & - \\
2 & | & u & u & u & u & \\
3 & | & l & u & u & u & u \\
1 & | & l & l & \times & \times & \blacksquare \\
9 & | & l & l & \times & \times & \times \\
4 & | & & l & \blacksquare & \times & \times
\end{pmatrix} \tag{B.2}$$

The front now consists of $(3, 1, 9, 4)$ and after elimination of variable 3 it is $(1, 9, 4)$. The next element to assemble is element (4). The next variable appearing in the matrix is variable 5. The front is $(1, 9, 4, 5)$ and the matrix looks like

$$\begin{pmatrix}
 & | & 2 & 3 & 1 & 9 & 4 & 5 \\
- & + & - & - & - & - & - & - \\
2 & | & u & u & u & u & & \\
3 & | & l & u & u & u & u & \\
1 & | & l & l & \times & \times & \times & \\
9 & | & l & l & \times & \times & \times & \times \\
4 & | & & l & \times & \times & \times & \times \\
5 & | & & & & \times & \times & \times
\end{pmatrix}. \tag{B.3}$$

Since variable 4 is fully summed it is eliminated. But in order to eliminate a variable, it must be in pivote place. Therefore, the matrix is permutated. The column and row containing variable 4 is swapped with the column and row of variable 1, yielding

$$\left(\begin{array}{cc|cccccc} & & 2 & 3 & 4 & 9 & 1 & 5 \\ - & + & - & - & - & - & - & - \\ 2 & | & u & u & & u & u & \\ 3 & | & l & u & u & u & u & \\ 4 & | & & l & \times & \times & \times & \times \\ 9 & | & l & l & \times & \times & \times & \times \\ 1 & | & l & l & \times & \times & \times & \\ 5 & | & & & \times & \times & & \times \end{array}\right) \tag{B.4}$$

where the front is also permuted to $(4, 9, 1, 5)$. The variable 4 is now eliminated giving a front $(9, 1, 5)$. In the next steps the matrices are permuted if necessary and fully summed variables are eliminated one by one. The subsequent fronts are given by

$$(9, 1, 5, 6),\ (5, 1, 9, 6),\ (1, 9, 6), \ldots, (8, 9, 1),\ (9, 1),\ (1). \tag{B.5}$$

The front is moving along the complete matrix and remains of the same size. The only storage in core is the frontal matrix which is here indicated by the entries $\times$. In this example the frontal matrix has size 4×4.

Appendix C Gaussian Elimination Process

In this appendix the most common direct method, Gaussian elimination is discussed. The decompositions for general (LU) and SPD (Cholesky) matrices are treated next.

Gaussian Elimination

The Gaussian elimination method transforms the system $Ax = b$ in an upper triangular matrix, requiring the pivots to be non-zero. The transformed system is then solved by backward substitution. To explain the method in more detail, assume that we have done $i-1$ steps of the algorithm already. Our matrix then looks like

$$\begin{pmatrix} a_{11} & a_{12} & \dots & & \dots & & a_{1n} \\ & \ddots & * & * & * & * & * \\ & & a^*_{i-1,i-1} & * & * & * & * \\ & & & a^*_{ii} & * & * & * \\ & \varnothing & & \bullet & \ddots & * & * \\ & & & \bullet & * & \ddots & * \\ & & & \bullet & * & * & a^*_{nn} \end{pmatrix} \tag{C.1}$$

where

- $*$ denotes updated elements of the matrix
- a^*_{kk} denotes updated diagonal entries
- $\bullet$ denotes elements to be made zero in column i.

In the next step the $\bullet-$ entries are made zero by multiplying the entry a_{ii} with a_{ji}/a_{ii} (multiplier) and subtracting it from row j, for $j = i+1, \dots, n$. The entries a_{kl} for $k, l = i+1, \dots, n$ will be updated also.

At the end, after $n-1$ steps, the matrix A is transformed in an upper triangular matrix U. The system is then solved by backward substitution starting by the last equation, since this equation for x_n does not contain any unknowns anymore, hence $x_n = b_n/a_{nn}$.

One last thing must be mentioned about this method. During the $n-1$ subsequent steps to construct an upper triangular matrix U from A, a zero diagonal element may be encountered. By means of re-ordering the equations this can be overcome.

Work

The amount of work for a direct method is measured in the amount of floating point operations that must be done to obtain the solution. The efficiency of different methods may be compared by the amount of work. For the Gaussian elimination process it has been shown (Ref. 18) that the amount of work for a matrix of size n is $2n^3/3$.

LU-decomposition

In this section another approach to the Gaussian elimination method is employed in which it is shown how the LU-decomposition from the last section can be used.

Observe that in the ith step the multipliers

$$l_{ji} = \frac{a_{ji}}{a_{ii}}, \ldots, l_{ni} = \frac{a_{ni}}{a_{ii}}.$$

are used. If these multipliers are stored in a lower triangular matrix L, and ones are added on the main diagonal of L, multiplying this matrix with the upper triangular matrix U of the last section, the matrix A is obtained again. This will not be shown here, but a more intuitive proof is given by considering the first two columns of L and the first two rows of U. The second row of L multiplied by the second column of U yields the second diagonal element of A which is a_{22}.

$$L = \begin{pmatrix} 1 & & & & \\ a_{21}/a_{11} & 1 & & \varnothing & \\ a_{31}/a_{11} & a_{32}/a_{22} & 1 & & \\ \vdots & \vdots & & \ddots & \\ a_{n1}/a_{11} & a_{n2}/a_{22} & \ldots & \ldots & 1 \end{pmatrix} \tag{C.2}$$

$$U = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ & & & & \\ 0 & a_{22} - \frac{a_{21}}{a_{11}} a_{12} & * & * & * \\ & 0 & \ddots & & \\ & & & & \end{pmatrix} \tag{C.3}$$

Compute

$$L_{(2,:)}\,U_{(:,2)} = \frac{a_{21}}{a_{11}}a_{12} + a_{22} - \frac{a_{21}}{a_{11}}a_{12} = a_{22}.$$

In the next section the nice properties of a symmetric positive definite matrix are exploited. Here the Cholesky-decomposition will be introduced.

Cholesky-decomposition

In this section it is assumed that the discretization matrix A is symmetric and positive definite. A matrix with this property has a nicer decomposition than just LU, known as the Cholesky decomposition. Consider the following theorem

Theorem C.1 (Cholesky-decomposition) *If $A \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, then there exists a* unique *lower triangular matrix $G \in \mathbb{R}^{n\times n}$ with positive diagonal entries such that $GG^{\top} = A$.*

The method may be written in template form. The template for the Cholesky decomposition is given below (Ref. 18)

for $k = 1, \ldots, n$

$$a_{kk} = \left(a_{kk} - \sum_{p=1}^{k-1} a_{kp}^2 \right)^{1/2}$$

for $i = k+1, \ldots, n$

$$a_{ik} = \left(a_{ik} - \sum_{p=1}^{k-1} a_{ip}a_{kp} \right) / a_{kk}$$

end

end

This is the Cholesky decomposition in *column* form. There are other algorithms possible (e.g. row-form), but this one is chosen, because it is very simple to implement.

Work

The work for the Cholesky decomposition is half of the work for Gaussian elimination, hence only $n^3/3$ flops have to be done. Since this amount of work is proportional to n^3 this is not satisfactory, since the system to be considered has $n \approx 10^8$. hence , the amount of work must be decreased. This can be done by exploiting the sparsity structure of the matrix A.

Appendix D Derivation of PCG

In this appendix the template for the PCG method is derived. If the conjugate gradient method is applied to the transformed system $\tilde{A}\tilde{u} = \tilde{b}$ the expression for α_k becomes

$$\alpha_k = \frac{(\tilde{r}^{k-1})^\top \tilde{r}^{k-1}}{(\tilde{p}^k)^\top \tilde{A}\tilde{p}^k} = \frac{(\tilde{r}^{k-1})^\top \tilde{r}^{k-1}}{(\tilde{p}^k)^\top P^{-1}AP^{-\top}\tilde{p}^k} \tag{D.1}$$

Hence, choosing $p^k = P^{-\top}\tilde{p}^k$, the denominator of the above expression is written as $(p^k)^\top Ap^k$.

Next, this new p^k is applied to the line where the residual is updated. This line now becomes

$$\tilde{r}^k = \tilde{r}^{k-1} - \alpha_k\, P^{-1}Ap^k. \tag{D.2}$$

Choosing $\tilde{r}^k = P^{-1}r^k$ and substitute this in (D.2) one obtains

$$r^k = r^{k-1} - \alpha_k\, Ap^k. \tag{D.3}$$

In much the same way $\tilde{x}^k = P^\top x^k$. The last part of the process is to take a closer look at the inner product $(\tilde{r}^{k-1})^\top \tilde{r}^{k-1}$. With the new definition for $\tilde{r}$ this becomes

$$(P^{-1}r^{k-1})^\top P^{-1}r^{k-1} = (r^{k-1})^\top P^{-\top}P^{-1}r^{k-1} = (r^{k-1})^\top M^{-1}r^{k-1}. \tag{D.4}$$

Hence, defining $z^{k-1} = M^{-1}r^{k-1}$ before the start of a new iteration yields the algorithm given in Section 4.4.2.1.