Computation of Thermo-Acoustic Instabilities in Combustors

Jan-Willem van Leeuwen

June 5th, 2007
1. The ARRIUS Engine

2. Combustion

3. CERFACS

4. Mathematics
   - Discretization
   - Eigenvalue Problems
   - Solution Methods
   - Implementation

5. Results

6. Conclusion
Computation of Thermo-Acoustic Instabilities in Combustors

Jan-Willem van Leeuwen

Outline
The ARRIUS Engine
Combustion
CERFACS
Mathematics
Discretization
Eigenvalue Problems
Solution Methods
Implementation
Results
Conclusion

Figure: The AS355 helicopter
Computation of Thermo-Acoustic Instabilities in Combustors

Jan-Willem van Leeuwen

Outline
The ARRIUS Engine
Combustion
CERFACS
Mathematics
Discretization
Eigenvalue Problems
Solution Methods
Implementation
Results
Conclusion

Figure: The Turbomeca ARRIUS 1A1 turbine engine
Engine, mesh of combustion chamber

**Figure:** The numerical grid of a part of the combustion chamber
Combustion

Combustion is a sequence of chemical reactions between a fuel and an oxidant accompanied by the production of heat and (sometimes) light.
Heat causes pressure change (1)

Figure: Heat causes movement of air
Heat causes pressure change (2)

Examples

- Balloon pops due to heat
- Warm air rising

Francis Bacon: "Heat itself, its essence and quiddity is motion and nothing else."
Combustion in an Engine

Pressure changes within a small space

- Modelled by wave equation
- Three Boundary Conditions
  - Opening in wall (constant pressure)
  - Inlet and solid walls (constant speed)
  - Outlet (acoustic impedance)
- Flame response
Helmholtz equation

Transformation from time to frequency domain.

The wave equation:

$$\nabla \cdot \left( \frac{1}{\rho_0} \nabla p_1 \right) - \frac{1}{\gamma p_0} \frac{\partial^2 p_1}{\partial t^2} = -\frac{\gamma - 1}{\gamma p_0} \frac{\partial q_1}{\partial t}$$

Becomes the Helmholtz equation:

$$\nabla \cdot \left( \frac{1}{\rho_0} \nabla \hat{p} \right) + \frac{\omega^2}{\gamma p_0} \hat{p} = i\omega \frac{\gamma - 1}{\gamma p_0} \hat{q}(x)$$
Internship

Figure: Signs at the entrance
Figure: A Sunblade 200
CERFACS ...

- is located in Toulouse, France
- employs ca. 100 researchers
- works in 5 fields:
  - Parallel Algorithms
  - Electromagnetism
  - Aviation & Environment
  - Computational Fluid Dynamics
  - Climate Modelling and Global Change
Overview

Three different mathematical topics:

- Discretization of the equation
- Eigenvalue problems
- Solution methods
  - Arnoldi
  - Jacobi-Davidson
Equation:

\[ \nabla \cdot \left( \frac{1}{\rho_0} \nabla \hat{p} \right) + \frac{\omega^2}{\gamma p_0} \hat{p} = i\omega \frac{\gamma - 1}{\gamma p_0} \hat{q}(x) \]

LHS: homogeneous Helmholtz equation

- Use finite elements and Galerkin method
- Use integration by parts

RHS: \( \hat{q}(x) = \) Non-linear in \( \omega \)
Discretization (2)

Galerkin method:

- divide the domain in elements
- define test functions $\phi_j$
- define $S_v$: the set of vertices outside the boundary where the pressure is 0.
- approximate $\hat{p}$ by $\hat{p}(x) \approx \sum_{j:v_j \in S_v} \hat{p}_j \phi_j(x)$
- multiply the LHS with the test function
- integrate over the domain $\Omega$
Discretization (3)

We started with:

$$\nabla \cdot \left( \frac{1}{\rho_0} \nabla \hat{p} \right) + \frac{\omega^2}{\gamma p_0} \hat{p} = 0.$$  

We obtain $\forall k : v_k \in S_v$:

$$\int_{\Omega} \phi_k \nabla \left( \frac{1}{\rho_0} \nabla \cdot \left( \sum_{j : v_j \in S_v} \hat{p}_j \phi_j(x) \right) \right) dx + \omega^2 \int_{\Omega} \frac{\phi_k}{\gamma p_0} \sum_{j : v_j \in S_v} \hat{p}_j \phi_j(x) dx = 0.$$  

Interchange summation and integration:

$$\sum_{j : v_j \in S_v} \int_{\Omega} \frac{1}{\rho_0} \phi_k \nabla \cdot (\nabla \phi_j) dx \hat{p}_j + \omega^2 \sum_{j : v_j \in S_v} \int_{\Omega} \frac{1}{\gamma p_0} \phi_k \phi_j dx \hat{p}_j = 0.$$
Discretization (4)

- Integrate the first integral by parts
- determine contributions from boundary conditions

Final equation:

$$\forall k : v_k \in S_v : \sum_{j:v_j \in S_v} \left( - \int_\Omega \frac{1}{\rho_0} \nabla \phi_k \nabla \phi_j dx \hat{p}_j \right) +$$

$$\omega \sum_{j:v_j \in S_v} \left( i \int_{\partial \Omega_Z} \frac{1}{\rho_0 c_0 Z} \phi_k \phi_j d\xi \hat{p}_j \right) +$$

$$\omega^2 \sum_{j:v_j \in S_v} \left( \int_\Omega \frac{1}{\gamma p_0} \phi_k \phi_j dx \hat{p}_j \right) = 0$$

or in matrix notation:

$$AP + \omega B(\omega)P + \omega^2 CP = 0$$
Different degrees of reality

- No impedance: solve $AP + \omega^2 CP = 0$ (generalized)
- Assume $1/Z = 1/Z_0 + Z_1 \omega + Z_2 / \omega$:
  solve $AP + \omega BP + \omega^2 CP = 0$ (quadratic)
- RHS $\neq 0$: solve $(A - D(\omega))P + \omega B(\omega)P + \omega^2 CP = 0$
  (fully non-linear)
Different types of Eigenvalue Problems

- Linear: \( A x = \lambda x \).
- Quadratic: \( A x + \lambda B x + \lambda^2 C x = 0 \).
- Linearized Quadratic (if \( C = I \)):
  \[
  \begin{pmatrix}
  -B & -A \\
  I & 0
  \end{pmatrix}
  \begin{pmatrix}
  \lambda x \\
  x
  \end{pmatrix}
  = \lambda
  \begin{pmatrix}
  C & 0 \\
  0 & I
  \end{pmatrix}
  \begin{pmatrix}
  \lambda x \\
  x
  \end{pmatrix}
  
- Non-linear: \( T(\lambda) x = 0 \).
Search space methods

- Problem: dimension N of the problem is big (>10000)
- Idea:
  - Look in a small subspace \( \mathcal{W} \) of \( \mathbb{R}^N \).
  - Use the approximation to construct a better subspace.
- How?
  - Construct base \( \nu_1 \ldots \nu_k \) of \( \mathcal{W} \).
  - Solve the projected eigenvalue problem: \( W' T(\omega) Wu = 0 \).
  - Check whether \( T(\omega)u < \text{tol} \).
  - Not good enough? Improve search space.
Arnoldi’s Method

- Choose starting vector $\mathbf{v}$ and max. subspace size $k$.
- Construct Krylov space: $\text{span}(\mathbf{v}, A\mathbf{v}, \ldots A^k\mathbf{v})$.
- Project the eigenvalue problem on the Krylov space, and solve it.
- Restart with a different starting vector, if necessary.
Choose starting vector $v$ and max. subspace size $k$.
Set $W = v$, and solve the projected eigenvalue problem.
Select a Ritz pair $(\omega, u)$, and calculate the residual $T(\omega)u$.
Solve $t \perp u$ (approximately) from
$$(I - uu^*)(A - \theta I)(I - uu^*)t = -r.$$ 
Orthogonalize $t$ against $W$ and set $W = [W \ t]$.
Restart after $k$ iterations with the latest Ritz vector.
Differences (1)

- History
  - Arnoldi exists since 1951, widely implemented and optimized
  - Jacobi-Davidson exists since 1996, needs optimization

- Method:
  - Arnoldi constructs search space immediately ($k$ iterations at once)
  - JD solves a small eigenvalue problem and an equation every iteration
Differences (2)

- Convergence speed
  - Arnoldi has linear convergence and a low workload per iteration
  - JD has quadratic convergence and a high workload per iteration
- Adaptability
  - Arnoldi is designed for linear problems
  - JD is designed for any problem
Implementations of the algorithms

Arnoldi has been implemented in ARPACK. Jacobi-Davidson has been implemented by Gerard Sleijpen for linear and generalized problems, and by Martin van Gijzen and Jan-Willem van Leeuwen for quadratic problems.

Issues:

- Must be matrix-free (in Fortran), only use matvec-subroutine.
- Stopping criterion must be equal.
- Maximal search space size must be (nearly) optimal.
- We need meshes for the tests.
Meshes

• Matlab
  • Rectangle of $0.5m \times 0.1m$, 2 grid densities

• Fortran
  • Rectangle of $1m \times 0.2m$, 4 grid densities
  • Rectangular box of $1m \times 0.2m \times 0.1m$, 4 grid densities
  • Combustion chamber ARRIUS, 22000 nodes
Stopping Criterion (1)

The Arnoldi Residual:
\[ \| r_{AR} \|_2 = \left\| \begin{pmatrix} -B & -A \\ I & 0 \end{pmatrix} \begin{pmatrix} \omega p_{AR} \\ p_{AR} \end{pmatrix} - \omega \begin{pmatrix} C & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \omega p_{AR} \\ p_{AR} \end{pmatrix} \right\|_2 \]

The Jacobi-Davidson Residual:
\[ \| r_{JD} \|_2 = \| Ap_{JD} + \omega B p_{JD} + \omega^2 C p_{JD} \|_2 \]
\[ = \left\| \begin{pmatrix} -B & -A \\ I & 0 \end{pmatrix} \begin{pmatrix} \omega p_{JD} \\ p_{JD} \end{pmatrix} - \omega \begin{pmatrix} C & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \omega p_{JD} \\ p_{JD} \end{pmatrix} \right\|_2 \]
\[ \left\| \begin{pmatrix} \omega p_{JD} \\ p_{JD} \end{pmatrix} \right\|_2 = \sqrt{1 + |\omega|^2}, \quad \left\| \begin{pmatrix} \omega p_{AR} \\ p_{AR} \end{pmatrix} \right\|_2 = 1. \]
Stopping Criterion (2)

The Arnoldi (ARPACK) criterion:
$$\| r_{AR} \|_2 < tol \cdot |\omega|$$

The Jacobi-Davidson criterion:
$$\| r_{JD} \|_2 < tol \cdot |\omega| \cdot \sqrt{1 + |\omega|^2}$$
Maximal Search space size

Hard to predict the optimal value.

Tests:

**Figure:** Left: CPU-time, Right: Matvecs
Results, overview

- Linear problems in MATLAB
- Quadratic and nonlinear problems in MATLAB
- Tests done with Fortran
- The ARRIUS chamber
Linear Problems, MATLAB

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>0.46 s</td>
<td>2.57 s</td>
</tr>
<tr>
<td>JD</td>
<td>3.18 s</td>
<td>65.35 s</td>
</tr>
</tbody>
</table>

Table: Results for linear problems using ARPACK and JDQZ

Figure: Convergence history for JDQZ
Quadratic and Nonlinear Problems, MATLAB

<table>
<thead>
<tr>
<th></th>
<th>Small</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR (quad)</td>
<td>0.775 s</td>
<td>11.74 s</td>
</tr>
<tr>
<td>JD (quad)</td>
<td>1.20 s</td>
<td>26.70 s</td>
</tr>
<tr>
<td>AR (NL)</td>
<td>177.5 s</td>
<td>2451 s</td>
</tr>
<tr>
<td>JD (NL)</td>
<td>3.14 s</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table: Results for Quadratic and Nonlinear Problems
Fortran Results, 2D academic case
Fortran Results, 3D academic case
## Fortran Results, 2D academic and ARRIUS

<table>
<thead>
<tr>
<th></th>
<th>$u = 0$</th>
<th>$y = 0.4 + 0.3i$</th>
<th>$y = 3 + 2i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AR</td>
<td>JD</td>
<td>AR</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>55.23</td>
<td>53.64</td>
<td>523.35</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>62.78</td>
<td>70.19</td>
<td>528.77</td>
</tr>
<tr>
<td>$\lambda_{10}$</td>
<td>76.49</td>
<td>116.46</td>
<td>746.36</td>
</tr>
</tbody>
</table>

**Table:** Results for the 2D academic testcase with 8000 nodes

<table>
<thead>
<tr>
<th></th>
<th>$u = 0$</th>
<th>$y = 0.4 + 0.3i$</th>
<th>$y = 3 + 2i$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AR</td>
<td>JD</td>
<td>AR</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>131</td>
<td>131</td>
<td>1158</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>131</td>
<td>223</td>
<td>1211</td>
</tr>
<tr>
<td>$\lambda_{10}$</td>
<td>197</td>
<td>513</td>
<td>1662</td>
</tr>
<tr>
<td>Matvecs</td>
<td>2669</td>
<td>7291</td>
<td>8018</td>
</tr>
</tbody>
</table>

**Table:** Results for the ARRIUS chamber with 22000 nodes
Conclusions

- Arnoldi is the best method for linear problems
- Jacobi-Davidson is better for quadratic problems
- Jacobi-Davidson has potential for nonlinear problems
Future Research

- Improve restart strategy
- Use preconditioning for the correction equation
- Improve JD method for nonlinear problems
- Implement parallel version
Questions