Comparison of Time Stepping Techniques for Compressible Gas Dynamics

Julian Köllermeier

RWTH Aachen University
May 3rd, 2012
Introduction
Aim of the Project

Implementation and comparison of efficient implicit time stepping schemes for non-linear PDE systems
Aim of the Project

Implementation and comparison of efficient implicit time stepping schemes for non-linear PDE systems

Steps
- implicit time discretization using adaptive time stepping
- non-linear solver and computation of Jacobian
- preconditioned linear solver
- comparison of methods
Aim of the Project

Implementation and comparison of efficient implicit time stepping schemes for non-linear PDE systems

Steps

- implicit time discretization using adaptive time stepping
- non-linear solver and computation of Jacobian
- preconditioned linear solver
- comparison of methods
  ⇒ runtime speed-up
Example: Heat Equation

\[ \frac{\partial T}{\partial t} + \nabla (\kappa \nabla T - q) = 0 \]
Example: Heat Equation

\[ \frac{\partial T}{\partial t} + \nabla \left( -\kappa \nabla T \right) = 0 \]

*Fourier’s law*

heat flux \( q \) proportional to negative temperature gradient over surface

\[ q = -\kappa(T) \nabla T \]
**Example: Heat Equation**

\[
\frac{\partial T}{\partial t} + \nabla \left( -\kappa \nabla T \right) = 0
\]

**Fourier’s law**

Heat flux \( q \) proportional to negative temperature gradient over surface

\[
q = -\kappa(T) \nabla T
\]

**Models for heat conductivity \( \kappa \)**

- \( \kappa = 1 = \text{const} \) (linear)
- \( \kappa(T) = c_1 + c_2 \cdot T^2 \) (non-linear), \( c_1, c_2 \in \mathbb{R}^+ \)
Explicit vs. Implicit schemes

Explicit schemes

+ implementation
+ no linear or non-linear solver needed

– CFL condition
Explicit vs. Implicit schemes

**Explicit schemes**

+ implementation
+ no linear or non-linear solver needed
- CFL condition

**Implicit schemes**

- implementation
- solution of large non-linear system of equations
- needs Jacobian
+ no time step constraints
Numerics
Spatial Discretization

\[
\frac{\partial Q}{\partial t} + L(Q) = 0
\]
Spatial Discretization

\[ \frac{\partial Q}{\partial t} + L(Q) = 0 \]

Spatial discretization scheme by Kapper
- least-squares reconstruction of fluxes
- second order in space
- six-point stencil for flux reconstruction
- nine-point stencil for each cell
Spatial Discretization

\[ \frac{\partial Q}{\partial t} + L(Q) = 0 \]

Spatial discretization scheme by Kapper

- least-squares reconstruction of fluxes
- second order in space
- six-point stencil for flux reconstruction
- nine-point stencil for each cell

\[ \frac{\partial Q}{\partial t} + R(Q) = 0 \]
Time Discretization

Apply different time stepping methods to

\[
\frac{\partial Q}{\partial t} + R(Q) = 0
\]
Time Discretization

Apply different time stepping methods to

$$\frac{\partial Q}{\partial t} + R(Q) = 0$$

**Time stepping methods**

- implicit Euler method
- implicit midpoint method
- implicit trapezoidal method
- BDF2 method
- Richardson extrapolation
Adaptive Time Stepping

Aim

calculate with largest possible time step $\Delta t$ subject to given error bound $\epsilon$ and stability of the method
Adaptive Time Stepping

Aim

calculate with largest possible time step $\Delta t$ subject to given error bound $\epsilon$ and stability of the method

Ingredients

- sensor for the error (error estimate)
- controller for $\Delta t$ or $h$ (time step adjustment strategy)
- use control theory model
Error Estimation

Comparison with higher order method

\[ \hat{r}_{n+1} = \| y_1 - y_2 \| \]
Error Estimation

Comparison with higher order method

\[ \hat{r}_{n+1} = \|y_1 - y_2\| \]

Comparison by step size variation

\[
\begin{align*}
y^h_{n+1} &= y(t_{n+1}) + \phi(t_n) h^k + O(h^{k+1}) \\
y^m_{n+1} &= y(t_{n+1}) + \phi(t_n) \frac{h^k}{m} + O\left(\frac{h^{k+1}}{m}\right)
\end{align*}
\]
Error Estimation

Comparison with higher order method

\[ \hat{r}_{n+1} = \| y_1 - y_2 \| \]

Comparison by step size variation

\[ y^h_{n+1} = y(t_{n+1}) + \phi(t_n) h^k + O(h^{k+1}) \]

\[ y^m_{n+1} = y(t_{n+1}) + \phi(t_n) \frac{h^k}{m} + O \left( \frac{h^{k+1}}{m} \right) \]

\[ \hat{r}_{n+1} = \left\| \frac{y^h_{n+1} - y^m_{n+1}}{1 - m^{-k}} \right\| \]
Elementary Error Control

Controller, see SöDERLIND

\[ h_{n+1} = \left( \frac{\epsilon}{\tilde{r}_{n+1}} \right)^{\frac{1}{k}} h_n \]
Elementary Error Control

Controller, see SöDERLIND

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{\frac{1}{k}} h_n \]

Properties

- error estimate larger \( \epsilon \Rightarrow \) decrease time step size
- error estimate smaller \( \epsilon \Rightarrow \) increase time step size
Integral Controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} h_n \]
Integral Controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} h_n \]

taking logarithm on both sides
Integral Controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} h_n \]

taking logarithm on both sides

\[ \log h_{n+1} = \log h_n + k_I (\log \epsilon - \log \hat{r}_{n+1}) \]
Integral Controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} h_n \]

taking logarithm on both sides

\[ \log h_{n+1} = \log h_n + k_I (\log \epsilon - \log \hat{r}_{n+1}) \]

- \( \log \epsilon - \log \hat{r}_{n+1} \): control error
- \( \epsilon \): setpoint
- \( k_I \): integral gain
Integral Controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_l} h_n \]

taking logarithm on both sides

\[ \log h_{n+1} = \log h_n + k_l (\log \epsilon - \log \hat{r}_{n+1}) \]

- \( \log \epsilon - \log \hat{r}_{n+1} \): control error
- \( \epsilon \): setpoint
- \( k_l \): integral gain (\( k_l k \in [0, 2] \) for stability)
Control Theory Model

Reference \[ \rightarrow \] Measured error \[ \rightarrow \] Controller \[ \rightarrow \] System input \[ \rightarrow \] System \[ \rightarrow \] System output

Measured output \[ \rightarrow \] Sensor

- when is the whole system stable?
- which controllers can we use?
Control Theory Model

Questions

- when is the whole system stable?
Control Theory Model

Questions

- when is the whole system stable?
- which controllers can we use?
Control Theory Model

Questions
- When is the whole system stable?
- Which controllers can we use?
Other Controllers

Integral controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_l} h_n \]
Other Controllers

Integral controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} h_n \]

Proportional-integral controller

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_I} \left( \frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_P} h_n \]
Other Controllers

**Integral controller**

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_l} h_n \]

**Proportional-integral controller**

\[ h_{n+1} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_l} \left( \frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_P} h_n \]

**Predictive controller**

\[ \frac{h_{n+1}}{h_n} = \left( \frac{\epsilon}{\hat{r}_{n+1}} \right)^{k_E} \left( \frac{\hat{r}_n}{\hat{r}_{n+1}} \right)^{k_R} \frac{h_n}{h_{n-1}} \]
Non-Linear System

\[
\frac{\partial Q}{\partial t} + R(Q) = 0
\]
Non-Linear System

\[
\frac{\partial Q}{\partial t} + R(Q) = 0 \quad \Rightarrow \quad \tilde{R}(Q^{n+1}) = 0
\]
Non-Linear System

\[ \frac{\partial Q}{\partial t} + R(Q) = 0 \quad \Rightarrow \quad \tilde{R}(Q^{n+1}) = 0 \]

Iterative solution is necessary. Define the update

\[ \Delta Q_k^{n+1} = Q_k^{n+1} - Q_k^{n+1} \]
Non-Linear System

\[
\frac{\partial Q}{\partial t} + R(Q) = 0 \quad \Rightarrow \quad \tilde{R}(Q^{n+1}) = 0
\]

Iterative solution is necessary. Define the update

\[
\Delta Q^{n+1}_{k+1} = Q^{n+1}_{k+1} - Q^{n+1}_k
\]

Newton's algorithm

\[
\frac{\partial \tilde{R}}{\partial Q} \Delta Q^{n+1}_{k+1} = -\tilde{R}(Q^{n+1}_k)
\]
Non-Linear System

\[ \frac{\partial Q}{\partial t} + R(Q) = 0 \quad \Rightarrow \quad \tilde{R}(Q^{n+1}) = 0 \]

Iterative solution is necessary. Define the update

\[ \Delta Q^{n+1}_{k+1} = Q^{n+1}_{k+1} - Q^{n+1}_k \]

Newton’s algorithm

\[ \frac{\partial \tilde{R}}{\partial Q} \Delta Q^{n+1}_{k+1} = -\tilde{R}(Q^{n+1}_k) \]

Dual time stepping

\[ \left( \frac{1}{\Delta \tau} I + \frac{\partial \tilde{R}}{\partial Q} \right) \Delta Q^{n+1}_{k+1} = -\tilde{R}(Q^{n+1}_k) \]
Non-linear solver needs Jacobian of discretized residual function $\tilde{R}$
Computation of Jacobian

Non-linear solver needs Jacobian of discretized residual function $\tilde{R}$

**Analytical computation**

+ exact derivative
  
- error prone, tedious
Computation of Jacobian

Non-linear solver needs Jacobian of discretized residual function $\tilde{R}$

Analytical computation

+ exact derivative
  - error prone, tedious

Finite differences

+ arbitrary right hand side function
  - only approximation of derivative
  - number of right hand side evaluations increases with unknowns
Computation of Jacobian

Non-linear solver needs Jacobian of discretized residual function $\tilde{R}$

### Analytical computation

- exact derivative
- error prone, tedious

### Finite differences

- arbitrary right hand side function
- only approximation of derivative
- number of right hand side evaluations increases with unknowns

### Solution

efficient finite differences
Solution of linear system
Solution of linear system

Iterative solver

- GMRES
- BiCG
- BiCGSTAB
Solution of linear system

Iterative solver
- GMRES
- BiCG
- BiCGSTAB

Preconditioner
- SSOR
- ILU
Results
Testcase

Linear test
- linear heat equation
- square domain
- quasi 1D setting, Dirichlet BC $Q = 0$, IC $Q = 0$
- right hand side sine function

Non-linear test
- non-linear heat equation
- square domain
- Dirichlet BC $Q = 0$, IC $Q = 1$
- with constant right hand side function
Testcase

Linear test

- linear heat equation
- square domain
- quasi 1D setting, Dirichlet BC $Q = 0$, IC $Q = 0$
- right hand side sine function
Testcase

Linear test
- linear heat equation
- square domain
- quasi 1D setting, Dirichlet BC $Q = 0$, IC $Q = 0$
- right hand side sine function

Non-linear test
- non-linear heat equation
- square domain
- Dirichlet BC $Q = 0$, IC $Q = 1$
- with constant right hand side function
Linear solver iterations and runtime
Error of Jacobian calculation

![Graph showing the error of Jacobian calculation for different mesh sizes (40x40, 80x80, 160x160). The x-axis represents the relative error, and the y-axis represents the error magnitude. The graph demonstrates that the error decreases as the mesh size increases, with the 160x160 mesh showing the least error.](image-url)
Comparison of Time Stepping Techniques

Linear test case

![Graph comparison of time stepping techniques](image.png)
Non-linear test case

Comparison of Time Stepping Techniques
Test case

Linear heat equation with right hand side function
Test case

Linear heat equation with right hand side function

Non-linear right hand side function

Comparison of Time Stepping Techniques
Test case

Linear heat equation with right hand side function

Non-linear right hand side function

- increasing time step at the beginning
- strong non-linearity near jump at $Q = 0.15$ leads to small timesteps
- large time steps in the end
Comparison of Time Stepping Techniques

I vs. PI controller

timesteps: 1170
rejects: 34
I vs. PI controller

Comparison of Time Stepping Techniques

I10:
- Timesteps: 1170
- Rejects: 34

PI42:
- Timesteps: 1169
- Rejects: 32
Comparison of Time Stepping Techniques

PI vs. PC controller

timesteps: 1169
rejects: 32
Comparison of Time Stepping Techniques

PI vs. PC controller

- **PI42**
  - Timesteps: 1169
  - Rejects: 32

- **PC47**
  - Timesteps: 1128
  - Rejects: 14
Test setting

- non-linear heat equation with constant right hand side
- calculate $Q$ at time $t_{end} = 1\text{sec}$
- expl. method: time step size limited by stability
- impl. methods: maximum time step size for 1% relative error
Comparison of Time Stepping Techniques

Runtime Measurements

<table>
<thead>
<tr>
<th>Technique</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>expl. Euler</td>
<td>18.2</td>
</tr>
<tr>
<td>impl. Euler</td>
<td>2.5</td>
</tr>
<tr>
<td>ATS Euler</td>
<td>2.0</td>
</tr>
<tr>
<td>Richardson Euler</td>
<td>0.7</td>
</tr>
<tr>
<td>ATS Richardson Euler</td>
<td></td>
</tr>
</tbody>
</table>
Runtime Measurements

Comparison of Time Stepping Techniques

- expl. Euler: 18.2 sec
- impl. Euler: 2.5 sec
- ATS impl. Euler: 2.0 sec
- Richardson Extrapolation: 0.7 sec
Conclusion
Successful implementation and comparison of different implicit time stepping techniques

- different implicit time discretizations
- adaptive time stepping
- non-linear and linear solvers as well as preconditioners
- comparison of methods for acceleration of simulation
- speedup of more than 90% with respect to explicit method
Thank you for your attention