Nonlinear Systems of Equations

Philipp Birken

Universität Kassel (SFB/TRR 30)
Soon: University of Lund

October 7th 2013
Efficient solution of large systems of non-linear PDEs in science
Lyon

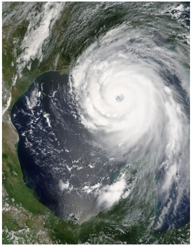
Outline

Outline

Steady flow problems: One system

Figure: Pershing-II, US Army, PD (left), A320 wing, Kudak, CC-by-SA 3.0 (right)

Unsteady flow problems: Sequence of Systems



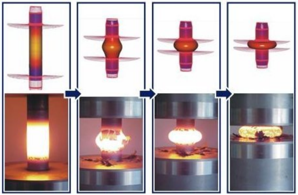
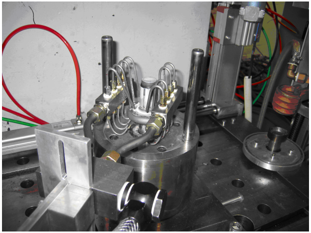



Figure: Hurricane Katrina, NASA, PD; Lillgrund Offshore windfarm, Mariusz Padziora, CC-by-sa 3.0 via Wikimedia Commons; Gas Quenching, Steinhoff

Example I: Navier-Stokes solver

- Consider unsteady 3D compressible viscous flow problems
- Important: Turbulence, Boundary Layer, Mach number
- Discretization in space via FV or DG
- Leads to initial value problem in time
- $\rightarrow$ huge number of unknowns, problem is typically stiff.
- $\rightarrow$ implicit time integration necessary.
- Goal: Thus sequence of nonlinear systems!

Overview: B., *Numerical methods for the unsteady compressible Navier-Stokes equations*, Habilitation Thesis, 2012, University of Kassel

Example II: Nonlinear Optimization

- Consider the problem

$$\min_{\mathbb{R}^m} \mathbf{f}(\mathbf{x})$$

  with $\mathbf{f}(\mathbf{x})$ nonlinear.
- Local minima are characterized by

$$\nabla \mathbf{f}(\mathbf{x}) = 0$$

- Again a nonlinear system!
- Let's solve them!

What kind of methods do we need?

Solver needs to respect hardware trend

- High degree of parallelism
- Low storage per process

Software needs to be used

- Modularity and Flexibility
- Ease of implementation

And: Superfast!

Figure: Cray Hermit in Stuttgart; Bild: ThE cRaCkEr, CC-by-sa 3.0, via Wikimedia Commons

Outline

Implicit time integration

- Discretized PDEs lead to stiff problems, thus a large stability region is required: A-stable methods!
- Unsteady problem: Higher order and time adaptivity.
- BDF often used, but does not fit the profile.
- Bijl et al (01,02): ESDIRK methods competitive!
- One explicit and subsequent backward Euler steps.
- ESDIRK: Specific diagonally implicit Runge-Kutta methods
- Time adaptivity via embedded method
- Software TEMPO (Time adaptivE iMPlicit cOnservation law solver)

ESDIRK schemes

- See Kennedy, Carpenter '01
- ESDIRK 3: 4 stages, order 3, embedding order 2
- ESDIRK 4: 6 stages, order 4, embedding order 3
- All A-stable, L-stable, stiffly stable
- First stage is equal to last stage from previous time step, thus comes for free

$$\mathbf{k}_1 = \mathbf{f}(\mathbf{u}_n)$$

$$\mathbf{k}_i = \mathbf{f}(\mathbf{u}^n + \Delta t \sum_{j=1}^{i} a_{ij}\mathbf{k}_j), \quad i = 2, ..., s$$

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \Delta t \sum_{i=1}^{s} a_{si}\mathbf{k}_i.$$

Rosenbrock-Wanner (ROW) schemes

- Linearize DIRK scheme, use fixed $\mathbf{W} \approx \frac{\partial \mathbf{f}(\mathbf{u}^n)}{\partial \mathbf{u}}$
- Impose additional order conditions
- ROS34PW2: 4 stages, order 3, embedding order 2
- RODASP: 6 stages, order 4, embedding order 3
- A-stability, L-stability possible

$$
\begin{aligned}
(\mathbf{I} - \gamma \Delta t \mathbf{W})\mathbf{k}_i &= \mathbf{f}(\mathbf{s}_i) + \Delta t \mathbf{W} \sum_{j=1}^{i-1} \gamma_{ij} \mathbf{k}_j, \quad i = 1, ..., s \\
\mathbf{s}_i &= \mathbf{u}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{k}_j, \quad i = 1, ..., s \\
\mathbf{u}^{n+1} &= \mathbf{u}^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{k}_i.
\end{aligned}
$$

The specific system

- Form of equation is independent of precise time integration scheme

$$\mathbf{u} = \alpha \Delta t \mathbf{f}(\mathbf{u}) + \psi.$$

- Rosenbrock case:

$$(\mathbf{I} - \gamma \Delta t \mathbf{W})\mathbf{k}_i = \psi$$

- This type of linear system also appears when applying Newton to the nonlinear equation!
- Thus same type of schemes relevant for that class as well

Basic Flow chart for DIRK scheme

Given error tolerance TOL, initial time t_0 and time step size Δt_0

- For $i = 1, ..., s$
  - For $k = 0, 1, ...$ until termination criterion with tolerance $TOL/5$ is satisfied or MAX_SOLVER_ITER has been reached
- If MAX_SOLVER_ITER has been reached, but the tolerance test has not been passed, repeat time step with $\Delta t_n = \Delta t_n / 4$
- Estimate local error and compute new time step size Δt_{n+1}
- $t_{n+1} = t_n + \Delta t_n$

Note: Puts additional bound on time step via nonlinear solver

Outline

How to solve nonlinear systems

- Except for very special problems no direct solvers or formulas available. Example: Quadratic Equations
- Thus iterative schemes needed
- Large number of algorithms for scalar problems
- Less so for systems
  - Fixed Point methods
  - Multigrid methods
  - Newton-Raphson method
  - Homotopy methods
- Need to be able to compare different schemes!

Speed of convergence

A method with iterates $x^{(k)}$, $k \in \mathbb{N}$, which converges to x^* is called

- linearly convergent to x^*, if $\|x^{(n+1)} - x^*\| \leq C\|x^{(k)} - x^*\|$, $0 < C < 1$,
- superlinearly convergent of order p to x^*, if $\|x^{(n+1)} - x^*\| \leq C\|x^{(k)} - x^*\|^p$ with $p > 1$, $C > 0$,
- superlinearly convergent to x^*, if $\lim_{n\to\infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|} = 0$,
- quadratically convergent to x^*, if $\|x^{(n+1)} - x^*\| \leq C\|x^{(k)} - x^*\|^2$ with $C > 0$.

Termination criteria

- Want to solve equation such that error in approximate solution is smaller than tolerance τ
- Problem: Don't know the error
- Solution: Use Residual as indicator of error
- Works if we are reasonably close to solution
- Relative criterion

$$\|\mathbf{r}(\mathbf{x}^k)\| \leq \tau_r \|\mathbf{r}(\mathbf{x}^0)\|$$

- Absolute criterion

$$\|\mathbf{r}(\mathbf{x}^k)\| \leq \tau_a\|$$

- Mixed

$$\|\mathbf{r}(\mathbf{x}^k)\| \leq \tau_r \|\mathbf{r}(\mathbf{x}^0)\| + \tau_a$$

Fixed point methods

- Fixed point equation
$$\mathbf{g}(\mathbf{x}) = \mathbf{x}$$
- Fixed point iteration
$$\mathbf{x}^{k+1} = \mathbf{g}(\mathbf{x}^k)$$
- Converges linearly provided that $\mathbf{g}$ is selfmap on compact domain and Lipschitz continuous with Lipschitz constant $L < 1$
- Easy to implement, parallelizes, etc.
- Not suitable for stiff problems, because $L < 1$ only if time step is small enough
- Therefore not fast!

Outline

Multigrid methods

- Represent discrete solution as sum of eigenvectors.
- These are discrete versions of the eigenfunctions of the differential operators.
- These are periodic functions like $\sin j\pi\Theta$, $\cos j\pi\Theta$, $e^{ij\Theta}$
- Idea: Decompose representation in low and high frequency parts.
- Reduce high frequency error components significantly using cheap iteration (smoother)
- Approximate low frequency errors on coarse grid $\rightarrow$ Reduction of problem size
- Recursive application $\rightarrow$ Multigrid method!

Multigrid: Linear variant

Given hierarchy of grids and discretization A_l of problem on all levels.

Function MG($\mathbf{x}_l, \mathbf{b}_l, l$)

- if $(l = 0)$, $\mathbf{x}_l = \mathbf{A}_l^{-1}\mathbf{b}_l$ (Exact solve on coarse grid)
- else
  - $\mathbf{x}_l = \mathbf{S}_l^{\nu_1}(\mathbf{x}_l, \mathbf{b}_l)$ (Presmoothing)
  - $\mathbf{r}_{l-1} = \mathbf{R}_{l-1,l}(\mathbf{b}_l - \mathbf{A}_l\mathbf{x}_l)$ (Restriction)
  - $\mathbf{v}_{l-1} = 0$
  - For $(j = 0;\ j < \gamma;\ j++)$ $MG(\mathbf{v}_{l-1}, \mathbf{r}_{l-1}, l-1)$ (Computation of coarse grid correction)
  - $\mathbf{x}_l = \mathbf{x}_l + \mathbf{P}_{l,l-1}\mathbf{v}_{l-1}$ (Correction via prolongation)
  - $\mathbf{x}_l = \mathbf{S}_l^{\nu_2}(\mathbf{x}_l, \mathbf{b}_l)$ (Postsmoothing)
- end if

Multigrid: Nonlinear variant

Given hierarchy of grids and discretization of problem on all levels.

Function FAS-MG($\tilde{\mathbf{u}}_l, \mathbf{u}_l, \mathbf{s}_l, l$)

- $\mathbf{u}_l = \mathbf{S}_l^{\nu_1}(\tilde{\mathbf{u}}_l, \mathbf{s}_l)$ (Presmoothing)
- if $(l > 0)$
  - $\mathbf{r}_l = \mathbf{s}_l - \mathbf{F}_l(\mathbf{u_l})$
  - $\tilde{\mathbf{u}}_{l-1} = \mathbf{R}_{l-1,l}\mathbf{u}_l$ (Restriction of solution)
  - $\mathbf{s}_{l-1} = \mathbf{F}_{l-1}(\tilde{\mathbf{u}}_{l-1}) + \mathbf{R}_{l-1,l}\mathbf{r}_l$ (Restriction of residual)
  - For $(j = 0;\ j < \gamma;\ j++)$ FAS-MG($\tilde{\mathbf{u}}_{l-1}, \mathbf{u}_{l-1}, \mathbf{s}_{l-1}, l-1$) (Computation of the coarse grid correction)
  - $\mathbf{u}_l = \mathbf{u}_l + \mathbf{P}_{l,l-1}(\mathbf{u}_{l-1} - \tilde{\mathbf{u}}_{l-1})$ (Correction via Prolongation)
  - $\mathbf{u}_l = \mathbf{S}_l^{\nu_2}(\mathbf{u}_l, \mathbf{s}_l)$ (Postsmoothing)
- end if

Possible smoothers

- Smoothers are problem dependent!
- Need to be designed specifically for your PDE
- Laplace: Jacobi, Gauß-Seidel, ILU
- Navier-Stokes: SGS, Runge-Kutta methods
- All important problems like parallel scaling, memory requirements, speed hinge on smoother
- With superb smoother, multigrid will converge linearly in 3-5 steps (textbook multigrid scheme)
- Finding good smoothers open problem for important PDEs
- Can later be used inside a Newton scheme

Multigrid convergence: UFLO103 on structured grid

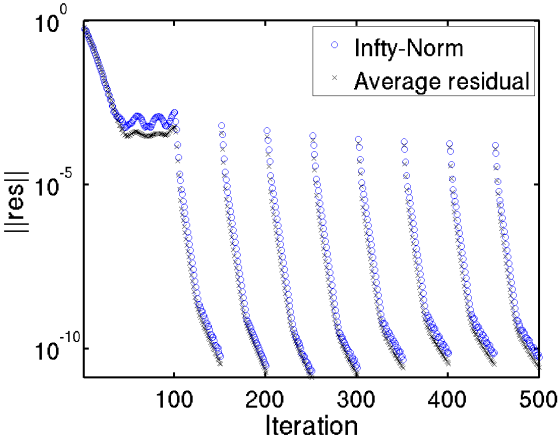


Figure: UFLO103 convergence, first 100 steps using the steady state solver, then 50 iteration of dual time stepping per time step (Jameson '91, Caughey & Jameson '01)

Multigrid convergence: DLR TAU

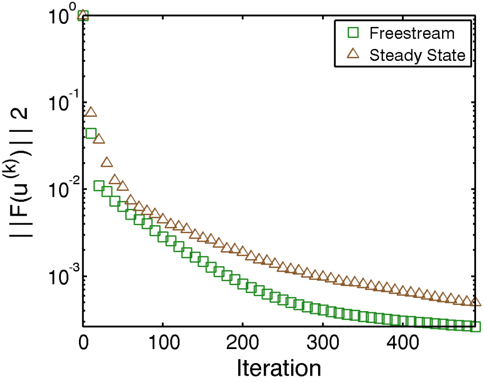


Figure: DLR TAU dual time stepping for two different systems.

Outline

Illustration of Newton's method

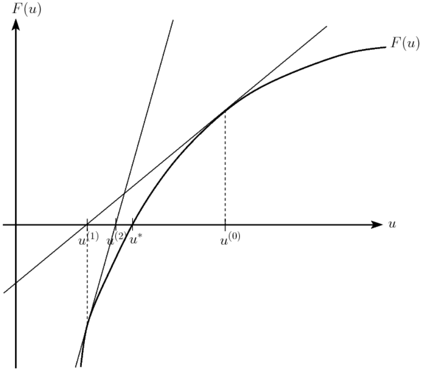


Figure: Illustration of Newton's method in one dimension (left); convergence curve (right)

Newton's method

$$\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}^{(k)}} \Delta \mathbf{x} = -\mathbf{F}(\mathbf{x}^{(k)}),$$
$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta \mathbf{x}.$$

- Use first order approximation of function
- Local quadratic convergence
- Outside no statement possible
- Nonlinear system transformed into sequence of linear systems

Flow-Chart for DIRK scheme with Newton

Given error tolerance TOL, initial time t_0 and time step size Δt_0

- For $i = 1, ..., s$
  - For $k = 0, 1, ...$ until termination criterion with tolerance $TOL/5$ is satisfied or MAX_NEWTON_ITER has been reached
    - Solve linear system up to certain tolerance
- If MAX_NEWTON_ITER has been reached, but the tolerance test has not been passed, repeat time step with $\Delta t_n = \Delta t_n/4$
- Estimate local error and compute new time step size Δt_{n+1}
- $t_{n+1} = t_n + \Delta t_n$

Note: Puts additional bound on time step via nonlinear solver

Summary

- Nonlinear systems arise particularly in the solution of nonlinear PDEs and nonlinear optimization
- They have to be solved iteratively
- Want schemes that scale in parallel and use little storage
- Fixed Point not suitable for stiff problems
- Multigrid needs to be adjusted to specific problem
- Newton local quadratic convergence