Scientific Computing WS 2019/2020

Lecture 24

Jürgen Fuhrmann

juergen.fuhrmann@wias-berlin.de

Nonlinear problems: motivation

- Assume nonlinear dependency of some coefficients of the equation on the solution. E.g. nonlinear diffusion problem

$$-\nabla \cdot (D(u)\vec{\nabla} u) = f \quad \text{in } \Omega$$
$$u = g \text{on} \partial\Omega$$

- FE+FV discretization methods lead to large nonlinear systems of equations

Nonlinear problems: caution!

This is a significantly more complex world:

- Possibly multiple solution branches
- Weak formulations in L^p spaces
- No direct solution methods
- Narrow domains of definition (e.g. only for positive solutions)

Finite element discretization for nonlinear diffusion

- Find $u_h \in V_h$ such that for all $w_h \in V_h$:

$$\int_\Omega D(u_h)\vec{\nabla} u_h \cdot \vec{\nabla} w_h \; dx = \int_\Omega f w_h \; dx$$

- Use appropriate quadrature rules for the nonlinear integrals
- Discrete system

$$A(u_h) = F(u_h)$$

Finite volume discretization for nonlinear diffusion

$$
\begin{aligned}
0 &= \int_{\omega_k} \left(-\nabla \cdot D(u)\vec{\nabla} u - f \right) d\omega \\
&= -\int_{\partial\omega_k} D(u)\vec{\nabla} u \cdot \mathbf{n}_k d\gamma - \int_{\omega_k} f d\omega && \text{(Gauss)} \\
&= -\sum_{L \in \mathcal{N}_k} \int_{\sigma_{kl}} D(u)\vec{\nabla} u \cdot \mathbf{n}_{kl} d\gamma - \int_{\gamma_k} D(u)\vec{\nabla} u \cdot \mathbf{n} d\gamma - \int_{\omega_k} f d\omega \\
&\approx \sum_{L \in \mathcal{N}_k} \frac{\sigma_{kl}}{h_{kl}} g_{kl}(u_k, u_l) + |\gamma_k| \alpha (u_k - w_k) - |\omega_k| f_k
\end{aligned}
$$

with

$$
g_{kl}(u_k, u_l) = \begin{cases} D(\frac{1}{2}(u_k + u_l))(u_k - u_l) \\ \text{or} \quad \mathcal{D}(u_k) - \mathcal{D}(u_l) \end{cases}
$$

where $\mathcal{D}(u) = \int_0^u D(\xi)\, d\xi$ (exact solution ansatz at discretization edge)

- Discrete system

$$
A(u_h) = F(u_h)
$$

Iterative solution methods: fixed point iteration

- Let $u \in \mathbb{R}^n$.
- Problem: $A(u) = f$:
- Assume $A(u) = M(u)u$, where for each u, $M(u) : \mathbb{R}^n \to \mathbb{R}^n$ is a linear operator.
- Iteration scheme:

  Choose u_0, $i \leftarrow 0$;
  while *not converged* **do**
      Solve $M(u_i)u_{i+1} = f$;
      $i \leftarrow i + 1$;
  end
- Convergence criteria:
  - residual based: $||A(u) - f|| < \varepsilon$
  - update based $||u_{i+1} - u_i|| < \varepsilon$
- Large domain of convergence
- Convergence may be slow
- Smooth coefficients not necessary

Iterative solution methods: Newton method

- Solve

$$A(u) = \begin{pmatrix} A_1(u_1 \dots u_n) \\ A_2(u_1 \dots u_n) \\ \vdots \\ A_n(u_1 \dots u_n) \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = f$$

- Jacobi matrix (Frechet derivative) for given u: $A'(u) = (a_{kl})$ with

$$a_{kl} = \frac{\partial}{\partial u_l} A_k(u_1 \dots u_n)$$

- Iteration scheme:

Choose u_0, $i \leftarrow 0$;
while *not converged* **do**
  Calculate residual $r_i = A(u_i) - f$;
  Calculate Jacobi matrix $A'(u_i)$;
  Solve update problem $A'(u_i)h_i = r_i$;
  Update solution: $u_{i+1} = u_i - h_i$;
  $i \leftarrow i + 1$;
end

Newton method II

- Convergence criteria:
  - residual based: $||r_i|| < \varepsilon$
  - update based $||h_i|| < \varepsilon$
- Limited domain of convergence
- Slow initial convergence
- Fast (quadratic) convergence close to solution

Linear and quadratic convergence

- Let $e_i = u_i - \hat{u}$.
- Linear convergence: observed for e.g. linear systems:
  Asymptically constant error contraction rate

$$\frac{||e_{i+1}||}{||e_i||} \sim \rho < 1$$

- Quadratic convergence: $\exists i_0 > 0$ such that $\forall i > i_0$,

$$\frac{||e_{i+1}||}{||e_i||^2} \leq M < 1.$$

$$\frac{\frac{||e_{i+1}||}{||e_i||}}{\frac{||e_i||}{||e_{i-1}||}} = \frac{||e_{i+1}||}{\frac{||e_i||^2}{||e_{i-1}||}} \leq ||e_{i-1}||M$$

  As $||e_i||$ decreases, the contraction rate decreases
- In practice, we can watch $||r_i||$ or $||h_i||$

Damped Newton method

- Remedy for small domain of convergence: damping

  Choose u_0, $i \leftarrow 0$, damping parameter $d < 1$;
  while *not converged* **do**
  Calculate residual $r_i = A(u_i) - f$;
  Calculate Jacobi matrix $A'(u_i)$;
  Solve update problem $A'(u_i)h_i = r_i$;
  Update solution: $u_{i+1} = u_i - dh_i$;
  $i \leftarrow i + 1$;
  end

- Damping slows convergence down from quadratic to linear

Damped Newton method II

- Better way: increase damping parameter during iteration:

  Choose u_0, $i \leftarrow 0$,damping $d_0 < 1$, growth factor $\delta > 1$;
  while *not converged* **do**
      Calculate residual $r_i = A(u_i) - f$;
      Calculate Jacobi matrix $A'(u_i)$;
      Solve update problem $A'(u_i)h_i = r_i$;
      Update solution: $u_{i+1} = u_i - d_i h_i$;
      Update damping parameter: $d_{i+1} = \min(1, \delta d_i)$;
      $i \leftarrow i + 1$;
  end

- Increase damping until it becomes 1 and quadratic convergence is achieved

Newton method: further issues

- In each iteration step we have to solve linear system of equations
  - Can be done iteratively, e.g. with the LU factorization of the Jacobi matrix from first solution step as a preconditioner
  - Linear iterative solution accuracy my be relaxed, but this may diminuish quadratic convergence
- Quadratic convergence yields very accurate solution with no large additional effort: once we are in the quadratic convergence region, convergence is very fast
- Monotonicity test: check if residual grows, this is often an sign that the iteration will diverge anyway.

Newton method: embedding

- Embedding method for parameter dependent problems.
- Solve $A(u_\lambda, \lambda) = f$ for $\lambda = 1$.
- Assume $A(u_0, 0)$ can be easily solved.
- Parameter embedding method:

  Solve $A(u_0, 0) = f$;
  Choose initial step size δ;
  Set $\lambda = 0$;
  while $\lambda < 1$ **do**
  | Solve $A(u_{\lambda+\delta}, \lambda + \delta) = 0$ with initial value1 u_λ;
  | $\lambda \leftarrow \min(\lambda + \delta)$;
  end

- Possibly decrease stepsize if Newton's method does not converge, increase it later
- Parameter embedding + damping + update based convergence control go a long way to solve even strongly nonlinear problems!

Dual numbers

Ring of dual numbers $\mathbb{D}$:

- Let $\varepsilon^2 = 0$.
  $\mathbb{D} = \{a + b\varepsilon \mid a, b \in \mathbb{R}\} = \left\{ \begin{pmatrix} a & b \\ 0 & a \end{pmatrix} \mid a, b \in \mathbb{R} \right\} \subset \mathbb{R}^{2\times 2}$
- Let $p(x) = \sum_{i=0}^{n} p_i x^i$. Then

$$
\begin{aligned}
p(a + b\varepsilon) &= \sum_{i=0}^{n} p_i a^i + \sum_{i=1}^{n} i p_i a^{i-1} b\varepsilon \\
&= p(a) + bp'(a)\varepsilon
\end{aligned}
$$

- Generalization to any analytical function
- $\Rightarrow$ automatic evaluation of function and derivative at once
  $\equiv$ forward mode automatic differentiation
- Multivariate dual numbers: generalization for partial derivatives

Automatic differentiation in Julia

- Julia: DualNumbers.jl as part of ForwardDiff.jl
- Write function once, evaluate for real or dual nubers $\Rightarrow$ get the evaluation of the derivative "for free"
- $\Rightarrow$ easy handling of complicated nonlinearities