Finite element methods in scientific computing

Wolfgang Bangerth, Colorado State University
Lecture 31.61:

Nonlinear problems

Part 3a: A “complete” Newton's method for the minimal surface equation, step-77
Recall from previous lectures: Our goal is to solve

\[-\nabla \cdot \left( A \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \right) = f \iff f + \nabla \cdot \left( A \frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \right) = 0\]

Newton's method: Iterate

\[\left[ R'(u_k) \right] \delta u_k = -R(u_k), \quad u_{k+1} = u_k + \alpha_k \delta u_k\]

step-15 does this. But we had some open questions:

- What is the variational formulation of this?
- How to choose the step length alpha?
- Could we make all of this a bit cheaper?
Newton's method for PDEs

Question: What is the variational formulation of

\[
\left[ R'(u_k) \right] \delta u_k = -R(u_k), \quad u_{k+1} = u_k + \alpha_k \delta u_k
\]

Answer: With...

\[
\langle \varphi, R'(u_k)(\delta u_k) \rangle := \left( \nabla \varphi, \frac{A}{\sqrt{1+|\nabla u_k|^2}} \nabla \delta u_k \right) - \left( \nabla \varphi, \frac{A(\nabla u_k \cdot \nabla \delta u_k)}{(1+|\nabla u_k|^2)^{3/2}} \nabla u_k \right)
\]

\[
\langle \varphi, R(u_k) \rangle := \langle \varphi, f \rangle + \left( \nabla \varphi, \frac{A}{\sqrt{1+|\nabla u_k|^2}} \nabla u_k \right)
\]

...we arrive at this in each Newton step:

\[
\left( \nabla \varphi, \frac{A}{\sqrt{1+|\nabla u_k|^2}} \nabla \delta u_k \right) - \left( \nabla \varphi, \frac{A(\nabla u_k \cdot \nabla \delta u_k)}{(1+|\nabla u_k|^2)^{3/2}} \nabla u_k \right) = -\langle \varphi, f \rangle - \left( \nabla \varphi, \frac{A}{\sqrt{1+|\nabla u_k|^2}} \nabla u_k \right) \quad \forall \varphi \in H_0^1
\]
**Question:** Newton's method does not always converge.

**Answer:** Yes. In many cases on needs a “globalization” strategy

\[ [R'(u_k)] \delta u_k = -R(u_k), \quad u_{k+1} = u_k + \alpha_k \delta u_k \]

where the step length is chosen anew in each iteration.

This is often done using a “line search” algorithm.

**But:** step-15 does not do this – because line search is not easy to implement.
Practical considerations

**Question:** How accurate do we have to be?

**Observation:** In the first few Newton steps, we are still far away from the solution!

- We could compute Newton updates $\delta u_k$ on a coarse mesh
- We could solve the linear system inaccurately

In practice, this is exactly what is done.

**But:** step-15 only does the former.
**Question:** Do we really have to assemble the Jacobian matrix in each iteration?

**Observation:** In the first few Newton steps, we are still far away from the solution!

- We could compute Newton updates $\delta u_k$ inaccurately, by not updating the Jacobian matrix in each step
- We can also avoid updating the expensive preconditioner

In practice, this is exactly what is done.

**But:** step-15 only does not do it.
Overview: What we want is to...

- Use optimal step lengths via a line search
- Only update the Jacobian matrix/preconditioner when necessary
- Only solve linear systems inexactlly

Step-77 does the first two points!

Question: How do we achieve this? This is going to take a lot of code.
Overview: What we want is to...
• Use optimal step lengths via a line search
• Only update the Jacobian matrix/preconditioner when necessary
• Only solve linear systems inexactely

Question: How do we achieve this? This is going to take a lot of code.

Answer: Let’s not re-invent the wheel – other people have already done all this, and better than we ever could!
Specifically: Use the KINSOL solver of the SUNDIALS project: https://computing.llnl.gov/projects/sundials