Lecture 8: Homotopy continuation method

COMP5930M Scientific Computation

Today

Recap

Algorithms for the initial state

- Current strategies
- Homotopy continuation

Summary

Next

The Newton algorithm

- Initial state: $\mathbf{x} = \mathbf{x}_0$
- while $|\mathbf{F}(\mathbf{x}_k)| > Tol$
  - $\mathbf{J}(\mathbf{x}_k)\delta = -\mathbf{F}(\mathbf{x}_k)$
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \lambda\delta$

- This first step is critical to the success of the algorithm

Guaranteeing convergence?

So far we have seen:

- Use *domain-knowledge* to choose $\mathbf{x}_0$
- Line-search update to improve convergence
- Gradient descent algorithm can be used to get started, switch to Newton once close enough

Alternative options?

Homotopy continuation

A more formal approach to improving initial guess $\mathbf{x}_0$:

- Define a continuous transformation, from a known solved state, to our desired nonlinear model
- Mathematically, we use a continuation parameter t to define the transformation
- Solve a sequence of nonlinear problems with the previous solution used as initial data for the next

Formulation

- We want to find $\mathbf{x}^*$ such that $\mathbf{F}(\mathbf{x}^*) = \mathbf{0}$ without restrictions on its closeness to the initial guess $\mathbf{x}_0$
- Define a one-parameter family of related problems:

$$\mathbf{G}(t, \mathbf{x}) = t\mathbf{F}(\mathbf{x}) + (1 - t)(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)) = 0$$

for $t \in [0, 1]$

  - At $t = 0$, $\mathbf{G}(0, \mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)$ and $\mathbf{x} = \mathbf{x}_0$ is the solution
  - At $t = 1$, $\mathbf{G}(1, \mathbf{x}) = \mathbf{F}(\mathbf{x})$ and $\mathbf{x} = \mathbf{x}^*$ is the solution

Formulation

- We want to find $\mathbf{x}^*$ such that $\mathbf{F}(\mathbf{x}^*) = \mathbf{0}$ without restrictions on its closeness to the initial guess $\mathbf{x}_0$
- Define a one-parameter family of related problems:

$$\mathbf{G}(t, \mathbf{x}) = t\mathbf{F}(\mathbf{x}) + (1 - t)(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)) = 0$$

  for $t \in [0, 1]$
  - At $t = 0$, $\mathbf{G}(0, \mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)$ and $\mathbf{x} = \mathbf{x}_0$ is the solution
  - At $t = 1$, $\mathbf{G}(1, \mathbf{x}) = \mathbf{F}(\mathbf{x})$ and $\mathbf{x} = \mathbf{x}^*$ is the solution
- Continuous path $\mathbf{x}(t)$ of solutions of nonlinear systems, starting from $\mathbf{G}(0, \mathbf{x})$ with a known solution $\mathbf{x}_0$, to the real nonlinear system $\mathbf{G}(1, \mathbf{x}) = \mathbf{F}(\mathbf{x})$.

Formulation

- We want to find $\mathbf{x}^*$ such that $\mathbf{F}(\mathbf{x}^*) = \mathbf{0}$ without restrictions on its closeness to the initial guess $\mathbf{x}_0$
- Define a one-parameter family of related problems:

$$\mathbf{G}(t, \mathbf{x}) = t\mathbf{F}(\mathbf{x}) + (1 - t)(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)) = 0$$

  for $t \in [0, 1]$
  - At $t = 0$, $\mathbf{G}(0, \mathbf{x}) = \mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)$ and $\mathbf{x} = \mathbf{x}_0$ is the solution
  - At $t = 1$, $\mathbf{G}(1, \mathbf{x}) = \mathbf{F}(\mathbf{x})$ and $\mathbf{x} = \mathbf{x}^*$ is the solution
- Continuous path $\mathbf{x}(t)$ of solutions of nonlinear systems, starting from $\mathbf{G}(0, \mathbf{x})$ with a known solution $\mathbf{x}_0$, to the real nonlinear system $\mathbf{G}(1, \mathbf{x}) = \mathbf{F}(\mathbf{x})$.

Deriving the continuation algorithm

We now solve for $\mathbf{x}(t)$

$$\mathbf{G}(t, \mathbf{x}(t)) \quad = \quad \mathbf{0}$$

for $t \in [0, 1]$ with initial conditions $\mathbf{x}(0) = \mathbf{x}_0$

$\mathbf{G}(t, \mathbf{x}(t)) = 0$ at every point on the continuation path $\mathbf{x}(t)$

Therefore, the total derivative of $\mathbf{G}$ with respect to t is:

$$\frac{d\mathbf{G}}{dt} \quad = \quad \frac{\partial \mathbf{G}}{\partial t} + \frac{\partial \mathbf{G}}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dt} \quad = \quad \frac{\partial \mathbf{G}}{\partial t} + \sum_{j=1}^{n} \frac{\partial \mathbf{G}}{\partial x_j} \frac{dx_j}{dt} \quad = \quad \mathbf{0}$$

which defines a system of differential equations for $\mathbf{x}(t)$

Deriving the algorithm

$$\begin{aligned}\mathbf{G}(t,\mathbf{x}) &= t\mathbf{F}(\mathbf{x}) + (1-t)\left(\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{x}_0)\right) \\ &= \mathbf{F}(\mathbf{x}) + (t-1)\mathbf{F}(\mathbf{x}_0)\end{aligned}$$

- $\frac{\partial \mathbf{G}}{\partial \mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} = \mathbf{J}(\mathbf{x})$
- $\frac{\partial \mathbf{G}}{\partial t} = \mathbf{F}(\mathbf{x}_0)$

From our previous slide

$$\begin{aligned}\mathbf{F}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x})\frac{d\mathbf{x}}{dt} &= \mathbf{0} \\ \frac{d\mathbf{x}}{dt} &= -\mathbf{J}^{-1}(\mathbf{x})\mathbf{F}(\mathbf{x}_0)\end{aligned}$$

Pseudo-timestepping

- Define a step size Δt and apply a difference approximation:

$$\frac{\partial \mathbf{x}}{\partial t} \approx \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{\Delta t}$$

- Leads to the method: $\mathbf{x}_{k+1} = \mathbf{x}_k - \Delta t \mathbf{J}^{-1}(\mathbf{x}_k)\mathbf{F}(\mathbf{x}_0)$

- An equivalent two-step algorithm:

$$\begin{aligned} \mathbf{J}(\mathbf{x}_k)\delta &= -\mathbf{F}(\mathbf{x}_0) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \Delta t\, \delta \end{aligned}$$

Continuation method

- Initial state: $\mathbf{x} = \mathbf{x}_0$
- $k = 1$ *to* nt
  - $\mathbf{J}(\mathbf{x}_k)\delta = -\mathbf{F}(\mathbf{x_0})$
  - $\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t\, \delta$

- Note similarity to Newton but with fixed right-hand side
- No line search required!
- A fixed number of steps from $t = 0$ to $t = 1$
- $\mathbf{x}(1)$ approximates $\mathbf{x}^*$

Notes on continuation method

- Computational cost comparable to Newton's method
- We can start from a poor initial guess
- We assume the *path* taken is well-defined
  - ie. $G(t, \mathbf{x})$ differentiable
- There may be some dependence on the step size Δt, take more steps if necessary

Summary

Two formal algorithms presented that can improve overall convergence from an approximate initial state

Implementation is more involved and convergence rate may be lower so should only be used if necessary