Class 14: Newton for Systems (II) / Quasi Newton

Friday, September 27, 2024 1:20 PM

Class 14: September 27, 2024

Recall: Last time, we wrapped up our discussion on the Fixed Point Iteration for systems of non-linear equations and the corresponding rootfinding problems. We then generalized the ideas behind the Newton method to show how linear approximation of F(x) at x = xk can be used to generate a formula for the Newton step. This gives us the equation:

J_F(x_k) (x_{k+1} - x_k) = - F(x_k)

We then compute p_k = np.linalg.solve(J_F(x_k), -F(x_k)), and compute x_{k+1} = x_k + p_k.

$$\boxed{\vec{x}_{k+1} = \vec{x}_k - J_F^{-1}(\vec{x}_k)\, F(\vec{x}_k).}$$

$\boxed{\text{DO NOT COMPUTE } J_F^{-1}(x_k)\,!}$

MORE EXPENSIVE AND LESS STABLE.

- $\vec{p}_k$ = np.linalg.solve $(J_F(x_k), -F(x_k))$
- $\vec{x}_{k+1} = \vec{x}_k + \vec{p}_k$. .

Why do we like Newton

↳ Quadratic convergence.

↳ Solution in few # of iter.

(How fast do I get my answer?)

COST = (# Iters)(Cost p / Iter)

NEWTON	LOW	GROWING $O(N^3)$
OTHER	HIGH	CHEAP!

- EVALUATE $J_F(x_k)$. $\longrightarrow$ EXPENSIVE
- SOLVE $J_F(x_k) p_k = -F(x_k)$ $\longrightarrow$ EXPENSIVE

LIMITATIONS OF NEWTON (COST PER ITER)

OTHER SOURCE:

OUTSIDE OF BASIN OF QUAD CONVERGENCE
↳ No guarantees.

ADD SOMETHING TO ENSURE CONV. (GLOBALLY CONV).

- Hybrid method
- Line-search methods.

QUASI-NEWTON:

□ WANT:
Preserve - for $\|x_0 - r\| < \delta$, I have that $x_k \to r$ quadratically. (superlinear)

□ AVOID:

- Computing $J_F(x_k)$.
- Make the linear solve cheaper.

"Lazy Newton" (Chord Iteration)

$$x_{k+1} = x_k - J_F(x_0)^{-1} F(x_k).$$

(p_k = np.linalg.solve ($J_F(x_0)$, $-F(x_k)$))

✓ Chord <u>does</u> avoid computing J_F except once.

✓ Solving a lot of Linear Systems for the same matrix.

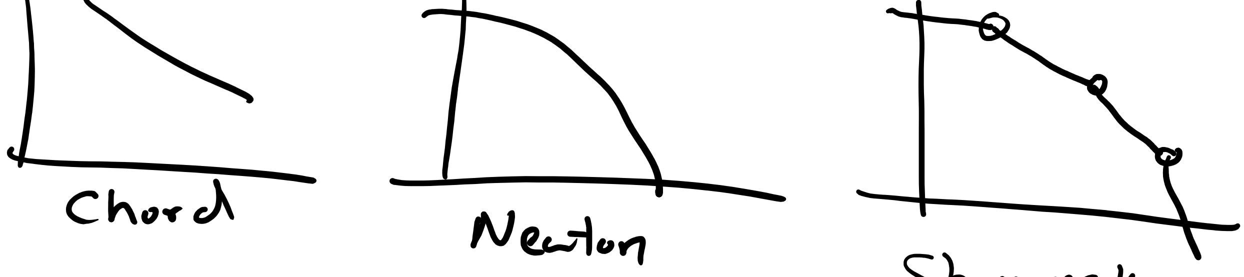
$$\underset{\text{LOWER TRIANG}}{J_F(x_0) = \mathcal{L}} \cdot \underset{\text{UPPER TRIANG.}}{\mathcal{U}} \longrightarrow \text{SOLVE } \underline{O(n^2)}$$

× If it converges, it does so <u>linearly</u>.

Between chord & Newton.

↳ Shaminskii method.

Update J_F every m steps.



Chord Newton Shaminskii

∘ Inexact Newton.

- Inexact Newton.
- "Secant" method (?)
  ↳ Quasi-Newton → Broyden

(Other methods — for optimization)

$$L_F(\vec{x}) = F(\vec{x}_0) + J_F(\vec{x}_0)(\vec{x} - \vec{x}_0)$$

$$L_s(\vec{x}) = F(x_0) + \underbrace{\mathbb{B}_0}_{\text{invertible.}}(\vec{x} - \vec{x}_0)$$