10.34, Numerical Methods Applied to Chemical Engineering
Professor William H. Green
Lecture #7: Introduction to Eigenvalues and Eigenvectors.

Newton's Method (Multi-dimensional)

$\underline{F}(\underline{x}^{true}) = \underline{0}$

Newton: Taylor expansion around x^{guess} If $\Delta\underline{x}$ is small. Works well when x^{guess} is close

$\underline{F}(\underline{x}^{guess}) + \underline{\underline{J}}(\underline{x}^{guess})\underline{\Delta x} \sim 0.0$ $\underline{x}^{true} \approx \underline{x}^{guess} + \Delta\underline{x}$

Select $\underline{x}^{guess}$; usually difficult to get a good guess

compute $\underline{F}(\underline{x}^{guess})$, $\underline{\underline{J}}(\underline{x}^{guess})$ $J_{mn} = \left.\dfrac{\partial F_m}{\partial x_n}\right|_{\underline{x}^{guess}}$

factorize $\underline{\underline{J}} \rightarrow \underline{\underline{L}}\ \underline{\underline{U}}$

solve $\underline{\underline{L}}\ \underline{\underline{U}}\ \underline{\Delta x} = -\underline{F}$ { backsub: $\underline{\underline{L}}\ \underline{V} = -\underline{F}$; $\underline{\underline{U}}\ \underline{\Delta x} = \underline{V}$

$\underline{x}^{new} = \underline{x}^{guess} + \Delta\underline{x}$

if $||\underline{x}^{new} - \underline{x}^{guess}|| < tolx$

if $||\underline{F}(\underline{x}^{new})|| < atolf$ { rtol doesn't work for $\underline{F}(\underline{x}) = 0$ } CONVERGENCE

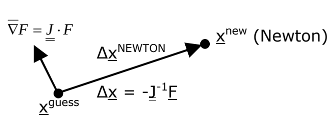
$\underline{x}^{guess} \leftarrow \underline{x}^{new}$

Iterate from compute $\underline{F}(\underline{x}^{guess})$

If $\underline{\underline{J}}$ is singular or poorly conditioned, will not be able to solve.
If $\Delta\underline{x}$ is big, method will not work.
In general, radius of convergence is small
- can bound $\Delta\underline{x}$ size
- can stop iteration after a certain number, for example, 20 iterations to see

Assumption of Newton's Method is $\underline{x}^{guess}$ is VERY GOOD
How close does $\underline{x}^{guess}$ have to be to guarantee convergence?
- radius of convergence

Backtrack Line Search



If you think $\underline{x}^{new}$ is too big, you can backtrack by looking at:
$||\underline{F}(\underline{x}^{guess})||$- $||\underline{F}(\underline{x}^{new})||$
$g(\lambda) = ||\underline{F}(\underline{x}^{guess})+\lambda\Delta\underline{x}^{NEWTON}||$
by minimizing $g(\lambda)$ using Bisection (etc.)

Figure 1. Trying to find x between x^{guess} and x^{new} that gives lower ||F||.

Maybe direction $\underline{F}(\underline{x}^{guess})$ to $\underline{F}(\underline{x}^{new})$ is wrong

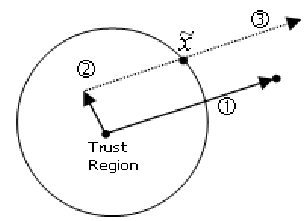
$\underline{f}(\underline{x}) = ||\underline{F}(\underline{x})||^2 = \Sigma|F_i(\underline{x})|^2$

Minimize scalar function: $\underline{\nabla f} = 2\sum \dfrac{\partial F_i}{\partial x_m} F_i = 2\underline{\underline{J}} \cdot \underline{F}$ works even when $\underline{\underline{J}}$ is singular

$\underline{\nabla f} = \underline{\underline{J}} \bullet \underline{f}$ is a move downhill

If x^{guess} is good, $\Delta x = -\underline{\underline{J}}^{-1}\underline{F}$ is the best direction but more risky (good if you can see the end)

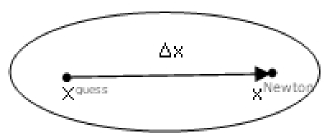
$\underline{\nabla} f$ is if "you are lost" → Brute force



1) Most risky method (Newton)
2) Safest method
3) Dogleg: compromise

Figure 2. The relationship of Newton's Method to Dogleg Method.

fsolve implements *Dogleg Method* using "Trust Region"



$\left.\begin{array}{l}\text{If } x^{Newton} \text{ is within the trust region,}\\ \text{the function will quickly converge}\end{array}\right\}$ Read 'doc fsolve'

Figure 3. If $\underline{F}(\tilde{x})$ is close to $\underline{F}(x_{guess})$, you can expand the trust region.

*fsolve has this all built in and is therefore much more powerful than simple Newton's method.

Optimization

$$\min_{\underline{x}} f(x)$$

$\underline{\nabla} f = 0$ is a bad way to do this (i.e. fsolve(gradf, x^{guess}))

The matrix is positive definite and

$$\frac{\partial^2 f}{\partial x_m \partial x_n} = \frac{\partial^2 f}{\partial x_n \partial x_m}.$$

Strategy: find regions where the problem can be considered optimization

f = $||\underline{F}||^2$ problem is there are local minimums

$\underline{\nabla} f = \underline{\underline{J}}\cdot\underline{F}$ can be zero if $\underline{\underline{J}}$ is singular and $\underline{F}$ is in "BAD DIRECTION"

$\underline{\underline{J}}$ singular → rank($\underline{\underline{J}}$) < N

$\underline{\underline{J}}\cdot\underline{v}^{bad} = \underline{0}$

$$\begin{pmatrix} ------ \\ ------ \\ ------ \end{pmatrix}(\vec{v}) = \begin{pmatrix} \overrightarrow{\text{row}}\,1\cdot\vec{v}^{\,\text{bad}} \\ \overrightarrow{\text{row}}\,2\cdot\vec{v}^{\,\text{bad}} \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

if $\underline{\nabla} f = 0$, no way of knowing which direction to go in.

$\underline{\underline{J}} \cdot \underline{v}^{bad} = 0 * \underline{v}^{bad}$

$\underline{\underline{J}} \cdot \underline{v}^{bad} = \lambda \underline{v}^{bad} \qquad \lambda = 0$

Poor conditioning

$\underline{\underline{A}} \cdot \underline{x} = \underline{b}$

$\underbrace{\underline{\underline{A}} \cdot \underline{v}^{bad} = \lambda \underline{v}^{bad}}_{\approx 0} \qquad \underline{\underline{A}}(\underline{x} + \underline{v}^{bad}) = \underbrace{\underline{\underline{A}} \cdot \underline{x}}_{\underline{b}} + \underbrace{\underline{\underline{A}} \cdot \underline{v}^{bad}}_{\delta \underline{b}}$

Certain linear combinations of values you can determine well. Other combinations you cannot determine.

$$\text{Sensitivity} \left\{ \begin{array}{l} \underline{\underline{A}} \cdot \underline{x} = \underline{b} \\ \underline{\underline{A}}^{-1} \rightarrow \underline{v}^{bad} \\ \underline{x} = \underline{\underline{A}}^{-1} \underline{b} \end{array} \right.$$

$$\left. \begin{array}{l} \underline{\underline{V}}^T = \underline{\underline{V}}^{-1} \\ \text{usually: } \underline{\underline{A}} = \underline{\underline{V}} \cdot \underline{\underline{\Lambda}} \cdot \underline{\underline{V}}^T \qquad \underline{\underline{A}} \cdot \underline{\underline{V}} = \underline{\underline{V}} \overset{(0) \quad \lambda 0}{\begin{pmatrix} \backslash & 0 & 0 \\ 0 & \backslash & 0 \\ 0 & 0 & \backslash \end{pmatrix}} \\ \qquad \qquad \underline{\underline{\Lambda}} = \begin{pmatrix} \backslash & 0 & 0 \\ 0 & \backslash & 0 \\ 0 & 0 & \backslash \end{pmatrix} \end{array} \right\} \text{Lecture 8 will discuss when you can do this factorization}$$

If you have large dimensional problem, it is difficult to give good x^{guess}
Look at $\underline{F}^{true}(\underline{x})$: can you change to a different problem?
$\underline{F}^{approx}(x^{guess})=0$ solvable (ideally, linear)

$F^{true} = F^{approx} + \lambda F^{perturb}$

You want to solve:
$\underline{F}^{true}(\underline{x}) = 0$

Is there an $\underline{F}^{approx}(\underline{x}) = 0$ that is soluble through linearization?

$\searrow \underline{x}^{guess}$

$\underline{F}^{true} = \underline{F}^{approx} + \lambda \underline{F}^{perturb}$

linear
or easy
$\underline{x}^{guess}$

$\underline{F}^{true} - \underline{F}^{approx}$

solve new problem with small λ:
$\underline{F}^{approx}(\underline{x}^{guess} = \underline{x}^{guess}{}_{approx}) \rightarrow \underline{x}^{guess,1}$
or linear
$\underline{F}^{approx} + 0.001\underline{F}^{perturb}(\underline{x}^{guess,1}) \rightarrow \underline{x}^{guess,2}$
$\underline{F}^{approx} + 0.01\underline{F}^{perturb}(\underline{x}^{guess,2}) \rightarrow \underline{x}^{guess,3}$
Increment λ until λ = 1

If the program crashes, need to step back and choose λ as a smaller value.