Unit 7.3 Nonlinear System Solutions

Numerical Analysis

EE/NTHU

May 3, 2017

Nonlinear Systems

- Newton's method is effective in solving nonlinear equations with only one variable.
- Need to have an effective method to solve nonlinear systems with more than one variable.
- Example. To find x and y that satisfy the following equations.

$$\begin{aligned} 2x^3 - y^2 - 1 &= 0 \\ xy^3 - y &= 0 \end{aligned} \tag{7.3.1}$$

- We can set $\mathbf{x} = [x_1, x_2]$ and

$$\begin{aligned} F_1(\mathbf{x}) &= 2x_1^3 - x_2^2 - 1 \\ F_2(\mathbf{x}) &= x_1 x_2^3 - x_2 \end{aligned} \tag{7.3.2}$$

- Then the problem becomes to find $\mathbf{x}^*$ such that

$$\mathbf{F}(\mathbf{x}^*) = \mathbf{0}. \tag{7.3.3}$$

- In this example, $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{F} : \mathbb{R}^2 \to \mathbb{R}^2$, and is a 2-dimensional nonlinear system problem.

Newton's Method in N-Dimension

- Consider one of the equation, $F_1(\mathbf{x})$, around the solution point, $\mathbf{x}^* = [x_1^*, x_2^*]$,

$$\begin{aligned} F_1(x_1^*, x_2^*) =& F_1(x_1, x_2) + (x_1^* - x_1)\frac{\partial F_1(x_1,x_2)}{\partial x_1} + (x_2^* - x_2)\frac{\partial F_1(x_1,x_2)}{\partial x_2} \\ &+ \frac{(x_1^* - x_1)^2}{2}\frac{\partial^2 F_1(\xi_1,\xi_2)}{\partial x_1^2} + \frac{(x_2^* - x_2)^2}{2}\frac{\partial^2 F_1(\xi_1,\xi_2)}{\partial x_2^2} \\ &+ \frac{(x_1^* - x_1)(x_2^* - x_2)}{2}\frac{\partial^2 F_1(\xi_1,\xi_2)}{\partial x_1 \partial x_2} \end{aligned} \tag{7.3.4}$$

  where (ξ_1, ξ_2) is in the neighborhood of (x_1^*, x_2^*).
- Suppose $|x_1^* - x_1| \ll 1$ and $|x_2^* - x_2| \ll 1$ then

$$F_1(x_1^*, x_2^*) \approx F_1(x_1, x_2) + (x_1^* - x_1)\frac{\partial F_1(x_1,x_2)}{\partial x_1} + (x_2^* - x_2)\frac{\partial F_1(x_1,x_2)}{\partial x_2} \tag{7.3.5}$$

- And we have

$$(x_1^* - x_1)\frac{\partial F_1(x_1,x_2)}{\partial x_1} + (x_2^* - x_2)\frac{\partial F_1(x_1,x_2)}{\partial x_2} = -F_1(x_1, x_2). \tag{7.3.6}$$

- By the same process, we also have

$$(x_1^* - x_1)\frac{\partial F_2(x_1,x_2)}{\partial x_1} + (x_2^* - x_2)\frac{\partial F_2(x_1,x_2)}{\partial x_2} = -F_2(x_1, x_2). \tag{7.3.7}$$

Newton's Method in N-Dimension, II

- Combining Eqs. (7.3.6) and (7.3.7), and arrange in matrix form

$$\begin{bmatrix} \frac{\partial F_1(x_1,x_2)}{\partial x_1} & \frac{\partial F_1(x_1,x_2)}{\partial x_2} \\ \frac{\partial F_2(x_1,x_2)}{\partial x_1} & \frac{\partial F_2(x_1,x_2)}{\partial x_2} \end{bmatrix} \begin{bmatrix} x_1^* - x_1 \\ x_2^* - x_2 \end{bmatrix} = \begin{bmatrix} -F_1(x_1,x_2) \\ -F_2(x_1,x_2) \end{bmatrix} \tag{7.3.8}$$

- Or

$$\begin{bmatrix} x_1^* - x_1 \\ x_2^* - x_2 \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1(x_1,x_2)}{\partial x_1} & \frac{\partial F_1(x_1,x_2)}{\partial x_2} \\ \frac{\partial F_2(x_1,x_2)}{\partial x_1} & \frac{\partial F_2(x_1,x_2)}{\partial x_2} \end{bmatrix}^{-1} \begin{bmatrix} -F_1(x_1,x_2) \\ -F_2(x_1,x_2) \end{bmatrix} \tag{7.3.9}$$

$$\begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} - \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -\begin{bmatrix} \frac{\partial F_1(x_1,x_2)}{\partial x_1} & \frac{\partial F_1(x_1,x_2)}{\partial x_2} \\ \frac{\partial F_2(x_1,x_2)}{\partial x_1} & \frac{\partial F_2(x_1,x_2)}{\partial x_2} \end{bmatrix}^{-1} \begin{bmatrix} F_1(x_1,x_2) \\ F_2(x_1,x_2) \end{bmatrix} \tag{7.3.10}$$

$$\begin{bmatrix} x_1^* \\ x_2^* \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} \frac{\partial F_1(x_1,x_2)}{\partial x_1} & \frac{\partial F_1(x_1,x_2)}{\partial x_2} \\ \frac{\partial F_2(x_1,x_2)}{\partial x_1} & \frac{\partial F_2(x_1,x_2)}{\partial x_2} \end{bmatrix}^{-1} \begin{bmatrix} F_1(x_1,x_2) \\ F_2(x_1,x_2) \end{bmatrix} \tag{7.3.11}$$

Newton's Method in N-Dimension, III

- The matrix is called Jacobian matrix and is defined as

$$\mathbf{J_F}(\mathbf{x}) = \left[\frac{\partial F_i}{\partial x_j}\right]. \tag{7.3.12}$$

It is an $n \times n$ matrix for n-dimensional problems.

- For the 2-dimensional problem above, we have

$$\mathbf{J_F}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial F_1(x_1,x_2)}{\partial x_1} & \dfrac{\partial F_1(x_1,x_2)}{\partial x_2} \\ \dfrac{\partial F_2(x_1,x_2)}{\partial x_1} & \dfrac{\partial F_2(x_1,x_2)}{\partial x_2} \end{bmatrix} \tag{7.3.13}$$

- Using Jacobian and matrix-vector notation, Eq. (7.3.11) can be rewritten as

$$\mathbf{x}^* = \mathbf{x} - \mathbf{J_F}^{-1}(\mathbf{x}) \cdot \mathbf{F}(\mathbf{x}). \tag{7.3.14}$$

- Thus, the n-dimensional Newton's iteration is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J_F}^{-1}(\mathbf{x}^{(k)}) \cdot \mathbf{F}(\mathbf{x}^{(k)}). \tag{7.3.15}$$

- The Newton's method in solving n-dimensional nonlinear system still have order 2 convergence property.

Newton's Method Example

- For the nonlinear system of Eq. (7.3.2) we have

$$F_1(\mathbf{x}) = 2x_1^3 - x_2^2 - 1 \qquad \partial F_1(\mathbf{x})/\partial x_1 = 6x_1^2$$
$$F_2(\mathbf{x}) = x_1 x_2^3 - x_2 \qquad \partial F_1(\mathbf{x})/\partial x_2 = -2x_2$$
$$\partial F_2(\mathbf{x})/\partial x_1 = x_2^3$$
$$\partial F_2(\mathbf{x})/\partial x_2 = 3x_1x_2^2 - 1$$

- Thus, the Jacobian matrix is

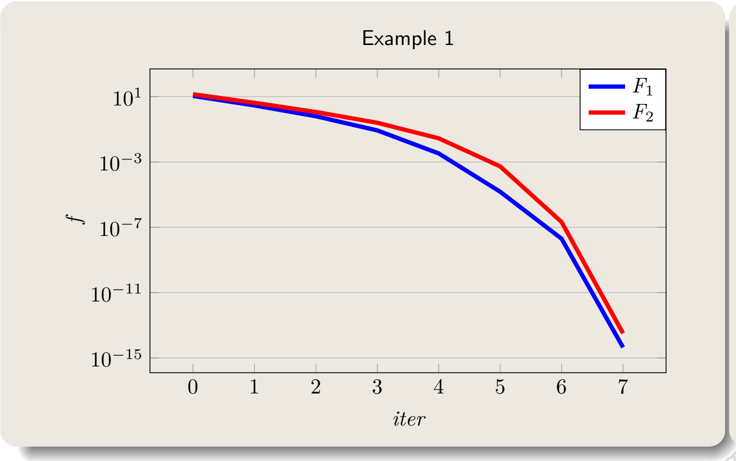
$$\mathbf{J_F}(\mathbf{x}) = \begin{bmatrix} 6x_1^2 & -2x_2 \\ x_2^3 & 3x_1x_2^2 - 1 \end{bmatrix}$$

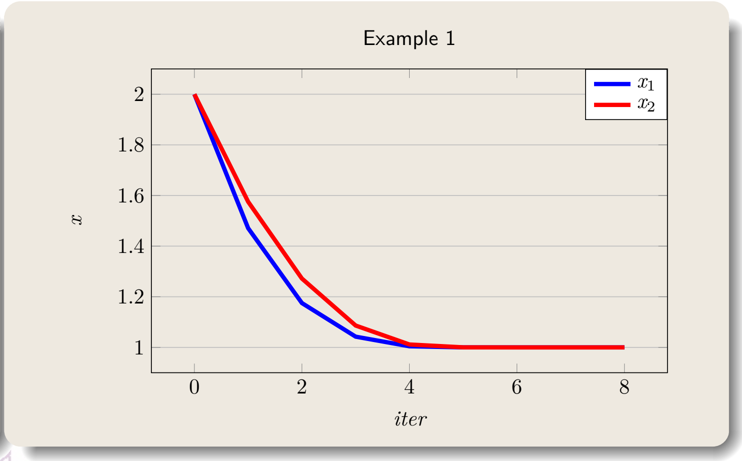
- Given an initial guess $\mathbf{x}^{(0)}$, the Newton's iteration to solve Eq. (7.3.1) is

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} x_1^{(k)} \\ x_2^{(k)} \end{bmatrix} - \begin{bmatrix} 6x_1^2 & -2x_2 \\ x_2^3 & 3x_1x_2^2 - 1 \end{bmatrix}^{-1} \begin{bmatrix} 2x_1^3 - x_2^2 - 1 \\ x_1x_2^3 - x_2 \end{bmatrix}$$

where the matrix inversion can be done using LU decomposition or any linear system solution method.

Newton's Method Example, II





- Newton's method is effective in solving nonlinear systems
- The initial guess, $\mathbf{x}^{(0)}$ is still an important issue that affects the convergence of the algorithm
  - Initial guess needs to be close to the solution $\mathbf{x}^*$
  - In some applications, other techniques to find an approximated solution is employed first before using Newton's method for accurate solution.
  - In this example, the initial guess is $\mathbf{x}^{(0)} = [2, 2]$.

Newton's Method in N-Dimension, IV

Theorem 7.3.1.

Let $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ be a C^1 function in a convex open set D of $\mathbb{R}^n$ that contains $\mathbf{x}^*$. Suppose that $\mathbf{J}_\mathbf{F}^{-1}(\mathbf{x}^*)$ exists and that there are positive constants R, C and L, such that $\|\mathbf{J}_\mathbf{F}^{-1}(\mathbf{x}^*)\| \leq C$ and

$$\|\mathbf{J}_\mathbf{F}(\mathbf{x}) - \mathbf{J}_\mathbf{F}(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\| \qquad \text{for all } \mathbf{x}, \mathbf{y} \in B(\mathbf{x}^*, R)$$

with the consistent vector and matrix norms, $\|\cdot\|$. Then, there is an $r > 0$ such that for any $\mathbf{x}^{(0)} \in B(\mathbf{x}^*, r)$, the Newton's iteration converges to $\mathbf{x}^*$ with

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| \leq CL\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2, \tag{7.3.16}$$

where $B(\mathbf{x}, r)$ is an open ball in $\mathbb{R}^n$ centered at $\mathbf{x}$ with radius r.

- If $\|\mathbf{J}_\mathbf{F}^{-1}(\mathbf{x}^*)\|$ and $\|\mathbf{J}_\mathbf{F}(\mathbf{x}^*)\|$ are bounded, then the Newton's method is convergent with order 2.

Newton's Method in N-Dimension, V

- Newton's iteration is

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(k)}) \cdot \mathbf{F}(\mathbf{x}^{(k)}). \tag{7.3.17}$$

- It can also be written as

$$\begin{aligned} \mathbf{J}_{\mathbf{F}}(\mathbf{x}^{(k)})\, \delta\mathbf{x}^{(k)} &= -\mathbf{F}(\mathbf{x}^{(k)}), \\ \mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \delta\mathbf{x}^{(k)}. \end{aligned} \tag{7.3.18}$$

- For each iteration
  - $\mathbf{F}(\mathbf{x}^{(k)})$ needs to be evaluated : n function evaluations
  - $\mathbf{J}_{\mathbf{F}}(\mathbf{x}^{(k)})$ needs to be computed : $n \times n$ function evaluations
    - Forming the linear system as Eq. (7.3.17) is sometimes known as linearizing the nonlinear system
    - Linearizing the nonlinear system is dominated by forming the Jacobian matrix.
  - $\mathbf{J}_{\mathbf{F}}^{-1}(\mathbf{x}^{(k)}) \cdot \mathbf{F}(\mathbf{x}^{(k)})$ needs to be computed
    - LU decomposition takes $\mathcal{O}(n^3)$ operations
    - Forward and backward substitutions takes $\mathcal{O}(n^2)$ operations
  - Overall computational complexity maybe dominated by LU decomposition $\mathcal{O}(n^3)$.
- Solving a large nonlinear system could be very time consuming.

Improving Nonlinear System Solution Time

- Solving nonlinear system could be slow
  - Linearizing the nonlinear system
    - Dominated by Jacobian matrix formation time
  - Linear system solution
    - Dominated by LU decomposition
- Some techniques have been developed to speed nonlinear system solution time
  - Cyclic update of Jacobian matrix
  - Difference approximations of the Jacobian matrix
  - Inexact solution of the linear system

Cyclic Updates of Jacobian Matrix

- Typical Newton's method

Algorithm 7.3.2. Newton's Method for N-Dimensional Problems.

Given $\mathbf{x}^{(0)}$ and a small $\epsilon \geq 0$, let

$k = 0$, $err^{(0)} = 1 + \epsilon$,

while $(err^{(k)} > \epsilon)$ {

evaluate $\mathbf{F}(\mathbf{x}^{(k)})$,

evaluate $\mathbf{J_F}(\mathbf{x}^{(k)})$,

solve $\mathbf{J_F}(\mathbf{x}^{(k)})\ \delta\mathbf{x} = -\mathbf{F}(\mathbf{x}^{(k)})$,

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}$,

$k = k + 1$,

$err^{(k)} = \|\mathbf{F}(\mathbf{x}^{(k)})\|$,

} .

- In the above algorithm, the iteration error is essentially the residue of the nonlinear system.
- It is also possible to use $\|\delta\mathbf{x}\|$ as the iteration error.
- Both methods have been adopted in real applications.

Cyclic Updates of Jacobian Matrix, II

- With cyclic update of Jacobian matrix

Algorithm 7.3.3. Cyclic Jacobian Updates

Given $\mathbf{x}^{(0)}$ and a small $\epsilon \geq 0$, let

$k = 0$, $err^{(0)} = 1 + \epsilon$,

while $(err^{(k)} > \epsilon)$ {

evaluate $\mathbf{F}(\mathbf{x}^{(k)})$,

if $(k\%p = 0)$ evaluate $\mathbf{J_F}(\mathbf{x}^{(\mathbf{k})})$,

solve $\mathbf{J_F}(\mathbf{x})\ \delta\mathbf{x} = -\mathbf{F}(\mathbf{x}^{(k)})$,

$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \delta\mathbf{x}$,

$k = k + 1$,

$err^{(k)} = \|\mathbf{F}(\mathbf{x}^{(k)})\|$,

} .

- The most expensive operation in linearizing the nonlinear system is not repeated at every iteration.
  - CPU time for each iteration is improved.
  - Convergent rate might be affected.
  - Careful selection of p can be critical.

Difference Approximation of the Jacobian Matrix

- In case that explicit Jacobian matrix is difficult to evaluate, then the Jacobian can approximate numerically

$$(\mathbf{J}_h^{(k)})_j = \frac{\mathbf{F}(\mathbf{x}^{(k)} + h_j^{(k)}\mathbf{e}_j) - \mathbf{F}(\mathbf{x}^{(k)})}{h_j^{(k)}}. \tag{7.3.19}$$

  where $\mathbf{e}_j$ is the j-th unit vector of the space $\mathbb{R}^n$ and $h_j^{(k)} > 0$ is a small increment at iteration k.
- $(\mathbf{J}_h^{(k)})_j$ is a column vector and for each j the subtraction and division need to be carried out n times, and the overall Jacobian still needs $n+1$ $\mathbf{F}(\mathbf{x})$ (vector functions) evaluations or $n \times (n+1)$ scalar function evaluations.
- The small increment h needs to be small for accurate partial derivative calculations.
  - But, if h is too small then computer round-off error might increase.
  - Selecting proper h is important.
  - For the first few iterations, Jacobian is known to be inaccurate and thus larger h can be used.
  - When $\mathbf{x}^{(k)} \to \mathbf{x}^*$, then more accurate Jacobian can improve convergence rate. Smaller h is preferred.

Inexact Solution of the Linear System

- In case that the LU decomposition time dominates the total solution time, faster but less accurate linear solution methods might be adopted.
- Iterative solution methods may need large number of iterations for accurate solution of the linear system.
- But the in the early phase of Newton's iteration, the solution needs not be very accurate.
- Each iteration of the linear iterative solution method improves the solution accuracy.
- Thus, one can perform linear iterative method for a fixed number of iterations or with a lower accuracy tolerance in the early phase of Newton's iterations.
- Note that the total number of function evaluations might be increased.
- Thus, this technique is valuable if linearizing the nonlinear system takes small portion of the CPU time.

Bairstow's Method

- Polynomial's quadratic factors can be found by Lin's quadratic method.
- But, Lin's method has linear convergence rate.
- Newton's method can be applied to speed up the convergence.
  - Bairstow's method.
- Recall that the n degree polynomial $P_n(x)$ is to be factorized as

$$\begin{aligned} P_n(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \\ &= (x^2 + px + q)(b_{n-2}x^{n-2} + b_{n-3}x^{n-3} + \cdots + b_1 x + b_0) + Rx + S. \end{aligned}$$

Where $b_{n-2}, b_{n-3}, \cdots, b_1, b_0, R, S$ have been shown to be

$$\begin{aligned} b_{n-2} &= a_n \\ b_{n-3} &= a_{n-1} - pb_{n-2} \\ b_{n-4} &= a_{n-2} - pb_{n-3} - qb_{n-2} \\ &\cdots \\ b_0 &= a_2 - pb_1 - qb_2 \\ R &= a_1 - pb_0 - qb_1 \\ S &= a_0 - qb_0 \end{aligned} \tag{7.3.20}$$

Bairstow's Method, II

- And we seek p and q such that

$$\begin{aligned} R &= 0 \\ S &= 0 \end{aligned}$$

Or

$$\begin{aligned} R &= a_1 - pb_0 - qb_1 = 0 \\ S &= a_0 - qb_0 = 0 \end{aligned}$$

- To apply Newton's method, we need to find $\frac{\partial R}{\partial p}, \frac{\partial R}{\partial q}, \frac{\partial S}{\partial p}, \frac{\partial S}{\partial q}$, to form the iterations.

$$\begin{bmatrix} p^{(k+1)} \\ q^{(k+1)} \end{bmatrix} = \begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} - \begin{bmatrix} \frac{\partial R}{\partial p} & \frac{\partial R}{\partial q} \\ \frac{\partial S}{\partial p} & \frac{\partial S}{\partial q} \end{bmatrix}^{-1} \begin{bmatrix} R(p^{(k)}, q^{(k)}) \\ S(p^{(k)}, q^{(k)}) \end{bmatrix} \tag{7.3.21}$$

Bairstow's Method, III

- From Eq. (7.3.20) we have

$$\frac{\partial b_{n-2}}{\partial p} = 0 \qquad \frac{\partial b_{n-2}}{\partial q} = 0$$

$$\frac{\partial b_{n-3}}{\partial p} = -b_{n-2} - p\frac{\partial b_{n-2}}{\partial p} \qquad \frac{\partial b_{n-3}}{\partial q} = -p\frac{\partial b_{n-2}}{\partial q}$$

$$\frac{\partial b_{n-4}}{\partial p} = -b_{n-3} - p\frac{\partial b_{n-3}}{\partial p} - q\frac{\partial b_{n-2}}{\partial p} \qquad \frac{\partial b_{n-4}}{\partial q} = -p\frac{\partial b_{n-3}}{\partial q} - b_{n-2} - q\frac{\partial b_{n-2}}{\partial q}$$

$$\cdots \qquad \cdots$$

$$\frac{\partial b_0}{\partial p} = -b_1 - p\frac{\partial b_1}{\partial p} - q\frac{\partial b_2}{\partial p} \qquad \frac{\partial b_0}{\partial q} = -p\frac{\partial b_1}{\partial q} - b_2 - q\frac{\partial b_2}{\partial q}$$

$$\frac{\partial R}{\partial p} = -b_0 - p\frac{\partial b_0}{\partial p} - q\frac{\partial b_1}{\partial p} \quad (7.3.22) \qquad \frac{\partial R}{\partial q} = -p\frac{\partial b_0}{\partial q} - b_1 - q\frac{\partial b_1}{\partial q} \quad (7.3.24)$$

$$\frac{\partial S}{\partial p} = -q\frac{\partial b_0}{\partial p} \quad (7.3.23) \qquad \frac{\partial S}{\partial q} = -b_0 - q\frac{\partial b_0}{\partial q} \quad (7.3.25)$$

Bairstow's Method, IV

- Let $c_j = \dfrac{\partial b_j}{\partial p}$ and $d_j = \dfrac{\partial b_j}{\partial q}$, then we have

$$c_{n-2} = 0 \qquad d_{n-2} = 0$$

$$c_{n-3} = -b_{n-2} - pc_{n-2} \qquad d_{n-3} = -pd_{n-2}$$

$$c_{n-4} = -b_{n-3} - pc_{n-3} - qc_{n-2} \qquad d_{n-4} = -pd_{n-3} - b_{n-2} - qd_{n-2}$$

$$\cdots \qquad \cdots$$

$$c_0 = -b_1 - pc_1 - qc_2 \qquad d_0 = -pd_1 - b_2 - qd_2$$

$$\frac{\partial R}{\partial p} = -b_0 - pc_0 - qc_1 \quad (7.3.26) \qquad \frac{\partial R}{\partial q} = -pd_0 - b_1 - qd_1 \quad (7.3.28)$$

$$\frac{\partial S}{\partial p} = -qc_0 \quad (7.3.27) \qquad \frac{\partial S}{\partial q} = -b_0 - qd_0 \quad (7.3.29)$$

- Thus, to find a quadratic factor $x^2 + px + q$ of an n degree polynomial, $P_n(x) = \sum_{k=0}^{n} a_k x^k$, we have the following algorithm

Bairstow's Method, V

Algorithm 7.3.4. Bairstow's Method

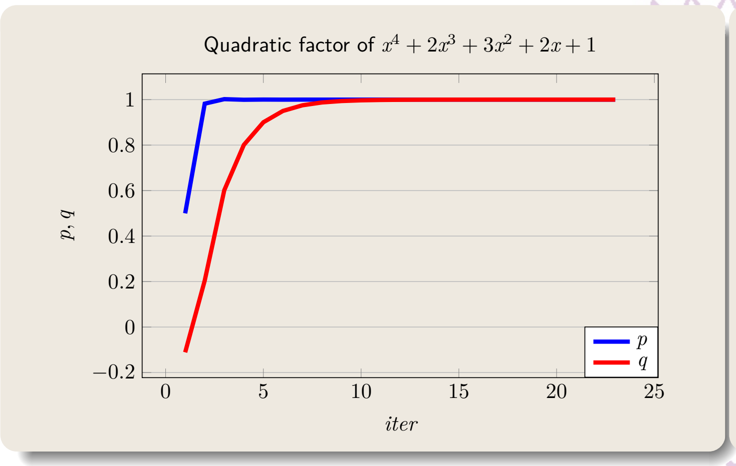
Given $p^{(0)}$, $q^{(0)}$, and integer *maxiter* and a small number ϵ,
let $err = 1 + \epsilon, k = 0$
while $(err \geq \epsilon)$ {

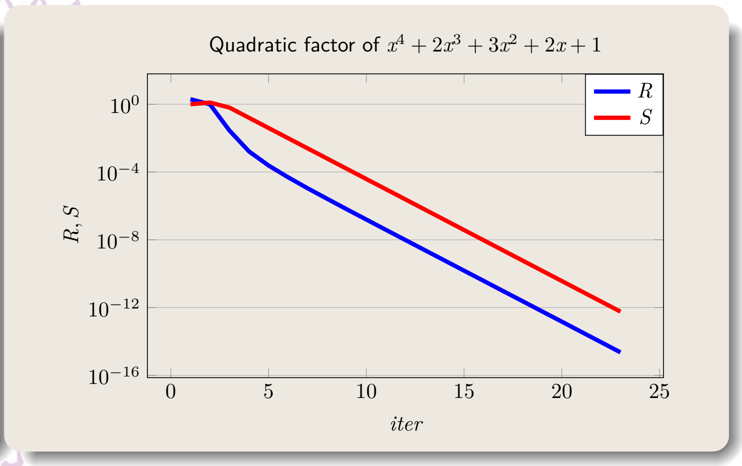
$$
\begin{aligned}
& b_{n-2} = a_n, \qquad b_{n-3} = a_{n-1} - p^{(k)} b_{n-2}, \\
& b_j = a_{j+2} - p^{(k)} b_{j+1} - q^{(k)} b_{j+2}, \qquad j = n-4, \ldots, 0, \\
& R = a_1 - p^{(k)} b_0 - q^{(k)} b_1, \qquad S = a_0 - q^{(k)} b_0, \\
& c_{n-2} = 0, \qquad c_{n-3} = -b_{n-2} - p^{(k)} c_{n-2}, \\
& c_j = -b_{j+1} - p^{(k)} c_{j+1} - q^{(k)} c_{j+2}, \qquad j = n-4, \ldots, 0, \\
& \frac{\partial R}{\partial p} = -b_0 - p^{(k)} c_0 - q^{(k)} c_1, \qquad \frac{\partial S}{\partial p} = -q^{(k)} c_0, \\
& d_{n-2} = 0, \qquad d_{n-3} = -p^{(k)} d_{n-2}, \\
& d_j = -p^{(k)} d_{j+1} - b_{j+2} - q^{(k)} d_{j+2}, \qquad j = n-4, \ldots, 0, \\
& \frac{\partial R}{\partial q} = -p^{(k)} d_0 - b_1 - q^{(k)} d_1, \qquad \frac{\partial S}{\partial q} = -b_0 - q^{(k)} d_0, \\
& \begin{bmatrix} p^{(k+1)} \\ q^{(k+1)} \end{bmatrix} = \begin{bmatrix} p^{(k)} \\ q^{(k)} \end{bmatrix} - \begin{bmatrix} \partial R/\partial p & \partial R/\partial q \\ \partial S/\partial p & \partial S/\partial q \end{bmatrix}^{-1} \begin{bmatrix} R \\ S \end{bmatrix}, \\
& k = k+1, \qquad err = \max(|R|, |S|),
\end{aligned}
$$

}

Bairstow's Method, VI

- Example: find the quadratic factor, $x^2 + px + q$ of $P_4 = x^4 + 2x^3 + 3x^2 + 2x + 1$ with initial guess $p^{(0)} = 0, q^{(0)} = 0$.



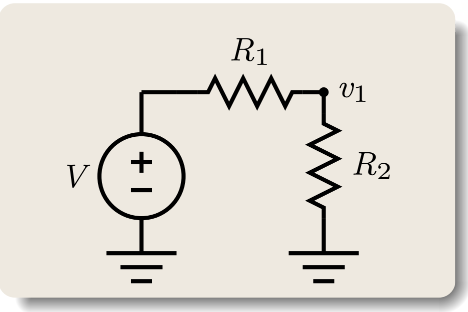


- Bairstow's method uses Newton's iteration to find the quadratic factor,
- Order of convergence is 2.
- It is still a local convergent algorithm, but with much larger convergence window than Lin's quadratic method.
- Some formulas in reference books have very small convergence window.
- Our method is much more robust to find the quadratic factors.

Application of Newton's Method

- Newton's method, also known as Newton-Raphson method, to solve nonlinear system of equations, is very popular
- Examples:
  - Circuit simulators: SPICE, etc
  - Device simulators: Medici, etc
- In most applications, analytical derivatives are usually employed for the best convergence rate
  - When analytical derivative is not available, the difference scheme is adopted
- Newton's method converges quickly if the initial guess is close to the solution.
  - In SPICE, special algorithms have been developed to find good initial guess before employing Newton's algorithm for accurate solutions.
- You are actually capable of solving OP, DC and AC analysis in SPICE already

Resistor Network Example



- Resistor voltage divider

$$R_i = R_{i0} + \kappa_i T_i, \qquad i = 1, 2. \tag{7.3.30}$$

where R_{i0} is the resistance at room temp, and T_i is the temperature increase when the resistor is consuming power. Thus,

$$T_i = \beta_i I_i V_i = \frac{\beta_i V_i^2}{R_i}, \qquad i = 1, 2. \tag{7.3.31}$$

And the current through the resistor is

$$I_i = \frac{V_i}{R_i} = \frac{V_i}{R_{i0} + \kappa_i T_i}, \qquad i = 1, 2. \tag{7.3.32}$$

- Current continuity at node v_1

$$\frac{v_1 - V}{R_{10} + \kappa_1 T_1} + \frac{v_1}{R_{20} + \kappa_2 T_2} = 0, \tag{7.3.33}$$

Resistor Network Example, II

- And two resistor temperature increase

$$T_1 - \frac{\beta_1 (v_1 - V)^2}{R_{10} + \kappa_1 T_1} = 0, \tag{7.3.34}$$

$$T_2 - \frac{\beta_2 v_1^2}{R_{20} + \kappa_2 T_2} = 0. \tag{7.3.35}$$

- Thus, the system has three variables, v_1, T_1 and T_2 with three equations

$$F_1(v_1, T_1, T_2) = \frac{v_1 - V}{R_{10} + \kappa_1 T_1} + \frac{v_1}{R_{20} + \kappa_2 T_2} = 0, \tag{7.3.36}$$

$$F_2(v_1, T_1, T_2) = T_1 - \frac{\beta_1 (v_1 - V)^2}{R_{10} + \kappa_1 T_1} = 0, \tag{7.3.37}$$

$$F_3(V_1, T_1, T_2) = T_2 - \frac{\beta_2 v_1^2}{R_{20} + \kappa_2 T_2} = 0. \tag{7.3.38}$$

Resistor Network Example, III

- To find the Jacobian, we have

$$\frac{\partial F_1}{\partial v_1} = \frac{1}{R_{10} + \kappa_1 T_1} + \frac{1}{R_{20} + \kappa_2 T_2}, \qquad \frac{\partial F_2}{\partial T_2} = 0,$$

$$\frac{\partial F_1}{\partial T_1} = -\frac{\kappa_1 (v_1 - V)}{(R_{10} + \kappa_1 T_1)^2}, \qquad \frac{\partial F_3}{\partial v_1} = -\frac{2\beta_2 v_1}{R_{20} + \kappa_2 T_2},$$

$$\frac{\partial F_1}{\partial T_2} = -\frac{\kappa_2 v_1}{(R_{20} + \kappa_2 T_2)^2}, \qquad \frac{\partial F_3}{\partial T_1} = 0,$$

$$\frac{\partial F_2}{\partial v_1} = -\frac{2\beta_1 (v_1 - V)}{R_{10} + \kappa_1 T_1}, \qquad \frac{\partial F_3}{\partial T_1} = 1 + \frac{\kappa_2 \beta_2 v_1^2}{(R_{20} + \kappa_2 T_2)^2}.$$

$$\frac{\partial F_2}{\partial T_1} = 1 + \frac{\kappa_1 \beta_1 (v_1 - V)^2}{(R_{10} + \kappa_1 T_1)^2},$$

- And, the Newton's iteration is

$$\begin{bmatrix} v_1^{(k+1)} \\ T_1^{(k+1)} \\ T_2^{(k+1)} \end{bmatrix} = \begin{bmatrix} v_1^{(k)} \\ T_1^{(k)} \\ T_2^{(k)} \end{bmatrix} - \begin{bmatrix} \partial F_1/\partial v_1 & \partial F_1/\partial T_1 & \partial F_1/\partial T_2 \\ \partial F_2/\partial v_1 & \partial F_2/\partial T_1 & \partial F_2/\partial T_2 \\ \partial F_3/\partial v_1 & \partial F_3/\partial T_1 & \partial F_3/\partial T_2 \end{bmatrix}^{-1} \begin{bmatrix} F_1(v_1^{(k)}, T_1^{(k)}, T_2^{(k)}) \\ F_2(v_1^{(k)}, T_1^{(k)}, T_2^{(k)}) \\ F_3(v_1^{(k)}, T_1^{(k)}, T_2^{(k)}) \end{bmatrix}.$$

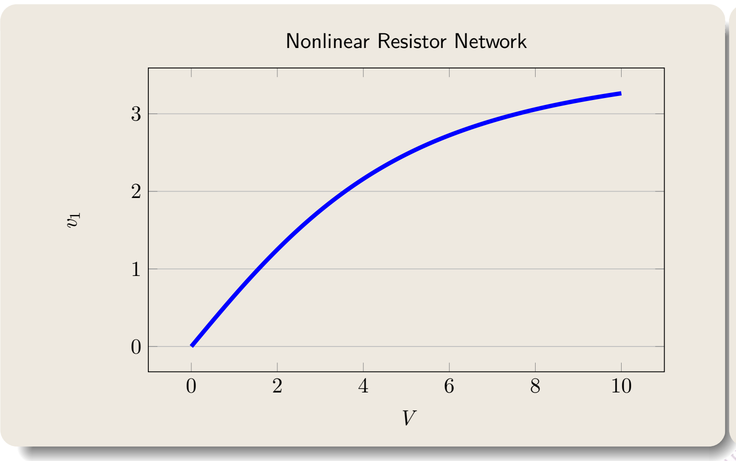
Resistor Network Example, IV

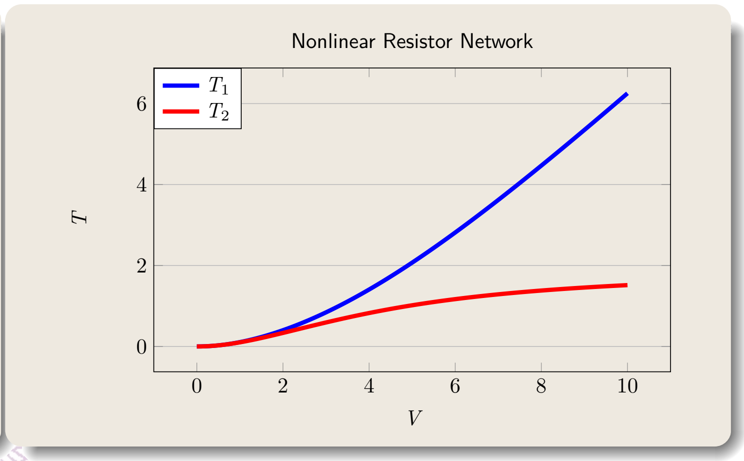
- Given any V, node voltage v_1 and the temperature increases of the two resistors can be found.
- If we have

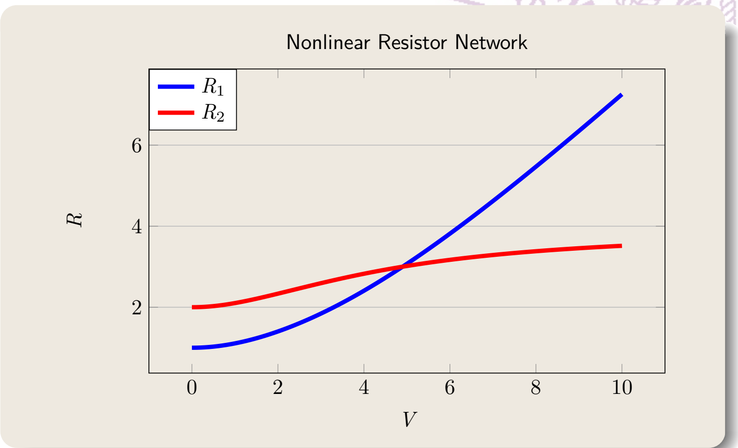
$$\begin{aligned} R_{10} &= 1, & \kappa_1 &= 1, & \beta_1 &= 1, \\ R_{20} &= 2, & \kappa_2 &= 1, & \beta_2 &= 0.5. \end{aligned}$$

- Then the voltage divider circuit can be solved given the supply voltage.
- One can increase the voltage supply and solve the circuit
  - SPICE DC analysis.

Resistor Network Example, IV







- Voltage v_1 is a nonlinear function of supply voltage V
- Newton's method is effective in solving nonlinear systems
- In solving the circuit at a new V, the converged solution at the previous step should be used as the initial guess.

Resistor Network Example, V

- The resistor network system can also be formulated as

$$
\begin{aligned}
v_0 &= V, \\
\frac{v_1 - v_0}{R_1} + \frac{v_1 - v_2}{R_2} &= 0, \\
v_2 &= 0, \\
T_1 - \frac{\beta_1 (v_1 - v_0)^2}{R_1} &= 0, \\
T_2 - \frac{\beta_2 (v_1 - v_2)^2}{R_2} &= 0,
\end{aligned}
$$

with the system unknowns $\mathbf{x} = (v_0, v_1, v_2, T_1, T_2)$, totally 5 variables. And Newton's iteration becomes

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (\mathbf{J}^{(k)})^{-1}\mathbf{F}$$

where $\mathbf{F}$ are the five equations defined above and $\mathbf{J}_{i,j} = \dfrac{\partial F_i}{\partial x_j}$.

Resistor Network Example, VI

- $\mathbf{J}$ can be derived as

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
-\frac{1}{R_1} & \frac{1}{R_1} + \frac{1}{R_2} & -\frac{1}{R_2} & \frac{v_1 - v_0}{R_1^2} \frac{\partial R_1}{\partial T_1} & \frac{v_1 - v_2}{R_2^2} \frac{\partial R_1}{\partial T_2} \\
0 & 0 & 1 & 0 & 0 \\
\frac{2\beta_1 (v_1 - v_0)}{R_1} & -\frac{2\beta_1 (v_1 - v_0)}{R_1} & 0 & 1 + \frac{\beta_1 (v_1 - v_0)^2}{R_1^2} \frac{\partial R_1}{\partial T_1} & 0 \\
0 & -\frac{2\beta_2 (v_1 - v_2)}{R_2} & \frac{2\beta_2 (v_1 - v_2)}{R_2} & 0 & 1 + \frac{\beta_1 (v_1 - v_0)^2}{R_1^2} \frac{\partial R_2}{\partial T_2}
\end{bmatrix}
$$

- And the right-hand side, $-\mathbf{F}$ is

$$
\begin{bmatrix} -v_0 + V & -\frac{v_1 - v_0}{R_1} - \frac{v_1 - v_2}{R_2} & 0 & -T_1 + \frac{\beta_1 (v_1 - v_0)^2}{R_1} & -T_2 + \frac{\beta_2 (v_1 - v_2)^2}{R_2} \end{bmatrix}^T
$$

Resistor Network Example, VII

- Assuming a nonlinear resistor, R_k, is connecting nodes i and j, then it contributes to the following stamps

$$\mathbf{J}_{ii} + = \frac{1}{R_k}, \qquad \mathbf{J}_{ik} = \frac{v_i - v_j}{R_k^2}\frac{\partial R_k}{\partial T_k},$$

$$\mathbf{J}_{ij} - = \frac{1}{R_k}, \qquad \mathbf{J}_{ki} = -\frac{2\beta_k(v_i - v_j)}{R_k},$$

$$\mathbf{J}_{ji} - = \frac{1}{R_k}, \qquad \mathbf{J}_{kj} = \frac{2\beta_k(v_i - v_j)}{R_k},$$

$$\mathbf{J}_{jj} + = \frac{1}{R_k}, \qquad \mathbf{J}_{kk} = 1 + \frac{\beta_k(v_i - v_j)^2}{R_k^2}\frac{\partial R_k}{\partial T_k},$$

- The left-hand stamps are identical to the linear resistor case, while the right-hand stamps are newly added.
- If the network has n nodes and m resistors, then it has totally $n+m$ unknowns.
- Using the stamping approach, the nonlinear system can be solved.
- General nonlinear network can be formulated similarly.

Summary

- Nonlinear systems
- Newton's method in N-dimension
- Improving nonlinear system solution time
  - Cyclic update of Jacobian matrix
  - Difference approximation of Jacobian matrix
  - Inexact solution of the linear system
- Finding a quadratic factor of n degree polynomial
  - Bairstow's method
- Application of Newton's method
- Example: nonlinear resistor network problem
  - SPICE DC analysis