Roots of polynomials: Bairstow Method

- Find a quadratic factor of the polynomial *f(x) as* $x^2 - \alpha_1 x - \alpha_0$
- Find the two roots (real or complex conjugates) as

$$r_{1,2} = 0.5\left(\alpha_1 \pm \sqrt{\alpha_1^2 + 4\alpha_0}\right)$$

- Algorithm: Express the given function as $f(x) = \sum_{j=0}^{n} c_j x^j$

- Perform a synthetic division by the quadratic factor

$$x^2 - \alpha_1 x - \alpha_0 \overline{\big)\, c_n x^n + c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + c_{n-3}x^{n-3} + \ldots + c_1 x + c_0}\left(c_n x^{n-2} + \left(c_{n\text{-}1} + \alpha_1 c_n\right)x^{n-3} + ..\right.$$

$$c_n x^n - \alpha_1 c_n x^{n-1} - \alpha_0 c_n x^{n-2}$$

$$\overline{\left(c_{n\text{-}1} + \alpha_1 c_n\right)x^{n-1} + \left(c_{n\text{-}2} + \alpha_0 c_n\right)x^{n-2} + c_{n-3}x^{n-3}}$$

$$\underline{\left(c_{n\text{-}1} + \alpha_1 c_n\right)x^{n-1} + \alpha_1\left(c_{n\text{-}1} + \alpha_1 c_n\right)x^{n-2} + \alpha_0\left(c_{n\text{-}1} + \alpha_1 c_n\right)x^{n-3}}$$

Bairstow Method: Algorithm

- For writing a recursive algorithm, we express the quotient as a polynomial of degree $n-2$ and the remainder as linear

$$f(x) = \sum_{j=0}^{n} c_j x^j = \left(x^2 - \alpha_1 x - \alpha_0\right) \sum_{j=0}^{n-2} \left(d_{j+2} x^j\right) + d_1\left(x - \alpha_1\right) + d_0$$

- Equating the coefficients of different powers of x, we get

$$\begin{aligned} d_n &= c_n \\ d_{n-1} &= c_{n-1} + \alpha_1 d_n \\ d_j &= c_j + \alpha_1 d_{j+1} + \alpha_0 d_{j+2} \qquad \text{for } j = n-2 \text{ to } 0 \end{aligned}$$

- The target is to choose α_0 and α_1 in such a way that d_0 and d_1 become zero
- Iterative solution using Newton method

Bairstow Method: Algorithm

- d_0 and d_1 are functions of α_0 and α_1. The values for α_0 and α_1 at the (i+1)th iteration are obtained from

$$\alpha_0^{(i+1)} = \alpha_0^{(i)} + \Delta\alpha_0^{(i)};\ \alpha_1^{(i+1)} = \alpha_1^{(i)} + \Delta\alpha_1^{(i)}$$

- And then choosing the increments to make the residual zero

$$d_0^{(i+1)} = 0 = d_0^{(i)} + \left[\frac{\partial d_0}{\partial \alpha_0}\Delta\alpha_0\right]^{(i)} + \left[\frac{\partial d_0}{\partial \alpha_1}\Delta\alpha_1\right]^{(i)}$$

$$d_1^{(i+1)} = 0 = d_1^{(i)} + \left[\frac{\partial d_1}{\partial \alpha_0}\Delta\alpha_0\right]^{(i)} + \left[\frac{\partial d_1}{\partial \alpha_1}\Delta\alpha_1\right]^{(i)}$$

Bairstow Method: Algorithm

- The partial derivatives could be written as

$$\frac{\partial d_n}{\partial \alpha_0} = 0$$

$$\frac{\partial d_{n-1}}{\partial \alpha_0} = 0$$

$$\frac{\partial d_j}{\partial \alpha_0} = d_{j+2} + \alpha_0 \frac{\partial d_{j+2}}{\partial \alpha_0} + \alpha_1 \frac{\partial d_{j+1}}{\partial \alpha_0} \qquad \text{for } j = n-2 \text{ to } 0$$

and

$$\frac{\partial d_n}{\partial \alpha_1} = 0$$

$$\frac{\partial d_{n-1}}{\partial \alpha_1} = d_n$$

$$\frac{\partial d_j}{\partial \alpha_1} = d_{j+1} + \alpha_0 \frac{\partial d_{j+2}}{\partial \alpha_1} + \alpha_1 \frac{\partial d_{j+1}}{\partial \alpha_1} \qquad \text{for } j = n-2 \text{ to } 0$$

Bairstow Method: Algorithm

- These may be combined in a single recursive equation by defining

$$\delta_j = \frac{\partial d_{j-1}}{\partial \alpha_0} = \frac{\partial d_j}{\partial \alpha_1}$$

  to obtain

$$\delta_{n-1} = d_n$$

$$\delta_{n-2} = d_{n-1} + \alpha_1 \delta_{n-1}$$

$$\delta_j = d_{j+1} + \alpha_1 \delta_{j+1} + \alpha_0 \delta_{j+2} \qquad \text{for } j = n-3 \text{ to } 0$$

- The new estimates of α_0 and α_1 are obtained by solving

$$\delta_1^{(i)} \Delta\alpha_0^{(i)} + \delta_0^{(i)} \Delta\alpha_1^{(i)} = -d_0^{(i)}$$

$$\delta_2^{(i)} \Delta\alpha_0^{(i)} + \delta_1^{(i)} \Delta\alpha_1^{(i)} = -d_1^{(i)}$$

- Repeat till convergence.

Bairstow Method: Example

$$d_n = c_n$$
$$d_{n-1} = c_{n-1} + \alpha_1 d_n$$
$$d_j = c_j + \alpha_1 d_{j+1} + \alpha_0 d_{j+2};\ j = n-2 \text{ to } 0$$

$$\delta_{n-1} = d_n$$
$$\delta_{n-2} = d_{n-1} + \alpha_1 \delta_{n-1}$$
$$\delta_j = d_{j+1} + \alpha_1 \delta_{j+1} + \alpha_0 \delta_{j+2};\ j = n-3 \text{ to } 0$$

$$\delta_1^{(i)} \Delta\alpha_0^{(i)} + \delta_0^{(i)} \Delta\alpha_1^{(i)} = -d_0^{(i)};\ \delta_2^{(i)} \Delta\alpha_0^{(i)} + \delta_1^{(i)} \Delta\alpha_1^{(i)} = -d_1^{(i)}$$

Solve: $x^5 - 5.05x^4 + 12.2x^3 - 16.48x^2 + 12.5644x - 4.28442 = 0$

j	Iteration 1 α_0= -1 α_1=1		Iteration 2 α_0= -1.174 α_1=1.467		Iteration 3 α_0= -1.582 α_1=1.936		Iteration 4 α_0= -1.986 α_1=2.196		Iteration 5 α_0= -2.018 α_1=2.198		Iteration 6 α_0= -2.02 α_1=2.2	
	d_j	δ_j	d_j	δ_j	d_j	δ_j	d_j	δ_j	d_j	δ_j	d_j	δ_j
0	1.130	-2.096	0.484	-0.283	0.204	0.154	0.010	-0.363	0.001	-0.475	0.000	
1	0.134	0.870	0.202	0.864	0.138	0.603	0.016	0.294	0.001	0.206	0.000	
2	-5.280	3.100	-3.809	1.491	-2.669	0.728	-2.145	0.516	-2.123	0.460	-2.121	
3	7.150	-3.050	5.770	-2.116	4.589	-1.177	3.947	-0.658	3.913	-0.653	3.910	
4	-4.050	1.000	-3.583	1.000	-3.114	1.000	-2.854	1.000	-2.852	1.000	-2.850	
5	1.000		1.000		1.000		1.000		1.000		1.000	
	$\Delta\alpha_0$=-0.174 $\Delta\alpha_1$=0.467		$\Delta\alpha_0$=-0.407 $\Delta\alpha_1$=0.470		$\Delta\alpha_0$=-0.404 $\Delta\alpha_1$=0.260		$\Delta\alpha_0$=-0.032 $\Delta\alpha_1$=0.002		$\Delta\alpha_0$=-0.002 $\Delta\alpha_1$=0.002			

The two roots are: 1.1±0.9 *i*

$$r_{1,2} = 0.5\left(\alpha_1 \pm \sqrt{\alpha_1^2 + 4\alpha_0}\right)$$

System of Linear Equations: Introduction

- Example:
- N machines make N types of strings, requiring t_{ms} time on machine m to produce unit length of string s
- If total times on each machine, T_i (i=1,2,..N) are given, find length of each type of string, l_j (j=1,2,…N)

$$\begin{bmatrix} t_{11} & t_{12} & \dots & t_{1N} \\ t_{21} & t_{22} & \dots & t_{2N} \\ . & . & . & . \\ t_{N1} & t_{N2} & \dots & t_{NN} \end{bmatrix} \begin{Bmatrix} l_1 \\ l_2 \\ . \\ l_N \end{Bmatrix} = \begin{Bmatrix} T_1 \\ T_2 \\ . \\ T_N \end{Bmatrix}$$

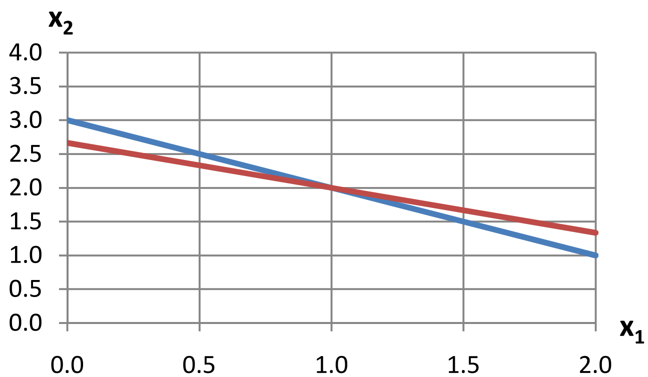
$$\text{Commonly used notation : } \quad [A]\{x\} = \{b\}$$

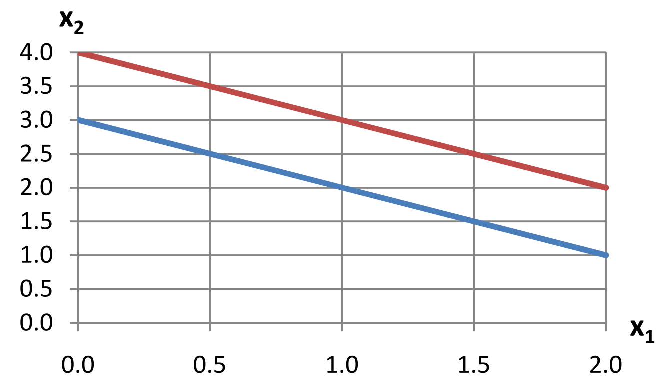
System of Linear Equations: Introduction

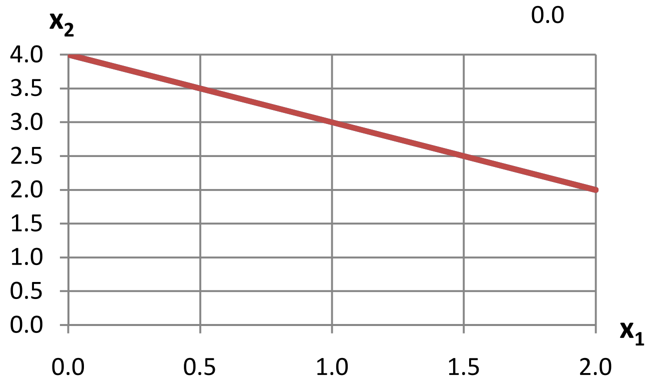
- Is a solution possible?
- Take a 2-dimensional example

$$a_{11}x_1 + a_{12}x_2 = b_1$$
$$a_{21}x_1 + a_{22}x_2 = b_2$$

- Solution exists for a_{11}=1, a_{12}=1, a_{21}=2,a_{22}=3,b_1=3,b_2=8
- No solution for a_{11}=1, a_{12}=1, a_{21}=2,a_{22}=2,b_1=3,b_2=8
- Infinite solutions for a_{11}=1, a_{12}=1, a_{21}=2,a_{22}=2,b_1=4,b_2=8







System of Linear Equations: Introduction

- We assume that a solution exists, and [A] is an n x n non-singular matrix
- How sensitive is the solution to small changes in [A] and/or {b}? (idea of a condition number)
- Small change in {b} : Already discussed Vector Norm
- Small change in [A] : A Matrix Norm is needed

Vector Norm	**Matrix Norm**
$\|x\| \geq 0$ (0 only for null vector)	$\|A\| \geq 0$ (0 only for null matrix)
$\|\alpha x\| = \|\alpha\| \|x\|$	$\|\alpha A\| = \|\alpha\| \|A\|$
$\|x_1 + x_2\| \leq \|x_1\| + \|x_2\|$	$\|A + B\| \leq \|A\| + \|B\|$
	$\|AB\| \leq \|A\| \|B\|$
For consistent (compatible) norm	$\|Ax\| \leq \|A\| \|x\|$

Matrix Norm

- Earlier we had defined L_p norm of a vector

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + |x_3|^p + \ldots + |x_n|^p\right)^{1/p} \qquad p \geq 1$$

which could be Euclidean distance (p=2), total distance (p=1), largest "axis distance" (p=∞) etc.

- For a matrix, we could define a norm based on what happens when the matrix is multiplied with a unit vector (known as the ***induced or subordinate norm***). Some other norms are element-wise norms, e.g. square-root of sum of squares of all elements (Frobenius norm).
- The norm will be large if the multiplication leads to a large magnitude vector.
- We define the p-norm of a matrix as

$$\|A\|_p = \max_{\|x\|_p = 1} \|Ax\|_p = \max_{\|x\|_p \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

Matrix Norm – The *1*-norm

- The *1*-norm of the matrix is written as $\|A\|_1 = \max_{\|x\|_1 = 1} \|Ax\|_1$
- We could write

$$Ax = x_1 \begin{Bmatrix} a_{11} \\ a_{21} \\ . \\ . \\ a_{n1} \end{Bmatrix} + x_2 \begin{Bmatrix} a_{12} \\ a_{22} \\ . \\ . \\ a_{n2} \end{Bmatrix} + \ldots + x_n \begin{Bmatrix} a_{1n} \\ a_{2n} \\ . \\ . \\ a_{nn} \end{Bmatrix}$$

- For $\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n| = 1$, what is the maximum L_1 norm of *Ax*?
- If we think of it as weighted sum of column-L_1 norms, maximum will occur when |x| corresponding to the column with maximum L_1 norm is 1 and all others 0
- Also known as the column-sum norm $\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}|$

Matrix Norm – The ∞-norm

- The ∞-norm of the matrix is written as $\|A\|_\infty = \max_{\|x\|_\infty = 1} \|Ax\|_\infty$
- Write

$$Ax = \begin{Bmatrix} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \\ . \\ . \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n \end{Bmatrix}$$

- For $\|x\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|) = 1$, what is maximum L_∞-norm of *Ax*?
- This will occur when all x are 1 with appropriate sign such that the row-sum is of maximum magnitude
- Also known as the row-sum norm $\|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$

Matrix Norm – The *2*-norm

- The *2*-norm of the matrix is written as $\left\|A\right\|_2 = \max_{\|x\|_2=1} \left\|Ax\right\|_2$
- If we transform a unit vector, {x}, to another vector, {b}, by multiplying with matrix [A], what is maximum "length" of {b}?
- This may be posed as a constrained optimization problem: Maximize $\{x\}^T[A]^T[A]\{x\}$ subject to $\{x\}^T\{x\} = 1$
- Use of the Lagrange multiplier method results in

$$\nabla\left[x^T A^T A x - \lambda\left(x^T x - 1\right)\right] = 0 \Rightarrow A^T A x = \lambda x$$

- Which leads to $\left\|A\right\|_2 = \sqrt{x^T A^T A x} = \sqrt{\lambda}$
- Also known as the spectral norm