Lecture 13.
Pole and Zero Analysis

Guoyong Shi, PhD
shiguoyong@ic.sjtu.edu.cn
School of Microelectronics
Shanghai Jiao Tong University
Fall 2010

Outline

- Computation of transfer function coefficients
- Discrete Fourier Transform
- Polynomial root finding algorithms
  - Muller's Method
- Implementation issues
  - Polynomial deflation

AC Analysis & Pole / Zero

- **Pole-zero (PZ) analysis is based on small-signal.**
- **A DC operating point must be simulated first.**
- **A stamp formulation is similar to AC analysis, except for ‘s’ is not replaced by ‘jω’.**
  - **That is, values of s other than ‘jω’ will be used in PZ analysis.**

AC Analysis

By AC stamping, the following equation is obtained:

$$A(s)\,X = bU(s)$$

Let the LU factorization of A(s) be:

$$A(s) = L\,U;$$

Assuming the diagonal elements of L are all '1'.

$$\det A(s) = u_{11} \cdot u_{22} \cdots u_{nn}$$

The determinant of A(s) will be used for PZ analysis.

Transfer Function

Define an output Y(s) = c X(s) = c A^{-1}(s) b U(s).

$$Y(s) = cA^{-1}(s)bU(s) = \frac{c \cdot \text{adj}(A(s))b}{\det(A(s))}U(s) = \frac{N(s)}{D(s)}U(s)$$

Adj(A) is the adjoint matrix of A.

D(s) and N(s) are polynomials of s.

$$D(s) = \det A(s)$$

The transfer function is

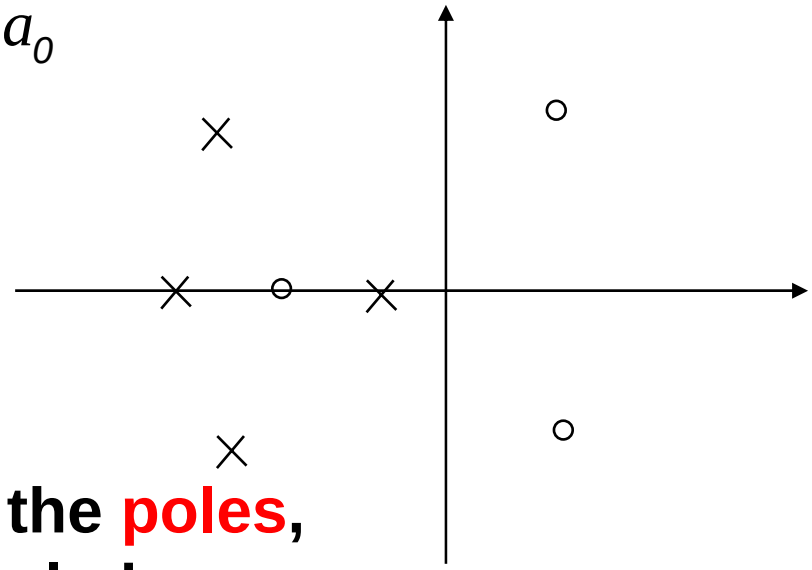
$$H(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

Factorized Form

- **Writing the polynomials in factorized form**

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

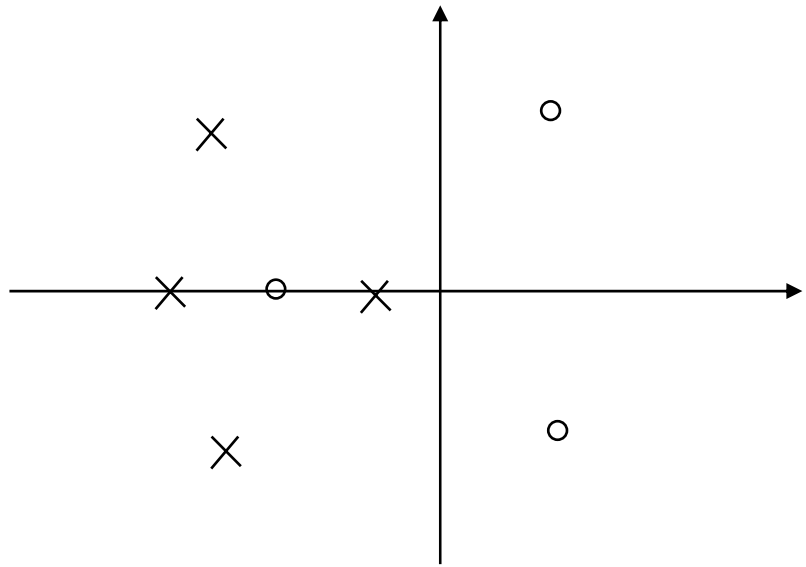
$$= \left(\frac{b_m}{a_n}\right) \frac{(s - z_1)\cdots(s - z_m)}{(s - p_1)\cdots(s - p_n)}$$



z_i's are the zeros and p_j's are the poles, suppose no z_i and p_j are canceled.

Conjugate Poles and Zeros

- **A real circuit has a transfer function with real coefficients.**
- **Hence, the complex poles and zeros must exist in conjugate form.**



Computation Flow

A(s) X = bU(s)

Step 1

$$H(s) = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

Step 2

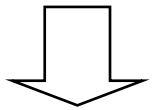
$$H(s) = \left(\frac{b_m}{a_n} \right) \frac{(s - z_1) \cdots (s - z_m)}{(s - p_1) \cdots (s - p_n)}$$

Computation Flow (Step 1)

- In Step 1: We compute the coefficients b_i and a_i.

A(s) X = bU(s)

Step 1



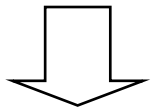
$$H(s) = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

Computation Flow (Step 2)

- **In Step 2: We compute the poles, zeros, and the dc gain.**

$$H(s) = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

Step 2



$$H(s) = \left(\frac{b_m}{a_n} \right) \frac{(s - z_1) \cdots (s - z_m)}{(s - p_1) \cdots (s - p_n)}$$

Computation of Step 1

- **We first compute all a_k.**
- **Then compute all b_i.**
- **The method is by "*sampling*" and "*fitting*".**

$$H(s) = \frac{N(s)}{D(s)} = \frac{b_m s^m + \cdots + b_1 s + b_0}{a_n s^n + \cdots + a_1 s + a_0}$$

$$D(s) = \det A(s)$$

$$D(s) = a_n s^n + \cdots + a_1 s + a_0 = \det A(s)$$

$$N(s) = b_m s^m + \cdots + b_1 s + b_0 = D(s)H(s) \qquad H(s) = \left[cA^{-1}(s)b \right]$$

Sampling 's'

- **The following eqn holds for all s.**
- **Hence, we can take some special samples of 's' to solve all a_k's.**

$$D(s) = a_n s^n + \cdots + a_1 s + a_0 = \det A(s)$$

Take (n+1) samples of s: $s_1, s_2, \ldots, s_k, s_{k+1}$.

$$\begin{bmatrix} 1 & s_1 & s_1^2 & \cdots & s_1^n \\ 1 & s_2 & s_2^2 & \cdots & s_2^n \\ 1 & s_3 & s_3^2 & \cdots & s_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s_{n+1} & s_{n+1}^2 & \cdots & s_{n+1}^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{d_1} \end{bmatrix} \qquad d_i = D(s_i)$$

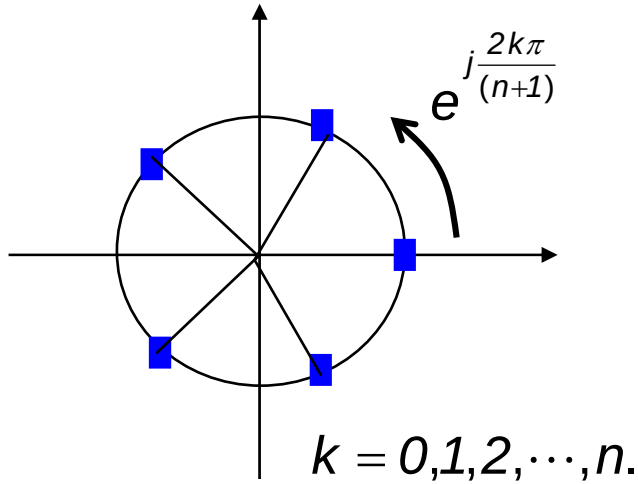
Numerical Stability

- **We have to be careful on the samples.**
- **The coefficient matrix might be ill conditioned if s is not sampled properly.**
- **We'll sample s from the "unit circle" (best).**

$$\begin{bmatrix} 1 & s_1 & s_1^2 & \cdots & s_1^n \\ 1 & s_2 & s_2^2 & \cdots & s_2^n \\ 1 & s_3 & s_3^2 & \cdots & s_3^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s_{n+1} & s_{n+1}^2 & \cdots & s_{n+1}^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{d_1} \end{bmatrix}$$

Unit-Circle Sampling

- **Sampling on the unit circle keep the norm $|s_i| = 1$.**
- **Uniformly spaced samples s_i^k make a Discrete Fourier Transform of the coefficients a_k.**



DFT of $(a_0, \ldots, a_n)$

$$a_0 + a_1\left(e^{j\frac{2\pi}{n+1}}\right) + a_2\left(e^{j\frac{4\pi}{n+1}}\right) + \cdots + a_n\left(e^{j\frac{2n\pi}{n+1}}\right) = d_k$$

Inverse DFT

- **The coefficients a_k can be obtained by Inverse DFT.**

$$a_0 + a_1\left(e^{j\frac{2\pi}{n+1}}\right) + a_2\left(e^{j\frac{4\pi}{n+1}}\right) + \cdots + a_n\left(e^{j\frac{2n\pi}{n+1}}\right) = d_k$$

$$a_k = \frac{1}{n+1}\sum_{i=0}^{n} d_i e^{-j\frac{2\pi ik}{n+1}}$$

(IDFT)

This is the best numerical algorithm with the least numerical error.

Order ‘n’

- **The polynomial order ‘n’ can be determined from the circuit: #L + #C**
  - **Count the C’s and L’s in circuit.**
- **n is usually less than the order of the MNA matrix.**
- **An overestimate of ‘n’ also is OK.**
  - **The IDFT algorithm will calculate very small coefficient values for a_i, i > n, which can be removed.**

Computing the b_k's

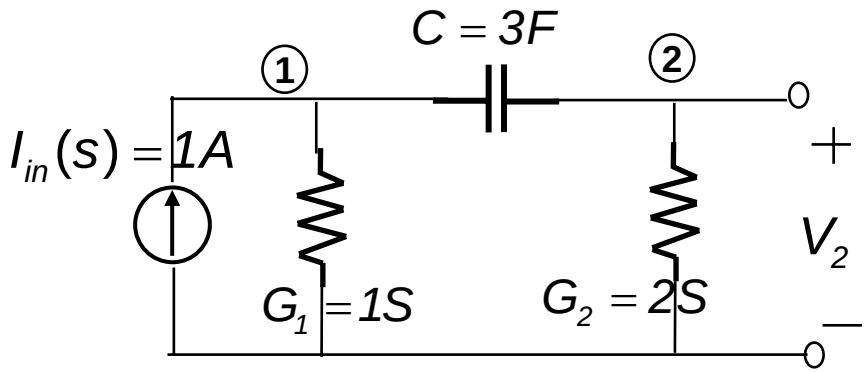
$$N(s_k) = b_m s_k^n + \cdots + b_1 s_k + b_0 = D(s_k)H(s_k)$$

- **We have to solve the output at $s = s_k$, which gives $H(s_k)$**
  - **Assuming the input is a unit impulse: U(s) = 1.**
- **Then apply the same IDFT to compute b_k's.**

Example 1

AC formulation:

$$\begin{bmatrix} G_1 + Cs & -Cs \\ -Cs & G_2 + Cs \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$



There is only one C, we know the degree n = 1.

$$H(s) = \frac{b_0 + b_1 s}{a_0 + a_1 s}$$

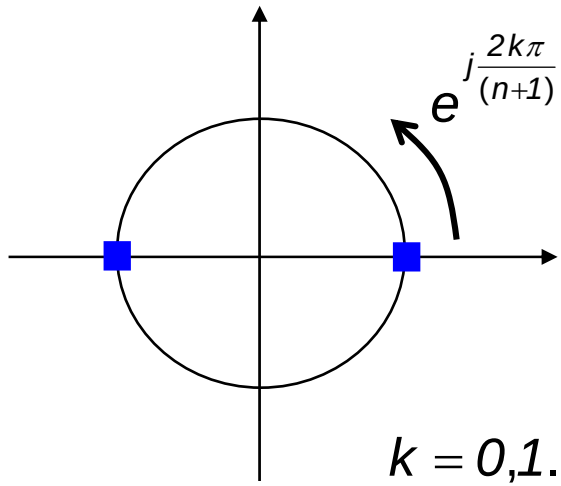
We'll determine the coefficients of H(s).

$$\begin{bmatrix} 1 + 3s & -3s \\ -3s & 2 + 3s \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Example 1 (cont'd)

s will be sampled at (+1) and (-1).

$$s_0 = 1; \quad s_1 = -1.$$



LU factorize for s_0 and s_1:

$$\begin{bmatrix} 1+3s & -3s \\ -3s & 2+3s \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$s_0 = 1 \Rightarrow \begin{bmatrix} 4 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$s_1 = -1 \Rightarrow \begin{bmatrix} -2 & 3 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} V_1(s) \\ V_2(s) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

Example 1 (cont'd)

LU factorizations: **V_2 is output.**

$s_0 = 1$ ⇒ $\begin{bmatrix} 4 & -3 \\ -3 & 5 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ⇒ $\begin{bmatrix} 1 & 0 \\ -\frac{3}{4} & 1 \end{bmatrix}\begin{bmatrix} 4 & -3 \\ 0 & \frac{11}{4} \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

⇒ $D(1) = 11;\ V_2(1) = \frac{3}{11};$ ⇒ $N(1) = D(1)V_2(1) = 11 \times \frac{3}{11} = 3$

$s_1 = -1$ ⇒ $\begin{bmatrix} -2 & 3 \\ 3 & -1 \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ ⇒ $\begin{bmatrix} 1 & 0 \\ -\frac{3}{2} & 1 \end{bmatrix}\begin{bmatrix} -2 & 3 \\ 0 & \frac{7}{2} \end{bmatrix}\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

⇒ $D(-1) = -7;\ V_2(-1) = \frac{3}{7};$ ⇒ $N(-1) = D(-1)V_2(-1) = (-7) \times \frac{3}{7} = -3$

Example 1 (cont'd)

$s_0 = 1$

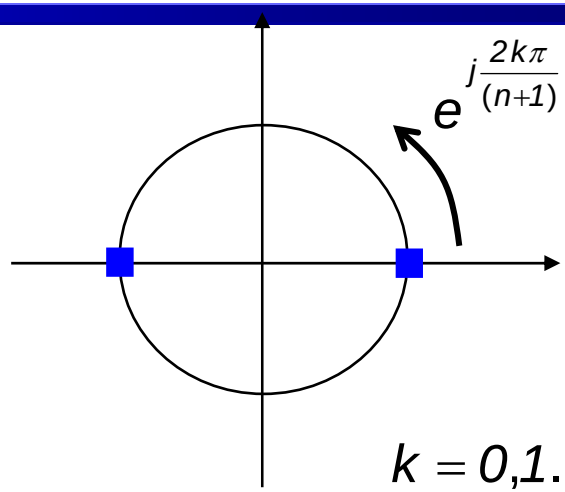
$d_0 = D(s_0) = 11;$

$n_0 = N(s_0) = 3;$

$s_1 = -1$

$d_1 = D(s_1) = -7;$

$n_1 = N(s_1) = -3;$



DFT(a_0, a_1) = (d_0, d_1) ⇒ **(a_0, a_1) = IDFT (d_0, d_1)**

DFT(b_0, b_1) = (n_0, n_1) ⇒ **(b_0, b_1) = IDFT (n_0, n_1)**

Example 1 (cont'd)

$$a_k = \frac{1}{2}\left(d_0 e^{-0} + d_1 e^{-j\frac{2\pi k}{2}}\right);\quad k = 0,1$$

(IDFT)

$$H(s) = \frac{3s}{2+9s}$$

$$d_0 = D(s_0) = 11; \qquad d_1 = D(s_1) = -7;$$

(a_0, a_1) = IDFT (d_0, d_1)

$$\Rightarrow \begin{cases} a_0 = \frac{1}{2}(11-7) = 2 \\ a_1 = \frac{1}{2}(11+7) = 9 \end{cases}$$

$$\Rightarrow D(s) = 2 + 9s;$$

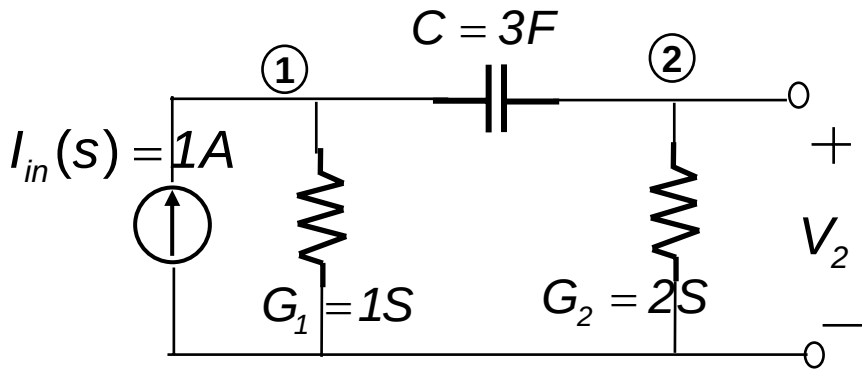
$$n_0 = N(s_0) = 3; \qquad n_1 = N(s_1) = -3;$$

(b_0, b_1) = IDFT (n_0, n_1)

$$\Rightarrow \begin{cases} b_0 = \frac{1}{2}(3-3) = 0 \\ b_1 = \frac{1}{2}(3+3) = 3 \end{cases}$$

$$\Rightarrow N(s) = 3s;$$

Check

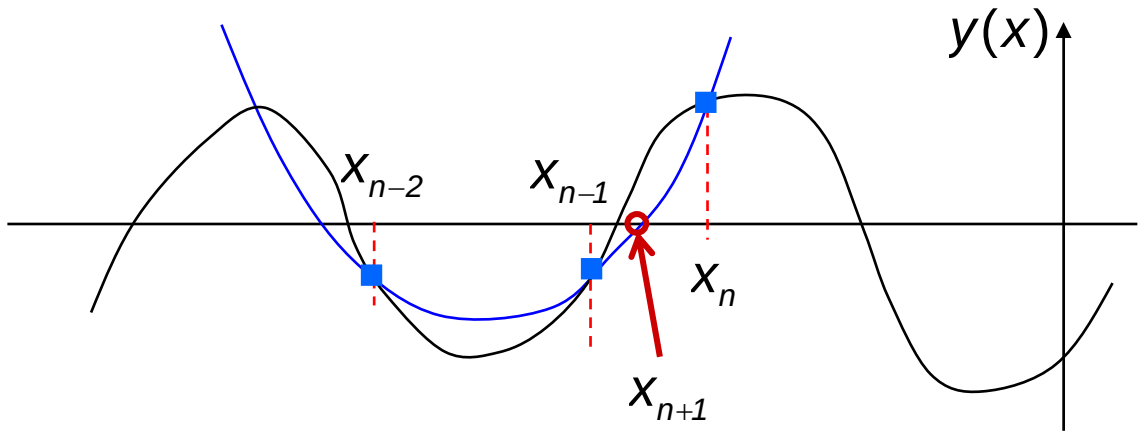


The symbolic transfer function is

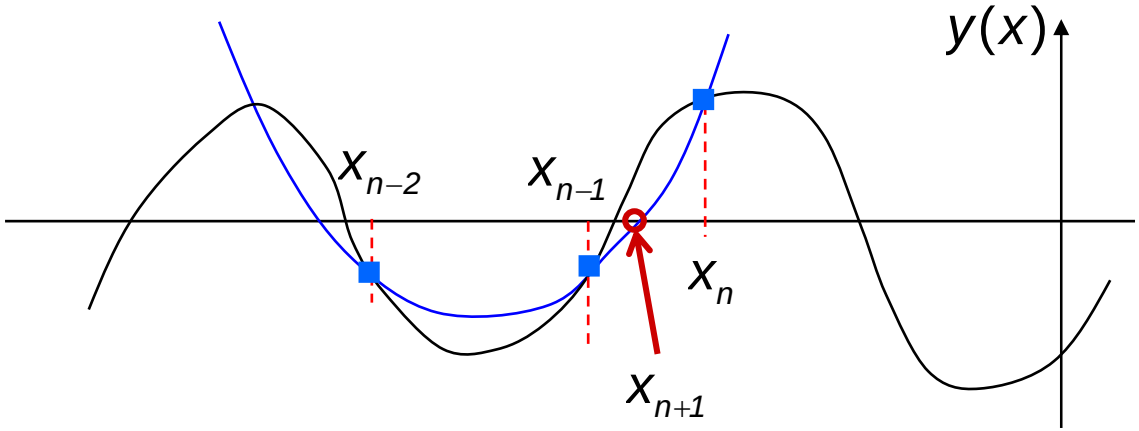
$$H(s) = \frac{Cs}{G_1G_2 + (G_1 + G_2)Cs} \quad \Rightarrow \quad H(s) = \frac{3s}{2 + 9s}$$

Muller's Method

- **Muller's method is an iteration algorithm using quadratic polynomial interpolation.**
- **Used in Berkeley Spice3f5 and HSPICE.**



Muller's Method



Given three points on the polynomial y(x) = 0:

(x_k, y_k), k = n-2, n-1, n.

Find a parabola h(x) passing the three points.

Find a root of h(x) = 0 as the next root estimate.

Muller's Method

- **Given three previous guesses for the root x_{n-2}, x_{n-1}, x_n, and their polynomial values y_{n-2}, y_{n-1}, y_n, the next approximation x_{n+1} is produced by the following formulas:**

$$q = \frac{x_i - x_{i-1}}{x_{i-1} - x_{i-2}}$$

$$a = qy_n - q(q+1)y_{n-1} + q^2 y_{n-2};$$

$$b = (2q+1)y_n - (q+1)^2 y_{n-1} + q^2 y_{n-2};$$

$$c = (q+1)y_n;$$

$$x_{n+1} = x_n - (x_n - x_{n-1})\frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

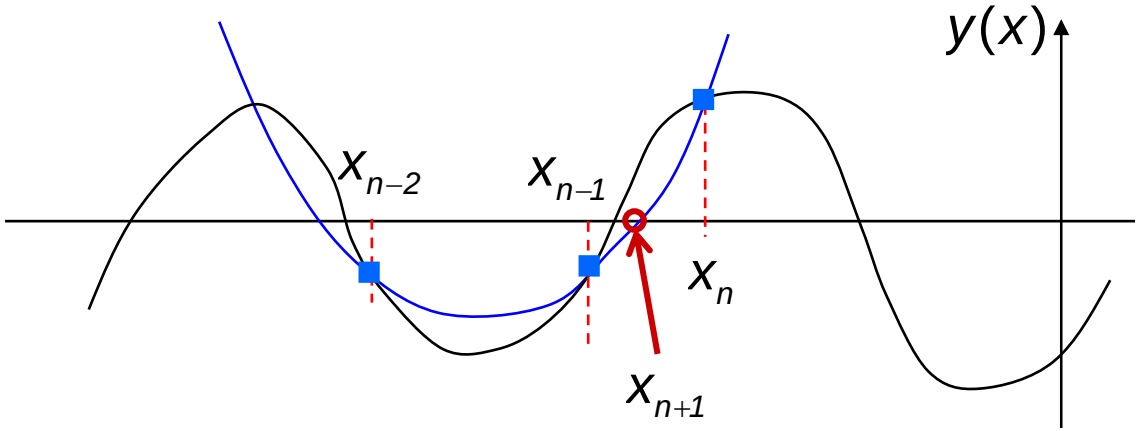
Choose +/- to make the modulus larger.

$$\left| b \pm \sqrt{b^2 - 4ac} \right|$$

Advantages of Muller's Method

- **Muller's method solves roots of a quadratic function, it can find complex pairs of roots.**
- **You may start the iteration with any three values of x, e.g., three equally spaced values on the x axis.**
- **The denominator could possibly be a complex number, hence your implementation must use complex number arithmetic.**

Derivation of Muller's Method



The Lagrange interpolation polynomial is

$$L(x) = y_n \frac{(x - x_{n-2})(x - x_{n-1})}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{(x - x_{n-2})(x - x_n)}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{(x - x_{n-1})(x - x_n)}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

This polynomial is a parabola passing the three points.

Derivation (cont'd)

$$L(x) = y_n \frac{(x - x_{n-2})(x - x_{n-1})}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{(x - x_{n-2})(x - x_n)}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{(x - x_{n-1})(x - x_n)}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

$L(x) = C + B(x - x_n) + A(x - x_n)^2$ another form of L(x)

Let's determine A, B, C.

$$L(x_n) = C = y_n$$

$$L'(x_n) = B = y_n \frac{(x_n - x_{n-2}) + (x_n - x_{n-1})}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{(x_n - x_{n-2})}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{(x_n - x_{n-1})}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

$$\frac{1}{2} L''(x) = A = y_n \frac{1}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{1}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{1}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

Derivation (cont'd)

Define $$q = \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}} \quad \Rightarrow \quad q+1 = \frac{x_n - x_{n-2}}{x_{n-1} - x_{n-2}}$$

$$B = y_n \frac{(x_n - x_{n-2}) + (x_n - x_{n-1})}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{(x_n - x_{n-2})}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{(x_n - x_{n-1})}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

$$= \left[y_n \underbrace{\frac{(x_n - x_{n-2}) + (x_n - x_{n-1})}{(x_n - x_{n-2})}} - y_{n-1} \underbrace{\frac{(x_n - x_{n-2})}{(x_{n-1} - x_{n-2})}} + y_{n-2} \underbrace{\frac{(x_n - x_{n-1})^2}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}} \right] \frac{1}{(x_n - x_{n-1})}$$

$$1 + \frac{q}{q+1} = \frac{2q+1}{q+1} \qquad -(q+1) \qquad \frac{q^2}{q+1}$$

$$B = \left[y_n \frac{2q+1}{q+1} - y_{n-1} \frac{(q+1)^2}{(q+1)} + y_{n-2} \frac{q^2}{(q+1)} \right] \frac{1}{(x_n - x_{n-1})}$$

Derivation (cont'd)

$$A = y_n \frac{1}{(x_n - x_{n-2})(x_n - x_{n-1})} + y_{n-1} \frac{1}{(x_{n-1} - x_{n-2})(x_{n-1} - x_n)} + y_{n-2} \frac{1}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)}$$

$$= \left[y_n \frac{(x_n - x_{n-1})}{(x_n - x_{n-2})} - y_{n-1} \frac{(x_n - x_{n-1})}{(x_{n-1} - x_{n-2})} + y_{n-2} \frac{(x_n - x_{n-1})^2}{(x_{n-2} - x_{n-1})(x_{n-2} - x_n)} \right] \frac{1}{(x_n - x_{n-1})^2}$$

$$\frac{q}{q+1} \qquad -q \qquad \frac{q^2}{q+1}$$

$$A = \left[y_n \frac{q}{(q+1)} - y_{n-1} \frac{q(q+1)}{(q+1)} + y_{n-2} \frac{q^2}{(q+1)} \right] \frac{1}{(x_n - x_{n-1})^2}$$

Derivation (cont'd)

$$C = y_n = \frac{c}{(q+1)}$$

$$B = \left[y_n \frac{2q+1}{q+1} - y_{n-1} \frac{(q+1)^2}{(q+1)} + y_{n-2} \frac{q^2}{(q+1)} \right] \frac{1}{(x_n - x_{n-1})} = \frac{b}{(q+1)(x_n - x_{n-1})}$$

$$A = \left[y_n \frac{q}{(q+1)} - y_{n-1} \frac{q(q+1)}{(q+1)} + y_{n-2} \frac{q^2}{(q+1)} \right] \frac{1}{(x_n - x_{n-1})^2} = \frac{a}{(q+1)(x_n - x_{n-1})^2}$$

$$L(x) = C + B(x - x_n) + A(x - x_n)^2 = 0$$

$$\Rightarrow \quad c + b\frac{(x - x_n)}{(x_n - x_{n-1})} + a\frac{(x - x_n)^2}{(x_n - x_{n-1})^2} = 0$$

Derivation (cont'd)

$$c + b\frac{(x - x_n)}{(x_n - x_{n-1})} + a\frac{(x - x_n)^2}{(x_n - x_{n-1})^2} = 0$$

$$\Rightarrow \quad \frac{(x_{n+1} - x_n)}{(x_n - x_{n-1})} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = -\frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

$$\Rightarrow \quad \boxed{x_{n+1} = x_n - (x_n - x_{n-1})\frac{2c}{b \pm \sqrt{b^2 - 4ac}}}$$

(Done)

Deflation of Polynomial

- **When a root r is found, the polynomial P(s) can be factorized as *P(s) = (s-r)Q(s)*.**
- ***Q(s)* is a reduced-order polynomial.**
- **It avoids finding again the same root already found.**
- ***Deflation* is just polynomial division.**
- **In the case of conjugate roots *(a±jb)*, we can deflate by a quadratic factor:**

$$s^2 - 2as + (a^2+b^2)$$

More on Deflation

- **Deflation must be used with care.**
- **Each new root is known with finite accuracy, error accumulates.**
- **The coefficients of the deflated polynomial Q(x) are not accurate.**
- **The roots can become increasingly inaccurate.**

Forward / Backward Deflation

$$P(s) = a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0$$

$$P(s) = (s - p)Q(s) \qquad Q(s) = q_{n-1} s^{n-1} + q_{n-2} s^{n-2} + \cdots + q_1 s + q_0$$

Stable deflation strategies:

- **Forward Deflation – Compute from q_{n-1} down to q_0 if the smallest |p| is divided out at each stage.**

- **Backward Deflation – Compute from q_0 up to q_{n-1} if the largest |p| is divided out at each stage.**

Stable Deflation

$$
\begin{aligned}
P(s) &= a_n s^n + a_{n-1} s^{n-1} + \cdots + a_2 s^2 + a_1 s + a_0 \\
sQ(s) &= q_{n-1} s^n + q_{n-2} s^{n-1} + \cdots + q_1 s^2 + q_0 s \\
pQ(s) &= \qquad\quad pq_{n-1} s^{n-1} + \cdots + pq_2 s^2 + pq_1 s + pq_0
\end{aligned}
$$

$$P(s) = (s - p)Q(s)$$

Forward deflation

(|p| smallest)

$$
\begin{aligned}
a_n &= q_{n-1} \\
a_{n-1} &= q_{n-2} - pq_{n-1} \\
&\vdots \\
a_1 &= q_0 - pq_1 \\
a_0 &= \quad - pq_0
\end{aligned}
$$

Backward deflation

(|p| largest)

Quadratic Deflation

$$
\begin{array}{rcl}
P(s) = & a_n s^n + a_{n-1} s^{n-1} + a_{n-2} s^{n-2} + \cdots + a_2 s^2 + a_1 s + a_0 \\
\hline
s^2 Q(s) = & q_{n-2} s^n + q_{n-3} s^{n-1} + q_{n-4} s^{n-2} + \cdots + q_0 s^2 \\
\alpha s Q(s) = & \alpha q_{n-2} s^{n-1} + \alpha q_{n-3} s^{n-2} + \cdots + \alpha q_1 s^2 + \alpha q_0 s \\
\beta Q(s) = & \beta q_{n-2} s^{n-2} + \cdots + \beta q_2 s^2 + \beta q_1 s + \beta q_0
\end{array}
$$

$$P(s) = (s^2 + \alpha s + \beta) Q(s)$$

$$Q(s) = q_{n-2} s^{n-2} + \cdots + q_1 s + q_0$$

Quadratic Deflation

$$a_n = q_{n-2}$$

$$a_{n-1} = q_{n-3} + \alpha q_{n-2}$$

$$a_{n-2} = q_{n-4} + \alpha q_{n-3} + \beta q_{n-2}$$

$$\vdots$$

$$a_2 = q_0 + \alpha q_1 + \beta q_2$$

$$a_1 = \alpha q_0 + \beta q_1$$

$$a_0 = \beta q_0$$

Forward deflation

(|p| smallest)

Backward deflation

(|p| largest)

Reference

- **J. Vlach and K. Singhal, Computer Methods for Circuit Analysis and Design, Van Nostrand Reinhold Company, 1983. (Chapter 7)**