Raíces de ecuaciones
Métodos abiertos y mixtos
Lección 06

Dr. Pablo Alvarado Moya

CE3102 Análisis Numérico para Ingeniería
Área de Ingeniería en Computadores
Tecnológico de Costa Rica

I Semestre 2018

Contenido

Métodos abiertos

- Los métodos abiertos requieren un único valor inicial o un par de valores pero que no necesitan encerrar la raíz buscada.
- Los métodos abiertos a veces son **divergentes**
- Si convergen, lo hacen más rápido que los métodos cerrados

Se revisará:

1. Iteración de punto fijo
2. Método de Newton-Raphson
3. Método de la secante

Iteración de punto fijo

La iteración de punto fijo parte de reformular la ecuación

$$f(x) = 0$$

en

$$x = g(x)$$

y permitir que un proceso iterativo

$$x_{i+1} = g(x_i)$$

converja a la raíz, con error aproximado calculado como:

$$\epsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| 100\,\%$$

Ejemplos de reformulación para iteraciones

- $x^2 - 2x + 3 = 0 \Rightarrow x = \dfrac{x^2 + 3}{2}$
- $\operatorname{sen} x = 0 \Rightarrow x = \operatorname{sen} x + x$
- $e^{-x} - x = 0 \Rightarrow e^{-x} = x$

Teorema del valor medio de la derivada

- Sean $g(x)$ y su derivada $g'(x)$ contínuas en un intervalo $[a, b]$
- El teorema del valor medio de la derivada establece que existe algún punto ξ sobre el que $g'(\xi)$ iguala a la pendiente de la recta trazada entre $g(a)$ y $g(b)$:

$$g'(\xi) = \frac{g(b) - g(a)}{b - a}$$

Convergencia del método de punto fijo (1)

La ecuación iterativa de búsqueda de la raíz es

$$x_{i+1} = g(x_i)$$

y para la raíz verdadera x_r se cumple

$$x_r = g(x_r)$$

Restando ambas ecuaciones se obtiene

$$x_r - x_{i+1} = g(x_r) - g(x_i)$$

Convergencia del método de punto fijo (2)

Con el teorema del valor medio se puede expresar con ξ en el intervalo entre x_i y x_r

$$g'(\xi) = \frac{g(x_r) - g(x_i)}{x_r - x_i}$$

y por tanto

$$g'(\xi)(x_r - x_i) = g(x_r) - g(x_i)$$

combinando esto con

$$x_r - x_{i+1} = g(x_r) - g(x_i)$$

se obtiene

$$x_r - x_{i+1} = g'(\xi)(x_r - x_i)$$

Convergencia del método de punto fijo (3)

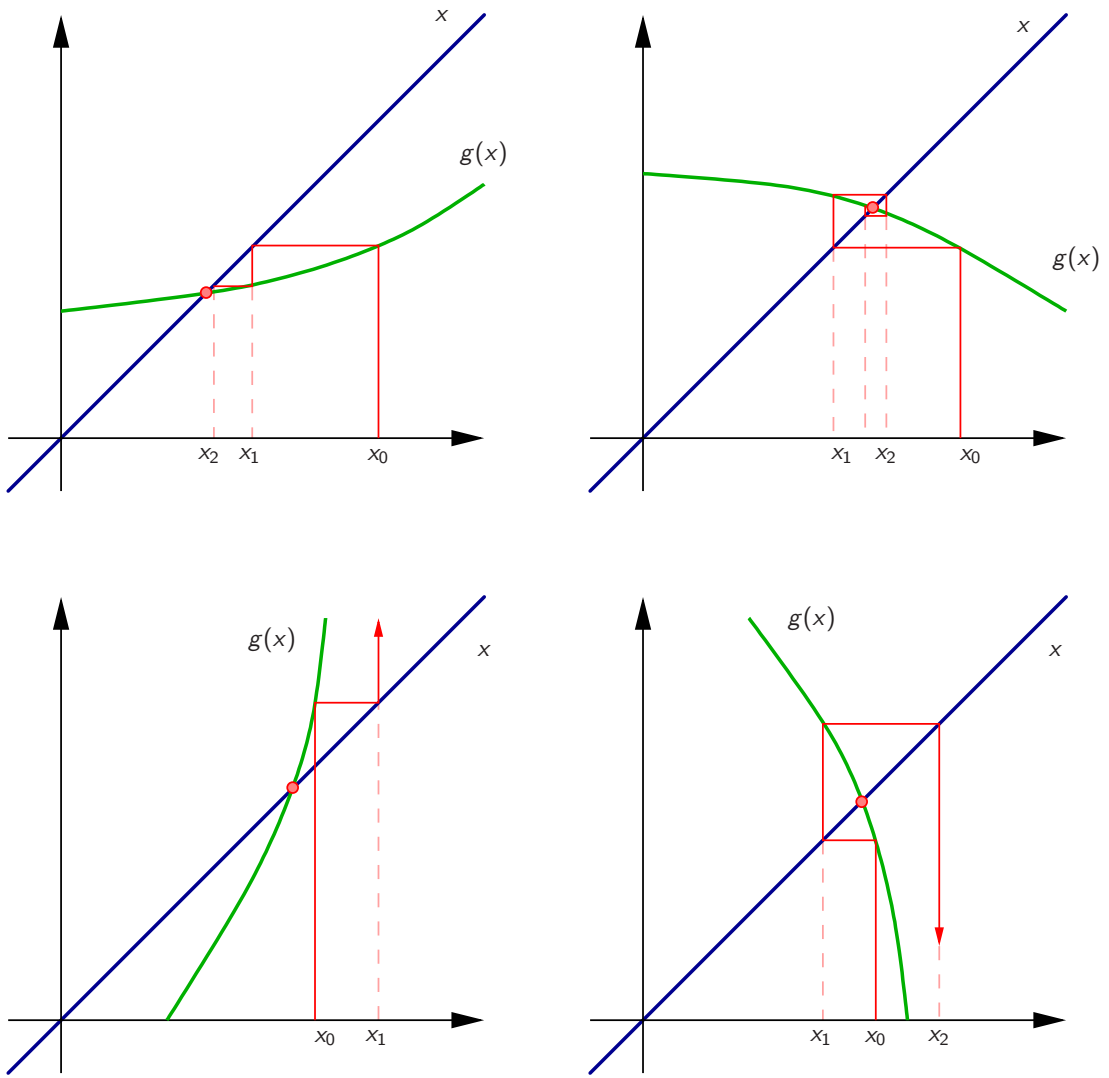
Si el error verdadero en la i-ésima iteración es $E_{t,i} = x_r - x_i$ entonces:

$$E_{t,i+1} = g'(\xi) E_{t,i}$$

de donde se deriva que la magnitud de la derivada $|g'(x)|$ debe ser menor que uno para asegurar **convergencia lineal**.

Si $|g'(x)| > 1$ entonces este método diverge.

Convergencia del método de punto fijo (4)



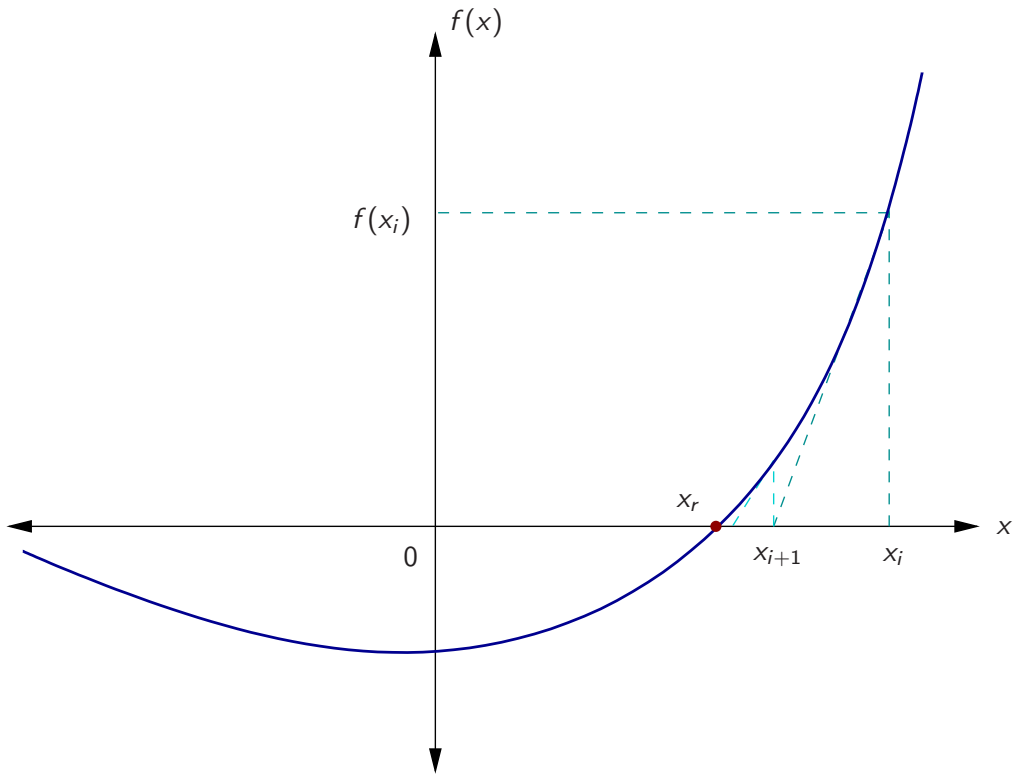
Estructura del código

```
template <typename T>
T fixpoint(T (*f)(const T),
           T x0,
           const T es=std::sqrt(std::numeric_limits<T>::epsilon())
           const int maxi=std::numeric_limits<T>::digits) {

  T xr=x0; // hay que iniciar con algo válido
  T ea=T();
  for (int i=maxi;i>0;--i){
    T xrold(xr); // lo necesitamos para el cálculo del error
    xr=f(xrold); // iteración de punto fijo
    // Evite una divisón por cero
    if (std::abs(xr) > std::numeric_limits<T>::epsilon()) {
      ea = std::abs((xr-xrold)/xr)*T(100);
    }
    if (ea < es) return xr;
  }
  // Retorne un NaN si no encontró ninguna raíz maxi iteraciones
  return std::numeric_limits<T>::quiet_NaN();
}
```

Método de Newton-Raphson

Concepto



- Encontrar x_r tal que $0 = f(x_r)$
- $f'(x_i) \stackrel{!}{=} \frac{f(x_i)}{x_i - x_{i+1}} \quad \Rightarrow \quad x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

Método de Newton-Raphson (1)

Derivación

Se desea encontrar el valor x_r para la función $y = f(x)$ que hace

$$0 = f(x_r)$$

Considere la expansión de primer orden de Taylor

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \underbrace{\frac{f''(\xi)}{2!}(x_{i+1} - x_i)^2}_{\text{Resíduo}}$$

con $\xi \in [x_i, x_{i+1}]$, donde se desea que x_{i+1} corresponda a la raíz x_r.

Método de Newton-Raphson (2)

Derivación

La aproximación de primer orden se utiliza para encontrar la raíz:

$$0 \stackrel{!}{=} f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

de donde se despeja

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

que es la iteración del método **Newton-Raphson**.

Obsérvese que este método requiere tanto $f(x)$ como $f'(x)$.

Estimación del error (1)

Método de Newton-Raphson

Con la serie de Taylor completa, para la raíz x_r se debe cumplir:

$$0 = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

y restando el resultado anterior que establece:

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i)$$

se obtiene

$$0 = f'(x_i)(x_r - x_{i+1}) + \frac{f''(\xi)}{2!}(x_r - x_i)^2$$

Estimación del error (2)

Método de Newton-Raphson

Con $E_{t,i} = x_r - x_i$ y $E_{t,i+1} = x_r - x_{i+1}$ se tiene entonces

$$0 = f'(x_i)E_{t,i+1} + \frac{f''(\xi)}{2!}E_{t,i}^2$$

Finalmente se reordena el error para obtener

$$E_{t,i+1} = -\frac{f''(\xi)}{2f'(x_i)}E_{t,i}^2$$

lo que indica que si el método converge, lo hace con orden **cuadrático**: el número de cifras significativas decimales se duplica (aproximadamente) en cada iteración.

El método en general converge si

$$\left|\frac{f''(\xi)}{2f'(x_i)}\right| < 1$$

Problemas del método de Newton-Raphson

- Debido a que en la iteración aparece la derivada en el denominador, el método se inestabiliza si pasos intermedios caen cerca de máximos o mínimos locales.
- Otra posibilidad es que el método oscile sin converger cuando hay puntos de inflexión ($f''(x) = 0$) cerca de la raíz.
- No hay un criterio de convergencia general para este método:
  - Depende de la naturaleza de la función
  - Depende de la elección del punto inicial x_0.
- Método puede converger a puntos que no son la raíz: debe confirmarse al final del proceso iterativo que $f(x_r) \approx 0$.

Problemas del método de Newton-Raphson

Método de la secante

- Equivale al método de Newton-Raphson sustituyendo la derivada por su aproximación hacia atrás:

$$f'(x_i) \approx \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}$$

- La iteración se convierte entonces en:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}$$

- Se requieren dos iteraciones anteriores: x_{i-1} y x_i
- No es necesario encerrar a x_r

Método de la secante

- Similitud con método de interpolación lineal (o falsa posición).

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \qquad \text{M. secante}$$

$$x_i = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \qquad \text{M. interpolación lineal}$$

- Convergencia es subcuadrática, orden $\approx 1,618$, es decir

$$E_{t,i+1} = \text{const} \times E_{t,i}^{1,618}$$

Diferencias

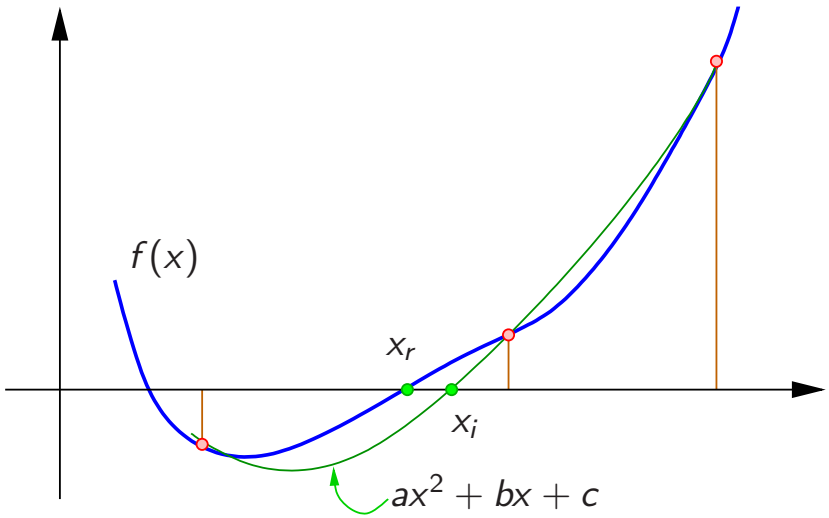
Métodos de la secante y de interpolación lineal

- El método de interpolación lineal sustituye uno de los extremos (x_u o x_l) por la última estimación de modo que siempre la raíz quede acorralada.
- El método de la secante reemplaza los valores en secuencia estricta, por lo que la raíz puede quedar fuera de los dos últimos valores utilizados.
  - **Peligro**: Esto no siempre lleva a convergencia.

Método de Brent

- O también método de van Wijngaarden-Dekker-Brent
- Garantiza convergencia si intervalo inicial encierra una raíz.
- Combina:
  1. Método de bisección
  2. Método de la secante
  3. Método de interpolación inversa cuadrática

Interpolación cuadrática (1)



Interpolación cuadrática (2)

Dados tres puntos x_i, x_{i-1} y x_{i-2} y sus correspondientes valores $y_i = f(x_i)$, $y_{i-1} = f(x_{i-1})$ e $y_{i-2} = f(x_{i-2})$ se asume que se cumple

$$y_i = ax_i^2 + bx_i + c$$
$$y_{i-1} = ax_{i-1}^2 + bx_{i-1} + c$$
$$y_{i-2} = ax_{i-2}^2 + bx_{i-2} + c$$

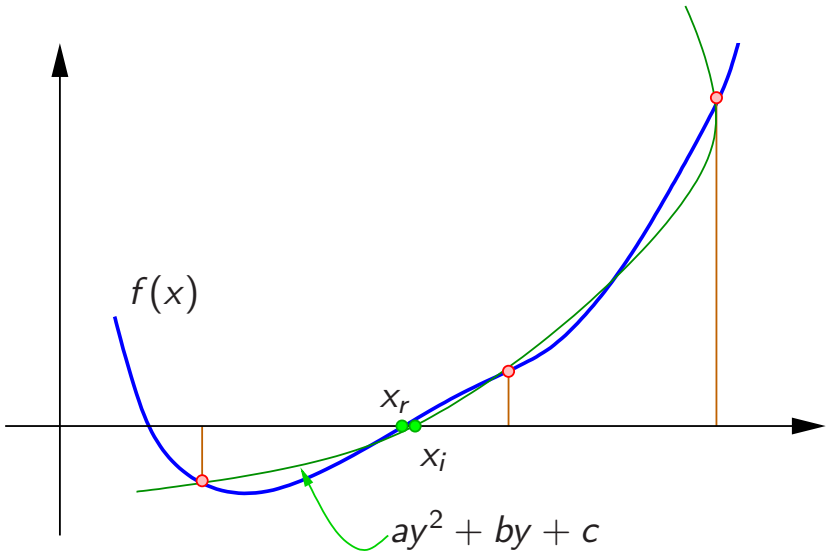
que en forma matricial

$$\begin{bmatrix} y_i \\ y_{i-1} \\ y_{i-2} \end{bmatrix} = \begin{bmatrix} x_i^2 & x_i & 1 \\ x_{i-1}^2 & x_{i-1} & 1 \\ x_{i-2}^2 & x_{i-2} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

se pueden despejar los coeficientes a, b y c.

Interpolación inversa cuadrática (1)

Es posible que la parábola interpolada no interseque del todo al eje real. Pero, esto no ocurre si se utiliza interpolación inversa cuadrática



Interpolación inversa cuadrática (2)

Dados de nuevo los tres puntos x_i, x_{i-1} y x_{i-2} y sus correspondientes valores $y_i = f(x_i)$, $y_{i-1} = f(x_{i-1})$ e $y_{i-2} = f(x_{i-2})$ se asume ahora que se cumple

$$x_i = ay_i^2 + by_i + c$$
$$x_{i-1} = ay_{i-1}^2 + by_{i-1} + c$$
$$x_{i-2} = ay_{i-2}^2 + by_{i-2} + c$$

que en forma matricial

$$\begin{bmatrix} x_i \\ x_{i-1} \\ x_{i-2} \end{bmatrix} = \begin{bmatrix} y_i^2 & y_i & 1 \\ y_{i-1}^2 & y_{i-1} & 1 \\ y_{i-2}^2 & y_{i-2} & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

se pueden despejar los coeficientes a, b y c.

Interpolación inversa cuadrática (3)

- Lo anterior funciona siempre y cuando no existan dos o más valores iguales de y, en cuyo caso el sistema de ecuaciones no tiene solución por existir varios valores x para un mismo y.
- En el método de Brent, si lo anterior es el caso entra en juego el método de la secante.

Interpolación inversa cuadrática (4)

Se puede demostrar que la función buscada es

$$x = \frac{(y-y_{i-1})(y-y_i)}{(y_{i-2}-y_{i-1})(y_{i-2}-y_i)}x_{i-2} + \frac{(y-y_{i-2})(y-y_i)}{(y_{i-1}-y_{i-2})(y_{i-1}-y_i)}x_{i-1} + \frac{(y-y_{i-2})(y-y_{i-1})}{(y_i-y_{i-2})(y_i-y_{i-1})}x_i$$

y sabiendo que se busca la intersección con $y = 0$ se obtiene

$$x_{i+1} = \frac{y_{i-1}y_i}{(y_{i-2}-y_{i-1})(y_{i-2}-y_i)}x_{i-2} + \frac{y_{i-2}y_i}{(y_{i-1}-y_{i-2})(y_{i-1}-y_i)}x_{i-1} + \frac{y_{i-2}y_{i-1}}{(y_i-y_{i-2})(y_i-y_{i-1})}x_i$$

Algoritmo de Brent

- El método de Brent utiliza el método de interpolación cuadrática cuando sea posible,
- de otro modo utiliza el método de la secante
- si ese método conduce a error, utilice bisección.

TAREA

Implementar el algoritmo de Brent en C++ para funtores que sigan las definiciones para funtores unarios (ver definiciones de funtores en la STL).

Raíces de polinomios

Generalidades

- Modelos de sistemas físicos se expresan por medio de ecuaciones diferenciales

Generalidades

- Modelos de sistemas físicos se expresan por medio de ecuaciones diferenciales
- Gran cantidad de sistemas (circuitos eléctricos lineales, sistemas de masas y resortes, sistemas neumáticos, térmicos, etc.) se expresan con ecuaciones diferenciales con coeficientes constantes.

Generalidades

- Modelos de sistemas físicos se expresan por medio de ecuaciones diferenciales
- Gran cantidad de sistemas (circuitos eléctricos lineales, sistemas de masas y resortes, sistemas neumáticos, térmicos, etc.) se expresan con ecuaciones diferenciales con coeficientes constantes.
- Transformada de Laplace permite mapear ecuación diferencial a polinomio en s

Generalidades

- Modelos de sistemas físicos se expresan por medio de ecuaciones diferenciales
- Gran cantidad de sistemas (circuitos eléctricos lineales, sistemas de masas y resortes, sistemas neumáticos, térmicos, etc.) se expresan con ecuaciones diferenciales con coeficientes constantes.
- Transformada de Laplace permite mapear ecuación diferencial a polinomio en s
- Solución de dichas ecuaciones diferenciales requiere encontrar **todas** las raíces (reales y complejas) de polinomios.

Generalidades

- Modelos de sistemas físicos se expresan por medio de ecuaciones diferenciales
- Gran cantidad de sistemas (circuitos eléctricos lineales, sistemas de masas y resortes, sistemas neumáticos, térmicos, etc.) se expresan con ecuaciones diferenciales con coeficientes constantes.
- Transformada de Laplace permite mapear ecuación diferencial a polinomio en s
- Solución de dichas ecuaciones diferenciales requiere encontrar **todas** las raíces (reales y complejas) de polinomios.
- Encontrar raíces de un polinomio es un problema mal condicionado: cambios leves de los coeficientes pueden producir fuertes cambios en la posición de las raíces.

Evaluación de polinomios

- La forma estándar de polinomios

$$f_n(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots a_n x^n$$

  requiere $n(n+1)/2$ multiplicaciones y n sumas.
- La forma anidada (esquema de Horner):

$$f_n(x) = a_0 + x(a_1 + x(a_2 + x(\ldots(a_{n-1} + a_n x)\ldots)))$$

  requiere n multiplicaciones y n sumas.
- Al requerir menos operaciones, la forma anidada produce menos errores de redondeo.

Implementación de forma anidada

Cálculo del polinomio

```
for (j=n,p=0; n>=0; --n) {
  p = p*x + a[j];
}
```

Cálculo del polinomio y su derivada

```
for (j=n,p=0,df=0; n>=0; --n) {
  df = df*x + p;
  p = p*x + a[j];
}
```

Esto último es útil por ejemplo con métodos como el Newton-Raphson, que requiere tanto el valor de la función como de su derivada.

Deflación polinomial

- Deflación polinomial: proceso de reducir grado de polinomio dividiéndolo por $(x - t_i)$, t_i raíz conocida del polinomio.
- Todo polinomio

$$\sum_{i=0}^{n} a_i x^i = a_n \prod_{i=1}^{n} (x - t_i)$$

con t_i, $i = 1 \ldots n$ las n raíces del polinomio

- Las raíces pueden ser algunas o todas iguales; reales o complejas.

Algoritmo de deflación polinomial

- Algoritmo para calcular la deflación polinomial:

```
r=a[n]; a[n]=0; // reducción del orden del polinomio
for (i=n-1; n>=0; --n) { s=a[i]; a[i]=r; r=s+r*t; }
```

- Si polinomio es divisible por $t \Rightarrow$ residuo $r = 0$.
- **División polinomial** necesaria para tratar el caso de pares de raíces complejas conjugadas, las cuales producen polinomios de segundo orden de coeficientes reales.

Proyecto 1
Primera parte

Plantear algoritmo de deflación polinomial, que funcione eficientemente tanto con raíces reales como complejas.

Problemas de la deflación

- Como raíces calculadas son aproximaciones, el método de deflación es sensible a los errores de redondeo.

Problemas de la deflación

- Como raíces calculadas son aproximaciones, el método de deflación es sensible a los errores de redondeo.
- El error de redondeo depende del orden de evaluación:

Problemas de la deflación

- Como raíces calculadas son aproximaciones, el método de deflación es sensible a los errores de redondeo.
- El error de redondeo depende del orden de evaluación:
  - En la deflación hacia **adelante** los coeficientes del polinomio tienen grado de potencias descendientes de x. En este caso se prefiere dividir primero por las raíces de valor absoluto pequeño.

Problemas de la deflación

- Como raíces calculadas son aproximaciones, el método de deflación es sensible a los errores de redondeo.
- El error de redondeo depende del orden de evaluación:
  - En la deflación hacia **adelante** los coeficientes del polinomio tienen grado de potencias descendientes de x. En este caso se prefiere dividir primero por las raíces de valor absoluto pequeño.
  - En la deflación hacia **atrás** los coeficientes del polinomio tienen grado de potencias ascendentes de x. En este caso se prefiere dividir primero por las raíces de valor absoluto grande.

Problemas de la deflación

- Como raíces calculadas son aproximaciones, el método de deflación es sensible a los errores de redondeo.
- El error de redondeo depende del orden de evaluación:
  - En la deflación hacia **adelante** los coeficientes del polinomio tienen grado de potencias descendientes de x. En este caso se prefiere dividir primero por las raíces de valor absoluto pequeño.
  - En la deflación hacia **atrás** los coeficientes del polinomio tienen grado de potencias ascendentes de x. En este caso se prefiere dividir primero por las raíces de valor absoluto grande.
- **Pulir** raíces es el proceso de, luego de la deflación y obtener nuevas raíces, utilizar éstas como puntos iniciales en la búsqueda de raíces pero con el polinomio original.

Aplicabilidad de métodos anteriores

- Todos los métodos cerrados y abiertos descritos anteriormente se pueden utilizar, siempre y cuando las raíces buscadas sean **reales**.
- A pesar de lo anterior, encontrar el intervalo o valor inicial es un problema complejo.
- Nuevos métodos son necesarios si raíces son **complejas**

Método de Müller

El método de Müller es similar al método de la secante, pero utiliza interpolación cuadrática, como se ilustró anteriormente, para encontrar la siguiente estimación de la raíz.

La aproximación cuadrática centrada en x_i:

$$\tilde{f}(x) = a(x - x_i)^2 + b(x - x_i) + c$$

a partir de tres puntos x_{i-2}, x_{i-1} y x_i tiene con

$$h_{i-2} = x_{i-1} - x_{i-2} \qquad h_{i-1} = x_i - x_{i-1}$$

$$\delta_{i-2} = \frac{f(x_{i-1}) - f(x_{i-2})}{x_{i-1} - x_{i-2}} \qquad \delta_{i-1} = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

coeficientes dados por:

$$a = \frac{\delta_{i-1} - \delta_{i-2}}{h_{i-1} - h_{i-2}}$$

$$b = ah_{i-1} + \delta_{i-1}$$

$$c = f(x_i)$$

Estimación de la raíz

- La estimación de la raíz se realiza con la fórmula cuadrática **alternativa**
- Permite reducir el error de redondeo potencial:

$$x_{i+1} - x_i = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}}$$

$$x_{i+1} = x_i - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

- Raíces **pueden** ser complejas.
- **Nótese** error aproximado dado por primera ecuación.

Problema del método de Müller

- Problema del método: produce dos raíces.
- Método establece elegir signo que coincida con el signo de b, para obtener la raíz estimada más cercana a x_i.

Iteración de algoritmo de Müller

La iteración del algoritmo de Müller depende si se buscan raíces puramente reales, o se permiten además raíces complejas.

1. Si se buscan raíces reales únicamente, se eligen los dos valores de $\{x_i, x_{i-1}, x_{i-2}\}$ más cercanos a la nueva raíz estimada x_{i+1}
2. Si se buscan también raíces complejas, se utiliza el método puramente secuencial (como en el método de la secante), y se usan $\{x_i, x_{i-1}\}$ junto a la nueva raíz x_{i+1}.

Proyecto 1
Segunda parte

Implemente los métodos de Müller y de Laguerre (Numerical Recipes) y compare sus convergencias en la búsqueda de las raíces, iniciando desde 0, entre otros para el polinomio en el enunciado.
Deben encontrarse todas las raíces, por lo que se requiere hacer deflación.
Utilice funtores y plantillas de C++, y evalúe además los efectos de utilizar precisiones simples o dobles, con y sin pulido.

Resumen

1. Métodos abiertos
   - Iteración de punto fijo
   - Método de Newton-Raphson
2. Métodos Mixtos
   - Método de Brent
3. Raíces de polinomios

Este documento ha sido elaborado con software libre incluyendo LaTeX, Beamer, GNUPlot, GNU/Octave, XFig, Inkscape, LTI-Lib-2, GNU-Make y Subversion en GNU/Linux