Métodos Numéricos

Francisco Frutos Alfaro
Escuela de Física
Centro de Investigaciones Espaciales
UCR

Curso Ingeniería Eléctrica y Computación
Enero 2014

Tópicos

- Introducción
- Interpolación y Extrapolación
- Solución de sistemas lineales: Vectores, matrices y autovalores
- Solución de (sistemas) ecuaciones no lineales
- Derivación e integración
- Ecuaciones diferenciales ordinarias
- Ecuaciones en derivadas parciales
- Ajuste de Curvas
- Optimización

Introducción

- Métodos numéricos juegan un rol importante en la investigación científica moderna
- Los procesos o sistemas se pueden representar matemáticamente.
- Las simulaciones o visualizaciones proveen un mejor entendimiento del sistema o proceso en estudio.

Introducción

- Internet
- http://www.ma.utexas.edu/CNA/NMC6/sample.html
- http://www.netlib.org/
- http://gams.nist.gov/
- http://cernlib.web.cern.ch/cernlib/
- http://www.gnu.org/software/gsl/
- http://jean-pierre.moreau.pagesperso-orange.fr/
- http://calgo.acm.org/
- http://people.sc.fsu.edu/~jburkardt%20/f_src/praxis/praxis.html

Introducción

- Lenguajes de programación

  Fortran, Pascal, C, C++, Python, f2c, cfortran, f90
- Software: GSL (Gnu Scientific Library)
- Computer Algebra Systems

  Reduce, Mathematica, Maple, Sage
- Simulación y Visualización

  Gnuplot, OpenGL, IDL, DX, GDL, Scilab

Errores

- Error absoluto, relativo y porcentual
- Error relativo: indica que tan buena es la medida con respecto a lo que se está midiendo

$$Error\ \ absoluto = |Valor - Valor\ \ medido|$$

$$Error\ \ relativo = \frac{Error\ \ absoluto}{Valor}$$

$$Error = 100 \times Error\ \ relativo$$

Definición

p' aproxima a p con t dígitos significativos

$$\frac{|p' - p|}{p} < 5 \times 10^{-t}$$

$$p - 5\,p \times 10^{-t} < p' < p + 5\,p \times 10^{-t}$$

Errores en Programas

- Suma de # de distinta magnitud
- Resta de # casi iguales
- Overflow y Underflow
- División por un # pequeño
- Error de discretización o cuantificación (Constantes)
- Errores de salida

Propagación de Errores

- Suma y resta
- Multiplicación y División
- Evaluación de funciones

Propagación de Errores

- Propagación lineal
- Propagación exponencial

$$Para\ n\ operaciones,\ con\ un\ error\ \epsilon\ en\ los\ cálculos$$

$$|\epsilon_n| = n c \epsilon$$

$$|\epsilon_n| = k^n \epsilon$$

$$donde\ c\ no\ depende\ de\ n\ y\ k>1$$

Interpolación

- Lagrange
- Newton
- Lagrange con puntos Chebychev
- Hermite
- Cubic Spline
- Tipos de interpolación: por trozos constantes, lineal, polinomial, spline

Extrapolación

- Exactamente lo mismo que interpolación excepto que se ajusta el polinomio afuera del ámbito

Sistemas Lineales

- Eliminación de Gauß (con pivoteo)

  Gauß-Jordan, Descomposición LU (A=LU)
- Métodos Jordan, Thomas, Doolitle, Crout, Cholesky
- Métodos Iterativos: Jacobi, Gauß-Seidel

  Successive-Over-Relaxation (SOR)

$$A\,X = B$$
$$\textit{donde } A \textit{ es una matriz } n \times n \textit{ y } B \textit{ es un vector}$$
$$X \textit{ es un vector}$$

Sistemas Lineales

- Memoria requerida es proporcional al cuadrado del orden de la matriz A
- Trabajo computacional es proporcional al cubo del orden de la matriz A
- Cuando no se tiene un vector solución inicial, se escoge X = 0
- $A_{ii} \neq 0$

Método de Jacobi

- Convergencia: A es Diagonal dominate

$$x_i^{k+1} = -\frac{1}{a_{ii}}\left[-b_i + \sum_{\underbrace{j=1}_{j\neq i}} a_{ij} x_j^k\right]$$

$$para\ 1 \leq i \leq n$$

$$k\ \ representa\ \ la\ \ iteracion$$

Algoritmo

Choose an initial guess x^0 to the solution
k = 0
check if convergence is reached
while convergence not reached do
for i := 1 step until n do
sigma = 0
for j := 1 step until n do
if $j \neq i$ then
sigma = sigma + a(i,j) x^k_j
end if
end (j-loop)
x^{k+1}_i = (b_i - sigma)/a(i,i)
end (i-loop)
check if convergence is reached
k = k + 1
loop (while convergence condition not reached)

Método de Gauß-Seidel

- Convergencia: A es Diagonal dominate y definida positiva

$$x_i^{k+1} = -\frac{1}{a_{ii}}\left[-b_i + \sum_{j=1}^{i-1} a_{ij} x_j^{k+1} + \sum_{j=i+1}^{n} a_{ij} x_j^{k}\right]$$

$$para\ 1 \leq i \leq n$$

$$k\ \ representa\ \ la\ \ iteracion$$

$$|a_{ii}| > \sum_{i \neq j} |a_{ij}|, \quad z^T A z > 0$$

Algoritmo

```
Inputs: A, b
Output: x

Choose an initial guess x  to the solution
repeat until convergence
    for i from 1 until n do
        sigma = 0
        for j from 1 until n do
            if j ≠ i then
                sigma = sigma + a(i,j) x_j
            end if
        end (j-loop)
       x_i = (b_i - sigma)/a(i,i)
    end (i-loop)
    check if convergence is reached
end (repeat)
```

Autovalores

- Método de Interpolación
- Método de la potencia,
  potencia inversa,
  potencia inversa corrida
- Matriz tridiagonal
- Iteración QR

Ecuaciones no Lineales

- Punto fijo, Newton-Raphson, Secante, Regula Falsi (Posición falsa), Bisección
- Aceleración de convergencia:

  Aitken, Steffensen, Illinois
- Raíces complejas: Müller
- Polinomios: Bairstow, Horner, Lin,

  Jenkins-Traub, Durand-Kerner, Aberth, Graeffe (Dandelin-Graeffe)
- Brent

Punto Fijo

- F(x) = x
- No requiere de un intervalo [a, b]
- Derivada de F(x) continua
- F(x) cualquiera
- Puede no converger

$$\epsilon_n = f'(r)\epsilon_{n-1}$$

Newton-Raphson

- No requiere de un intervalo [a, b]
- Derivada de F(x) continua
- F(x) cualquiera
- Se necesita calcular derivada de F(x)
- Aplicable a raíces complejas

$$\epsilon_n \simeq -\frac{f''(r)}{2f'(r)}\epsilon_{n-1}^2$$

Newton-Raphson

- Serie de Taylor

$$f(x) = f(x_0) + f'(x_0)[x - x_0] + O([x - x_0]^2) = 0$$

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

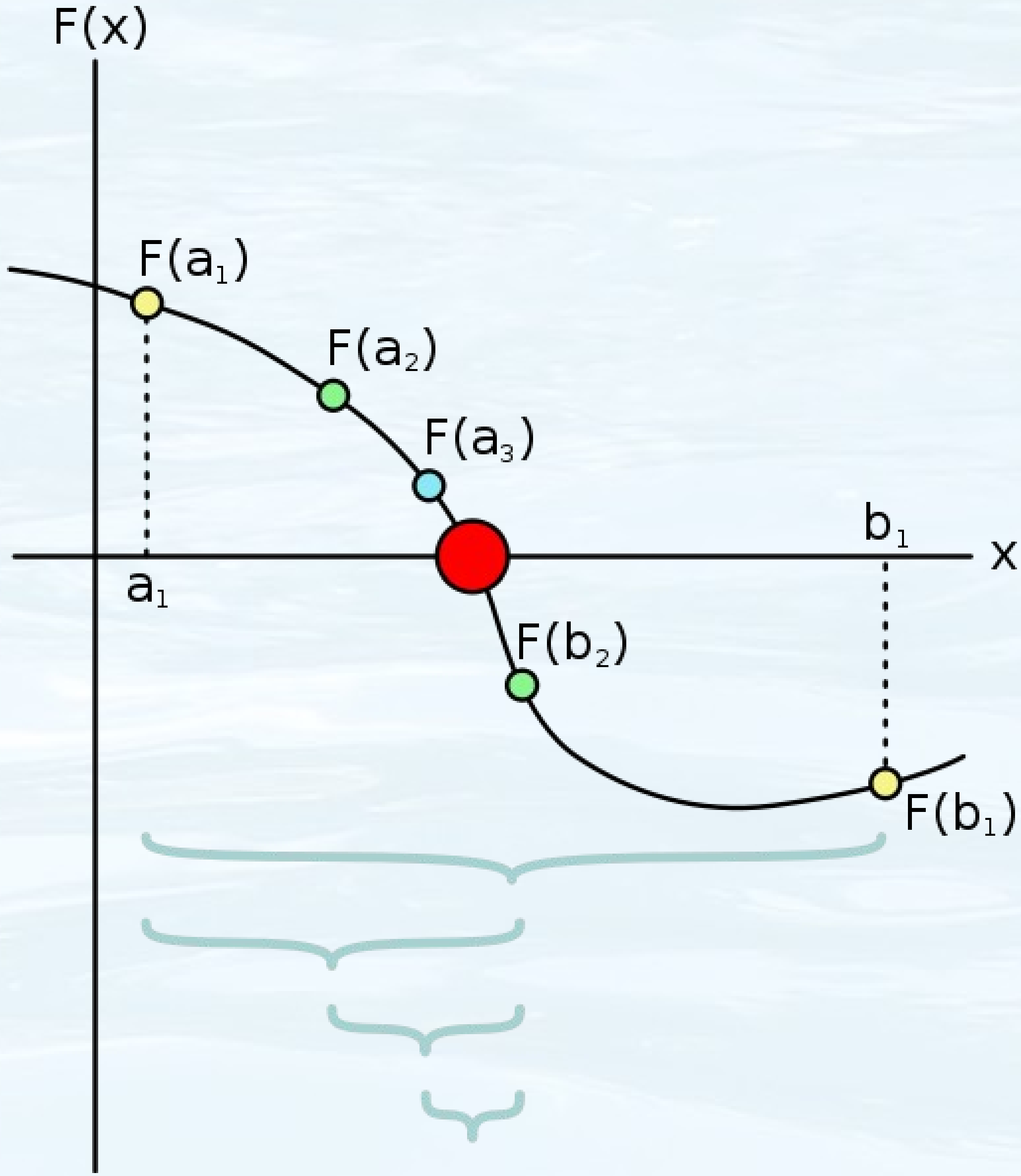
```
%These choices depend on the problem being solved
x0 = 1                          %The initial value
f = x^2 - 2             %The function whose root we are trying to find
fprime =  2*x           %The derivative of f(x)
tolerance = 10^(-7)          %7 digit accuracy is desired
epsilon = 10^(-14)           %Don't want to divide by a number smaller than this

maxIterations = 20          %Don't allow the iterations to continue indefinitely
haveWeFoundSolution = false %Were we able to find the solution to the desired tolerance? not yet.

for i = 1 : maxIterations

    y = f(x0)
    yprime = fprime(x0)

    if(abs(yprime) < epsilon)                          %Don't want to divide by too small of a number
        fprintf('WARNING: denominator is too small\n')
        break;                                         %Leave the loop
    end

    x1 = x0 - y/yprime                                 %Do Newton's computation

    if(abs(x1 - x0)/abs(x1) < tolerance)               %If the result is within the desired tolerance
        haveWeFoundSolution = true
        break;                                         %Done, so leave the loop
    end

    x0 = x1                                            %Update x0 to start the process again

end

if (haveWeFoundSolution) % We found a solution within tolerance and maximum number of iterations
    fprintf('The root is: %f\n', x1);
else %If we weren't able to find a solution to within the desired tolerance
    fprintf("Warning: Not able to find solution to within the desired tolerance of %f\n", tolerance);
    fprintf("The last computed approximate root was %f\n", x1)
end
```

Regula Falsi

- Se requiere de un intervalo [a, b]
- Derivada de F(x) continua
- F(x) cualquiera

Bisección

- Se requiere de un intervalo [a, b]
- Derivada de F(x) no necesariamente continua
- F(x) cualquiera
- Robusto
- Aplicable a funciones no analíticas



F(x)
F(a1)
F(a2)
F(a3)
b1
x
a1
F(b2)
F(b1)

INPUT: Function f, endpoint values a, b, tolerance TOL, maximum iterations NMAX
CONDITIONS: a < b, either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0
OUTPUT: value which differs from a root of f(x)=0 by less than TOL

```
N ← 1
While N ≤ NMAX # limit iterations to prevent infinite loop
  c ← (a + b)/2 # new midpoint
  If f(c) = 0 or (b – a)/2 < TOL then # solution found
    Output(c)
    Stop
  EndIf
  N ← N + 1 # increment step counter
  If sign(f(c)) = sign(f(a)) then a ← c else b ← c # new interval
EndWhile
Output("Method failed.") # max number of steps exceeded
```

Brent

- Complicado
- Combinación de los métodos de bisección, secante e interpolación cuadrática inversa
- Interpolación cuadrática inversa (ICI)

  Basada en la interpolación de Lagrange
- Basado en el algoritmo de Dekker

Interpolación de Lagrange

$$f^{-1}(y)=x_{n+1}=\frac{(y-f_{n-1})(y-f_n)}{(f_{n-2}-f_{n-1})(f_{n-2}-f_n)}x_{n-2}$$

$$+\frac{(y-f_{n-2})(y-f_n)}{(f_{n-1}-f_{n-2})(f_{n-1}-f_n)}x_{n-1}$$

$$+\frac{(y-f_{n-2})(y-f_{n-1})}{(f_n-f_{n-2})(f_n-f_{n-1})}x_n$$

$$donde\ \ x_{n-2},\ x_{n-1},\ x_n,\ f_{n-2},\ f_{n-1},\ f_n\ son\ dados$$

$$Si\ \ y=f(x)=0\rightarrow ICI$$

```
input a, b, and (a pointer to) a function for f
calculate f(a), calculate f(b)
if f(a) f(b) >= 0 then exit,  because the root is not bracketed.
if |f(a)| < |f(b)| then swap (a,b) end if
c := a, set mflag
repeat until f(b or s) = 0 or |b − a| is small enough (convergence)
  if f(a) ≠ f(c) and f(b) ≠ f(c) then (inverse quadratic interpolation)
  else (secant rule)  end if
  if (condition 1) s is not between (3a+b)/4 and b  or
     (condition 2) (mflag is set and |s−b| ≥ |b−c|/2) or
     (condition 3) (mflag is cleared and |s−b| ≥ |c−d|/2) or
     (condition 4) (mflag is set and |b−c| < |δ|) or
     (condition 5) (mflag is cleared and |c−d| < |δ|)
  then  (bisection method) set mflag
  else clear mflag  end if
  calculate f(s)
  d := c  (d is assigned for the first time here; it won't be used above on the first
iteration because mflag is set)
  c := b
  if f(a) f(s) < 0 then b := s else a := s end if
  if |f(a)| < |f(b)| then swap (a,b) end if
end repeat
output b or s (return the root)
```

Solución de sistemas no lineales

- Punto fijo, Newton-Raphson (N-R), N-R modificado, Broyden, Brent, Método del descenso rápido

Derivación

- Desarrollo o expansión de Taylor
- Diferenciación por interpolación polinomial

Derivación (Forward)

$$f'(x) \simeq \frac{f(x+h) - f(x)}{h}$$

$$f'_i = \frac{f_{i+1} - f_i}{h} + O(h)$$

$$f'_i = \frac{-f_{i+2} + 4 f_{i+1} - 3 f_i}{2h} + O(h^2)$$

$$f'_i = \frac{2 f_{i+3} - 9 f_{i+2} + 18 f_{i+1} - 11 f_i}{6h} + O(h^3)$$

Derivación (Backward)

$$f'(x) \simeq \frac{f(x) - f(x-h)}{h}$$

$$f'_i = \frac{f_i - f_{i-1}}{h} + O(h)$$

$$f'_i = \frac{3f_i - 4f_{i-1} + f_{i-2}}{2h} + O(h^2)$$

$$f'_i = \frac{11f_i - 18f_{i-1} + 9f_{i-2} - 2f_{i-3}}{6h} + O(h^3)$$

Derivación (Central)

$$f'(x) \simeq \frac{f(x+h) - f(x-h)}{2\mathrm{h}}$$

$$f'_i = \frac{f_{i+1} - f_{i-1}}{2\mathrm{h}} + O(h^2)$$

$$f'_i = \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{2h} + O(h^4)$$

Integración

- Regla Trapezoidal
- Regla de Simpson, 1/3, 3/8
- Formulas de Newton-Cotes
- Cuadraturas de Gauß
- Double exponential transformation
- Monte Carlo

Ecs. Diferenciales Ordinarias

- Tipos stiff (rígido) y nonstiff
- Euler (backward, forward y modificado)
- Runge-Kutta segundo, tercer y cuarto orden
- Runge-Kutta-Fehlberg
- Predictor Corrector

  (Adams-Moulton, Adams-Bashforth-Moulton, Adams)
- Stiff: Método implícito transformación exponencial

Ecs. Diferenciales Ordinarias

- Exactitud del método Euler es baja
- Exactitud de los métodos de RK se incrementa al usar puntos intermedios en cada etapa o paso del intervalo.

Runge-Kutta Cuarto Orden

$$y(x+h)-y(x) \simeq a k_1 + b k_2 + c k_3 + d k_4$$
$$k_1 = h f(x, y)$$
$$k_2 = h f(x + m h, y + m k_1)$$
$$k_3 = h f(x + n h, y + n k_2)$$
$$k_4 = h f(x + p h, y + p k_3)$$
$$Solucion$$
$$m = n = 1/2,\ \ p = 1,\ \ a = d = 1/6,\ \ b = c = 1/3$$

Ecs. en Derivadas Parciales

- Condiciones de Frontera

  Cauchy (valores en la frontera y sus derivadas)

  Dirichlet (valores en la frontera),

  Neumann (valores de la derivada en la frontera)

- Tipos de ecuaciones

  Elípticas, hiperbólicas, parabólicas,

Ecs. en Derivadas Parciales

$$A(x,y)\frac{\partial^2 U}{\partial x^2}+B(x,y)\frac{\partial^2 U}{\partial x\,\partial y}+C(x,y)\frac{\partial^2 U}{\partial y^2}=F\left(x,\ y,\ U,\ \frac{\partial U}{\partial x},\ \frac{\partial U}{\partial y}\right)$$

$$donde\ \ U=U(x,y)$$

$$Si\ B^2-4AC<0\rightarrow Eliptica\ \ en\ [a,b]$$

$$Si\ B^2-4AC=0\rightarrow Parabolica\ \ en\ [a,b]$$

$$Si\ B^2-4AC>0\rightarrow Hiperbolica\ \ en\ [a,b]$$

Ejemplos

- Ecuación de Laplace → Elíptica
- Ecuación de onda → Hiperbólica
- Ecuación de Difusión → Parabólica

$$\frac{\partial^2 U}{\partial x^2}+\frac{\partial^2 U}{\partial y^2}=0$$

$$\frac{\partial^2 U}{\partial x^2}=\frac{1}{c^2}\frac{\partial^2 U}{\partial t^2}$$

$$\frac{\partial^2 U}{\partial x^2}=\frac{1}{a^2}\frac{\partial U}{\partial t}$$

Dominios o Grillas

- Grilla cuadrada o rectangular
- Sector circular o esférico
- Combinaciones
- Dominios no rectangulares

Discretización

- Diferencias finitas
- Elementos finitos (FEM)

Ecuaciones Elípticas:Laplace

- Diferencias finitas
- Jacobi, Gauß-Seidel, SOR
- 5 puntos

Ecuación de Laplace

$$\frac{\partial f}{\partial x} \simeq \frac{f(x+\Delta x, y) - f(x-\Delta x, y)}{2\Delta x}$$

$$\frac{\partial f}{\partial y} \simeq \frac{f(x, y+\Delta y) - f(x, y-\Delta y)}{2\Delta y}$$

$$\frac{\partial^2 f}{\partial x^2} \simeq \frac{f(x+\Delta x, y) - 2f(x, y) - f(x-\Delta x, y)}{\Delta x^2}$$

$$\frac{\partial^2 f}{\partial y^2} \simeq \frac{f(x, y+\Delta y) - 2f(x, y) + f(x, y-\Delta y)}{\Delta y^2}$$

Ecuación de Laplace

- 5 puntos

$$\frac{\partial f}{\partial x} \simeq \frac{f(i+1,j) - f(i-1,j)}{2\Delta x}$$

$$\frac{\partial f}{\partial y} \simeq \frac{f(i,j+1) - f(i,j-1)}{2\Delta y}$$

$$\frac{\partial^2 f}{\partial x^2} \simeq \frac{f(i+1,j) - 2f(i,j) + f(i-1,j)}{\Delta x^2}$$

$$\frac{\partial^2 f}{\partial y^2} \simeq \frac{f(i,j+1) - 2f(i,j) + f(i,j-1)}{\Delta y^2}$$

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \simeq \frac{f(i+1,j) - 2f(i,j) + f(i-1,j)}{\Delta x^2} + \frac{f(i,j+1) - 2f(i,j) + f(i,j-1)}{\Delta y^2} \text{¿}$$

Ecuaciones Hiperbólicas: Onda

- Diferencias finitas
- FTCS y BTCS
- Métodos Lax, Lax-Wendroff (Richtmyer)
- Método de MacCormack
- Métodos Upwind

Ecuación de Difusión

- Diferencias finitas
- Forward-time centered space (FTCS)
- Backward-time centered space (BTCS)
- Métodos de Richardson y Dufort-Frankel
- Métodos implícitos

  Crank-Nicolson

Ecuación de Difusión

$$\frac{f(n+1,i)-f(n,i)}{\Delta t}=\alpha\frac{f(n,i+1)-2f(n,i)+f(n,i-1)}{\Delta x^2}$$

$$f(n+1,i)=f(n,i)+\beta[f(n,i+1)-2f(n,i)+f(n,i-1)]$$

$$\textit{donde } \beta=\alpha\Delta t/\Delta x^2 \textit{ es el numero de difusion}$$

Ajuste de Curvas

- Regresión lineal
- Polinomial
- Combinación de funciones

Optimización

- Máximos y mínimos
- Globales y locales
- Con o sin restricciones

Optimización

- Método del descenso rápido
- Camino aleatorio
- Simplex (lineal y no lineal)
- Praxis (Brent)
- Simulated Annealing, Pikaia
- Ant Colony, Bee Algorithm
- Particle swarm optimization
- Genéticos

Bibliografía

- Brandt, Datenanalyse, Spektrum, 1999.
- Brent, Algorithms for Minimization without Derivatives, Dover, 2002
- Dennis, Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Siam, 1996.
- Faires, Burden, Numerische Methoden, Spektrum, 1994.
- Engeln-Müllges, Uhlig, Numerical Algorithms with C (Fortran), Springer, 1996.

Bibliografía

- Gould, Tobochnik, An Introduction to Computer Simulation, Addison-Wesley, 1996.
- Hoffman, Numerical Methods for Engineers and Scientists, Marcel Dekker, 2001.
- Künzi, Tzschach, Zehnder, Numerical Methods of Mathematical Optimization,Academic Press, 1968.
- Matsumoto, Omura, Computer Space Plasma Physics, Terra Scientific, 1996.
- McCormick, Salvadori, Numerical Methods in Fortran, Prentice-Hall, 1964.

Bibliografía

- Nakamura, Applied Numerical Methods in C, Prentice-Hall, 1993.
- Nieves, Domínguez, Métodos Numéricos, CECSA, 1998.
- Rayna, REDUCE Software for Algebraic Computation, Springer, 1987.
- Sadiku, Numerical Techniques in Electromagnetics, CRC, 2000.
- Scheid, Numerical Analysis, Schaum, 1988.
- Schwefel, Numerical Optimization of Computer Models, John Wiley & Sons, 1981.

Gracias!