Informe Tarea 4

Ronald Cardona Anderson Grajales
Sebastian Valencia Julian Sanchez

21 de septiembre de 2018

1. Solucion numerica de ecuaciones de una variable

En el presente documento se describen algunos de los algoritmos numericos principales para resolver ecuaciones de una variable. Estos algoritmos nos permiten ya sea encontrar intervalos en los que existe una ra z de la ecuacion $f(x) = 0$ o encontrar propiamente la ra z.

1.1. Busqueda Incremental

Leer x_0, $delta$, $niter$
$fx0$ $f(x_0)$
si $fx0 = 0$ entonces
| x_0 es ra z
en otro caso
| x_1 $x_0 + delta$
| $contador$ 1
| $fx1$ $f(x_1)$
| mientras $fx0$ $fx1 > 0$ y $contador < niter$ hacer
| | x_0 x_1
| | $fx0$ $fx1$
| | x_1 $x_0 + delta$
| | $fx1$ $f(x_1)$
| | $contador$ $contador + 1$
| n
| si $fx1 = 0$ entonces
| | x_1 es ra z
| si no, si $fx0$ $fx1 < 0$ entonces
| | Hay ra z entre x_0 y x_1
| en otro caso
| | Fracaso en $niter$ iteraciones
| n
n

Algoritmo 1: Algoritmo de Busqueda Incremental

1.2. Metodo de la biseccion

Leer x_i, x_s, $tolerancia$, $niter$
$tabla$ [i, x inf, x sup, x med, fxm, Error Abs, Error Rel]
$fxi \quad f(x_i)$
$fxs \quad f(x_s)$
si $fxi = 0$ entonces
x_i es ra z
si no, si $fxs = 0$ entonces
x_s es ra z
si no, si $fxi \quad fxs < 0$ entonces
$x_m \quad \frac{x_i + x_s}{2}$
$fxm = f(x_m)$
$contador \quad 1$
$tabla$ [$contador$, x_i, x_s, x_m, fxm, No existe, No existe]
$error \quad tolerancia + 1$
mientras $error > tolerancia$ y $fxm \neq 0$ y $contador < niter$ hacer
si $fxi \quad fxm < 0$ entonces
$x_s \quad x_m$
$fxs \quad fxm$
en otro caso
$x_i \quad x_m$
$fxi \quad fxm$
n
$x_{aux} \quad x_m$
$x_m \quad \frac{x_i + x_s}{2}$
$fxm \quad f(x_m)$
$error \quad j x_m \quad x_{aux} j$
$errorRel \quad \frac{error}{x_m}$
$tabla$ [$contador$, x_i, x_s, x_m, fxm, $error$, $errorRel$]
$contador \quad contador + 1$
n
si $fxm = 0$ entonces
x_m es ra z
si no, si $error < tolerancia$ entonces
x_m es aproximacion a una ra z con una tolerancia = $tolerancia$
en otro caso
Fracaso en $niter$ iteraciones
n
en otro caso
El intervalo es inadecuado
n
imprimir tabla

Algoritmo 2: Metodo de la Biseccion

1.3. Metodo de la regla falsa

Leer x_i, x_s, *tolerancia*, *niter*
tabla [i, x inf, x sup, x mi, fxm, Error Abs, Error Rel]
$fxi \quad f(x_i)$
$fxs \quad f(x_s)$
si $fxi = 0$ entonces
| x_i es ra z
si no, si $fxs = 0$ entonces
| x_s es ra z
si no, si $fxi \quad fxs < 0$ entonces
| $x_m \quad x_i \quad \frac{f(x_i) \ (x_s \ \ x_i)}{f(x_s) \ \ f(x_i)}$
| $fxm = f(x_m)$
| *contador* 1
| *tabla* [*contador*, x_i, x_s, x_m, fxm, No existe, No existe]
| *error* *tolerancia* + 1
| mientras *error* > *tolerancia* *y* $fxm \neq 0$ *y* *contador* < *niter*
| hacer
| | si $fxi \quad fxm < 0$ entonces
| | | $x_s \quad x_m$
| | | $fxs \quad fxm$
| | en otro caso
| | | $x_i \quad x_m$
| | | $fxi \quad fxm$
| | n
| | $x_{aux} \quad x_m$
| | $x_m \quad x_i \quad \frac{f(x_i) \ (x_s \ \ x_i)}{f(x_s) \ \ f(x_i)}$
| | $fxm \quad f(x_m)$
| | $error \quad jx_m \quad x_{aux}j$
| | $errorRel \quad \frac{error}{x_m}$
| | *tabla* [*contador*, x_i, x_s, x_m, fxm, *error*, *errorRel*]
| | *contador* *contador* + 1
| n
| si $fxm = 0$ entonces
| | x_m es ra z
| si no, si *error* < *tolerancia* entonces
| | x_m es aproximacion a una ra z con una tolerancia = *tolerancia*
| en otro caso
| | Fracaso en *niter* iteraciones
| n
en otro caso
| El intervalo es inadecuado
n
imprimir tabla

Algoritmo 3: Metodo de la Regla Falsa

1.4. Metodo de Punto Fijo

Leer $tolerancia$, x_a, $niter$
$tabla$ [i, xn, fxn, Error Abs, Error Rel]
fx $f(x_a)$
$contador$ 0
$error$ $tolerancia + 1$
$tabla$ [$contador$, x_n, fxn, No existe, No existe]
mientras $fx \neq 0$ y $error > tolerancia$ y $contador < niter$ hacer
| x_n $g(x_a)$
| fx $f(x_n)$
| $error$ jx_n x_aj
| $errorRel$ ($\frac{error}{x_n}$ x_a x_n
| $tabla$ [$contador$, x_n, fxn, $error$, $errorRel$]
| $contador$ $contador + 1$
n
si $fx = 0$ entonces
| x_a es ra z
si no, si $error < tolerancia$ entonces
| x_a es aproximacion con una tolerancia = $tolerancia$
en otro caso
| El metodo fracaso en $niter$ iteraciones
n
imprimir tabla

Algoritmo 4: Metodo de Punto Fijo

1.5. Metodo de Newton

Leer tol, x_0, $niter$
$tabla$ [i, xn, fxn, Error Abs, Error Rel]
fx $f(x_0)$
$dfx = f^0(x_0)$
$contador$ 0
$error$ $tol + 1$
$tabla$ [$contador$, x_i, fxn, No existe, No existe]
mientras $fx \neq 0$ y $dfx \neq 0$ y $error > tol$ y $contador < niter$ hacer
| x_1 x_0 $\frac{fx}{dfx}$
| fx $f(x_1)$
| dfx $f^0(x_1)$
| $error$ jx_1 x_0j
| $errorRel$ $\frac{error}{x_n}$
| x_0 x_1
| $contador$ $contador + 1$
| $tabla$ [$contador$, x_1, fxn, $error$, $errorRel$]
n
si $fx = 0$ entonces
| x_a es ra z
si no, si $error < tol$ entonces
| x_a es aproximacion con una tolerancia = $tolerancia$
si no, si $dfx = 0$ entonces
| x_1 es una posible ra z multiple
en otro caso
| El metodo fracaso en $niter$ iteraciones
n
imprimir tabla

Algoritmo 5: Metodo de Newton

1.6. Metodo de la Secante

Leer x_0, x_1, $niter$, tol
$tabla$ [i, xn, fxn, Error Abs, Error Rel]
fx_0 $f(x_0)$
si $fx_0 = 0$ entonces
| $imprimir$ x_0 es ra z
en otro caso
| fx_1 $f(x_1)$
| $contador = 0$
| $error = tol + 1$
| $denominador$ fx_1 fx_0
| mientras $error > tol$ y $fx_1 \neq 0$ y $denominador \neq 0$ y
| $contador < niter$ hacer
| | x_2 x_1 $\frac{fx_1 (x_1 \; x_0)}{denominador}$
| | $error$ $j\frac{x_2 \; x_1}{x_2}j$
| | $error$ ($\frac{error}{x_2}$
| | x_0 x_1
| | fx_0 fx_1
| | x_1 x_2
| | fx_1 $f(x_1)$
| | $denominador$ fx_1 fx_0
| | $contador$ $contador + 1$
| | $tabla$ [$contador$, x_1, $fx1$, $error$, $errorRel$]
| n
| si $fx_1 = 0$ entonces
| | $imprimir$ x_1 es ra z
| si no, si $error < tol$ entonces
| | $imprimir$ x_1 es una aproximacion con tolerancia = tol
| si no, si $denominador = 0$ entonces
| | $imprimir$ probablemente existe una raiz multiple
| en otro caso
| | $imprimir$ fracaso en $niter$ iteraciones
| n
n
imprimir tabla

Algoritmo 6: Metodo de la Secante

1.7. Metodo de Raices Multiples

Leer x_0, *niter*, *tol*

tabla ([i, xn, fx, dfx, ddfx, Error Abs, Error Rel]

$fx \quad f(x_0)$

$dfx \quad f'(x_0)$

$ddfx \quad f''(x_0)$

$denominador \quad dfx^2 \quad fx \quad ddfx$

$contador \quad 0$

tabla ([$contador$, x_n, fx, dfx, $ddfx$, No existe, No existe]

$error \quad tol + 1$

mientras $error > tol$ *y* $fx \neq 0$ *y* $denominador \neq 0$ *y* $contador < niter$ hacer

$x_1 \quad x_0 \quad \frac{fx \; dfx}{denominador}$

$fx \quad f(x_1)$

$dfx \quad f'(x_1)$

$ddfx \quad f''(x_1)$

$denominador \quad dfx^2 \quad fx \quad ddfx$

$error \quad |x_1 \quad x_0|$

$x_0 \quad x_1$

$errorRel \quad \frac{error}{x_0}$

$contador \quad contador + 1$

tabla ([$contador$, x_n, fx, dfx, $ddfx$, $error$, $errorRel$]

n

si $fx = 0$ entonces

imprimir x_0 es ra z

si no, si $error < tol$ entonces

imprimir x_0 es una aproximacion con tolerancia = *tol*

en otro caso

imprimir fracaso en *niter* iteraciones

n

imprimir tabla

Algoritmo 7: Metodo de Raices Multiples

2. Metodos de optimizacion para la solucion numerica de ecuaciones de una variable

En algunos casos es necesario optimizar algunos metodos numericos con la nalidad de alcanzar una tolerancia en el menor numero de iteraciones posible. Es por esto, que aca se presenta un metodo de optimizacion a el metodo de Punto Fijo, que hace que la convergencia sea mucho mas rapida y exacta.

2.1. Metodo de Ste ensen

// Tomado de:
https://en.wikipedia.org/wiki/Steffensen%27s_method
Leer x_0, *niter*, *tol*
tabla ([i, x0, x1, x2, x, fx, Error]
error = 1+tol
contador = 0
tabla = [*contador*, x_0, x_1, x_2, x, $f_{(x)}$, No existe, No existe] **mientras** *contador* < *niter* *y* *error* > *tol* **hacer**
| $x_1 = f(x_0)$
| $x_2 = f(x_1)$
| $f(x) = f(x_0)$
| $p = x_0 \quad \frac{(x_1 \quad x_2)^2}{x_2 \quad 2\; x_1 + x_0}$
| $error = jx \quad x_0 j$
| $x_0 = p$
| *contador* = *contador* + 1
| *tabla* = [*contador*, x_0, x_1, x_2, x, $f_{(x)}$, *error*]
n
si *error* *tol* **entonces**
| Hay una ra z en p
en otro caso
| El metodo fracaso despues de *niter* iteraciones
n
imprimir tabla

Algoritmo 8: Metodo de Ste ensen

2.2. Escalonamiento de matrices

Sea $A = [a_{ij}]$ una matriz de $n \times n$. Decimos que A esta escalonada si $\forall i; j; 1 \quad i \quad j \quad n$, $a_{ij} = 0$.

```
Leer A, b
si A ∉ ℜ^{n×n} o b ∉ ℜ^n entonces
  | A debe ser cuadrada y b debe ser un arreglo de n posiciones
si no, si det A = 0 entonces
  | A debe ser invertible
en otro caso
  | para k = 1 a n − 1 hacer
  |   | si A_kk = 0 entonces
  |   |   | j ← k + 1
  |   |   | mientras j < n y A_jk = 0 hacer
  |   |   |   | j ← j + 1
  |   |   | n
  |   |   | si j < n entonces
  |   |   |   | para l = k a n hacer
  |   |   |   |   | A_kl ← A_kl + A_jl
  |   |   |   | n
  |   |   |   | b_k ← b_k + b_j
  |   |   | n
  |   | n
  |   | para i = k + 1 a n hacer
  |   |   | si A_ki ≠ 0 entonces
  |   |   |   | m ← A_ik / A_kk
  |   |   |   | para l = k a n hacer
  |   |   |   |   | A_il ← A_il − m × A_kl
  |   |   |   | n
  |   |   |   | b_i ← b_i − m × b_k
  |   |   | n
  |   | n
  | n
n
La solucion (A; b)
```

Algoritmo 9: Algoritmo para escalonar matrices