Ejercicios - Taller 1

Diego Alejandro Gómez Parra
diego.gomezp@javeriana.edu.co

Luis Manuel Peñaranda Ramirez
penaranda-lm@javeriana.edu.co

Camilo Andrés Moreno Colorado
camilomoreno@javeriana.edu.co

15 de febrero de 2020

1. Numero de operaciones

1.1. Marco teórico

Las operaciones fundamentales de la aritmética de valores reales, son la suma + y el producto . Estas también, son las operaciones necesarias para evaluar un polinomio P(x) en un valor particular x. Es por esto, que es importante saber cómo se evalúa un polinomio y más aun hacerlo de manera eficiente, para que el número de sumas y productos requeridas sea mínima.

1.2. Utilice el método de inducción matemática para demostrar el resultado del método 3.

- Sea $P_0(x) = a_0x^0 = a_0$. El numero de multiplicaciones para calcular $P_0(x)$ es cero. Por esto el caso $k = 0$ se cumple.
- Se asume que $P_k(x) = a_0 + a_1x + ... + a_kx^k$ y que $P_k(x_0) = a_0 + a_1x_0 + ... + a_kx^0$ tiene un total de k multiplicaciones.
- Se asume que el método Horner se expresa de la siguiente manera:

$$b_k = a_k$$
$$b_k = a_{k-1} + b_k * x_0$$
$$b_0 = a_0 + b_1 * x_0$$

- Se debe llegar a la forma k+1 del polinomio:

$$P_{k+1}(x_0) = a_0 + x(a_1 + x(a_2 + ... + x(a_k + xa_{k+1})))$$

- Se reemplaza a_k por b_k. Esta es la primera instrucción del método de Horner:

$$P_{k+1}(x_0) = a_0 + x_0(a_1 + x(a_2 + ... + x(a_k + b_k)))$$

- También se añade la iteración en el método de Horner para b_{k+1}:

$$b_k = a_k$$
$$b_k = a_{k-1} + b_{k+1} * x_0$$
$$b_0 = a_0 + b_1 * x_0$$

- Se reemplaza b_k en $P_k(x_0)$:

$$P_{k+1}(x_0) = a_0 + x_0(a_{k+1} + x_0(a_{k-1} + x_0(a_k + b_{k+1} * x_0))$$

  Lo anterior es equivalente a $P_{k+1}(x_0)$, donde se demuestra que el numero de multiplicaciones es k el cual es el grado del polinomio.

1.3. Implemente en R o Python para verificar los resultados del método de Horner

Para verificar el resultado del método Horner se utiliza la misma implementación realizada en la parte de problemas de este mismo taller.

Entrada:

- coeficiente = 2, 0, -3, 3, -4. coeficientes del polinomio.
- $x_0 = -2$. Valor evaluado.

Salida:

- Resultado = 10. El numero mínimo de operaciones es 8, y se de 4 sumas y 4 multiplicaciones.

```
horner <- function(coeficientes, x){
  y <- coeficientes[1]
  i <-0

  for(k in coeficientes[2:length(coeficientes)]){
    y <- x*y + k
    i <- i + 2
  }
  return(cat("El numero de operaciones realizadas fueron de: ", i, " siendo ", i/2 , "el
      numero de multiplicaciones y ", i/2 , "el numero de sumas realizadas"))
}
x0 <- -2
coeficientes <- c(2,0,-3,3,-4)
horner(coeficientes,x0)
```

1.4. Evaluar en $x = 1{,}0001$ con $P(x) = 1 + x + x^2 + ... + x^{50}$. Encuentre el error de calculo al compararlo con la expresión equivalente $Q(x) = (x^{51} - 1)/(x - 1)$

Entrada:

- $x = 1{,}0001$. Valor a evaluar.
- $Q(x) = (x^{51} - 1)/(x - 1)$. expresión equivalente.

Salida:

- Resultado 1 = 51.1277085003.
- Resultado 2 = 51.1277085001. Usando la expresión equivalente.

1.5. Números binarios

1.5.1. Ejercicios

1.5.1.a. Encuentre los primeros 15 bits en la representación binaria de π

Entrada:

- π.

Salida:

- 11 . 10110000111. Numero en binario de π

1.5.1.b. Convertir los siguientes números binarios a base 10: 1010101; 1011.101; 10111.010101...; 111.1111...

Entrada:

- 10
- 1010101
- 1011.101
- 10111.010101
- 111.1111

Salida:

- 2
- 85
- 11 . 5
- 23 . 21
- 7 . 15

1.5.1.c. Convierta los siguientes numeros de base 10 a binaria: 11.25; 2/3; 30.6; 99.9

Entrada:

- 11.25
- 2/3
- 30.6
- 99.9

Salida:

- 1011 . 11001
- 0. 1000010
- 11110 . 110
- 1100011 . 1001

1.6. Representación del Punto Flotante de los Números Reales

1.6.1. ejercicios

1.6.1.1. ¿ Cómo se ajusta un número binario infinito en un numero finito de bits?

Para ajustar un número binario infinito, se utiliza el método de truncamiento o el método de redondeo para representar el valor en un número finito de bits. Mediante estas técnicas se pretende representar un valor infinito en una aproximación finita de bits.

1.6.1.2. ¿Cuál es la diferencia entre redondeo y recorte?

En el método de truncamiento se eliminan los bits que van después de una posición determinada. Mientras que, en el método de redondeo, se aproxima hacia arriba o hacia abajo el bit en una posición determinada en función del bit siguiente. Es decir, si el último bit eliminado es 1 entonces se aproxima hacia arriba, en caso contrario se aproxima hacia abajo.

1.6.1.3. Indique el número de punto flotante (IEEE) de precisión doble asociado a x, el cual se denota como $fl(x)$; para x(0,4)

Lo primero que debe realizarse para resolver esta pregunta es pasar de entero a binario:

$$(0{,}4)_{10} = (100110011001100\ldots)_b$$

ó

$$(1, 100110011001100\ldots)_b X 10^{-1}$$

Luego es posible definir que el exponente es '-1', por lo tanto, la característica es 1023 - 1 = 1022 lo que equivale a:

$(1022)_{10}$ ó (1111111110) en binario

La mantisa corresponde a los 52 primeros dígitos después de la ',' dado que hay infinitos dígitos se emplea el método de redondeo de inserte aquí método obteniendo:

$$Mantisa = 1001100110011001100110011001100110011001100110011001100110010$$

Se puede observar en los dos últimos dígitos como fueron redondeados haca arriba. Al decir que el bit es par debido a que el signo es 0 dado que el valor es positivo. El número 0.4 en el estándar 754-IEEE corresponde a:

$$fl(x) = (0111111111010011001100110011001100110011001100110011001100110011010)_b$$

Y en base decimal seria:

$$fl(0{,}4) = (0{,}4000000000000000022204460492503130808472633361816406 25)_{10}$$

1.6.1.4. Error de redondeo En el modelo de la aritmética de computadora IEEE, el error de redondeo relativo no es mas de la mitad del épsilon de máquina, encuentre el error de redondeo para x = 0.4:

El error relativo corresponde a:

$$e = \frac{!fl(0{,}4) - 0{,}4!}{|0{,}4|} = 5{,}551115X10^{-17}$$

El cual es menor que la mitad del error de maquina:

$$e_{maq} = \frac{1}{2} * 2^{-52} = 1{,}110223X10^{-16}$$

1.6.1.5. Verificar si el tipo de datos básico de R y Python es de precisión doble IEEE y Revisar en R y Phython el format long

Las máquinas actuales usan aritmética de punto flotante IEEE-754 para el mapeo de los datos se usa "doble precisión" de IEEE-754 lo que significa convertir 0.1 por ejemplo a la fracción más cercana a J/2***N* donde J es un entero que contiene exactamente 53 bits.

1.6.1.6. Encuentre la representación en número de máquina hexadecimal del número real 9.4

Para cambiar el número '9.4' de decimal a hexadecimal se usa la función

"as.hexmode(numero)"

la cual se usa reemplazando el número por 9.4 dando un resultado igual a (9.4)hex.

1.6.1.7. Encuentre las dos raíces de la ecuación cuadrática $x^2+9^{12}x = 3$ Intente resolver el problema usando la aritmética de precisión doble, tenga en cuenta la pérdida de significancia y debe contrarestarla

La respuesta obtenida después del uso de la herramienta wolfram alpha se obtuvo dos raíces:

$$x = \frac{6}{282429536481+\sqrt{443076872509863373}}$$

$$x = \frac{-28242953648}{2} - \frac{\sqrt{7976644307687250986337 3}}{2}$$

Al comparar ambos resultados obtenidos con precisión doble (debido al uso de wolfram) los resultados al restarlos serían:

$$-2{,}8242953648100000000000000000000000000000000000000000x10^{11}$$

Lo que indica la vasta diferencia entre ambos puntos y ratifica su lugar como respuestas independientes.

2. Raíces de una ecuación

2.1. Marco teorico

La solución de ecuaciones no lineales o sistemas es un problema importante en el cálculo científico y la solución de ecuaciones es un tema central en análisis numérico. Existen métodos iterativos y métodos directos para solucionar el problema $f(x) = 0$.

- Métodos iterativos: Consiste en acercarse a la solución mediante aproximaciones sucesivas, a partir de un valor inicial estimado. En cada iteración se puede usar el resultado anterior para obtener una mejor aproximación.

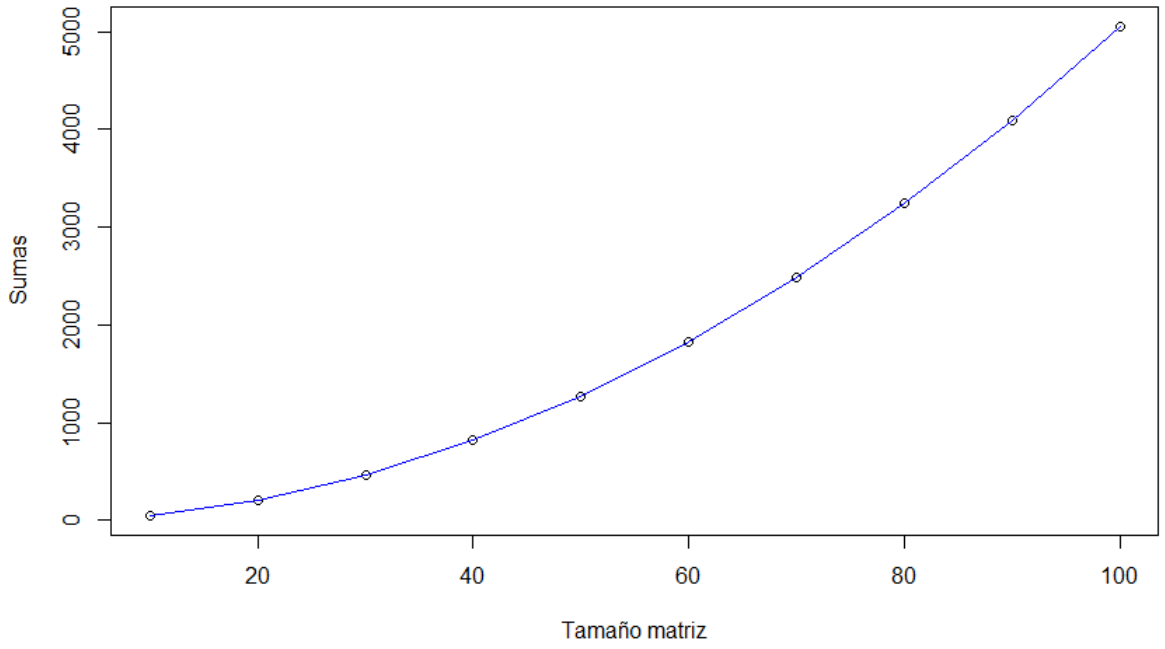
- Métodos directos: La aproximación a la respuesta se produce a través de una serie finita de operaciones aritméticas y la eficiencia del método depende de la cantidad de operaciones el cual se puede asociar al tamaño del problema, en notación O() .

2.2. Implemente en R o Python un algoritmo que le permita sumar únicamente los elementos de la sub matriz triangular superior o triangular inferior, dada la matriz cuadrada A_n. Imprima varias pruebas, para diferentes valores de n y exprese f(n) en notación O() con una gráfica que muestre su orden de convergencia.

Entrada:

- $x = (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)$. Tamaños de diferentes matrices a evaluar.

Salida:



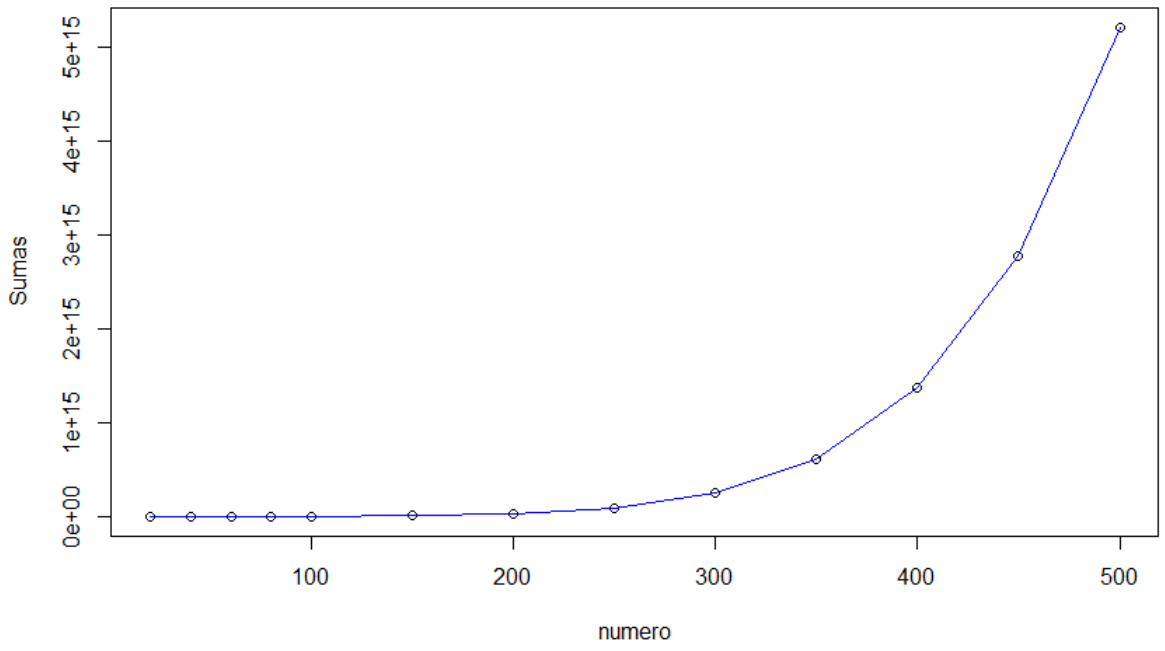
Gráfica 1: Gráfica convergencia de suma sub matriz triangular.

2.3. Implemente en R o Python un algoritmo que le permita sumar los n^2 primeros números naturales al cuadrado. Imprima varias pruebas, para diferentes valores de n y exprese f(n) en notación O() con una gráfica que muestre su orden de convergencia.

Entrada:

- $x = (20, 40, 60, 80, 100, 150, 200, 250, 300, 350, 400, 450, 500)$. Números naturales a evaluar.

Salida:



Gráfica 2: Gráfica suma de los n^2 primeros números naturales al cuadrado.

2.4. Para describir la trayectoria de un cohete se tiene el modelo: $y(t) = 6 + 2{,}13t^2 - 0{,}0013t^4$ Donde, y es la altura en [m] y t tiempo en [s]. El cohete esta colocado verticalmente sobre la tierra. Utilizando dos métodos de solución de ecuación no lineal, encuentre la altura máxima que alcanza el cohete.

Se utiliza un algoritmo donde se va evaluando t y t+1, hasta que el primero sea mayor que el segundo, en ese momento, se calcula el punto mas alto en la ecuación.

Entrada:

- $y(t) = 6 + 2{,}13t^2 - 0{,}0013t^4$. La ecuación a evaluar.

Salida:

- $m = 877{,}8647$. Altura máxima en metros.

3. Convergencia de Métodos Iterativos

3.1. Marco teórico

Este capítulo se dedica a investigar el orden de convergencia de los esquemas de iteración funcional y, con la idea de obtener una convergencia rápida, reutilizar el método por ejemplo, de Newton. Consideraremos también maneras de acelerar la convergencia del método de Newton en circunstancias especiales. Pero, antes, tenemos que definir un procedimiento para medir la rapidez de la convergencia.

3.2. Ejercicios

Para cada uno de los siguientes ejercicios implemente en R o Python, debe determinar el número de iteraciones realizadas, una gráfica que evidencie el tipo de convergencia del método y debe expresarla en notación O().

3.2.1. Sea $f(x) = e^x - x - 1$

1. Demuestre que Tiene un cero de multiplicidad 2 en x = 0

2. Utilizando el método de Newton con p0 = 1 verifique que converge a cero pero no de forma cuadrática

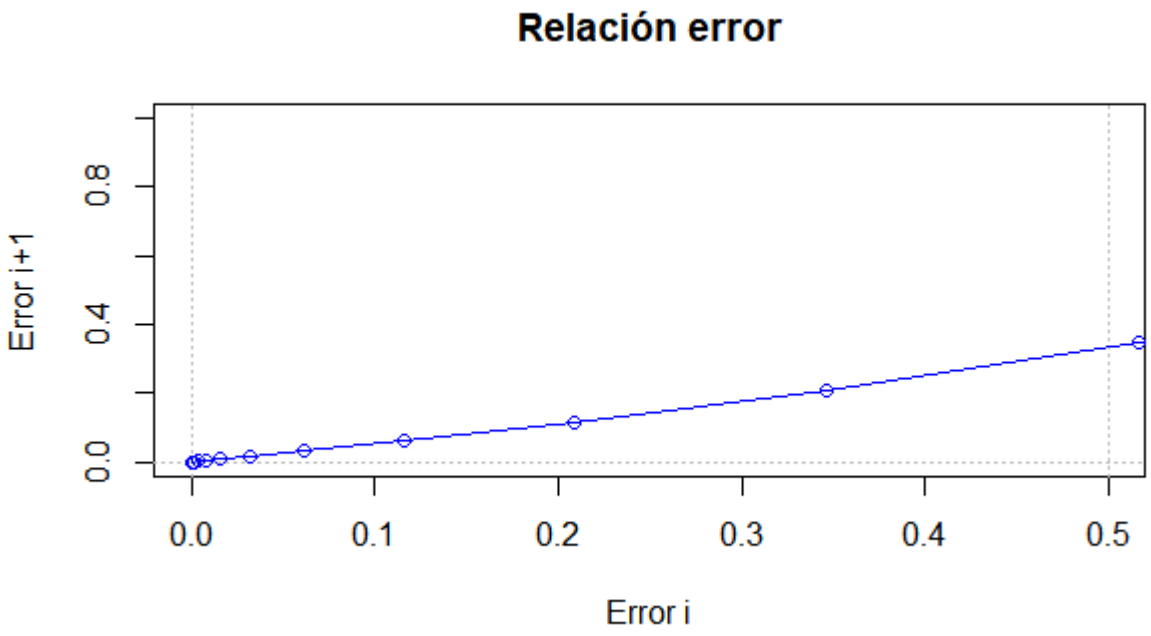
3. Utilizando el método de Newton generalizado, mejora la tasa de rendimiento

Entrada:

- Funcion $f(x) = e^x - x - 1$
- Derivada $f'(x) = e^x - 1$
- Valor inicial $x_0 = 0, x_2 = 0$
- Tolerancia $1x10^{-8}$

Salida:

- Raices: 1.848537046e-08 y 1.63852842



Gráfica 3: Gráfica del método Newton.

x_k	$f(x_k)$	Error est.	Error ant.
1.313035285499331	1.404404811735363	0.686964714500669	0.000000000000000
0.796223594538966	0.420928638564356	0.516811690960366	0.686964714500669
0.450392870069002	0.118535579385603	0.345830724469964	0.516811690960366
0.242044035520060	0.031806250720908	0.208348834548941	0.345830724469964
0.125899366995155	0.008268660808180	0.116144668524905	0.208348834548941
0.064270222231248	0.002110297323536	0.061629144763907	0.116144668524905
0.032479309209032	0.000533209867719	0.031790913022216	0.061629144763907
0.016327561852855	0.000134023067785	0.016151747356177	0.031790913022216
0.008185996600725	0.000033596882257	0.008141565252130	0.016151747356177
0.004098582505826	0.000008410675969	0.004087414094899	0.008141565252130
0.002050691117366	0.000002104105074	0.002047891388460	0.004087414094899
0.001025696003140	0.000000526206039	0.001024995114226	0.002047891388460
0.000512935672600	0.000000131573997	0.000512760330540	0.001024995114226
0.000256489761567	0.000000032896311	0.000256445911033	0.000512760330540
0.000128250363524	0.000000008224429	0.000128239398043	0.000256445911033
0.000064126553251	0.000000002056151	0.000064123810273	0.000128239398043
0.000032063617683	0.000000000514043	0.000032062935568	0.000064123810273
0.000016031895484	0.000000000128511	0.000016031722199	0.000032062935568
0.000008015975363	0.000000000032128	0.000008015920121	0.000016031722199
0.000004008013049	0.000000000008032	0.000004007962314	0.000008015920121
0.000002004026709	0.000000000002008	0.000002003986339	0.000004007962314
0.000001002070328	0.000000000000502	0.000001001956382	0.000002003986339
0.000000501064973	0.000000000000126	0.000000501005355	0.000001001956382
0.000000250244777	0.000000000000031	0.000000250820196	0.000000501005355
0.000000125134132	0.000000000000008	0.000000125110645	0.000000250820196
0.000000061253836	0.000000000000002	0.000000063880295	0.000000125110645
0.000000032253909	0.000000000000000	0.000000028999927	0.000000063880295
0.000000018485370	0.000000000000000	0.000000013768539	0.000000028999927
0.000000018485370	0.000000000000000	0.000000000000000	0.000000013768539

Cuadro 1: Proceso del metodo Newton

4. Convergencia Acelerada

4.1. Metodo de Aitken

Técnica que se usa para acelerar la convergencia de cualquier sucesión que converja linealmente, independientemente de su origen.

4.2. Metodo de Steffersen

Aplicando el método de Aitken a una sucesión que converge linealmente obtenida de la iteración de punto fijo, podemos acelerar la convergencia a cuadrática. Este procedimiento es conocido como el método de Steffersen y difiere un poco de aplicar el de Aitken directamente a una sucesión de iteración de punto fijo que sea linealmente convergente.

4.2.1. Utilice el algoritmo de Steffersen para resolver $x^2 - cosx$ y compararlo con el metodo de Aitken

Entrada:

- $f(x) = x^2 - cosx$
- Valores iniciales $x_0 = 0, x_0 = 1$
- Tolerancia $1x10^{-8}$

Salida:

- Resultado sin aceleracion: 0.8241329
- Resultado con aceleracion: 0.8241374
- Valor real de la solucion: 0.8241323123