Métodos Numéricos

http://www.famaf.unc.edu.ar/∼serra/metodos_numericos.html

Guía 1

Marzo de 2015

Problema 1: En una terminal, cree un directorio nuevo. Descargue desde el navegador el archivo de esta guía y mueva el mismo al nuevo directorio. Cambie el nombre del archivo a *guia1.pdf* y copie el archivo a otro con nombre *Copia.pdf*. Luego, liste el contenido del directorio y verifique los nombres y el tamaño y fecha de los mismos. Averigüe qué espacio ocupa el directorio usando el comando *du -h*. Finalmente, borre todos los archivos y el directorio.

Problema 2: Utilice el programa *gnuplot* para graficar la función $f(x) = \sin(2\pi x)$, para $x \in [0 : 5]$. Ajuste la cantidad de puntos que usa gnuplot para graficar (*samples*) a 500 para mejorar la calidad del gráfico. De nombre a los ejes, título al gráfico y defina la leyenda para que aparezca $f(x)$.

Problema 3: Abra una terminal y realice las siguientes acciones:

a) Cree un directorio con el nombre *MetNum*

b) En ese nuevo directorio ejecute el editor *kate* liberando la línea de comandos (utilice el símbolo & o suspenda el proceso con *Ctrl + Z* y luego ingrese *bg*).

c) En el editor de texto ingrese los siguientes datos en tres columnas, separados por dos espacios:

0.0	1.0	1.2
1.0	1.5	1.6
2.0	2.0	2.0
3.0	2.5	2.4

Grabe esta tabla en el archivo *datos.dat*. Asegúrese que esté en el directorio *MetNum*. No cierre el editor.

d) Ahora utilice *gnuplot* para graficar la 2da y la 3er columna versus la primera. Grafique usando solo puntos. En el mismo gráfico incluya las funciones $y = 0.5x + 1$ y $y = 0.4x + 1.2$. Nombre al eje horizontal x y al vertical y. Habilite la grilla. Titule el gráfico *Funciones lineales*. Cree archivos de salida en formato postscript color y blanco y negro (elija nombres con terminación .ps). Abra otra terminal y con la aplicación *okular* visualice los archivos de salida, a medida que los va generando). Lluego, experimente graficar cambiando el tamaño de los puntos y el grosor de las líneas, etc. Modique el archivo en la ventana de *kate* (que está abierta) y actualice el gráfico para constatar los cambios. Finalmente, cierre *gnuplot*.

e) Realice un gráfico similar usando la aplicación *xmgrace*. Explore las distintas alternativas que ofrece esta apliación.

f) Liste los archivos del directorio mostrando su tamaño y fecha. Averigüe qué espacio ocupa el directorio *MetNum*. Finalmente, cierre la ventana del editor. Borre los archivos postscript y cierre la terminal.

Problema 4: Evaluar (en papel y lápiz) las siguientes expresiones, anticipando el resultado de estas operaciones en Fortran 90. Verifique programando las mismas en Fortran 90.

a) `5 / 2 + 20/ 6`

b) `4 * 6 / 2 - 15 / 2`

c) `5 * 15 / 2 / (4 - 2)`

d) `1 + 1/4`

e) `1.  + 1/4`

f) `1 + 1./4`

g) `1.  + 1./4.`

Problema 5: Evaluar (en papel y lápiz) las siguientes operaciones usando matemática exacta y *matemática de enteros de 2 bytes* (Integer(2)). Verifique sus resultados escribiendo un código en Fortran 90.

a) `32767 + 1`

b) `30000*2`

c) `-30000 - 10000`

Problema 6: Escriba un programa que comprueba las reglas de la matemática de no detención con $\pm$`Infinity` , ± 0 y `NaN`. Luego, compile el mismo programa con el flag `-ffpe-trap=zero,invalid,overflow` y compare los resultados.

Problema 7: De ejemplos para mostrar que la matemática de punto flotante

a) no es cerrada respecto a la suma ni a la multiplicación,

b) no es asociativa respecto a la suma ni a la multiplicación

c) la multiplicación no es distributiva respecto a la suma.

Haga el programa Fortran correspondiente.

Problema 8: Escriba el siguiente programa y explique por qué los valores obtenidos no son iguales:

```
program test
implicit none
real(4) :: sum0,sum1
integer(4) :: i
!
sum0 = 0. ; sum1 = 1.
do i=1,10000
   sum0 = sum0 + 1.e-8
   sum1 = sum1 + 1.e-8
end do
sum0 = sum0 + 1.
write(*,*) sum0, sum1
end program test
```

Problema 9: Escriba un programa que calcule ϵ_m de su máquina, en REAL(4) y REAL(8).

Problema 10: Escriba un programa que pida dos números reales e imprima en la pantalla el mayor de ellos. El programa debe indicar si los números son iguales.

Problema 11: Escriba un programa que pida un número entero y determine si es múltiplo de 2 y de 5.

Problema 12: Escriba una programa que ingrese los coeficientes A, B y C de un polinomio real de segundo grado $(Ax^2 + Bx + C)$, calcule e imprima en pantalla las dos raíces del polinomio en formato complejo $a + ib$, sin utilizar algebra compleja.

Problema 13: Escriba un programa que permita convertir números naturales con base 10 a la base $b <= 16$. El programa debe pedir como entrada b y el número natural a convertir (que debe estar en base 10).

Problema 14: Se pretende calcular las sumas $S_N = \sum_{k=1}^{N} a_k$, donde N es un número natural. Llamemos S'_N al valor calculado que se logra de hacer `(float)`$(S_{N-1} + a_N)$. Sea $S_N = \sum_{k=1}^{N} 1/k$. Mostrar que S'_N se estaciona a partir de algún N suficientemente grande. Deducir que a partir de entonces $S_N \neq S'_N$. Hacer un programa que determine el valor a partir del cual S'_N se estaciona.

Problema 15: Escriba un programa para calcular un valor aproximado de π utilizando la productoria de Wallis

$$\frac{\pi}{2} = \prod_{n=1}^{\infty} \frac{(2n)^2}{(2n)^2 - 1} = \frac{2}{1}\frac{2}{3}\frac{4}{3}\frac{4}{5}\frac{6}{5}\frac{6}{7}\cdots$$

Calcule el valor de π truncando la productoria a 10^6 factores.

Problema 16: Modifique el programa que convierte numeros decimales a la base b, con $b \leq 16$, de manera que utilice `SELECT CASE` para asignar los dígitos, permitiendo así escribir números en base mayor o igual a 10. Utilice `CASE DEFAULT`.

Problema 17: Dado el arreglo `a`, declarado de la siguiente manera

```
real (kind(1.0)), dimension(50,20) :: a
```

escriba *secciones* de este arreglo representando:

a) la primera fila de `a`;

b) la última columna de `a`;

c) un elemento de por medio en cada fila y columna.

Problema 18: Escriba un programa, que utilizando una subrutina, multiplique un vector de N elementos, por una matriz de $N \times N$. El programa debe preguntar el valor de N y luego definir los arreglos, y darle valores iniciales tal que, la matriz sea triangular superior, con todos sus elementos igual a 1, excepto los de la diagonal que toman valor 3, el vector tendra todos sus elementos pares igual a 2, y los impares igual a 3. No utilice `DO` para las inicializar el vector, ni `DO` anidados para inicializar la matriz.

Problema 19: Dado el siguiente programa y su correspondiente subrutina:

```
program test
implicit none
real(kind(1.)) :: x, y,s
integer :: i,j
i = 6; j = 3; x = 4.; y = 8.
call sum(i,x,s)
write(*,'(G4.1," + ",G4.1," = ",F5.1)') i,x,s
end program test
```

```
subroutine sum(z,w,ss)
implicit none
real(kind(1.)), INTENT(IN) :: z, w
real(kind(1.)), INTENT(OUT) :: ss
ss = z + w
end subroutine sum
```

Escriba, compile y ejecute el programa de las siguiente manera:

a) con el programa tal como esta, incluyendo la subrutina en el mismo archivo o en archivo separado.

b) modificando el programa con la cláusula `CONTAINS`, para que *contenga* a la subrutina.

c) creando un modulo que contenga la subrutina, y usando este en el programa.

Verifique que en el primer caso compila sin errores y produce *resultados incorrectos*, y en los otros casos no.

Problema 20: Escriba un programa para calcular la posición y la velocidad en función del tiempo, para un problema de tiro oblicuo. Debe preguntar el ángulo (en grados centígrados) y la velocidad inicial (en m/seg.), asumiendo que el proyectil parte del origen. Elija el incremento temporal (Δt) de manera que la gráfica tenga 600 puntos y abarque el intervalo entre el disparo y el instante en que el proyectil vuelve a tener altura 0. Utilice *funciones* para calcular la posición y la velocidad. Escriba la salida del programa en un archivo de texto, con 5 columnas (t, $x(t)$, $y(t)$,$v_x(t)$,$v_y(t)$). La primera línea debe comenzar con #, e incluir la descripción de los datos de cada columna. Los datos en la tabla deben tener 6 cifras significativas y estar escritos en notación exponencial. Grafique $x(t)$, $y(t)$ y $v_y(t)$ en función de t, y la trayectoria del proyectil, utilizando *gnuplot* y *xmgrace*.

Ejercicios Complementarios

Problema 21: Para calcular un valor aproximado de π utilizaremos la siguiente serie infinita alternante

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4} \tag{1}$$

Recordemos que una cota superior para el error cometido al truncar una serie alternante (de valor absoluto decreciente) está dado por el valor absoluto del primer término despreciado. Escriba un programa que ingrese el número de cifras decimales exactas con que se desea el valor de π (entre 1 y 5 cifras) y devuelva en pantalla el número de térrminos que deben incluirse en la serie (1) para obtener dicha precisión y a renglón siguiente el valor de π obtenido de esta forma, truncado el resultado al número de cifras pedido.

Problema 22: Escribir un programa que pida una contraseña de tres dígitos y permita leer tres intentos. Si el usuario da la contraseña correcta responde responde "Correcto" y queda inactivo, con este mensaje. En caso contrario el programa escribe "Lo siento, contraseña equivocada" y se cierra de inmediato.

Problema 23: Escribir un programa que, dado un año y el nombre de un mes, saque por pantalla el número de días del mes (tenga en cuenta que algunos años son bisiestos).

Problema 24: Escriba un programa que genere secuencialmente 10 archivos con nombre de salida diferentes (dependiendo del valor que tome algún parámetro). En cada archivo, escriba bajo la forma de dupla $(x, f(x))$ una función evaluada en N puntos y que también dependa del parámetro (por ejemplo $y = sin(\omega\pi x)$, con $\omega = 1, 2, 3, \ldots$). El *loop* debe cerrar cada archivo luego de escribir en él. En el mismo programa o en otro, construya otro *loop* que abra secuencialmente los archivos y que (sin borrar los datos escritos) agregue otros N puntos de la función $(x, f(x))$.