.

UNIVERSIDAD NACIONAL AGRARIA
DPTO. Estadística e Informática.
Métodos Numéricos y Simulación.
Parcial, Mayo del 2014

1. **Dada la función:** $exp(-x^2) - x^2 - 2x + 10 = 0$

   (a) **Hallar una aplicación contraída $g(x)$ en el intervalo (2,3) para hallar el punto fijo.**

```
x <- NULL
g <- function(x) sqrt(exp(-x^2) - 2 * x + 10)
x[1] <- 2.5
x[2] <- g(x[1])
x[3] <- g(x[2])
x[4] <- g(x[3])
x[5] <- g(x[4])
print(x)
```

```
## [1] 2.500 2.236 2.352 2.302 2.324
```

```
cat("\nPuntos que pertencen al intervalo I(2,3)")
```

```
##
## Puntos que pertencen al intervalo I(2,3)
```

   (b) **Probar que cumple la condición de Lipschitz.**

```
cat("\nCondicion de Lipschitz\n")
```

```
##
## Condicion de Lipschitz
```

```
abs(g(x[1]) - g(x[2]))
```

```
## [1] 0.1159
```

```
abs(x[1] - x[2])
```

```
## [1] 0.2635
```

```
k <- abs(g(x[1]) - g(x[2]))/abs(x[1] - x[2])
print(k)
```

```
## [1] 0.4398
```

```
cat("\nnuevos puntos\n")
```

```
##
## nuevos puntos
```

```
abs(g(x[2]) - g(x[3]))
```

```
## [1] 0.05039
```

```
abs(x[2] - x[3])
```

```
## [1] 0.1159
```

```r
k <- abs(g(x[2]) - g(x[3]))/abs(x[2] - x[3])
print(k)
```

```
## [1] 0.4348
```

```r
cat("\nnuevos puntos\n")
```

```
##
## nuevos puntos
```

```r
abs(g(x[3]) - g(x[4]))
```

```
## [1] 0.02201
```

```r
abs(x[3] - x[4])
```

```
## [1] 0.05039
```

```r
k <- abs(g(x[3]) - g(x[4]))/abs(x[3] - x[4])
print(k)
```

```
## [1] 0.4368
```

```r
cat("\nCumple la condicion de Lipschitz\n")
```

```
##
## Cumple la condicion de Lipschitz
```

(c) Hallar el punto fijo mediante el algoritmo $X_{i+1} = g(X_i)$

```r
cat("\nAplicando el algoritmo\n")
```

```
##
## Aplicando el algoritmo
```

```r
for (i in 1:5) {
    x[i + 1] <- g(x[i])
}
cbind(x)
```

```
##          x
## [1,] 2.500
## [2,] 2.236
## [3,] 2.352
## [4,] 2.302
## [5,] 2.324
## [6,] 2.314
```

2. la ecuacion no lineal $exp(-x) + x - 2$ tiene como solución el valor: -1.146193208. Si se aplica el método de secante con tres iteraciones dando como valores iniciales: -1.2 y -1.1, hallar:

(a) El error absoluto y el nmero de cifras significativas exactas de la aproximación.

```r
f <- function(x) exp(-x) + x - 2
x[1] <- -1.2
x[2] <- -1.1
```

```r
# primera iteracion
x[3] <- (x[1] * f(x[2]) - x[2] * f(x[1]))/(f(x[2]) - f(x[1]))
# segunda iteracion
x[4] <- (x[2] * f(x[3]) - x[3] * f(x[2]))/(f(x[3]) - f(x[2]))
# tercera iteracion
x[5] <- (x[3] * f(x[4]) - x[4] * f(x[3]))/(f(x[4]) - f(x[3]))
cat("\nAproximacion de la raiz:", x[5])
```

```
##
## Aproximacion de la raiz: -1.146
```

(b) Que porcentaje de error tiene la aproximacin.

```r
# Aproximacion de un nuevo valor para el calculo del error
Ex <- abs(x[5] - -1.146193208)
cat("\nError absoluto:", Ex)
```

```
##
## Error absoluto: 7.084e-08
```

(c) Si este valor es utilizado en la función $exp(-x^2)$, cual es la magnitud de error que se estara cometiendo.

```r
f <- expression(exp(-X^2))
fx <- D(f, "X")
print(fx)
```

```
## -(exp(-X^2) * (2 * X))
```

```r
X <- x[5]
Ef <- abs(eval(fx)) * Ex
cat("\nError de la funcion:", Ef)
```

```
##
## Error de la funcion: 4.365e-08
```

3. En el sistema de ecuaciones no lineales $f(x, y) = x - sen(y)$; $g(x, y) = y - cos(x)$, la solución en la una iteración fue: $x = 0.694$, $y = 0.768$. Si se utilizo el mtodo de Newton, Cuales serían los siguientes valores de x e y como aproximaciń a la solución del sistema.

```r
f <- expression(x - sin(y))
g <- expression(y - cos(x))
x <- 0.694
y <- 0.768
fx <- D(f, "x")
fy <- D(f, "y")
gx <- D(g, "x")
gy <- D(g, "y")
fx
```

```
## [1] 1
```

```r
fy
```

```
## -cos(y)

gx

## sin(x)

gy

## [1] 1

# Jacobiano
J <- cbind(c(eval(fx), eval(fy)), c(eval(gx), eval(gy)))
F0 <- c(eval(f), eval(g))
print(J)

##         [,1]   [,2]
## [1,]  1.0000 0.6396
## [2,] -0.7193 1.0000

print(F0)

## [1] -0.0006980 -0.0006937

X <- c(x, y)
# Segun newton-raphson
X <- X - solve(J) %*% F0
print(X)

##        [,1]
## [1,] 0.6942
## [2,] 0.7688
```

4. Dado el siguiente sistema de ecuaciones:

$$2x + 4y + 6z = 0$$
$$x + y + 2z = 2$$
$$3x + 4y + 2z = -6$$

Aplicar operaciones elementales para convertir en un sistema diagonal y halla la solución.

```
A <- rbind(c(2, 4, 6, 0), c(1, 1, 2, 2), c(3, 4, 2, -6))
# matriz original
print(A)

##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    0
## [2,]    1    1    2    2
## [3,]    3    4    2   -6
```

```r
# Diagonalizando Paso 1
A[2, ] <- 2 * A[2, ] - A[1, ]
A[3, ] <- 2 * A[3, ] - 3 * A[1, ]
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    4    6    0
## [2,]    0   -2   -2    4
## [3,]    0   -4  -14  -12
```

```r
# Paso 2
A[1, ] <- A[1, ] + 2 * A[2, ]
A[3, ] <- A[3, ] - 2 * A[2, ]
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    0    2    8
## [2,]    0   -2   -2    4
## [3,]    0    0  -10  -20
```

```r
# Paso 3
A[1, ] <- 5 * A[1, ] + A[3, ]
A[2, ] <- 5 * A[2, ] - A[3, ]
print(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   10    0    0   20
## [2,]    0  -10    0   40
## [3,]    0    0  -10  -20
```

```r
# Resultado final
x <- A[1, 4]/A[1, 1]
y <- A[2, 4]/A[2, 2]
z <- A[3, 4]/A[3, 3]
print(c(x, y, z))
```

```
## [1]  2 -4  2
```

Puntaje 5 c/u