Solución del examen de Métodos Numéricos del 22 de Febrero de 2002

1 Ejercicio 1

1.1 Parte a

```
function [A]=escalerizar(A)

[n,m]=size(A);
for i=2:n
    m=A(i,i-1)/A(i-1,i-1);
    A(i,i-1:i)=A(i,i-1:i)-m*A(i-1,i-1:i);
end
```

1.2 Parte b

Escalerización

En cada iteración se debe modificar sólo la fila i, pues las filas siguientes ya tienen 0's en la columna $(i-1)$. Para modificar cada una de estas filas se hacen 3 operaciones. Estas son: una división para calcular el multiplicador, **m**, y una suma y una multiplicación[1] para calcular cada uno de los dos elementos que se modifican (el elemento de la diagonal principal y el de la subdiagonal inferior. En las demás posiciones o bien ya hay un cero o bien el elemento correspondiente de la fila $(i-1)$ es nulo). Por lo tanto, la cantidad de operaciones necesarias para escalerizar es $3(n-1) \simeq 3n$.

Sustitución hacia atrás

La matriz escalerizada tiene a lo más, dos elementos distintos de cero en cada fila (y un solo elemento no nulo en la última fila). Esto significa que para despejar cada uno de los elementos del vector incógnita, x, debo realizar una multiplicación, una resta y una división. Es decir, la cantidad de operaciones necesarias para sustituir hacia atrás es $2(n-1)+1 \simeq 2n$.

[1] una suma y una multiplicación se consideran una sola operación

Observación: la cantidad de operaciones necesarias para sustituir *hacia adelante* es totalmente equivalente.

1.3 Parte c

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ l_{2,1} & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & l_{3,2} & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & l_{4,3} & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & l_{n,n-1} & 1 \end{pmatrix}$$

$$\mathbf{U} = \begin{pmatrix} u_{1,1} & u_{1,2} & 0 & 0 & \dots & 0 & 0 \\ 0 & u_{2,2} & u_{2,3} & 0 & \dots & 0 & 0 \\ 0 & 0 & u_{3,3} & u_{3,4} & \dots & 0 & 0 \\ 0 & 0 & 0 & u_{4,4} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & u_{n,n} \end{pmatrix}$$

1.4 Parte d

La inversa de una matriz esparsa es generalmente una matriz llena. Al invertir una matriz tridiagonal obtendremos (salvo casos muy particulares) una matriz sin estructura ninguna donde la mayoría de los elementos son distintos de cero. Para efectuar el cálculo $(A^{-1} \times b)$ necesario para resolver el sistema $A \times x = b$, se realizan del orden de n^2 cuentas.
Sin embargo para resolver el sistema usando la descomposición LU, sólo se necesitan $2n$ cuentas para la sustitución hacia atrás y otro tanto para la sustitución hacia adelante (esta cota se puede mejorar aun más: como la matriz L tiene 1's en la diagonal, sólo se necesitan del orden de n cuentas para realizar la sustitución hacia adelante).

Por lo tanto, es más conveniente utilizar la descomposición LU para resolver el sistema.

2 Ejercicio 2

2.1 Parte a

La consistencia de un método numérico para la resolución de EDO se define como:

Si $E_n(h)$ es el error global cometido en un paso genérico n, entonces $E_n(h) \to 0$ cuando $h \to 0$. La idea es que tomar pasos más pequeños mejore la solución, y en el caso límite coincide con la solución exacta.

La estabilidad numérica requiere que $|E_n(h)|$ permanezca acotado cuando aumentamos n. La idea es que los errores que se acumulan al resolver paso por paso no hagan diverger a la solución numérica.

2.2 Parte b

Para deducir el método del trapecio, planteamos:

$$\int_{x_i}^{x_{i+1}} y'(x)\, dx = \int_{x_i}^{x_{i+1}} f(x, y(x))\, dx \simeq \frac{h}{2}\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1})\right)$$

Por lo que el paso de iteración será:

$$y_{i+1} = y_i + \frac{h}{2}\left(f(x_i, y_i) + f(x_{i+1}, y_{i+1})\right)$$

siendo y_{i+1} la aproximación a la solución en el punto $x_{i+1} = x_i + h$. Obsérvese que este método es *implícito* por lo que deberá resolverse la ecuación anterior por algún método general de resolución de ecuaciones no lineales.

Estimemos el error local observando que el error por aproximar linealmente la función $f(x, y(x))$ es:

$$E(x) = \frac{\partial f}{\partial x}(\xi)(x - x_i)(x - x_{i+1}) \leq M(x - x_i)(x_{i+1} - x)$$

por lo que el error en el método del trapecio será:

$$e_{local} = \int_{x_i}^{x_{i+1}} E(x)\, dx \leq M \int_0^h u(h - u)\, du = M\left(\frac{h^3}{2} - \frac{h^3}{3}\right) = \frac{M}{6}h^3$$

Por lo que el error local será de orden $O\left(h^3\right)$, es decir, el orden de consistencia será $O\left(h^2\right)$.

2.3 Parte c

Se plantea el problema test de resolver la ecuación:

$$y'(x) = qy(x)$$

Por lo tanto, $f(x,y) = qy$. La ecuación de iteración queda:

$$\begin{aligned} y_{i+1} &= y_i + \frac{h}{2}(qy_i + qy_{i+1}) \\ y_{i+1} &= \left(\frac{1+\frac{h}{2}q}{1-\frac{h}{2}q}\right) y_i \end{aligned}$$

Por lo que la región de estabilidad es:

$$\left|\frac{1+\frac{h}{2}q}{1-\frac{h}{2}q}\right| < 1$$

2.4 Parte d

Se propone el siguiente código Matlab:

```
[x,y]=function(x0,y0,x1,h,tol)
x=[x0];
y=[y0];
i=1;

while(x(i)+h)$<$x1
    x(i+1)=x(i)+h;
    u_i=y(i)+1/2*h*func(x(i),y(i));
    y(i+1)=y(i)+h*func(x(i),y(i)); %predictor
    aux=0;
    while abs(y(i+1)-aux)$<$tol
        aux=y(i+1);}
        y(i+1)=u\_i+1/2*h*func(x(i+1),y(i+1));
    end
    i=i+1;
end
```

3 Ejercicio 3

3.1 Parte a

Dado x_i el punto x_{i+1} se define como el cero de la línea tangente a $f(x)$ en x_i. De esta forma se construye la sucesión $x_0, x_1, x_2, \ldots$ donde x_0 debe ser dado al algoritmo.
Para deducir la fórmula del método de Newton-Raphson, se escribe el desarrollo de Taylor de $f(x)$ alrededor del punto x_i:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \ldots.$$

La ecuación de la tangente viene dada por los primeros dos términos de la serie de Taylor:

$$y = f(x_i) + f'(x_i)(x - x_i)$$

y haciendo $y = 0$ se tiene:

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

3.2 Parte b

Sea e_i el error cometido en la iteración i definido como $e_i = x_i - \alpha$, donde α es la raíz buscada. Si se cumple que

$$\lim_{i \to \infty} \frac{|e_{i+1}|}{|e_i|^r} = C \neq 0$$

se dice que el orden de convergencia del método es r.
Desarrollamos $f(x)$ en su serie de Taylor:

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \frac{1}{2} f''(\zeta)(x - x_i)^2.$$

siendo ζ un punto intermedio entre x y x_i.
Si $x = \alpha$, entonces:

$$0 = f(\alpha) = f(x_i) + f'(x_i)(\alpha - x_i) + \frac{1}{2} f''(\zeta)(\alpha - x_i)^2.$$

Dividiendo entre $f'(x_i)$ y reordenando, obtenemos:

$$\alpha - \left(x_i - \frac{f(x_i)}{f'(x_i)} \right) = \frac{f''(\zeta)}{2f'(x_i)} (\alpha - x_i)^2.$$

El lado izquierdo de la ecuación es justamente $\alpha - x_{i+1} = e_{i+1}$, por lo tanto tenemos que

$$e_{i+1} = \left(\frac{f''(\zeta)}{2f'(x_i)} \right) {e_i}^2.$$

Si la iteración converge tenemos que:

$$\lim_{i \to \infty} \frac{|e_{i+1}|}{|e_i|^2} = \frac{f''(\alpha)}{2f'(\alpha)} = C.$$

Con esto se prueba que el método de Newton-Raphson es de orden $r = 2$ o cuadrático.

3.3 Parte c

Uno de los métodos más simples y confiables para la resolución de ecuaciones de la forma $f(x) = 0$ es el método de bipartición. Para usar este método deben darse dos puntos a y b tales que: $a < b$ y $f(a) \cdot f(b) < 0$. El método de bipartición parte el intervalo $[a, b]$ a la mitad en cada iteración, de la siguiente forma:

- Sea $m = (a + b)/2$, el punto medio del intervalo $[a, b]$. Calcule $f(m)$. Si $|f(m)| < tol_1$, termina la iteración.
- Si $f(a) \cdot f(m) < 0$, sea $b = m$, si no $a = m$.
- Si $b - a < tol_2$, termina la iteración.

Este algoritmo tiene orden de convergencia lineal pues en cada iteración $\frac{|e_{i+1}|}{|e_i|} = \frac{1}{2}$. Es por lo tanto, de convergencia más lenta que el algoritmo de NR que tiene convergencia cuadrática.
Desde el punto de vista de la complejidad del algoritmo es claro que el método de NR es bastante más caro que el método de bipartición pues si bien ambos requieren que se evalúe una vez la función en cada iteración, el primero necesita también que se calcule $f'(x_i)$.
El método de bipartición converge siempre a alguna de las raíces de orden impar que se encuentren dentro del intervalo, mientras que el método de NR podría no converger. NR no converge si $f'(\alpha) = 0$ y necesita de una buena primera aproximación para converger.
Según estas características el método de bipartición es adecuado para proporcionar una buena semilla al método de NR el cual convergerá rápidamente a la solución, logrando mayor precisión en menos iteraciones.