

Nome:

No. USP:

Prova 5 (bis): Sistemas não lineares, interpolação, integração.**Lembrete:** O método de Newton é implementado em Octave como

```

k=1; x=x0; err=1e+10;
while ((k<=kmax) && (err > tol))
    f = .....;
    J = .....;
    dx = -J\f;
    err = .....;
    x = x + dx;
    k = k + 1;
endwhile

```

onde **f** e **J** correspondem à função cujo zero se quer determinar e à sua matriz Jacobiana, respectivamente. O código faltante depois de **err** depende do critério de parada desejado.

1. **(3 pontos)** Escrever um código baseado no método de Newton que calcule o valor x^* que minimiza a distância (euclidiana usual) entre o gráfico da função $f(x) = e^x$ e o ponto $(x, y) = (-1, 1)$. Utilize critério de parada de “teste do resíduo”.

Hint: Minimizar a distância ao quadrado.

A distância (ao quadrado) para cada x é, simplesmente,

$$F(x) = (x + 1)^2 + (e^x - 1)^2 .$$

A função a minimizar é a derivada dela,

$$f(x) = 2(x + 1) + 2e^x(e^x - 1) .$$

e a Jacobiana será $f'(x)$.

2. **(3 pontos)** Escrever um código baseado no método de Newton que calcule o valor x^* que minimiza a distância (euclidiana usual) entre o gráfico da função $f(x) = \frac{1}{x}$ e o ponto $(x, y) = (1, 3)$. Utilize critério de parada de “erro relativo”.

Hint: Minimizar a distância ao quadrado.

A distância (ao quadrado) para cada x é, simplesmente,

$$F(x) = (x - 1)^2 + (x^{-1} - 3)^2 .$$

A função a minimizar é a derivada dela,

$$f(x) = 2(x - 1) - 2x^{-2}(x^{-1} - 3) .$$

e a Jacobiana será $f'(x)$.

3. **(3 pontos)** Se deseja uma regra de integração da forma

$$I = A_1 y(x_1) + A_2 y(x_2)$$

sendo que $x_1 = 0$ e $x_2 = 2$, e que se deseja aproximar a integral

$$\int_0^{+\infty} e^{-x} y(x) dx .$$

Calcule A_1 e A_2 de maneira que I integre exatamente qualquer polinômio $y(x)$ de primeiro grau.

$$(A_1, A_2) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

4. **(3 pontos)** Se deseja uma regra de integração da forma

$$I = A_1 y(x_1) + A_2 y(x_2)$$

sendo que $x_1 = 1$ e $x_2 = 3$, e que se deseja aproximar a integral

$$\int_0^{+\infty} e^{-x} y(x) dx .$$

Calcule A_1 e A_2 de maneira que I integre exatamente qualquer polinômio $y(x)$ de primeiro grau.

$$(A_1, A_2) = (1, 0)$$

Nome:

No. USP:

Prova 6: Solução numérica de EDOs.

Lembrete: Como lembrete de programação, a função pêndulo embaixo resolve pelo método de Euler explícito a equação do pêndulo $\theta'' = -\omega^2 \sin \theta$ pelo método de Euler explícito.

```
function [y time]=pendulo(y0,t0,dt,nt)
ome=1.;
time(1)=t0;
y(:,1)=y0;
for n=1:nt
    y(1,n+1)=y(1,n)+dt*y(2,n);
    y(2,n+1)=y(2,n)-dt*ome*ome*sin(y(1,n));
    time(n+1)=time(n)+dt;
endfor
```

As definições dos métodos são:

Euler explícito:

$$y^{n+1} = y^n + \delta t f(t_n, y^n)$$

Euler implícito:

$$y^{n+1} = y^n + \delta t f(t_{n+1}, y^{n+1})$$

Trapezoidal implícito:

$$y^{n+1} = y^n + \frac{\delta t}{2} (f(t_n, y^n) + f(t_{n+1}, y^{n+1}))$$

RK2 trapezoidal explícito:

$$y^{n+1} = y^n + \delta t (c_1 k_1 + c_2 k_2)$$

onde $k_1 = f(t_n, y_n)$, $k_2 = f(t_n + \delta t, y^n + \delta t k_1)$, $c_1 = c_2 = 1/2$.**RK2 Euler melhorado:**

$$y^{n+1} = y^n + \delta t (c_1 k_1 + c_2 k_2)$$

onde $k_1 = f(t_n, y_n)$, $k_2 = f(t_n + \frac{\delta t}{2}, y^n + \frac{\delta t}{2} k_1)$, $c_1 = 0$, $c_2 = 1$.
Adams-Bashforth de 3 passos:

$$y^{n+1} = y^n + \frac{\delta t}{12} [23f(t_n, y^n) - 16f(t_{n-1}, y^{n-1}) + 5f(t_{n-2}, y^{n-2})]$$

1. **(3 pontos)** A equação de um circuito RLC é dada, em termos da corrente $I(t)$, por

$$\frac{d^2}{dt^2} I(t) + 2\alpha \frac{d}{dt} I(t) + \omega_0^2 I(t) = \sin 3t, \quad (*)$$

onde α é chamada de frequência de Neper e ω_0 é a frequência natural.

Escreva um código em Octave que, a partir dos dados $\mathbf{a}=\alpha$, $\mathbf{w2}=\omega_0^2$, $\mathbf{i0}=I(t=0)$, $\mathbf{di0}=I'(0)$, $\mathbf{dt}=\delta t$ e $\mathbf{N}$, resolva numericamente a equação até o tempo $T=N\delta t$ pelo método

RK2 Euler melhorado.

A equação é

$$\begin{pmatrix} I \\ D \end{pmatrix} = \begin{pmatrix} D \\ -\omega_0^2 I - 2\alpha D + \sin 3t \end{pmatrix}$$

Assim,

```
K(1:2,1)=[y(2);-w2*y(1)-2*a*y(2)+sin(3*time(n))];
yy=y(:,n)+dt*K(:,1);
K(1:2,2)=[yy(2);-w2*yy(1)-2*a*yy(2)+sin(3*time(n))+dt/2];
y(:,n+1)=y(:,n)+dt*K(:,2);
```

2. **(3 pontos)** A equação de um circuito RLC é dada, em termos da corrente $I(t)$, por

$$\frac{d^2}{dt^2} I(t) + 2\alpha \frac{d}{dt} I(t) + \omega_0^2 I(t) = \sin 5t, \quad (*)$$

onde α é chamada de frequência de Neper e ω_0 é a frequência natural.

Escreva um código em Octave que, a partir dos dados $\mathbf{a}=\alpha$, $\mathbf{w2}=\omega_0^2$, $\mathbf{i0}=I(t=0)$, $\mathbf{di0}=I'(0)$, $\mathbf{dt}=\delta t$ e $\mathbf{N}$, resolva numericamente a equação até o tempo $T=N\delta t$ pelo método

RK2 trapezoidal explícito.

A primeira parte igual ao exercício anterior.

```
K(1:2,1)=[y(2);-w2*y(1)-2*a*y(2)+sin(3*time(n))];
yy=y(:,n)+dt*K(:,1);
K(1:2,2)=[yy(2);-w2*yy(1)-2*a*yy(2)+sin(3*time(n))+dt/2];
y(:,n+1)=y(:,n)+dt*0.5*(K(:,1)+K(:,2));
```

3. **(3 pontos)** A equação da velocidade V de queda de um objeto é dada por

$$\frac{d}{dt} V(t) + \beta V(t) - g = 0, \quad (*)$$

onde β é um coeficiente constante de arrasto e g é a gravidade.

Escreva um código em Octave que, a partir dos dados $\mathbf{b}=\beta$, $\mathbf{g}=g$, $\mathbf{v0}=V(t=0)$, $\mathbf{dt}=\delta t$ e $\mathbf{N}$, resolva numericamente a equação até o tempo $T=N\delta t$ pelo método de Euler implícito.

$$\frac{V^{n+1} - V^n}{\delta t} + \beta V^{n+1} - g = 0 ,$$

$$(1 + \beta \delta t) V^{n+1} = V^n + g \delta t ,$$

e assim,

$$V^{n+1} = \frac{V^n + g \delta t}{1 + \frac{\beta \delta t}{2}}$$

Só resta programar.

4. (3 pontos) A equação da velocidade V de queda de um objeto é dada por

$$\frac{d}{dt} V(t) + \beta V(t) - g = 0 ,$$

onde β é um coeficiente constante de arrasto e g é a gravidade.

Escreva um código em Octave que, a partir dos dados $\mathbf{b}=\beta$, $\mathbf{g}=g$, $\mathbf{v0}=V(t=0)$, $\mathbf{dt}=\delta t$ e $\mathbf{N}$, resolva numericamente a equação até o tempo $T = \mathbf{N}\delta t$ pelo método **trapezoidal implícito**.

$$\frac{V^{n+1} - V^n}{\delta t} + \frac{\beta}{2} (V^{n+1} + V^n) - g = 0 ,$$

$$\left(1 + \frac{\beta \delta t}{2}\right) V^{n+1} = \left(1 - \frac{\beta \delta t}{2}\right) V^n + g \delta t ,$$

e assim,

$$V^{n+1} = \frac{1 - \frac{\beta \delta t}{2}}{1 + \frac{\beta \delta t}{2}} V^n + \frac{g \delta t}{1 + \frac{\beta \delta t}{2}}$$

Só resta programar.

5. (2 pontos) Responda brevemente: No exercício anterior, acha que o método programado terá alguma restrição de estabilidade no passo de tempo? Qual seria essa restrição em termos dos dados?

Para os métodos **explícitos** esperamos uma restrição de estabilidade do tipo $\delta t < 2/\lambda$, mas para os **implícitos** não! Só estimar o maior autovalor da Jacobiana no primeiro caso.

6. (2 pontos) Para a equação $y' = y - t^3 + 1$, considere o método numérico

$$y^{n+1} = y^n + \delta t(y^n - t_n^3 + 1) + \frac{\delta t^2}{2} (y^n - t_n^3 - 3t_n^2 + 1)$$

qual a ordem de consistência desse método?

Rta: 2

7. (2 pontos) Para a equação $y' = y - t^3 + 1$, considere o método numérico

$$y^{n+1} = y^n + \delta t(y^n - t_n^3 + 1) + \frac{\delta t^2}{2} (y^n - t_n^3 + 1)$$

qual a ordem de consistência desse método?

Rta: 1

8. (2 pontos) Para a equação $y' = y - t^3 + 1$, considere o método numérico

$$y^{n+1} = \frac{4}{3}y^n - \frac{1}{3}y^{n-1} + \frac{2\delta t}{3}(y^{n+1} - t_{n+1}^3 + 1)$$

qual a ordem de consistência desse método?

Rta: 2

9. (2 pontos) Para a equação $y' = y - t^3 + 1$, considere o método numérico

$$y^{n+1} = \frac{4}{3}y^n - \frac{1}{3}y^{n-1} + \frac{2\delta t}{3}(y^n - t_n^3 + 1)$$

qual a ordem de consistência desse método?

Rta: 1

10. (3 pontos) Responda Verdadeiro ou Falso à esquerda de cada item.

- (a) Os métodos explícitos tem uma restrição no passo de tempo para resolver $y' = f(t, y)$ desde $t = 0$ até $t = T$, que é

$$\delta t < \frac{2}{M}$$

onde M é o maior valor, em módulo, que f pode atingir entre o tempo 0 e o tempo T . F

- (b) Os métodos explícitos tem uma restrição no passo de tempo para resolver $y' = f(t, y)$ desde $t = 0$ até $t = T$, que é

$$\delta t < \frac{2}{M}$$

onde M é o maior valor, em módulo, que os autovalores da matriz Jacobiana de f podem atingir entre o tempo 0 e o tempo T . V

- (c) Os métodos explícitos tem uma restrição no passo de tempo para resolver $y' = f(t, y)$ desde $t = 0$ até $t = T$, que é

$$\delta t < \frac{2}{M}$$

onde M é o menor valor, em módulo, que os autovalores da matriz Jacobiana de f podem atingir entre o tempo 0 e o tempo T . F

- (d) O custo computacional de um método de três passos explícito é menor que o de um método de Runge-Kutta de três estágios. V

- (e) O custo computacional de um método de três passos explícito é maior que o de um método de Runge-Kutta de três estágios. F

- (f) O custo computacional de um método de três passos explícito é igual ao de um método de Runge-Kutta de três estágios. F

- (g) Para a mesma ordem de precisão, os métodos implícitos são menos custosos que os explícitos, mas menos estáveis. F

- (h) Para a mesma ordem de precisão, os métodos explícitos são menos custosos que os implícitos, mas menos estáveis. V
 - (i) O método de Euler implícito é mais preciso que o método de Euler explícito. F
 - (j) O método de Euler explícito é mais preciso que o método de Euler implícito. F
 - (k) Quanto maior o número de estágios de um método de Runge-Kutta, maior sua estabilidade. F
 - (l) Quanto maior o número de estágios de um método de Runge-Kutta, maior sua precisão. V
-