Instituto Tecnológico de Aeronáutica – ITA
Matemática Computacional – CCI-22
Laboratório 10 – PVI com Métodos de Passo Múltiplo e PVC

Professor: Marcos Ricardo Omena de Albuquerque Maximo

28 de maio de 2019

1 Tarefas

1.1 Implementação

As seguintes funções em MATLAB são fornecidas (cada uma em um arquivo .m separado):

1. `y = PassoAdamsBashforth4(fv, h, y)`: realiza um avanço com Método de Adams-Bashforth de ordem 4 a partir do ponto $\mathbf{y}$ com passo h. A matriz $\mathbf{f}_v$ contém os valores da função derivada necessários para o cálculo:

$$\mathbf{f}_v = \begin{bmatrix} \mathbf{f}_v(1,:) \\ \mathbf{f}_v(2,:) \\ \mathbf{f}_v(3,:) \\ \mathbf{f}_v(4,:) \end{bmatrix} = \begin{bmatrix} f(x(i-3), \mathbf{y}(i-3,:)) \\ f(x(i-2), \mathbf{y}(i-2,:)) \\ f(x(i-1), \mathbf{y}(i-1,:)) \\ f(x(i), \mathbf{y}(i,:)) \end{bmatrix} \tag{1}$$

2. `y = PassoAdamsMoulton4(fv, h, y)`: realiza um avanço com Método de Adams-Moulton de ordem 4 a partir do ponto $\mathbf{y}$ com passo h. A matriz $\mathbf{f}_v$ contém os valores da função derivada necessários para o cálculo:

$$\mathbf{f}_v = \begin{bmatrix} \mathbf{f}_v(1,:) \\ \mathbf{f}_v(2,:) \\ \mathbf{f}_v(3,:) \\ \mathbf{f}_v(4,:) \end{bmatrix} = \begin{bmatrix} f(x(i-2), \mathbf{y}(i-2,:)) \\ f(x(i-1), \mathbf{y}(i-1,:)) \\ f(x(i), \mathbf{y}(i,:)) \\ f(x(i+1), \mathbf{y}(i+1,:)) \end{bmatrix} \tag{2}$$

3. `y = PassoRungeKutta4(f, h, x, y)`: realiza um avanço com Método de Runge-Kutta de ordem 4 a partir do ponto $(x, \mathbf{y})$ com passo h. A função f é a função derivada do sistema de EDOs.

4. `x = SolucaoTridiagonal(a, b, c, d)`: utiliza o algoritmo de Thomas, cuja complexidade é $\mathcal{O}(n)$, para resolver um sistema tridiagonal na seguinte forma:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, i = 1, 2, ..., n \tag{3}$$

Em que $a_1 = c_n = 0$. Ou em representação matricial:

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_3 & b_3 & c_3 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_4 & b_4 & c_4 & \cdots & 0 & 0 \\ 0 & 0 & 0 & a_5 & b_5 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix} \tag{4}$$

As entradas da função são os vetores coluna $\mathbf{a} = [a_1, a_2, ..., a_n]^T$, $\mathbf{b} = [b_1, b_2, ..., b_n]^T$, $\mathbf{c} = [c_1, c_2, ..., c_n]^T$ e $\mathbf{d} = [d_1, d_2, ..., d_n]^T$, enquanto a saída é o vetor coluna $\mathbf{x} = [x_1, x_2, ..., x_n]^T$.

A partir destas funções, implementar as seguintes funções em MATLAB:

1. `y = PVIAdamsBashforth4(f, x, y0)`: utiliza o Método de Adams-Bashforth de ordem 4 para resolver o problema de valor inicial (PVI) associado ao sistema de equações diferenciais ordinárias (EDOs) $\mathbf{p}'(t) = f(t, \mathbf{p})$ e à condição inicial $\mathbf{p}(t_0 = a) = \mathbf{y}_0$, no intervalo $[a = x_0, b = x_n]$. A solução é calculada nos pontos contidos no vetor coluna $\mathbf{x} = [x_0, x_1, x_2, \ldots, x_n]^T$, em que $h = x_{i+1} - x_i$ (i.e. considera-se pontos igualmente espaçados no intervalo $[a, b]$). A função vetorial $f(t, \mathbf{p})$ é definida como um vetor linha:

$$f(t, \mathbf{p}) = \begin{bmatrix} f_1(t, \mathbf{p}) & f_2(t, \mathbf{p}) & \cdots & f_l(t, \mathbf{p}) \end{bmatrix} \tag{5}$$

em que:

$$\mathbf{p}(t) = \begin{bmatrix} p_1(t) & p_2(t) & \cdots & p_l(t) \end{bmatrix} \tag{6}$$

Desse modo, o retorno da função é a matriz $\mathbf{y}$ definida da seguinte forma:

$$\mathbf{y} = \begin{bmatrix} \mathbf{p}(x_0) \\ \mathbf{p}(x_1) \\ \mathbf{p}(x_2) \\ \vdots \\ \mathbf{p}(x_n) \end{bmatrix} = \begin{bmatrix} p_1(x_0) & p_2(x_0) & \cdots & p_l(x_0) \\ p_1(x_1) & p_2(x_1) & \cdots & p_l(x_1) \\ p_1(x_2) & p_2(x_2) & \cdots & p_l(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(x_n) & p_2(x_n) & \cdots & p_l(x_n) \end{bmatrix} \tag{7}$$

Para inicializar o método de passo múltiplo, a função utiliza o Método de Runge-Kutta de ordem 4.

2. `y = PVIPrevisorCorretor4(f, x, y0)`: utiliza um Método de Previsão-Correção para resolver o problema de valor inicial (PVI) associado ao sistema de equações diferenciais ordinárias (EDOs) $\mathbf{p}'(t) = f(t, \mathbf{p})$ e à condição inicial $\mathbf{p}(t_0 = a) = \mathbf{y}_0$, no intervalo $[a = x_0, b = x_n]$. A solução é calculada nos pontos contidos no vetor coluna $\mathbf{x} = [x_0, x_1, x_2, \ldots, x_n]^T$, em que $h = x_{i+1} - x_i$ (i.e. considera-se pontos

igualmente espaçados no intervalo $[a, b]$). A função vetorial $f(t, \mathbf{p})$ é definida como um vetor linha:

$$f(t, \mathbf{p}) = \begin{bmatrix} f_1(t, \mathbf{p}) & f_2(t, \mathbf{p}) & \cdots & f_l(t, \mathbf{p}) \end{bmatrix} \tag{8}$$

em que:

$$\mathbf{p}(t) = \begin{bmatrix} p_1(t) & p_2(t) & \cdots & p_l(t) \end{bmatrix} \tag{9}$$

Desse modo, o retorno da função é a matriz $\mathbf{y}$ definida da seguinte forma:

$$\mathbf{y} = \begin{bmatrix} \mathbf{p}(x_0) \\ \mathbf{p}(x_1) \\ \mathbf{p}(x_2) \\ \vdots \\ \mathbf{p}(x_n) \end{bmatrix} = \begin{bmatrix} p_1(x_0) & p_2(x_0) & \cdots & p_l(x_0) \\ p_1(x_1) & p_2(x_1) & \cdots & p_l(x_1) \\ p_1(x_2) & p_2(x_2) & \cdots & p_l(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(x_n) & p_2(x_n) & \cdots & p_l(x_n) \end{bmatrix} \tag{10}$$

A função utiliza o Método de Adams-Bashforth de ordem 4 como previsor e o Método de Adams-Moulton de ordem 4 como corretor. Apenas um passo de correção é realizado. Para inicializar o método de passo múltiplo, a função utiliza o Método de Runge-Kutta de ordem 4.

3. `y = ProblemaValorContorno(g1, g2, r, x, y0, yn)`: resolve numericamente um problema de valor de contorno (PVC) da seguinte forma:

$$y''(x) + g_1(x)y'(x) + g_2(x)y(x) = r(x), y(a) = y_0, y(b) = y_n, x \in [a, b] \tag{11}$$

O PVC é resolvido nos pontos igualmente espaçados do vetor $\mathbf{x} = [x_0, x_1, \ldots, x_n]^T$, em que $h = x_{i+1} - x_i, i = 0, 1, 2, \ldots, n-1$. Utiliza-se esquemas de diferenças centradas para aproximar as derivadas:

$$y''(x) \approx \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \tag{12}$$

$$y'(x) \approx \frac{y_{i+1} - y_{i-1}}{2h} \tag{13}$$

Desse modo, a solução do PVC é obtida através da solução do sistema tridiagonal montado a partir das equações:

$$\alpha_i y_{i-1} + \beta_i y_i + \gamma_i y_{i+1} = \delta_i, i = 1, 2, 3, \ldots, n-1 \tag{14}$$

em que:

$$\alpha_i = 1 - \frac{g_1(x_i)h}{2} \tag{15}$$

$$\beta_i = -2 + g_2(x_i)h^2 \tag{16}$$

$$\gamma_i = 1 + \frac{g_1(x_i)h}{2} \tag{17}$$

$$\delta_i = r(x_i)h^2 \tag{18}$$

1.2 Análise

1. Considere o seguinte PVI:

$$y''(x) + y(x) = 0, [y(0), y'(0)] = [1, 1] \tag{19}$$

cuja solução analítica é:

$$y^*(x) = \cos(x) + \sin(x) \tag{20}$$

Resolva este PVI numericamente no intervalo $[0, 100]$ com $h = 0,05$, utilizando os 2 métodos de passo múltiplo implementados. Então, compare os resultados obtidos numericamente com a solução analítica. Para isto, calcule o erro absoluto entre cada solução numérica e a analítica, em cada ponto. Então, trace os erros obtidos num único gráfico, utilizando escala logarítmica para o eixo Y. O resultado é o esperado? Comente.

Observação: a função `ComparacaoPVIPassoMultiplo`, fornecida juntamente com o roteiro, traça o gráfico pedido neste item, desde que as funções `PVIAdamsBashforth4` e `PVIPrevisorCorretor4` estejam implementadas.

2. Seja o seguinte problema de valor de contorno (PVC):

$$y''(x) + 2y'(x) + y(x) = x, y(0) = 0, y(1) = -1, x \in [0, 1] \tag{21}$$

cuja solução analítica é:

$$y^*(x) = 2e^{-x}(1 - x) + x - 2 \tag{22}$$

Resolva este PVC numericamente utilizando passo $h = 0,02$. Em seguida, compare graficamente a solução numérica com a solução analítica. Mostre o gráfico gerado e comente.

Observação: a função `ComparacaoPVC`, fornecida juntamente com o roteiro, traça o gráfico pedido neste item, desde que aa função `ProblemaValorContorno` esteja implementada.

2 Instruções

- A primeira etapa do processo de correção consistirá em submeter as funções implementadas a vários casos de teste de forma automatizada. Assim, os cabeçalhos das funções devem ser seguidos **rigorosamente**. Arquivos .m com os nomes destas funções e os cabeçalhos já implementados foram fornecidos juntamente com este roteiro. Dê preferência a implementar seu laboratório a partir destes arquivos .m fornecidos para evitar erros.
- **Não** é permitido o uso de funções ou comandos prontos do MATLAB que realizem toda a funcionalidade atribuída a uma certa função. Entretanto, o uso destas funções para verificação das implementações realizadas é encorajado. Em caso de dúvida quanto à permissão de uso de alguma função ou comando, recomenda-se consultar o professor.
- Não é necessário reimplementar métodos de laboratórios anteriores. Assim, funções implementadas em laboratórios anteriores podem ser utilizadas. Caso prefira, também é permitido utilizar funções equivalentes do MATLAB.
- Não é necessário se preocupar com verificação dos dados de entrada.
- Os arquivos .m implementados devem ser entregues juntamente com um relatório.
- No relatório, não é necessário demonstrar que as funções implementadas funcionam corretamente (isto será verificado separadamente). Basta incluir resultados e conclusões relativos à **Análise**.
- Para facilitar a correção da Análise, inclua os gráficos diretamente no relatório. Nos gráficos, coloque títulos, legendas e nomes nos eixos.

3 Dicas:

- Utilize o operador `@` para definir funções quando necessário.
- Utilize `semilogy(x,y)` para plotar gráficos com escala logarítmica no eixo Y.
- Perceba que a primeira e a última equações devem ser tratadas de forma especial na formulação da função `ProblemaValorContorno` devido à presença de y_0 e y_n (condições de contorno), respectivamente.
- Cuidado com o formato requerido para vetores para evitar problemas na correção automática.
- Para criar gráficos com alta qualidade em formato PNG para inclusão em arquivos do Microsoft Word, utilize o comando: `print -dpng -r300 grafico.png`.
- Se utilizar LaTeX, dê preferência para incluir gráficos em formato vetorizado. No Linux, utilizando `pdflatex`, você pode gerar um gráfico em formato EPS usando "`print -depsc2 grafico.eps`" e depois convertê-lo para PDF usando o comando de terminal "`epstopdf grafico.eps`". O arquivo PDF é aceito pelo `pdflatex`.