Métodos numéricos para EDPs – terceira lista de exercícios

Leonardo de Oliveira Carvalho

13 de maio de 2010

Exercício 1

O método de *Leapfrog* é definido utilizando-se as aproximações:

$$u_t(x_j, t_{n+1}) \approx \frac{u(x_j, t_{n+1}) - u(x_j, t_{n-1})}{2\Delta t} = \frac{1}{2\Delta t}\left(U_j^{n+1} - U_j^{n-1}\right),$$
$$u_x(x_j, t_{n+1}) \approx \frac{u(x_{j+1}, t_n) - u(x_{j-1,t_n})}{2\Delta x} = \frac{1}{2\Delta x}\left(U_{j+1}^n - U_{j-1}^n\right).$$

Para a equação da onda unidirecional $u_t + cu_x = 0$ teremos então:

$$\frac{1}{2\Delta t}\left(U_j^{n+1} - U_j^{n-1}\right) + \frac{c}{2\Delta x}\left(U_{j+1}^n - U_{j-1}^n\right) = 0,$$

ou ainda

$$U_j^{n+1} = U_j^{n-1} - \sigma\left(U_{j+1}^n - U_{j-1}^n\right), \qquad \text{onde } \sigma = \frac{c\Delta t}{\Delta x}.$$

Este é um método de dois passos, no instante inicial podemos utilizar outro método de mesma ordem para obter U^1, a partir de então usar o *Leapfrog*.

Para a análise de Von Neumann, vamos usar a transformada de Fourier inversa para cada instante de tempo n:

$$U_j^n = \frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta xk}\widehat{U}^n(k)\,dk.$$

Usando esta relação na equação de diferenças do método teremos:

$$\begin{aligned}
\frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta xk}\widehat{U}^{n+1}(k)\,dk &= \frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta xk}\widehat{U}^{n-1}(k)\,dk \\
&\quad - \sigma\left(\frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{i(j+1)\Delta xk}\widehat{U}^n(k)\,dk\right. \\
&\quad \left. - \frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{i(j-1)\Delta xk}\widehat{U}^n(k)\,dk\right) \\
&= \frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta xk}\left(\widehat{U}^{n-1}(k) - \sigma\widehat{U}^n(k)\left(e^{i\Delta xk} - e^{i\Delta xk}\right)\right)dk \\
&= \frac{1}{\sqrt{2\pi}}\int_{-\pi/\Delta x}^{\pi/\Delta x} e^{ij\Delta xk}\left(\widehat{U}^{n-1}(k) - 2i\sigma\widehat{U}^n(k)\,\mathrm{sen}(\Delta xk)\right)dk.
\end{aligned}$$

De onde teremos

$$\widehat{U}^{n+1}(k) = \widehat{U}^{n-1}(k) - 2i\sigma\widehat{U}^n(k)\,\mathrm{sen}(\Delta xk).$$

Fazendo $\widehat{U}^n(k) = (g_{\Delta t}(k))^n$ teremos

$$(g_{\Delta t}(k))^{n+1} = (g_{\Delta t}(k))^{n-1} + 2i\sigma(g_{\Delta t}(k))^n \operatorname{sen}(\Delta x k)$$
$$g_{\Delta t}(k) = (g_{\Delta t}(k))^{-1} - 2i\sigma \operatorname{sen}(\Delta x k)$$
$$(g_{\Delta t}(k))^2 = 1 - 2i\sigma \operatorname{sen}(\Delta x k) g_{\Delta t}(k)$$
$$(g_{\Delta t}(k))^2 + 2i\sigma \operatorname{sen}(\Delta x k) g_{\Delta t}(k) - 1 = 0.$$

Cujas soluções são

$$g^+_{\Delta x}(k) = \sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} - i(\sigma \operatorname{sen}(\Delta x k))$$
$$g^-_{\Delta x}(k) = -\sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} - i(\sigma \operatorname{sen}(\Delta x k))$$

Quando $g^+_{\Delta x}(k) = g^-_{\Delta x}(k) = g(k)$ a solução para $\widehat{U}^n(k)$ pode ser escrita como

$$\widehat{U}^n(k) = h_1(k) g(k)^n + h_2(k) n g(k)^{n-1},$$

onde $h_1(k)$ e $h_2(k)$ são determinados pelas codições iniciais.

Quando $g^+_{\Delta x}(k) \neq g^-_{\Delta x}(k)$ teremos

$$\widehat{U}^n(k) = h^+(k) \left(g^+_{\Delta x}(k)\right)^n + h^-(k) \left(g^-_{\Delta x}(k)\right)^n,$$

com $h^+(k)$ e $h^-(k)$ determinados pelas condições iniciais. Esta equação pode ser reorganizada na forma

$$\widehat{U}^n(k) = h_1(k) \left(g^+_{\Delta x}(k)\right)^n + h_2(k) \left(\frac{\left(g^-_{\Delta x}(k)\right)^n - \left(g^+_{\Delta x}(k)\right)^n}{g^-_{\Delta x}(k) - g^+_{\Delta x}(k)} \right),$$

para $h_1(k)$ e $h_2(k)$ determinadas pelas codições iniciais.

Nos dois caso ($g^+_{\Delta x}(k)$ e $g^-_{\Delta x}(k)$ iguais ou não), vemos que para $n = 0$ temos

$$\widehat{U}^0(k) = h_1(k),$$

e quando $n = 1$

$$\widehat{U}^1(k) = h_1(k) g(k) + h_2(k) = \widehat{U}^0(k) g^+_{\Delta x}(k) + h_2(k).$$

Para que o sistema seja estável precisamos que exista algum inteiro I tal que para qualquer tempo T exista uma constante C tal que

$$\Delta x \sum_{j=-\infty}^{\infty} |U^n_j|^2 \leqslant C \Delta x \sum_{i=0}^{I} \sum_{j=-\infty}^{\infty} |U^i_j|^2$$

para $0 \leqslant n\Delta t \leqslant T$, e com $(\Delta x, \Delta t)$ numa região Λ.

Para a estabilidade do *Leapfrog* vamos usar $I = 1$. No caso em que $g^+_{\Delta x}(k) \neq g^-_{\Delta x}(k)$, se escolhermos dados iniciais U_0 e U_1 tais que $h_2(k) = 0$ teremos então

$$|\widehat{U}^1(k)| = |h_1(k)||g^+_{\Delta x}(k)|^n.$$

Para o método ser estável neste caso é necessário então que $g^+_{\Delta x}(k)$ satisfaça

$$|g^+_{\Delta x}(k)(k)| \leqslant 1 + \mathcal{O}(\Delta t),$$

da mesma forma como foi visto para esquemas de passo único. De forma semelhante, podemos escolher dados iniciais de forma que cheguemos à mesma conclusão para $g^-_{\Delta x}(k)$, isto é

$$|g^-_{\Delta x}(k)(k)| \leqslant 1 + \mathcal{O}(\Delta t).$$

Como estamos utilizando passos constantes a estabilidade pode ser determinada por

$$|g^{\pm}_{\Delta x}(k)| \leqslant 1.$$

Se $|\sigma| \leqslant 1$ então

$$|g^{+}_{\Delta x}(k)|^2 = |g^{-}_{\Delta x}(k)|^2 = 1 - \sigma^2 \operatorname{sen}^2(\Delta x k) - \sigma^2 \operatorname{sen}^2(\Delta x k) = 1$$

Caso contrário, quando $|\sigma| > 1$, é fácil ver que quando $\Delta x k = \pi/2$ teremos $|g^{-}_{\Delta x}(\pi/2)| > 1$, e portanto, neste caso o método é instável.

Considerando agora o caso $g^{+}_{\Delta x}(k) = g^{-}_{\Delta x}(k)$, teremos

$$\begin{gathered}
\sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} - i\,(\sigma \operatorname{sen}(\Delta x k)) = -\sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} - i\,(\sigma \operatorname{sen}(\Delta x k)) \\
\sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} = -\sqrt{(1 - \sigma^2 \operatorname{sen}^2(\Delta x k))} \\
1 - \sigma^2 \operatorname{sen}^2(\Delta x k) = 0 \\
\sigma^2 \operatorname{sen}^2(\Delta x k) = 1 \\
|\sigma \operatorname{sen}(\Delta x k)| = 1
\end{gathered}$$

Como $|\sigma| \leqslant 1$, teremos $g^{+}_{\Delta x}(k) = g^{-}_{\Delta x}(k)$ apenas quando $\Delta x k = \pm\pi/2$, onde teremos $g^{+}_{\Delta x}(k) = g^{-}_{\Delta x}(k) = \pm i$. Assim teremos

$$\widehat{U}^n \left(\pm\frac{\pi}{2\Delta x}\right) = h_1 \left(\pm\frac{\pi}{2\Delta x}\right) (\pm i)^n + h_2 \left(\pm\frac{\pi}{2\Delta x}\right) n(\pm i)^{n-1}.$$

Observe que quando os valores de $h_2\left(\pm\frac{\pi}{2\Delta x}\right)$ são diferentes de zero teremos um crescimento linear de $\widehat{U}^n$, logo neste caso o método será instável, mesmo estando aparentemente cotado.

Assim o método *Leapfrog* será estável se $\sigma < 1$.

Para verificar agora a ordem do método, vamos analisar as aproximações utilizadas para $u_t(x_j, t_n)$ e $u_x(x_j, t_n)$, onde u é uma solução do problema contínuo. Temos:

$$\begin{gathered}
u(x_j, t_{n+1}) = u(x_j, t_n) + \Delta t u_t(x_j, t_n) + \frac{\Delta t^2}{2} u_{tt}(x_j, t_n) + \frac{\Delta t^3}{6} u_{ttt}(x_j, t_n) + \mathcal{O}\left(\Delta t^4\right) \\
u(x_j, t_{n-1}) = u(x_j, t_n) - \Delta t u_t(x_j, t_n) + \frac{\Delta t^2}{2} u_{tt}(x_j, t_n) - \frac{\Delta t^3}{6} u_{ttt}(x_j, t_n) + \mathcal{O}\left(\Delta t^4\right) \\
u(x_j, t_{n+1}) - u(x_j, t_{n-1}) = 2\Delta t u_t(x_j, t_n) + \mathcal{O}\left(\Delta t^3\right) \\
u_t(x_j, t_n) = \frac{u(x_j, t_{n+1}) - u(x_j, t_{n-1})}{2\Delta t} + \mathcal{O}\left(\Delta t^2\right)
\end{gathered}$$

De forma similar obtemos

$$u_x(x_j, t_n) = \frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2\Delta x} + \mathcal{O}\left(\Delta x^2\right)$$

Portanto os termos truncados são quadráticos, logo o método *Leapfrog* é de ordem 2.

A molécula do método é ilustrada na Figura 1. O nome do método se refere a uma brincadeira de crianças na qual os jogadores saltam sobre as costas uns dos outros. O método tem esse nome já que em cada interação não se utiliza o valor obtido na mesma posição no instante anterior, mas utiliza-se o valor de dois passos atrás no tempo, ou seja há "saltos" de posições assim como há no jogo de criança. Estes saltos de posições indicam que o método usa valores em células distintas da grade, sugerindo assim que ele tenha uma ordem mais alta do que

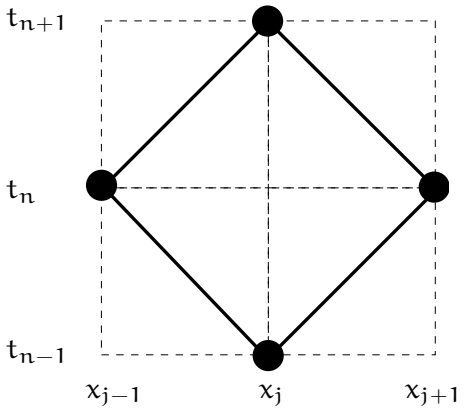


Figura 1: Molécula do método.

métodos que usam valores de vértices de uma mesma célula, que são geralmente são métodos de ordem 1, e como verificamos, o *Leapfrog* é um método de ordem 2.

Para saber se o método é dissipativo ou dispersivo vamos fazer uma análise da equação modificada. Seja u uma função contínua que satisfaça a equação de diferenças, teremos então

$$\frac{u(x_j, t_{n+1}) - u(x_j, t_{n-1})}{2\Delta t} + c\frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2\Delta x} = 0 \tag{1}$$

Pela série de Taylor temos

$$\begin{aligned}\frac{u(x_j, t_{n+1}) - u(x_j, t_{n-1})}{2\Delta t} &= \frac{1}{2\Delta t}\left(2\Delta t u_t(x_j, t_n) + 2\frac{\Delta t^3}{6}u_{ttt}(x_j, t_n) + \mathcal{O}\left(\Delta t^5\right)\right) \\ &= u_t(x_j, t_n) + \frac{\Delta t^2}{6}u_{ttt}(x_j, t_n) + \mathcal{O}\left(\Delta t^4\right).\end{aligned}$$

Da mesma forma temos

$$\frac{u(x_{j+1}, t_n) - u(x_{j-1}, t_n)}{2\Delta x} = u_x(x_j, t_n) + \frac{\Delta x^2}{6}u_{xxx}(x_j, t_n) + \mathcal{O}\left(\Delta x^4\right).$$

Substituindo esses resultados na equação 1 temos

$$\begin{aligned}u_t + \frac{\Delta t^2}{6}u_{ttt} + \mathcal{O}\left(\Delta t^4\right) + c\left(u_x + \frac{\Delta x^2}{6}u_{xxx} + \mathcal{O}\left(\Delta x^4\right)\right) &= 0 \\ u_t + cu_x = \frac{-\Delta t^2}{6}u_{ttt} + \mathcal{O}\left(\Delta t^4\right) - c\left(\frac{\Delta x^2}{6}u_{xxx} + \mathcal{O}\left(\Delta x^4\right)\right),\end{aligned}$$

onde todos os termos estão avaliados em (x_j, t_n).

Vemos então que o método *Leapfrog* produz uma aproximação de ordem mais alta do que 2 para a equação

$$u_t + cu_x = \frac{-\Delta t^2}{6}u_{ttt} - c\left(\frac{\Delta x^2}{6}u_{xxx}\right). \tag{2}$$

Pela série de Taylor, e pelo fato que u satisfaz a equação de diferenças teremos

$$u_t + cu_x = \mathcal{O}\left(\Delta t^2\right) + \mathcal{O}\left(\Delta x^2\right).$$

Teremos então

$$\begin{aligned} u_t &= -cu_x + \mathcal{O}\left(\Delta x^2\right) \\ u_{tt} &= -cu_{xt} = -cu_{tx} \\ &= -c\left(-cu_x + \mathcal{O}\left(\Delta x^2\right)\right)_x \\ &= c^2 u_{xx} \\ u_{ttt} &= c^2 u_{xxt} = c^2 u_{txx} \\ &= c^2\left(-cu_x + \mathcal{O}\left(\Delta x^2\right)\right)_{xx} \\ &= -c^3 u_{xxx} \end{aligned}$$

Substituindo este resultado na equação 2, obtemos

$$\begin{aligned} u_t + cu_x &= \frac{-\Delta t^2}{6}\left(-c^3 u_{xxx}\right) - c\left(\frac{\Delta x^2}{6} u_{xxx}\right) \\ &= \frac{c^3 \Delta t^2}{6}\left(u_{xxx}\right) - c\left(\frac{\Delta x^2}{6} u_{xxx}\right) \\ &= \left(\frac{c^3 \Delta t^2 - c\Delta x^2}{6}\right) u_{xxx} \end{aligned}$$

Chegamos assim à conclusão de que o método é dispersivo.

Exercício 2

Item a

O fenômeno de *aliasing* acontece quando um sinal é amostrado de forma que a restrição de duas ou mais ondas com diferentes frequências são equivalentes em pontos da grade de amostragem.

Podemos verificar este fenômeno através da função $f(x) = \text{sen}(2\pi x)$. Utilizando a amostragem $v_k = f(x_k) = f(1,05k)$, para $k = 0, 1, \ldots$, estaremos utilizando um espaçamento $\Delta x = 1,05$. Neste caso teremos

$$\begin{aligned} v_k &= \text{sen}(2\pi x_k) \\ &= \text{sen}(2\pi(1,05k)) \\ &= \text{sen}(2\pi k + 0.1\pi k) \\ &= \text{sen}(2\pi k)\cos(0.1\pi k) + \cos(2\pi k)\,\text{sen}(0.1\pi k) \\ &= \text{sen}(0.1\pi k) \end{aligned}$$

Portanto a amostra será equivalente a uma função seno com frequência menor do que a de $f(x)$. A Figura 2 mostra uma comparação entre os pontos obtidos através desta amostragem (em linha sólida) com a função f (linha tracejada).

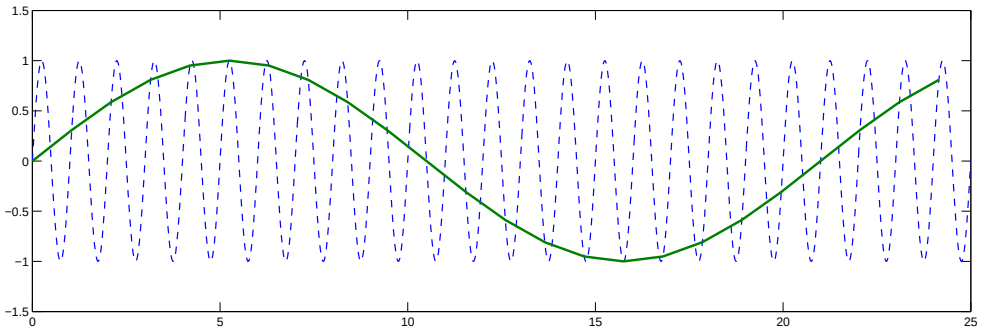


Figura 2: Ilustração de *aliasing*.

Este gráfico foi gerado em MATLAB com os comandos (e mais algumas edições):

```
x = 0:0.01:25;
y = 0:1.05:25;
plot(x, sin(2*pi*x), y, sin(2*pi*y))
```

Conforme utilizamos outros espaçamentos, obtemos outros resultados. Por exemplo, ao tomarmos $x_k = k$ ($\Delta x = 1$), as amostras serão $v_k = \text{sen}(2\pi k) = 0$, ou seja, a frequência original é perdida e a amostra parece ser de uma função constante. A Figura 3 mostra o resultado obtido em MATLAB (usando mesmo padrão de linhas da figura anterior; o código também é semelhante, mudando apenas a variação em y).

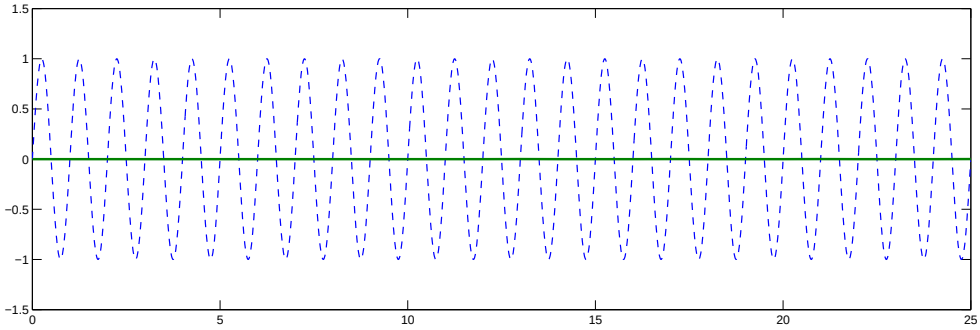


Figura 3: Ilustração de *aliasing*.

Item b

Seja u uma função periódica de período 2π. Vamos tomar N amostras igualmente espaçadas para x no intervalo $[0, 2\pi)$, fazendo $\Delta x = 2\pi/N$ e $x_j = j\Delta x$, com $j = 0, \ldots, N-1$.

Neste exercício vamos derivar e interpolar u ao mesmo tempo, isto pode ser feito utilizando as propriedades:

$$\mathcal{F}[u](x + x_0) = e^{ikx_0}\hat{u}(k)$$
$$\mathcal{F}[u_x](x) = ik\hat{u}(k),$$

onde $\mathcal{F}[u](x) = \hat{u}(k)$ é a transformada de Fourier de u. Queremos encontrar os valores $u_x(x_j + 0,5\Delta x)$, com $j = 0, \ldots, N-1$. Usando as propriedades acima teremos:

$$\mathcal{F}[u_x](x + 0,5\Delta x) = ike^{0,5\Delta xik}\hat{u}(k).$$

Calculando esta transformada, poderemos então obter o resultado desejado através da transformada de Fourier inversa.

O seguinte código de MATLAB implementa esta operação

```
function v = spectral_interp_deriv(u, x0)
    N = size(u,2);
    i = complex(0,1);
    k = [-N/2:N/2-1];

    ui = fftshift(fft(u));
    vi = i*ui.*k.*(exp(i*k*x0));
    v = real(ifft(ifftshift(vi)));
end
```

A função `fft` retorna a transformada de Fourier discreta com frequências de 0 a $N-1$, ou seja, no intervalo $[0, 2\pi/\Delta x)$, usamos `fftshift` para trabalhar com frequências de $-N/2$ a $N/2-1$, isto é, no intervalo $[-\pi/\Delta x, \pi/\Delta x)$.

O seguinte código aplica a operação descrita acima à função $u(x) = \sin(x)$, discretizada em 16 pontos, gerando o gráfico da Figura 4.

```
% Sampling on x.
N = 16;
deltax = 2*pi/N;
x = deltax*[0:N-1];

% Discretization of function.
u = sin(x);

% Interpolate and derive u.
v = spectral_interp_deriv(u, 0.5*deltax);

% Plot results.
xx = [0:0.01:2*pi];
plot(x,u, '*', x+0.5*deltax, v, 'o', xx, cos(xx),'--');
ylim([-1.1, 1.1])
```

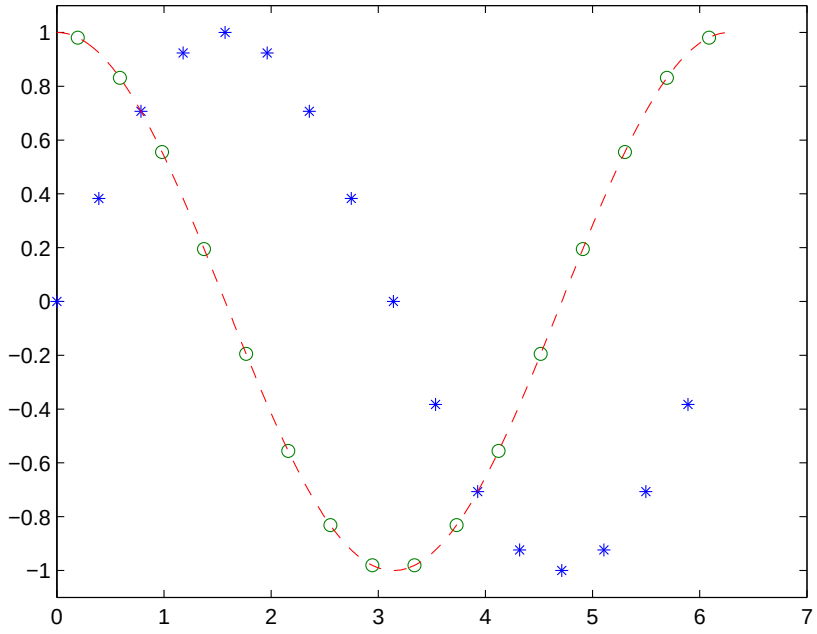


Figura 4: Interpolação e diferenciação espectral.

Nesta figura podemos ver os valores $u(x_j)$ com $*$, os valores encontrados para $u_x(x_j + 0,5\Delta x)$ com $\circ$, e também valores de $u_x(x) = \frac{d(\sin(x))}{dx} = \cos(x)$ numa malha mais fina, em linha tracejada. Pode-se perceber que os valores de $u_x(x_j + 0.5\Delta x)$ conferem com o resultado obtido pela função `spectral_interp_deriv`.

O próximo exemplo utiliza a função $u(x) = \sin(x) + 2\cos(x-1) - 4\,\mathrm{sen}(2x)$, também discretizada em 16 pontos no intervalo $[0, 2\pi)$, gerando o gráfico da Figura 5.

```
% Sampling on x.
N = 16;
deltax = 2*pi/N;
x = deltax*[0:N-1];

% Discretization of function.
u = sin(x) + 2*cos(x-1) - 4*sin(2*x);

% Interpolate and derive u.
v = spectral_interp_deriv(u, 0.5*deltax);

% Plot results.
xx = [0:0.1:2*pi];
plot(x,u, '*', x + 0.5*deltax, v, 'o', ...
    xx, cos(xx) - 2*sin(xx-1)-8*cos(2*(xx)),'--');
```

Esta figura utiliza o mesmo padrão de apresentação dos dados utilizado na figura anterior. Neste exemplo também é possível comparar o resultado obtido através da solução exata da derivada com aquele obtido pela função spectral_interpolation.

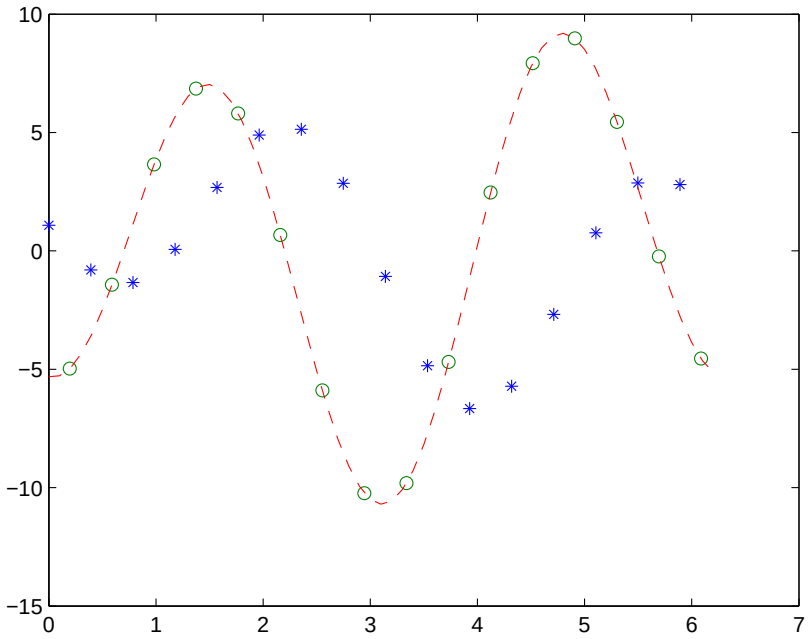


Figura 5: Interpolação e diferenciação espectral.

Item c

Seja $u \in L^2(\mathbb{R})$, e v a função de grade em $\Delta x\mathbb{Z}$ definida por $v_j = u(x_j)$. A versão discreta do teorema de Paley-Wiener nos diz que se u puder ser estendida como uma função analítica numa tira complexa $|\mathrm{Im} z| < a$, $a > 0$, com $\|u(\cdot + iy)\| < c$ uniformemente para todos $y \in (-a, a)$, $c > 0$, então

$$|\hat{v}(k) - \hat{u}(k)| = \mathcal{O}\left(e^{-\pi(a-\epsilon)/\Delta x}\right) \qquad \Delta x \to 0 \qquad \forall \epsilon > 0.$$

Pela identidade de Parseval temos também

$$\|\hat{u}\|_{\Delta x}^2 \equiv \int_{-\pi/\Delta x}^{\pi/\Delta x} |\hat{u}(k)|^2 dk = \sum_{j=-\infty}^{\infty} |u_j|^2 \Delta x \equiv \|u\|_{\Delta x}^2.$$

Juntando o resultado de Paley-Wiener com a identidade de Parseval vemos que u também estará próximo de sua discretização v. Vamos verificar este fato calculando a derivada da função $f(x) = 1/(1 + \mathrm{sen}^2(x/2))$, que é analítica na faixa do plano complexo $|\mathrm{Im} z| < 1$, e portanto está dentro das hipóteses do teorema de Paley-Wiener.

Vamos comparar a derivada de $f(x)$ calculada por um método espectral com a derivada exata (a menos de erro de arredondamento). Isto pode ser feito calculando-se a diferença entre a derivada de f em N pontos pelo método espectral e pela fórmula desta derivada calculada manualmente, e variando o valor de N.

O seguinte código em MATLAB executa esta tarefa.

```
eps = 1e-10;
for j = 1:20
     % Number of samples.
     N = j*2+1;

     % Sampling on x.
     deltax = 2*pi/N;
     x = deltax*[0:N-1];

     % Discretization of function.
     u = 1./(1+sin(x/2).^2);

     % Calculate derivative using spectral method.
     uprime = spectral_derivative(u);

     % Calculate the true derivative.
     actualuprime = -sin(x/2).*cos(x/2).*u.^2;

     % Get difference.
     dif(j) = norm(uprime - actualuprime);

     % Exponential function to be compared with difference.
     expon(j) = exp(-pi*(1-eps)/deltax);
end

% Plot results.
plot([1:20]*2+1,dif, '-o', [1:20]*2+1, expon, '--');
ylim([-0.01, 0.2])
```

Onde a função `spectral_derivative` é dada pelo código:

```
function du = spectral_derivative(u)
    M = size(u,2);
    i = complex(0,1);
    k = [-M/2:M/2-1];

     ui = fftshift(fft(u));
     vi = i*ui.*k;
     du = real(ifft(ifftshift(vi)));
end
```

Ao executar, é gerado o gráfico da Figura 6, onde encontram-se com $\circ$ ligados por linhas contínuas as diferenças entre a derivada de f calculada analiticamente com a aproximação encontrada pelo método espectral, encontra-se também em linhas tracejadas o gráfico da função $e^{-\pi(1-\epsilon)/\Delta x}$, com $\epsilon = 10^{-10}$, calculada utilizando o Δx correspondente em cada iteração. Percebe-se que esta diferença decai rapidamente, numa taxa maior do que a exponencial (isto pode ser verificado também para qualquer outro valor positivo de ϵ).

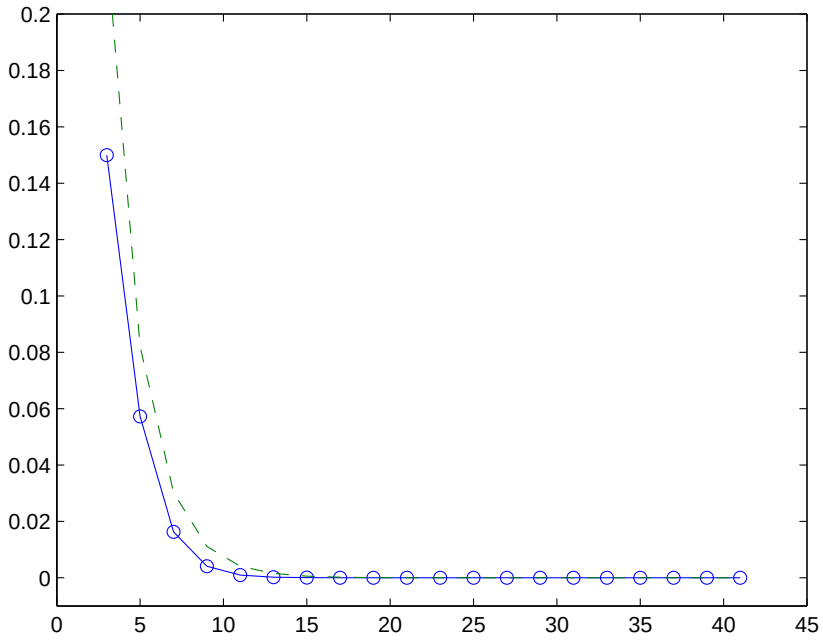


Figura 6: Análise entre o número de pontos amostrados e a diferença entre a derivada de $f(x)$ exata e a obtida pelo método espectral.

Exercício 3

3.1

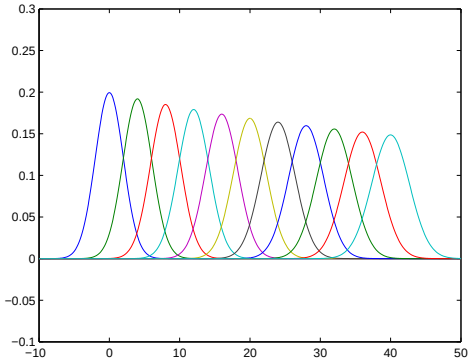
Vamos analisar como o esquema *upwind* se comporta ao violarmos (ou não) a CFL.

A CFL é safisfeita quando $\sigma \leqslant 1$. No caso $\sigma < 1$ a função muda de forma ao longo do tempo, teremos um resultado que é influenciado pelo termo difusivo do *upwind*. O caso ideal para este método é quando $\sigma = 1$, quando a velocidade da onda é igual à velocidade numérica de cada método. O termo difusivo dispersivo se anula neste caso, portanto a solução não apresenta alterações numéricas no formato da função. Vamos verificar a aplicação do método para a equação $u_t + 2u_x = 0$.

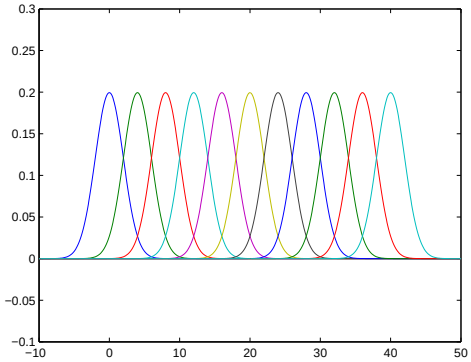
Vamos utilizar inicialmente $\Delta x = 0,1$. A Figura 7(a) mostra o resultado obtido com 2000 passos no tempo com o método aplicado a uma Gaussiana, utilizando $\Delta t = 0,01$, temos $\sigma = 0,2$ (para facilitar a visualização a figura ilustra apenas o resultado em alguns passos no tempo), pode-se perceber que a solução decai no tempo.

A Figura 7(b) mostra o resultado obtido com 400 passos no tempo com o método aplicado à mesma Gaussiana utilizando $\Delta t = 0,05$, neste caso teremos $\sigma = 1$, ou seja, é o caso ideal, a solução não apresenta decaimento espúrio, como é possível verificar pelo gráfico.

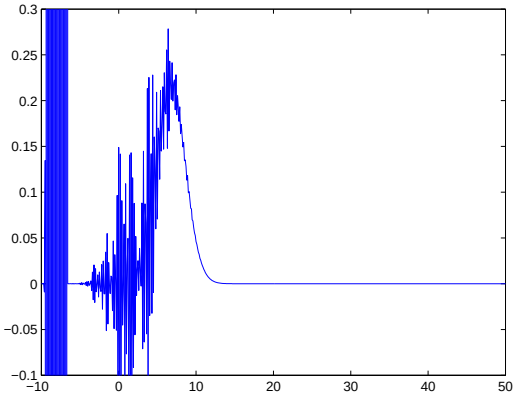
Para valores maiores de Δt, teremos $\sigma > 1$, e estaremos fora da região de estabilidade do método. A Figura 7(c) ilustra o resultado do método com $\Delta t = 0,1$ ($\sigma = 2$) após 34 passos no tempo, onde já é possível perceber a instabilidade do método, após este ponto a forma da Gaussiana é alterada rapidamente.



(a) $\Delta x = 0,1$, $\Delta t = 0,01$, $\sigma = 0,2$.



(b) $\Delta x = 0,1$, $\Delta t = 0,05$, $\sigma = 1$.



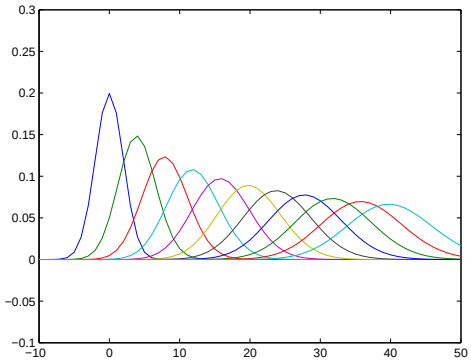
(c) $\Delta x = 0,1$, $\Delta t = 0,1$, $\sigma = 2$.

Figura 7: Análise da CFL.

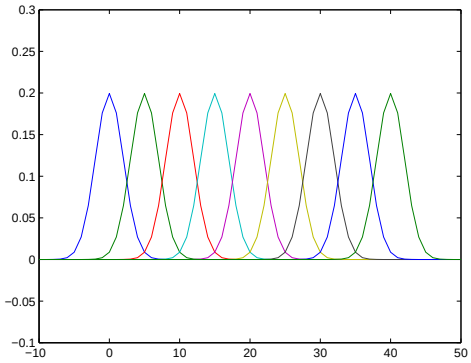
Vamos considerar agora um passo maior no espaço. Seja $\Delta x = 1$. Testando com $\Delta t = 0,1$, teremos $\sigma = 0,2$, a Figura 8(a) ilustra o resultado de 400 passos no tempo com estes parâmetros (a figura mostra apenas alguns passos de tempo facilitar a visualização). Pode-se perceber claramente o efeito difusivo neste caso.

Se fizermos agora $\Delta t = 0,5$ voltaremos à condição ideal $\sigma = 1$, situação esta mostrada na Figura 8(b) após 40 passos no tempo, percebe-se que embora a solução esteja mais grosseira em relação ao caso $\Delta x = 0,1$, como estamos na situação ideal a solução não apresenta difusão.

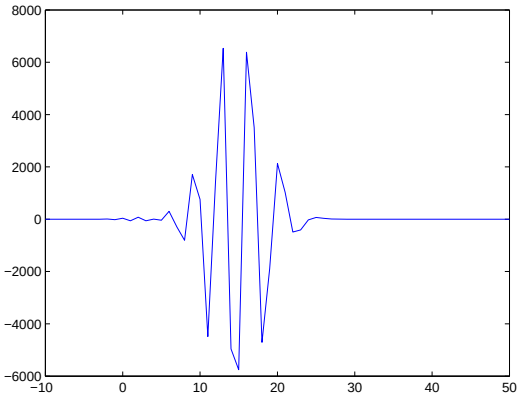
Quando $\Delta t > 0,5$ estaremos violando a CFL. A Figura 8(c) mostra o resultado após 20 iterações do método usando $\Delta t = 0,1$ ($\sigma = 2$), onde percebe-se que a forma do gráfico da Gaussiana foi bastante modificada.



(a) $\Delta x = 1, \Delta t = 0,1, \sigma = 0,2$.



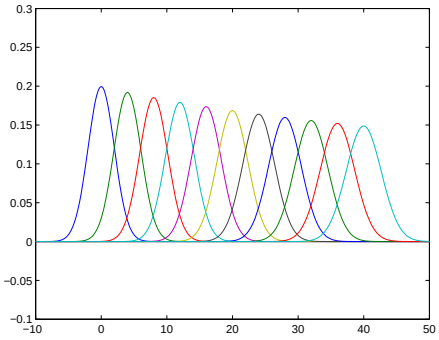
(b) $\Delta x = 1, \Delta t = 0,5, \sigma = 1$.



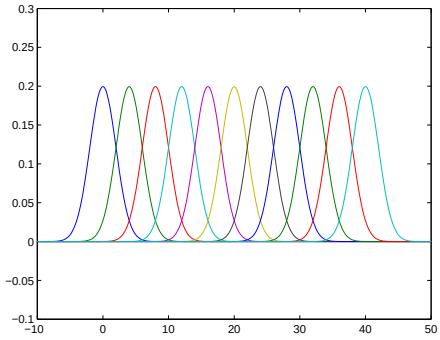
(c) $\Delta x = 0,1, \Delta t = 0,1, \sigma = 2$.

Figura 8: Análise da CFL.

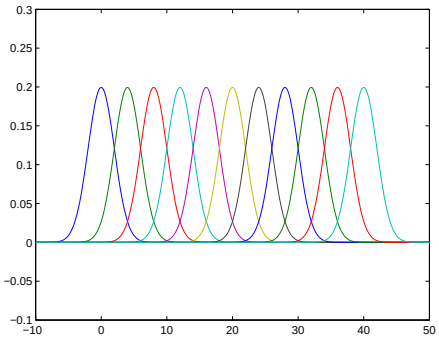
Através do método Semi-Lagrangeano podemos violar a CFL mantendo a estabilidade. As Figuras 9(a) a 9(c) ilustram o comportamento deste método utilizando os mesmos parâmetros da Figura 7. Pode-se perceber que para $\sigma \leqslant 1$ o resultado é equivalente ao obtido pelo método *upwind*, mas para $\sigma > 1$ o Semi-Lagrangeano se mantém estável, isto acontece mesmo para valores altos de σ, a Figura 9(d) mostra o resultado para $\sigma = 100$, foram necessários apenas 4 passos no tempo para chegar ao mesmo instante de tempo final $t = 20$ que foi usado nas outras figuras, neste caso, mesmo com σ grande não foi observado nenhum fenômeno espúrio pois as características "cairam" em posições próximas aos vértices da grade de discretização. Porém em casos onde as características caem entre vértices da grade pode-se observar uma difusão numérica, como ilustra a Figura 9(e).



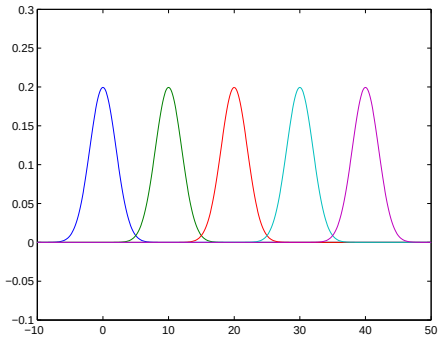
(a) $\Delta x = 0,1$, $\Delta t = 0,01$, $\sigma = 0,2$.



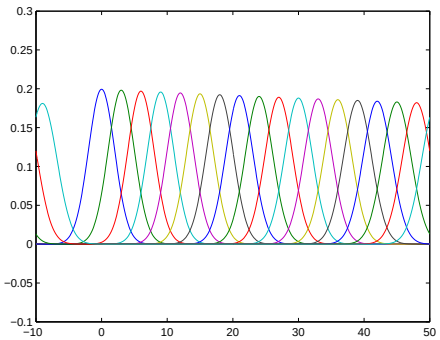
(b) $\Delta x = 0,1$, $\Delta t = 0,05$, $\sigma = 1$.



(c) $\Delta x = 0,1$, $\Delta t = 0,1$, $\sigma = 2$.



(d) $\Delta x = 0,1$, $\Delta t = 5$, $\sigma = 100$.



(e) $\Delta x = 0,1$, $\Delta t = 0,1$, $c = 1,5$, $\sigma = 1,5$.

Figura 9: Método Semi-Lagrangeano.

3.2

Vamos considerar agora o *upwind* com condições de contorno periódicas. O seguinte código em MATLAB executa este procedimento.

```
% Solves the wave equation u_t + c*u_x = 0.
function u = wave_upwind(c, u0, x, niter, deltat)
     N = size(x,2);
     deltax = (x(N) - x(1))/(N-1);
     sigma = c*deltat/deltax;

     u = zeros(niter, N);
     u(1,:) = u0;

     j = 2:N;
     for n = 2:niter
          u(n,j) = u(n-1,j) - sigma*(u(n-1,j) - u(n-1,j-1));

          % Periodic Boundary condition.
          u(n,1) = u(n-1,1) - sigma*(u(n-1,1) - u(n-1,N));
     end
end
```

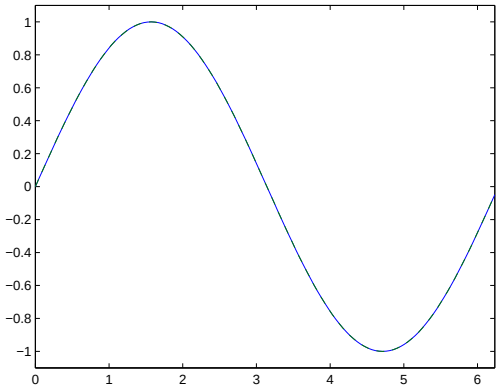
Queremos analisar a velocidade de fase numérica, isto pode ser feito comparando a solução numérica com a solução real da equação da onda.

Tomemos a onda com perfil inicial $u_0(x) = \text{sen}(x)$, com x entre 0 e 2π. A solução da equação da onda $u_t + cu_x = 0$ é $u(x,t) = \text{sen}(x - ct)$, que possui velocidade de fase c. A Figura 10 ilustra o resultado em diversos instantes de tempo, foi utilizado $c = 0,2454$, $\Delta x = 2\pi/128$, e $\Delta t = 0,1$ ($\sigma = 0,5$), a solução exata está plotada em linhas tracejadas, e a numérica em linhas contínuas. Podemos observar que a solução numérica mantém a velocidade de fase da solução exata, mesmo após várias "voltas" pelo domínio. Os gráficos foram gerados por comandos como

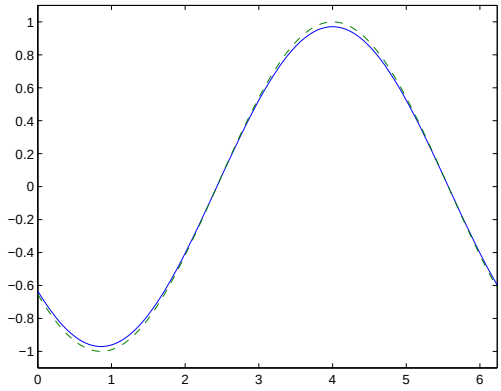
```
...
u = wave_upwind(c, u0, x, niter, deltat);

n = 10000;
plot(x,u(n,:), x,sin(x-c*(n-1)*deltat), '--');
axis([x(1), x(N), -1.1, 1.1]);
```

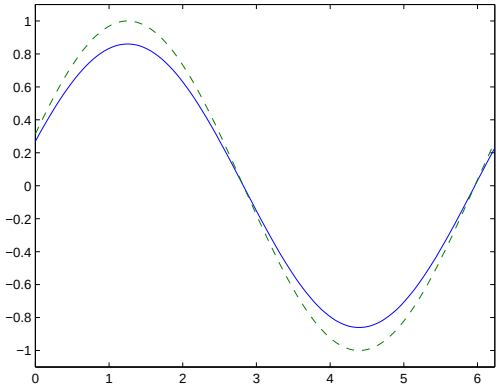
Onde as variáveis devem ser devidamente definidas indicando o número de iterações, a amostragem em x, a função inicial u_0, a variação Δt de cada passo de tempo e a velocidade da onda c.



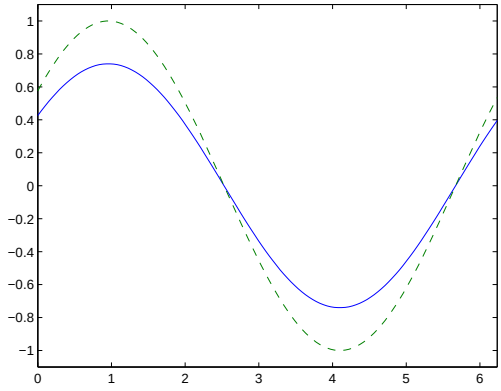
(a) Instante inicial.



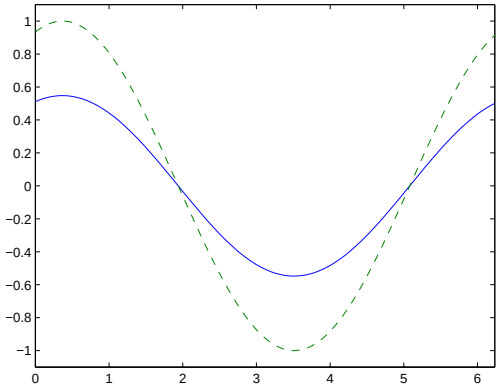
(b) 100 passos.



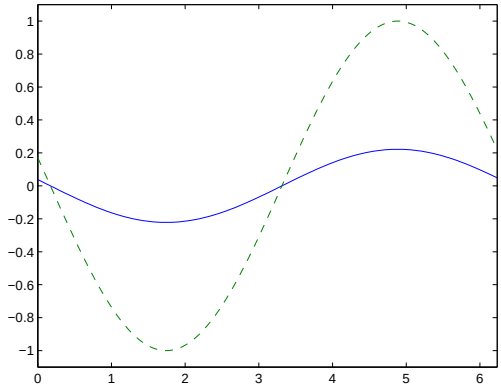
(c) 500 passos.



(d) 1000 passos.



(e) 2000 passos.



(f) 5000 passos.

Figura 10: Comparação entre velocidades de fase, solução real versus solução numérica.

3.3

Neste exercício vamos implementar um método espectral para resolver cada um dos problemas: a equação da onda $u_t + cu_x = 0$, a equação do calor $u_t = cu_{xx}$ e a KdV linearizada $u_t + cu_{xxx} = 0$.

Equação da onda

O nosso método espectral consiste em calcular u_x utilizando uma aproximação espectral, desta forma a equação da onda se torna

$$u_t = -c\bar{u}_x,$$

onde $(\bar{u}_x)$ é a aproximação espectral de u_x. Portanto o problema se resume a resolver um sistema de EDOs. Para resolver este sistema podemos utilizar um esquema de diferenças finitas, é importante usar um método que nos dê uma boa aproximação, quanto menor a ordem do método mais impreciso é resultado obtido no cálculo de u_x. Neste exercício utilizamos o método de Runge-Kutta de quarta ordem, nos dando bons resultados.

O seguinte código em MATLAB resolve a equação da onda:

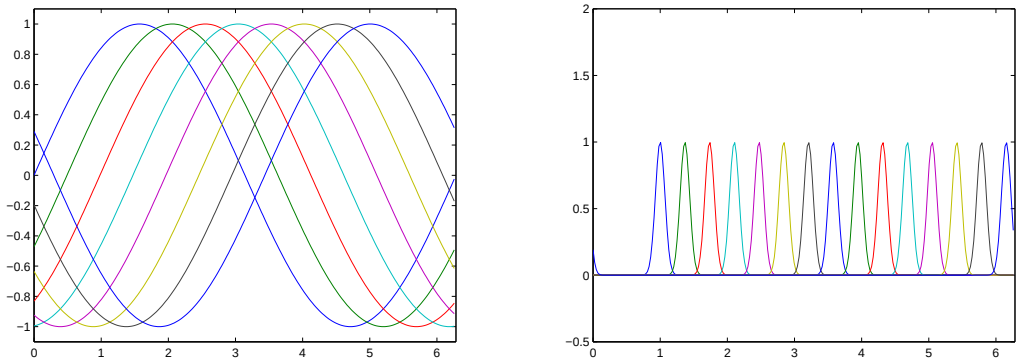
```
% Solves the wave equation u_t + c*u_x = 0, u(0,x) = u0(x).
function u = wave_spectral(c, u0, niter, deltat)
     N = size(u0,2);

     % Initialize solution.
     u = zeros(niter, N);
     u(1,:) = u0;

     % Runge-Kutta iterations.
     for n = 2:niter
          oldu = u(n-1,:);
          F1 = -c*deltat*(spectral_derivative(oldu));
          F2 = -c*deltat*(spectral_derivative(oldu+0.5*F1));
          F3 = -c*deltat*(spectral_derivative(oldu+0.5*F2));
          F4 = -c*deltat*(spectral_derivative(oldu+F3));
          u(n, :) = oldu + 1/6*(F1 + 2*F2 + 2*F3 + F4);
     end
end
```

A função `spectral_derivative` é a mesma utilizada na questão 2 (item c).

A Figura 11 ilustra o resultado obtido ao aplicar este método a um perfil inicial senoidal e ao perfil gaussiano $u_0(x) = e^{-100(x-1)^2}$. Foi utilizada uma velocidade $c = 2$, amostragem de 256 pontos igualmente espaçados entre 0 e 2π.



(a) $u_0(x) = \text{sen}(x)$.

(b) $u_0(x) = e^{-100(x-1)^2}$.

Figura 11: Método espectral, equação da onda.

Equação do calor

Para cálcular derivadas mais altas, observamos que

$$\mathcal{F}\left[\frac{\partial^n u}{\partial x^n}\right](x) = (ik)^n \hat{u}(k)$$

Portanto podemos usar esta fórmula no código:

```
function du = spectral_n_derivative(u, n)
    M = size(u,2);
    i = complex(0,1);
    k = [-M/2:M/2-1];

     ui = fftshift(fft(u));
     vi = (i*k).^n.*ui;
     du = real(ifft(ifftshift(vi)));
end
```

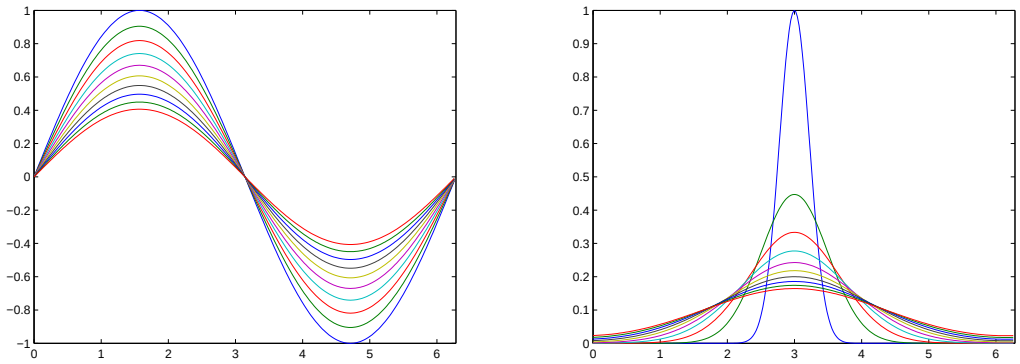
Podemos então usar esta função para resolver a equação do calor de forma análoga à utilizada na equação da onda.

```
% Solves the heat equation u_t = c*u_x, u(0,x) = u0(x).
function u = heat_spectral(c, u0, niter, deltat)
     N = size(u0,2);

     % Initialize solution.
     u = zeros(niter, N);
     u(1,:) = u0;

     % Runge-Kutta iterations.
     for n = 2:niter
          oldu = u(n-1,:);
          F1 = c*deltat*(spectral_n_derivative(oldu,2));
          F2 = c*deltat*(spectral_n_derivative(oldu+0.5*F1,2));
          F3 = c*deltat*(spectral_n_derivative(oldu+0.5*F2,2));
          F4 = c*deltat*(spectral_n_derivative(oldu+F3,2));
          u(n, :) = oldu + 1/6*(F1 + 2*F2 + 2*F3 + F4);
     end
end
```

A Figura 12 mostra o resultado da aplicação deste código às funções iniciais $u_0(x) = \text{sen}(x)$ e $u_0(x) = e^{-10(x-3)^2}$, utilizando $c = 10^{-3}$, $\Delta t = 0,1$ em amostras com 256 pontos cada.



(a) $u_0(x) = \text{sen}(x)$.

(b) $u_0(x) = e^{-100(x-3)^2}$.

Figura 12: Método espectral, equação do calor.

Equação KdV linearizada

Da mesma forma podemos aplicar o método descrito para resolver uma KdV linearizada $u_t + cu_{xxx} = 0$, basta utilizar a função `spectral_n_derivative` para calcular a terceira derivada. Desta forma teremos a função:

```
% Solves the kdv equation u_t = c*u_x, u(0,x) = u0(x).
function u = kdv_spectral(c, u0, niter, deltat)
     N = size(u0,2);

     % Initialize solution.
     u = zeros(niter, N);
     u(1,:) = u0;

     % Runge-Kutta iterations.
     for n = 2:niter
          oldu = u(n-1,:);
          F1 = c*deltat*(spectral_n_derivative(oldu,3));
          F2 = c*deltat*(spectral_n_derivative(oldu+0.5*F1,3));
          F3 = c*deltat*(spectral_n_derivative(oldu+0.5*F2,3));
          F4 = c*deltat*(spectral_n_derivative(oldu+F3,3));
          u(n, :) = oldu + 1/6*(F1 + 2*F2 + 2*F3 + F4);
     end
end
```

A Figura 13 mostra o resultado da aplicação deste código à função inicial $u_0(x) = e^{-10(x-3)^2}$, utilizando $c = 10^{-5}$, $\Delta t = 0,1$ em amostras com 256 pontos cada. Para facilitar a visualização, a figura mostra apenas o resultado de alguns passos no tempo.

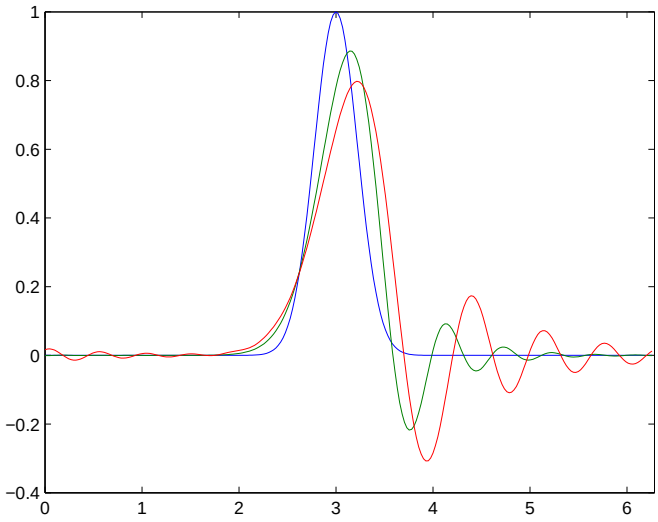


Figura 13: Método espectral, equação KdV linearizada.