Capítulo 23
Integração e diferenciação

Integração

- O cálculo da integral de uma função é o processo de se aproximar a área compreendida entre a curva da função e o eixo das abcissas; outro nome para esse processo é *quadratura* da função.

- O **MATLAB** possui funções pré-definidas para aproximar numericamente a integral de funções, algumas das quais veremos aqui. As funções alvo podem estar descritas em arquivos M ou em funções *in line*.
  - $quad$: integral simples;
  - $quadl$: integral simples;
  - $dblquad$: integral dupla.

Integração

- Considere a função $humps$ e seu gráfico construído no arquivo *mm2301.m*:

```
::::::::::::::
mm2301.m
::::::::::::::
x = -1:.17:2;
y=humps(x);

xx=[x;x;x];
yy=[y;zeros(size(y));y];
xx=xx(:)';
yy=yy(:)';

xi=linspace(-1,2);
yi=humps(xi);
plot(xx,yy,':',xi,yi,[-1,2],[0 0],'k')
title('Figure 23.1: Integration Approximation with Trapezoids')
```

- A soma das áreas trapezoidais aproxima-se da integral da função.

A função $trapz$

- A função $trapz$ aproxima a integral via método trapezoidal.
- Para o nosso exemplo, utilizando valores tabulados da função humps, podemos obter uma aproximação pela área dos trapézios. A precisão varia com a discretização utilizada:

```
>> x = -1:.17:2;
>> y = humps(x);
>> area = trapz(x,y)
area =
  25.91740000817554
```

```
>> x = linspace(-1,2,100);
>> y = humps(x);
>> area = trapz(x,y)
area =
  26.34473119524596
```

A função $cumtrapz$

- Calcula uma aproximação da integral cumulativa pelo método trapezoidal. É usada quando queremos calcular a integral em função de x, isto é, dado x_1 conhecido:

$$\int_{x_1}^{x} f(x)d(x).$$

- Vamos ver $cumtrapz$ no nosso exemplo:

```
% Arquivo mm2302.m
x = linspace(-1,2,100);
y = humps(x);
z = cumtrapz(x,y);
size(z)
plotyy(x,y,x,z)
grid on
xlabel('x')
ylabel('humps(x) and integral of humps(x)')
title('Figure 23.2: Cumulative Integral of humps(x)')
```

As funções $quad$ e $quadl$

- Considerando a aproximação pelo método trapezoidal, pode ser difícil determinar uma largura ótima de trapézio.

- As funções matemáticas $quad$ e $quadl$ baseiam-se no conceito matemático de *quadratura*. Do Help:
  - $quad$: Numerically evaluate integral, adaptive Simpson quadrature. Q = QUAD(FUN,A,B) tries to approximate the integral of function FUN from A to B to within an error of 1.e-6 using recursive adaptive Simpson quadrature.
  - $quadl$: Numerically evaluate integral, adaptive Lobatto quadrature. Q = QUADL(FUN,A,B) tries to approximate the integral of function FUN from A to B to within an error of 1.e-6 using high order recursive adaptive quadrature.

- A função a ser integrada deve aceitar um vetor como argumento de entrada, retornando um vetor de dados de saída. Isto significa utilizar operadores pontuados (.$*, , /,$, etc).

$quad$ e $quadl$ - exemplo

- Retomemos o nosso exemplo:

```
>> disp(z(end)) % resultado de cumtrapz
  26.34473119524596

>> disp(quad(@humps,-1,2))
  26.34496049276723

>> disp(quadl(@humps,-1,2))
  26.34496047137897
```

- A precisão de $quad$ e $quadl$ é de oito dígitos significativos, mas a função $quadl$ é mais rigorosa que a $quad$.
- Algumas vezes, e dependendo da versão de **MATLAB**, estas duas funções podem retornar o mesmo resultado.
- É possível especificar uma tolerância para o erro como um quarto argumento de entrada. O padrão é 1e-6.

A função $dblquad$

- Utilizada para calcular a integral dupla. Isto é

$$\int_{y_{min}}^{y_{max}} \int_{x_{min}}^{x_{max}} f(x,y)dxdy.$$

- Vamos exemplificar o uso de $dblquad$ com a seguinte função:

```
function z=myfun(x,y)
%MYFUN(X,Y) an example function of two variables
z = sin(x).*cos(y) + 1;
```

- cuja representação gráfica pode ser obtida através dos comandos:

mm2303.m

```
x = linspace(0,pi,20); y = linspace(-pi,pi,20);
[xx,yy] = meshgrid(x,y); zz = myfun(xx,yy);
mesh(xx,yy,zz)
xlabel('x'), ylabel('y'), title('Figure 23.3: myfun.m plot')
```

$dblquad$ - exemplo

- O volume desta função pode ser estimado usando a função $dblquad$.

```
>> area = dblquad('myfun',0,pi,-pi,pi)
area =
  19.73920880217871

>> errel = (area-2*pi^2)/(2*pi^2)
errel =
    -1.799825775391955e-16
```

- O resultado é bastante preciso.
- A função $dlbquad$ faz chamadas à função $quad$.
- Também é possível estabelecer uma tolerância com um argumento de entrada a mais (a tolerância padrão é a mesma que a das funções $quad$ e $quadl$: 1e-6).

Diferenciação

- Diferenciação numérica é mais difícil que integração.
- A diferenciação representa a inclinação de uma função.
- Diferenciação é muito mais sensível a alterações, ainda que pequenas, na função.

- A diferenciação numérica é evitada, sempre que possível, devido a essas dificuldades inerentes.
- Se os dados são obtidos experimentalmente, em geral, é melhor realizar um ajuste de curva polinomial, usando quadrados mínimos, e então derivar o polinômio resultante.
- Outra opção é ajustar splines cúbicas aos dados e depois encontrar a representação por splines da derivada (vide capítulo 20).

Diferenciação

- Considere novamente o exemplo de ajuste de curva do capítulo 19:

mm2304.m

```
x = [0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
y = [-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2]; % data
n = 2; % order of fi t
p = polyfi t(x,y,n) % fi nd polynomial coeffi cients
xi = linspace(0,1,100);
yi = polyval(p,xi); % evaluate polynomial
plot(x,y,'-o',xi,yi,'--')
xlabel('x'), ylabel('y=f(x)')
title('Figure 23.4: Second Order Curve Fitting')
```

- Os dados retornados pela seqüência de comandos acima, que representa os coeficientes do polinômio, são

```
p =
  -9.81083916083916  20.12929370629370  -0.03167132867133
```

Diferenciação

- Podemos usar a função $polyder$ para encontrar a derivada analítica no nosso exemplo:

```
>> polyder(p)
ans =
 -19.62167832167832  20.12929370629370
```

- Vamos agora trabalhar na diferenciação numérica. Lembremos que:

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \approx \frac{\Delta x}{\Delta y} = \frac{f(x+\Delta x) - f(x)}{\Delta x},$$

que é a *diferença finita avançada*, que é o mesmo que

$$\frac{dy}{dx} \approx \frac{y_{i+1} - y_i}{\Delta x}$$

A função $diff$

- Esta função calcula a diferença entre os elementos de um vetor.
- Pode ser utilizada para calcular a derivada aproximada a partir de dados tabulados que descrevem certa função.
- Vamos usá-la para calcular a derivada aproximada pelo método das diferenças finitas.
- Como $diff$ calcula a diferença entre os elementos de um vetor, a saída possui um elemento a menos que a entrada. Assim, para imprimir o gráfico, um dos elementos de x devem ser desconsiderados.
  - Se o elemento x_1 for desconsiderado obtemos a *diferença finita atrasada* ($y' \approx (y_i - y_{i-1})/\Delta x)$).
  - Se o elemento descartado for x_n obtemos a *diferença avançada*.

$diff$ com dados imprecisos

- Retornemos ao nosso exemplo:

mm2305.m

```
x = [0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
y = [-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2]; % data
n = 2; % order of fi t
p = polyfi t(x,y,n); % fi nd polynomial coeffi cients
dp=polyder(p);
dyp=polyval(dp,x);
dy = diff(y)./diff(x); % compute differences and use array division
xd = x(1:end-1);% create new x axis array since dy is shorter than y
plot(xd,dy,x,dyp,':')
ylabel('dy/dx'), xlabel('x')
title('Figure 23.5: Forward Difference Derivative Approximation')
```

- Comparado o resultado obtido com o da aproximação polinomial, observamos como a aproximação por diferenças finitas pode ser ruim (principalmente quando os dados são experimentais).

$diff$ com dados mais precisos

- Se os dados utilizados forem mais precisos, a utilização de $diff$ pode conduzir a resultados aceitáveis, pelo menos para fins de visualização.

mm2305.m

```
x = linspace(0,2*pi);
y = sin(x);
dy = diff(y)/(x(2)-x(1));
xd = x(2:end);
plot(x,y,xd,dy)
xlabel('x'), ylabel('sin(x) and cos(x)')
title('Figure 23.6: Backward Difference Derivative Approximation')
```

- Os componentes do exemplo anterior são igualmente espaçados; porisso dividimos por $x(2) - x(1)$. Se este não fosse o caso teríamos que usar $diff$ no denominador.
- Descartamos o primeiro elemento de x, o que conduziu à formula da diferença atrasada.

Método das diferenças finitas

- O erro da derivada anterior é pequeno:

```
>> disp(max(abs(cos(xd)-dy)))
   0.0317
```

- Podemos também usar diferenças centradas. Neste caso, precisamos executar as operações vetoriais necessárias diretamente:

$$\frac{dy}{dx} \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} \qquad \frac{dy(x_n)}{dx} \approx \frac{f(x_{n+1}) - f(x_{n-1})}{x_{n+1} - x_{n-1}}$$

```
>> dy = (y(3:end)-y(1:end-2)) /(x(3)-x(1));
>> xd = x(2:end-1);
>> disp(max(abs(cos(xd)-dy)))
    6.708600433286138e-04
```

- Neste caso, as derivadas do primeiro e último ponto do intervalo não possuem uma aproximação.

Dados bidimensionais

- A função $gradient$ utiliza diferenças centradas para estimar a inclinação em cada direção, em cada ponto tabulado.
- A saída possui o mesmo número de pontos da entrada porque diferenças avançadas são usadas nos pontos iniciais e atrasadas nos finais.
- Esta função é empregada principalmente para visualização gráfica de dados.

mm2307.m

```
[x,y,z] = peaks(20); % simple 2-D function
dx = x(1,2) - x(1,1); % spacing in x direction
dy = y(2,1) - y(1,1); % spacing in y direction
[dzdx,dzdy] = gradient(z,dx,dy);
contour(x,y,z)
hold on
quiver(x,y,dzdx,dzdy)
hold off
title('Figure 23.7: Gradient Arrow Plot')
```

A função $del2$

- Em algumas situações é útil saber a curvatura de uma superfície, isto é, a mudança de inclinação em cada ponto.
- Esta curvatura é estimada pela função $del2$, que calcula a aproximação discreta do Laplaciano.

$$\nabla^2 z(x,y) = \frac{d^2 z}{dx^2} + \frac{d^2 z}{dy^2}.$$

- Considere o exemplo a seguir em que a curvatura absoluta da superfície influencia a cor da superfície.

mm2307.m

```
[x,y,z] = peaks; % default output of peaks
dx = x(1,2) - x(1,1); % spacing in x direction
dy = y(2,1) - y(1,1); % spacing in y direction
L = del2(z,dx,dy);
surf(x,y,z,abs(L))
shading interp
title('Figure 23.8: Discrete Laplacian Color')
```