2.1 Equações diferenciais ordinárias da primeira ordem

A solução de uma equação diferencial é uma *função* que satisfaz a equação diferencial sobre algum intervalo aberto. Uma equação diferencial *ordinária* tem a forma geral

$$\varphi(x,y,y',y'',y''',..., d^ny/dx^n) = 0 \qquad (1)$$

Esta equação é de n-ésima ordem é tem somente uma variável independente, x.

A função y = F(x) é uma solução de (1) se ela é n vêzes diferençável e se satisfaz a Eq. (1).

As equações y':= dy/dx = x+y; y''+(1-y^2)y' + y = 0 são exemplos de equações diferenciais ordinárias. Uma equação diferencial φ(x,y(x),dy/dx) = 0 pode geralmente ser escrita como

$$dy/dx = y' = f(x,y) \qquad (2)$$

As equações diferenciais ordinárias têm várias soluções. Para escolher uma única solução, são necessárias informações adicionais, normalmente n para uma eq. de n-ésima ordem. Se todas as n condições adicionais forem especificadas para um mesmo valor de x, por exemplo x_o, temos um *Problema de Valor Inicial*, **PVI**. Caso estas n condições adicionais sejam dadas para mais de um valor de x, temos um *Problema de Valor de Contorno*, **PVC**.

O gráfico de uma solução da equação diferencial chama-se de *curva solução*. (Uma curva solução é também uma curva integral.)

Existem métodos gráficos e numéricos para obter uma idéia sobre a forma da solução, e aos quais pode-se recorrer se não existe nenhuma fórmula explícita da solução ou se a fórmula é complicada demais para ser útil.

Para MuPAD existe um "Arbeitsblatt" (*folha de trabalho*) com o título *Das Euler-Cauchy Verfahren*... (*O método de Euler-Cauchy...*) de Agnes Havasi e Kai Gehrs, onde os autores explicam detalhadamente o método gráfico.

Para as aplicações na Física precisamos de métodos numéricos que aproximam uma solução exata com praticamente qualquer precisão.

2.1.1 Método de Euler para y' = f(x,y)

Vamos considerar agora a equação diferencial ordinária de primeira ordem y' = f(x,y) junto com uma condição inicial $y(x_0) = y_0$.

O nosso objetivo é obter numericamente uma solução y(x) que satisfaça a equação diferencial e as condições iniciais. O método numérico mais simples é o de Euler (1707-1783) e baseia-se na idéia de aproximar os valores de y(x) pela reta tangente, como é ilustrado na figura 2.1-1.

(Euler descreveu o seu método em 1768 em "*Institutiones Calculi Integralis*, sectio secunda, capu VII")

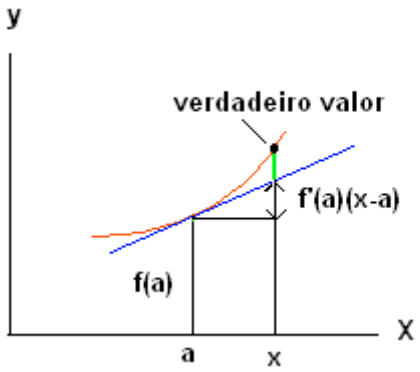

Fig.: 2.1-1

Se fizermos um zoom no gráfico de uma função lisa de uma variável y = f(x) perto de um ponto x = a, o gráfico parece cada vez mais uma reta e assim se torna indistinguível de sua reta tangente nesse ponto. A inclinação da reta tangente é a derivada f'(a) e a reta passa pelo ponto (a,f(a)) de modo que sua equação é

$$y = f(a) + f'(a)(x-a) \qquad (3)$$

Agora aproximamos os valores de f pelos valores-y da reta tangente. Para valores de x próximos de a, podemos escrever para o verdadeiro valor f(x) da função f no ponto x

$$f(x) \approx f(a) + f'(a)(x-a) \qquad (4)$$

O traço verde na figura indica o erro que fazemos aproximando f(x) pelo valor de f(a) + f'(a)(x-a). O fato de f ser aproximadamente uma função linear em x perto de a é expresso dizendo que f é *localmente linear* perto de x = a.

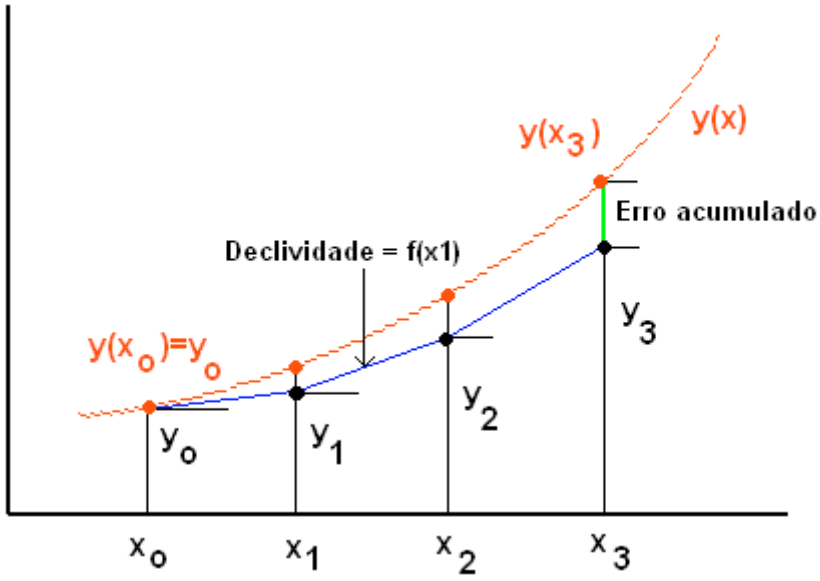

Fig.: 2.1-2

Vimos que o método de Euler se baseia na suposição que a reta tangente à curva solução (curva integral) de y' = f(x,y) com $y(x_0) = y_0$ em $(x_i,y(x_i))$ aproxima a curva solução sobre o intervalo $[x_i , x_{i+1}]$.

Visto que a inclinação (ou declividade) da curva solução em $(x_i, y(x_i))$ é $y'(x_i) = f(x_i, y(x_i))$, a equação da reta tangente à curva integral em $(x_i, y(x_i))$ é

$$y = y(x_i) + f(x_i, y(x_i))(x-x_i) \qquad (5)$$

Fazendo $x = x_{i+1} = x_i + h$, obtemos

$$y_{i+1} = y(x_i) + h\cdot f(x_i, y(x_i)) \qquad (6)$$

sendo h o tamanho do passo e y_{i+1} o valor de y *até a reta tangente* no ponto x_{i+1}. y_{i+1} tomamos como uma aproximação a $y(x_{i+1})$.

Já que foi dado $y(x_0) = y_0$, podemos usar (6) com i = 0 para calcular y_1

$$y_1 = y(x_0) + h\cdot f(x_0, y(x_0)) = y_0 + h\cdot f(x_0, y_0) \qquad (7)$$

Agora fazemos i =1 e Eq. (6) se torna

$$y_2 = y(x_1) + h \cdot f(x_1, y(x_1)) \qquad (8)$$

mas esta equação não é útil pois não conhecemos $y(x_1)$. (Só $y(x_0)$ está conhecido e o chamamos y_0.) Bem, vem aqui a aproximação:

O valor $y(x_1)$, que não conhecemos, substituimos pelo valor y_1, que só chega até a reta tangente e que é, no exemplo da figura 2.1-2, nitidamente inferior ao valor real da função y(x) em x_1.

$$y(x_2) \approx y_2 = y_1 + h \cdot f(x_1, y_1)$$

No próximo passo vamos substituir $y(x_2)$ por y_2:

$$y_3 = y_2 + h \cdot f(x_2, y_2)$$

O processo pode ser repetido até que o valor desejado de x seja alcançado.

Em geral, o método de Euler começa com o valor conhecido $y(x_0) = y_0$ e calcula y_1, y_2, ... y_n por meio da fórmula de recorrência

$$\mathbf{y_{i+1} = y_i + h \cdot f(x_i, y_i)}, \quad 0 <= i <= n - 1 \qquad (9)$$

(Trata-se de uma fórmula de recorrência pois no caso de uma iteração, que é um caso especial da recorrência, se busca, em geral, um valor limite para o processo. Mas, o uso das palavras neste sentido estrito não é muito comum.)

Os números y_1, y_2, y_3 etc. são aproximações de $y(x_1)$, $y(x_2)$, $y(x_3)$ etc.

Exemplo:

y' = 1-x+4y, y(0) = 1; solução exata: $y(x) = x/4 - 3/16 + (19/16) \cdot e^{4x}$

Solução segundo Euler: (h = 0.1)

f(x,y) = 1 - x +4y

$y_1 = y_0 + h \cdot f(x_0, y_0) = 1 + 0.1 \cdot (1-0+4 \cdot 1) = 1+0,5 =$ **1,5** ; $x = x_1 = h = 0.1$

valor exato: y(0,1) = 1,609041828

$y_2 = y_1 + h \cdot f(x_1, y_1) = 1,5 + 0,1 \cdot (1-0,1+4 \cdot 1,5) = 1,5 +0,69 =$ **2,19**; $x = x_2 = 0,2$

valor exato: y(0,2)= 2.505329853

$y_3 = y_2 + h \cdot f(x_2, y_2) = 2,19 + 0,956 =$ **3,146**; $x = x_3 = 0,3$

valor exato: y(0,3) = 3,830138846

Compare estes cálculos com os resultados do seguinte programa:

Os resultados aparecem como: i, x_{i+1}, y_{i+1}, $y(x_{i+1})$

Programa 1:

```
x:=0:y:=1:h:=0.01://Euler para y'=f(x,y)
DIGITS:=10:
for i from 0 to 5 do
f:=1-x+4*y://f(xi,yi)
y:=y+h*f://yi+1
x:=x+h://xi+1
//xi:=x-h://valor anterior xi
//yi:=y-h*f:// valor anterior yi

F0:=x/4-3/16+(19/16)*exp(4*x)://solução analítica
print(i,x,y,F0):
end_for:
```

i x_{i+1} y_{i+1} $y(x_{i+1})$

0, 0.1, 1.5, 1.609041828

1, 0.2, 2.19, 2.505329853

2, 0.3, 3.146, 3.830138846

3, 0.4, 4.4744, 5.794226004

4, 0.5, 6.32416, 8.712004117

5, 0.6, 8.903824, 13.05252195

2.1.2 Métodos melhorados de Euler

O método de Euler na formulação $\mathbf{y_{i+1} = y_i + h \cdot f(x_i,y_i)}$ utiliza sempre a inclinação da reta tangente à curva solução no começo do intervalo $[x_i , x_{i+1}]$ e supõe que esta inclinação permaneça constante durante o intervalo inteiro. Mas, normalmente vemos que a curva solução de y(x) muda a inclinação da reta tangente à curva solução no intervalo $[x_i , x_{i+1}]$. Pode-se esperar um melhoramento do método se se toma a inclinação no centro do intervalo o quando se toma uma média de vários inclinações em $[x_i , x_{i+1}]$.

Como **Método melhorado de Euler** ou **Método de Heun** (pron.: hoin) se conhece um procedimento, onde se calcula primeiro a média das inclinações das retas tangentes à curva integral nos extremos do intervalo $[x_i , x_{i+1}]$, depois se segue os passos que nos levaram às fórmulas (5) até (9). Voltemos, então, à Eq. (5) e substituamos a inclinação $(f(x_i, y(x_i))$ pela média

$$m_i = (f(x_i,y(x_i)) + f(x_{i+1},y(x_{i+1}))/2 \qquad (10)$$

A Eq. (6) reza agora $y_{i+1} = y(x_i) + h\ (f(x_i,y(x_i)) + f(x_{i+1},y(x_{i+1})))/2$ e é uma aproximação a $y(x_{i+1})$. Como antes, aproximamos $y(x_i)$ por seu valor aproximado y_i quando i>0. Pero $y(x_{i+1})$ também será, normalmente, desconhecido e vamos substituí-lo pela aproximação $y(x_{i+1}) \approx y_{i+1} = y_i + h\ f(x_i,y_i)$.

A fórmula de recursão do *método melhorado de Euler* ou a *fórmula de Heun* fica finalmente assim

$$y_{i+1} = y_i + h/2 \cdot (f(x_i,y_i) + f(x_{i+1}, y_i + h\ f(x_i,y_i))) \qquad (11)$$

Para o cálculo prático é útil a introdução das seguintes expressões

$$k_{1i} = f(x_i,y_i) \qquad (12)$$

$$k_{2i} = f(x_i+h, y_i + hk_{1i}) \qquad (13)$$

$$y_{i+1} = y_i + h\ (k_{1i} + k_{2i})/2 \qquad (14)$$

O método de Heun produz resultados bastante mais exatos do que o método de Euler simples, especialmente una variante del processo que melhora em cada ponto x_i primeiro os resultados por meio de uma iteração interna antes de avançar ao próximo ponto x_{i+1}. No programa de Heun com iteração realizamos isso.

(O método de Heun possui um erro de truncamento global da ordem de $O(h^2)$ e podemos verificar que, se diminuirmos o tamanho do passo de h para h/2, o erro será reduzido de um fator 1/4, e assim sucessivamente. O símbolo $O(h^2)$ quer dizer que o método de Heun coincide com a série de Taylor até os termos de ordem h^2.)

- ```
  reset():Método de Heun simples
  Heun:=proc(h,passos)
  local i,n,x,y,dy,f,k1,k2,F0;
  begin
  x:=0:y:=1:
  DIGITS:=6:
  f:=(x,y)->-2*y+x^3*exp(-2*x):
  tabela:=array(1..passos,1..4):
  for i from 1 to passos do
  k1:=f(x,y):
  x:=x+h:
  k2:=f(x,y+h*k1):
  dy:=h*(k1+k2)/2:
  y:=y+dy:
  F0:=exp(-2*x)*(x^4+4)/4://solução analítica
  tabela[i,1]:=x:
  tabela[i,2]:=y:
  tabela[i,3]:=F0:
  tabela[i,4]:=F0-y:
  end_for:
  return(tabela)
  end_proc:

  Heun(0.1,5)
  ```

  Resultados:

X	Y	exato	F0-y
0.1,	0.820041,	0.818751,	-0.00128972
0.2,	0.672734,	0.670588,	-0.00214627
0.3,	0.552598,	0.549923,	-0.00267466
0.4,	0.455161,	0.452205,	-0.00295597
0.5,	0.376681,	0.373628,	-0.00305369
  ```

O programa está construido como procedimento (procedure). A declaração das variáveis locais não é estritamente necessária. Porém, um bom estilo de programação pede o uso de variáveis locais para evitar conflitos no momento em que você quer incorporar este procedimento como módulo dentro de outro programa.

O método *modificado* de Euler utiliza a reta tangente com a inclinação (aproximada) no centro do intervalo. Chega-se de esta maneira à seguinte fórmula de recursão (também chamada de *fórmula do ponto médio*).

$$y_{i+1} = y_i + h\, f(x_i+h/2,\ y_i + f(x_i,y_i)\cdot h/2) \qquad (14)$$

(A equação da reta tangente pelo ponto (x_0,y_0) é, no método de Euler, $y = y_0 + f(x_0,y_0)(x-x_0)$. O *método do ponto médio* substitui a inclinação $f(x_0,y_0)$ pela do centro do intervalo, ou seja pela inclinação da reta tangente no ponto $(x_0+h/2,\ y(x_m))$. O valor $y(x_m)$ é desconhecido, mais pode ser aproximado pelo valor médio (média aritmética) $(y_0 + y_1)/2$. O valor desconhecido de y_1 calculamos por meio da equação de Euler: $y_1 = y_0 + h\cdot f(x_0,y_0)$. A fórmula modificada será, então, para o primeiro intervalo

$$y_1 = y_0 + h\, f(x_m,\ (y_0 + (y_0+h\cdot f(x_0,y_0))) = y_0 + h\cdot f(x_0+h/2,\ y_0+f(x_0,y_{,0})\cdot h/2)$$

daí segue em geral a Eq. (14).

No seguinte programa encontramos este algoritmo incorporado e documentado.

```
reset()://Método do ponto médio
Midpoint:=proc(h,passos)
local i,n,x,y,dy,f,k1,k2,ym,xm,F0;
begin
x:=0:y:=1:
DIGITS:=8:
f:=(x,y)->-2*y+x^3*exp(-2*x):
tabela:=array(1..passos,1..4):
for i from 1 to passos do
k1:=f(x,y)://inclinação em xi
ym:=y+h*k1/2://y_médio calculado com Euler
xm:=x+h/2://x_médio
k2:=f(xm,ym)://inclinação no centro (aprox.)
x:=xm+h/2://xi+1
y:=y+h*k2://yi+1
F0:=exp(-2*x)*(x^4+4)/4://fórmula exata
tabela[i,1]:=x:
tabela[i,2]:=y:
tabela[i,3]:=F0:
tabela[i,4]:=F0-y:

end_for:
return(tabela)
end_proc:

Midpoint(0.1,5)
```

```
0.1, 0.82001131, 0.81875122, -0.0012600891

0.2, 0.67265111, 0.67058817, -0.0020629394

0.3, 0.55246799, 0.54992298, -0.0025450119

0.4, 0.45500468, 0.45220467, -0.0028000139

0.5, 0.37652114, 0.37362756, -0.0028935785
```

Você pode ver que o método é bastante exato e, neste exemplo, mesmo ligeiramente melhor do que o método de Heun. (Estruturas parecidas)

Resultados:

X	Y	exato	F0 - Y
0.1,	0.82001131,	0.81875122,	-0.0012600891
0.2,	0.67265111,	0.67058817,	-0.0020629394
0.3,	0.55246799,	0.54992298,	-0.0025450119
0.4,	0.45500468,	0.45220467,	-0.0028000139
0.5,	0.37652114,	0.37362756,	-0.0028935785

Agora vamos unir os métodos de Euler e Heun para obter o seguinte algoritmo bastante efetivo:

- `reset():`**`//Método Euler_Heun`**

```
Euler_Heun:=proc(h,passos)
local i,t,y,v,f,F0;
begin
t:=1:y:=2:v:=0:
DIGITS:=8:
f:=(t,y)->-v/t+4*y/t^2:
tabela:=array(1..passos,1..4):

for i from 1 to passos do
```

```
va:=v:
a:=f(t,y):
v:=va+h*a:
y:=y+(h/2)*(va+v):
t:=t+h:
F0:=t^2+1/t^2:
tabela[i,1]:=t:
tabela[i,2]:=v:
tabela[i,3]:=y:
tabela[i,4]:=F0:
end_for:

return(tabela)

end_proc:

Euler_Heun(0.05,20)
```

```
1.05, 0.4, 2.01, 2.0095295
1.1, 0.74557823, 2.0386395, 2.0364463
1.15, 1.0486535, 2.0834952, 2.0786437
1.2, 1.3181442, 2.1426652, 2.1344444
1.25, 1.5608139, 2.2146391, 2.2025
1.3, 1.7818552, 2.2982059, 2.281716
1.35, 1.9852993, 2.3923847, 2.3711968
1.4, 2.1743085, 2.4963749, 2.4702041
1.45, 2.3513868, 2.6095173, 2.5781243
1.5, 2.5185344, 2.7312653, 2.6944444
```

Veja agora o *algoritmo de Heun com iterção*:

- ```
  reset():://Método de Heun com iteração interna
  Heun:=proc(h,passos)
  local i,j,n,x,y,ya,ye,dy,f,k1,k2,F0;
  begin
  x:=0:y:=1:
  DIGITS:=6:
  ```
  ```

```
f:=(x,y)->2*(x^2+y):
tabela:=array(1..passos,1..3):
for i from 1 to passos do
ya:=y:// antigo valor de y
k1:=f(x,y):
x:=x+h:
ye:=y+h*k1:
//print(ye)://valor Euler

for j from 1 to 4 do//4 iterações internas
k2:=f(x,ye):
dy:=h*(k1+k2)/2:
y:=ya+dy:
//print(y):

ye:=y:
end_for:

ya:=y:
F0:=1.5*exp(2*x)-x^2-x-0.5://solução analítica
//print(i,x,y,F0):

tabela[i,1]:=x:
tabela[i,2]:=y:
tabela[i,3]:=F0:// a tabela mostra x,y,valor exato

end_for:

return(tabela)

end_proc:

Heun(0.1,5)
```

```
0.1, 1.22333, 1.2221
0.2, 1.50073, 1.49774
0.3, 1.84867, 1.84318
0.4, 2.28726, 2.27831
0.5, 2.84109, 2.82742
```

O **método de Runge-Kutta** é o rei entre aqueles métodos apropriados para resolver os problemas de valor inicial (C. Runge 1856-1927, W. Kutta 1867-1944). Os seus atrativos são simplicidade, alta precisão e versatilidade. Não é de estranhar ouvir que também MuPAD tem uma espécie de "Runge-Kutta" incorporado. A idéia detrás do método RK é bastante parecido ao raciocínio detrás do *método melhorado de Euler* (fórmula de Heun), mas agora calculamos a função f(x,y) não apenas duas vezes, como no método de Heun, antes quatro vezes, reduzindo assim o erro global de truncamento para $O(h^4)$! (O método de RK, que mais adiante vamos usar, é também conhecido como RK de quarta ordem. O método de Heun é chamado de RK de segunda ordem e o método de Euler como RK de primeira ordem.)

Não vamos desenvolver rigorosamente as fórmulas do método RK. Vou, porém, apresentar o algoritmo como se fosse uma simples modificação do método de Euler.

Algoritmo Runge-Kutta de quarta ordem para y' = f(t,y).

(Já que nas aplicações temos geralmente o tempo como variável independente, utilizaremos t em vez de x e v em vez de y'. O símbolo <v> indica o valor médio de 4 derivadas -velocidades- do método de RK.)

$$t_{n+1} = t_n + h$$

$$y_{n+1} = y_n + h\langle v\rangle,$$

onde

$$\langle v\rangle := (v_1+2v_2+2v_3+v_4)/6 \qquad (15)$$

As quatro derivadas calculam-se segundo o seguinte esquema:

$$v_1 := f(t,y)$$

$$v_2 := f(t+h/2,\ y+ v_1 \cdot h/2)$$

$$v_3 := f(t+h/2,\ y+ v_2 \cdot h/2)$$

$$v_4 := f(t+h,\ y+ v_3 \cdot h) \qquad (16)$$

(Mais adiante será muito simples a generalização do método para equações de segunda ordem y" = f(t,x,y').)

Agora vamos ver como é simples a implementação computacional das esquemas (15) e (16).

```
reset():
//Runge-Kutta y'(t,y)
t:=0:
y:=0:
h:=0.1:
DIGITS:=5:
f:=(t,y)->1/(1+t^2)-2*y^2:
for i from 1 to 10 do
v1:=f(t,y):
v2:=f(t+h/2,y+v1*h/2):
v3:=f(t+h/2,y+v2*h/2):
v4:=f(t+h,y+v3*h):

y:=y+h*(v1+2*v2+2*v3+v4)/6:

t:=t+h:
F0:=t/(1+t^2):

print(t,y,F0,F0-y):
end_for:
```

```
0.1, 0.099009, 0.09901, 4.1895e-7
0.2, 0.19231, 0.19231, 9.5851e-7
0.3, 0.27523, 0.27523, 1.7105e-6
0.4, 0.34482, 0.34483, 2.6161e-6
0.5, 0.4, 0.4, 3.5041e-6
```

Os resultados mostram que os erros são apenas da ordem de 10^{-6}, no caso do método de Euler encontramos erros da ordem de 10^{-2}. O método de Euler tem, porém, o seu valor didático e ajuda no entendimento dos métodos mais exatos.