Lab 5 – EDO e Integração (Queda Livre)

Material necessário:

- Bola
- Celular com câmera para filmar

Experiência:

O objetivo é encontrar a solução da equação de segunda ordem que descreve o movimento de uma bola em queda livre, utilizando métodos para solucionar EDOs e extrapolando o movimento com MMQ.

A EDO de segunda ordem que descreve o sistema é dada por

$$\frac{d^2y}{dt^2} = -g,$$

onde y é a distância percorrida pela bola e t o tempo.

Sendo assim, o sistema de EDOs de primeira ordem equivalente será

$$\begin{bmatrix} y` \\ z` \end{bmatrix} = \begin{bmatrix} z \\ -g \end{bmatrix}$$

Em que *g* é a aceleração da gravidade e *z* é uma variável auxiliar.

Um componente do grupo irá lançar a bola para cima e esperar esta tocar no chão. O segundo integrante do grupo deverá posicionar a câmera de modo a filmar todo o possível trajeto da bola sem mover a câmera. Lembre de enquadrar algum referencial de medida conhecida.

No Windows, abrir o vídeo com o Media Player Classic e capturar uma imagem da bola ao atingir a altura máxima (velocidade em "*y*" igual a zero) e uma no momento em que a bola toca no chão. Lembre de anotar o tempo com precisão de milissegundos. Para capturar as imagens use o atalho "Alt + i". Descubra a altura (em pixels) que a bola toca o chão.

Precisamos do valor da gravidade em px/s², sabendo que g=9,81 m/s², vamos converter para m/s² e depois achar uma correspondência entre px (pixel) e metro. Utilize algum referencial para calcule o valor da gravidade em pixels/ s².

Descubra quando a bola irá tocar no chão utilizando os métodos de resolução de EDO e MMQ.

O relatório deve ter, além dos códigos, as chamadas utilizadas para cada implementação.

1- Gráfico da solução (momento em que a bola toca no chão) pelo método de EULER com passo h=0.1s e h=0.001s, partindo da primeira imagem. (dois gráficos)

2 – Encontre a solução por MMQ linear e de segundo grau utilizando a malha de pontos gerada no item anterior. (quatro gráficos)

3- Gráfico da solução (momento em que a bola toca no chão) pelo método de RK2 com passo h=0.1s e h=0.001s, partindo da primeira imagem. (dois gráficos)

4 – Encontre a solução por MMQ linear e de segundo grau utilizando a malha de pontos gerada no item anterior. (quatro gráficos)

5- Gráfico da solução (momento em que a bola toca no chão) pelo método de RK4 com passo h=0.1s e h=0.001s, partindo da primeira imagem. (dois gráficos)

6 – Encontre a solução por MMQ linear e de segundo grau utilizando a malha de pontos gerada no item anterior. (quatro gráficos)

7 – Comparação e comentários entre os métodos e a solução real do problema.

Programa:

Os algoritmos abaixo implementam os métodos de EULER de primeiro grau e segundo grau, e RK2 de primeiro grau, respectivamente.

Algoritmo do método de Euler:
Entrada: malha [a,b] , espaçamento h e valor inicial y0
Saída: vetores x e y com pares de pontos da solução do PVI

```
x = a:h:b;
y(1) = y0 ;
para i=2:comprimento(x)
      y(i) = y(i-1) + h*df(x(i-1),y(i-1))
fim para
```

Teste do código:

$$\frac{dy}{dt} = (12 - y)$$

Condições iniciais: y(0)=0;
Chamada: [x,y]=euler(a,b,h,y0)

```
[x,y]=euler(0,10,1,0)
resultado:
x= [1 2 3 4 5 6 7 8 9 10]
y=[ 0 12 12 12 12 12 12 12 12 12]
plot(x,y,'o-')
```

Novo teste:

```
[x,y]=Euler(0,10,0.25,0)
plot(x,y,'o-r')
```

Algoritmo do método de Euler (2ª. ordem):
Entrada: malha [a,b], espaçamento h e valor inicial y0, z0..
Saída: vetores x, y e z com pares de pontos da solução do PVI.
 	
x = a:h:b;
y(1) = y0;
z(1) = z0;
para i=2:comprimento(x)
    y(i)=y(i-1)+h*df1(x(i-1),y(i-1),z(i-1));
    z(i)=z(i-1)+h*df2(x(i-1),y(i-1),z(i-1));
fim para
	Teste do código:
	$\frac{dy}{dt} = z$
	$\frac{dz}{dt} = (12 - y - z)$
	
	Condições iniciais: y(0)=z(0)=0;
	Chamada: [x,y,z]=Euler2(a,b,h,y0,z0)
	[x,y,z]=Euler(0,10,1,0,0)
	resultado:
	x= [1 2 3 4 5 6 7 8 9 10]
	y=[0 0 12 24 24 12 0 0 12 24 24]
	z=[0 12 12 0 -12 -12 0 12 12 0 -12]
	plot(x,y,'o-')

Algoritmo do método de RK2 (Euler Modificado):
Entrada: malha [a,b], espaçamento h e valor inicial y0;
Saída: vetores x e y com pares de pontos da solução do PVI
 	
x = a:h:b;
y(1) = y0;
para i=2:comprimento(x)
    k1 = df(x(i-1),y(i-1))
    k2 = df(x(i), y(i-1) + h*k1)
    y(i) = y(i-1) + h/2*(k1 + k2)
fim para
	Teste do código:
	$\frac{dy}{dt} = (12 - y)$
	Condições iniciais: y(0)=0;
	Chamada: [x,y]=RK2(a,b,h,y0)
	
	[x,y]=RK2(0,10,1,0)
	resultado:
	x= [1 2 3 4 5 6 7 8 9 10]
	y=[0 6 9 10.5 11.25 11.63 11.81 11.91 11.95 11.98 11.99]
	plot(x,y,'o-')
	
	Novo teste:
	[x,y]=RK2(0,10,0.25,0)
	plot(x,y,'o-r')

Algoritmo Ajuste Linear Polinomial
 	
Entrada: Vetores dos pares de pontos "x" e "y" que definem o polinômio "f" a ser ajustado e o grau "k" do polinômio.
Saída: Coeficientes "a_i" do polinômio.
 	
n = comprimento(x)
para i = 1 até n
        para j = 1 até k+1
        v(i,j) = $x(i)^{j-1}$
        fim_para
        b(i) = y(i)
fim_para
a=gauss(v^T*v,v^T*y) //ou a = inv(v^Tv)*(v^Tb)
	Teste de Código:
	
	x=[1 7 14 16 23 25];
	y=[19 - 4 - 33 - 40 - 71 - 75];
	Chamada: a=poli(x,y,3)
	Saída:
	a=[22.077585
	- 3.1762855
	- 0.0821912
	0.0020815]