MAC0209 - modelagem e simulação

anotações para P1

semana 1

MODELOS MATEMÁTICOS E SOLUÇÕES COMPUTACIONAIS

fenômeno → sensor → dados observados (modelo matemático; teoria científica) → análise computacional → síntese computacional (simulação) → visualização

MÉTODO CIENTÍFICO

1. coloque-se uma questão
2. formule uma hipótese
3. formule um experimento
4. observe (coleta de dados)
5. analise os resultados
6. volte para o passo 2 se a hipótese não for correta
7. relate os resultados.

COMO SE FAZ UM MODELO?

1. identifique o problema e as questões científicas
2. identifique as variáveis relevantes (e irrelevantes)
3. identifique o tipo de modelo matemático
    - função
    - equação de diferença
    - equação diferencial, etc.
4. monte o modelo (relacionando as variáveis)
5. simplifique até que seja tratável
    - analiticamente: as equações precisam ser solúveis
    - computacionalmente: o problema precisa ser solúvel

semana 1

6. resolva as equações
7. responda as questões científicas
8. modifique o modelo, compare soluções
9. estude a sensibilidade do modelo
10. compare as saídas com as observações experimentais.

VELOCIDADE MÉDIA

$x(t) = a + bt$ (espaço)

$$v = \frac{\Delta x}{\Delta t} = \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$ (velocidade do movimento)

$x(t) = x_0 + v(t - t_0)$ (lei horária)

JUPITER NOTEBOOK

- introdução ao Python

semana 2

VELOCIDADE MÉDIA

$x(t) = a + bt$

a velocidade instantânea $v(t)$ num instante t qualquer, num movimento descrito por $x = x(t)$, é dada por

$$v(t) = \frac{dx}{dt}$$

PROBLEMA INVERSO

queremos calcular o espaço a partir de $v(t)$

consideremos um movimento cuja velocidade $v(t)$ é dada por:

$$v(t) = 2at + b$$

a área a calcular neste caso é o trapézio sombreado

$$\frac{dx}{dt} = 2at + b$$

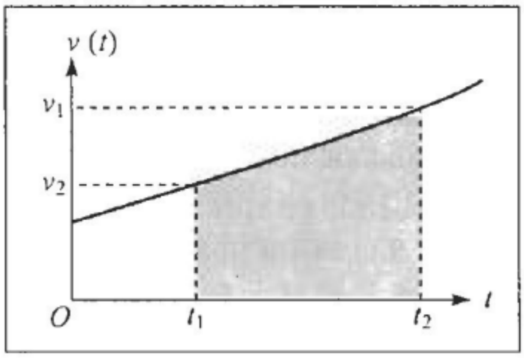


Figura 2.12 Exemplo de integração.

ACELERAÇÃO

a aceleração instantânea é a derivada em relação ao tempo da velocidade instantânea

$x(t) \rightarrow$ deriva $\rightarrow v(t) \rightarrow$ deriva $\rightarrow a(t)$

$$a(t) = \frac{d}{dt}\left(\frac{dx}{dt}\right) = \frac{d^2x}{dt^2}$$

LEI HORÁRIA DO MOVIMENTO RETILÍNEO UNIFORMEMENTE ACELERADO

$$x(t) = x_0 + v_0(t - t_0) + \frac{1}{2}a(t - t_0)^2$$

ou

$$s = s_0 + v_0 t + \frac{at^2}{2}$$

VELOCIDADE DO MOVIMENTO UNIFORMEMENTE ACELERADO

$$v^2 = v_0^2 + 2a(x - x_0)$$

ou

$$v^2 = v_0^2 + 2a\Delta S$$

ALGORITMO DE EULER

e quando não soubermos a solução analítica?

- métodos analíticos representam soluções baseadas em fórmulas matemáticas.
- métodos numéricos são aplicações de algoritmos pelas quais é possível formular e resolver problemas matemáticos usando operações aritméticas menos complexas.

O método de Euler é uma forma de resolver numericamente uma equação diferencial ordinária.

As EDOs são equações que não só lidam com diversas variáveis, como também lidam com funções e suas derivadas.

Assume-se ser conhecidas a derivada de uma função que se quer encontrar ("resolver") e um valor inicial da equação a ser integrada.

Por exemplo, no caso do movimento uniformemente acelerado:

$a = b =$ constante $\quad v(t) = x'(t) = \frac{dx}{dt} = 2at + b$

$$x(0) = 0$$

A ideia do método de Euler é substituir a derivada por uma aproximação de Taylor, desprezando-se os termos maiores que segunda ordem. Isto é:

$$x'(t) = \frac{x(t + \Delta t) - x(t)}{\Delta t} = 2at + b$$

Portanto, podemos escrever

$$x(t+\Delta t) = x(t) + \Delta t\,(2at+b)$$

Isso permite implementar o algoritmo de Euler para calcular a posição futura $x(t+\Delta t)$ a partir da posição inicial $x(t)$.

Nesse caso, $a, b, \Delta t$ são parâmetros de entrada, bem como a posição inicial $x(0) = 0$

Observamos que para esse caso, é possível calcular a solução $x(t)$ analiticamente:

$$v(t) = x'(t) = \frac{dx(t)}{dt} = 2at + b \Rightarrow x(t) = at^2 + bt + c$$

JUPITER NOTEBOOK

- implementando o algoritmo de Euler

Para a implementação ficar organizada, crie duas funções:

- nextXeuler(x,t,params,dt): que recebe um vetor de parâmetros iniciais, params, o tempo e a posição atual, t e x, respectivamente, e o delta de tempo, dt (note que com isso desacoplamos os índices do incremento). A função retorna a nova posição $x(t+\Delta t)$ conforme explicado acima.
- nextXa(t,params): que recebe um vetor de parâmetros iniciais, params, e o tempo atual. A função retorna a nova posição x_t de acordo com a equação integrada analiticamente: $x(t) = at^2 + bt + c$.

params pode ser implementada como uma lista $[a, b, c]$.

Crie uma função main que itera essas duas funções entre os tempos 0s e 2s (com um dt de 0.1s) calcula e imprime a diferença absoluta entre elas (erro) e as grafique.

semana 3

MEDIDAS

- "um conjunto de operações com o objetivo de determinar o valor de uma quantidade"
- quantidade é um atributo de um fenômeno, corpo, ou substância que pode ser distinguido qualitativamente e determinado quantitativamente.
- uma medida é algo que fazemos que resulta num número e numa unidade.
- para obter a medida, comparamos nosso sistema desconhecido com um sistema conhecido.

ERROS E INCERTEZAS

- o erro é a diferença entre o resultado de uma medição e o valor real do que está sendo medido.
- existem 3 tipos de erro
- erro grosseiro ou por engano: não tem como prevenir e pode comprometer significamente o resultado (medição), de difícil tratamento.
- erro sistemático: o método pode estar errado e não importa quantas vezes repitamos o experimento, não melhoraremos o resultado, desvio consistente do valor verdadeiro.
- erro aleatório: é todo tipo de erro que não é por engano e nem sistemático.

PRECISÃO E ACURÁCIA

- acurácia é um conceito relacionado a obter a resposta certa com incerteza aceitável
- precisão é inversamente proporcional à incerteza.

JUPITER NOTEBOOK

- formato JSON

Exercício: limpeza e simplificação do JSON

Usando o arquivo 'extracted_sample1.json' gerado na seção anterior, execute a célula anterior e crie um novo arquivo JSON chamado 'cleaned_sample1.json' em que cada objeto da sequência 'photos' contém somente os campos:

- 'lat'
- 'lng'
- 'heading'
- 'shot_date'

Ou seja, o novo arquivo gerado a partir do 'extracted_sample1.json' deverá seguir o modelo:

`{ "photos" : [ { 'lat': '32.188423', 'lng' : '-81.195239', 'heading' : '72.76266', 'shot_date': '2018-03-03 20:29:36' }, ... ] }`

Medindo a distância entre dois pontos

Neste exercício você deve implementar a função distancia_euclidiana.

Esta função recebe dois vetores (i.e. array numpy) cada um com duas dimensões e retorna a distância euclidiana entre eles.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} = \text{distância euclidiana}$$

Exercício - Visualizando o comprimento de cada trecho

Usando a função criada no exercício anterior vamos agora ver a distância percorrida pelo veículo a cada passo do sub-trajeto selecionado:

A ideia aqui é que todos os pontos da faixa de pontos selecionada sejam percorridos, e para cada par de pontos subsequentes seja calculada a distância entre eles e impressa essa distância.

Imprima uma lista como o exemplo a seguir usando a sua função de medir distância entre dois pontos:

- trecho 1000 a 1001 ; d = 0
- trecho 1001 a 1002 ; d = 0
- trecho 1002 a 1003 ; d = 0
- trecho 1003 a 1004 ; d = 0
- ...
- trecho 1097 a 1098 ; d = 0
- trecho 1098 a 1099 ; d = 0
- trecho 1099 a 1100 ; d = 0

Dica, você pode usar a linha abaixo para imprimir uma vez que a distância foi calculada:

```
print(f'trecho {i} {i+1} ; d = {dist}')
```

semana 4

FORÇAS EM EQUILÍBRIO

o movimento é afetado pela ação do que costumamos chamar de "forças".

2ª lei de Newton:

$$F = m.a$$

"a variação do momento é proporcional à força impressa, e tem a direção da força."

QUEDA LIVRE

a força P que atua sobre um corpo na vizinhança da superfície da Terra devido à atração gravitacional por ela exercida sobre o corpo é:

$$P = m.g$$

Para uma partícula em queda livre, a 2ª lei de Newton leva à

$$a = g$$

- partícula em queda livre

$$F_g = -mg$$

- segunda lei de Newton

$$m\frac{d^2y}{dt^2} = F$$

JUPITER NOTEBOOK

- analise dos dados do acelerômetro nos experimentos de queda livre

Exercícios

Crie um programa em Python que analise os dados do acelerômetro e detecte automaticamente os pontos de:

1. início da estabilização (repouso)
2. início da queda livre
3. início da fase de forte influência do atrito do ar
4. final da queda livre
5. início da estabilização (repouso)
6. início da passagem do Bob Esponja na fila indiana
7. final da passagem do Bob Esponja na fila indiana

Calcule quanto tempo demorou cada período acima.

Calcule a velocidade alcançada pelo Bob Esponja no final da queda.

Semana 5

SISTEMAS DINÂMICOS

Sistemas dinâmicos são sistemas fora do equilíbrio, caracterizados por estados que mudam com o tempo.

São usados para modelar e fazer previsões de sistemas físicos, biológicos, financeiros, etc.

- visão computacional
- estado
  - variáveis de estado
  - vetor de estado (vetor de variáveis: $\vec{v}(t)$)

Processos que evoluem em função de alguma dimensão independente (ex: Tempo)

A lei de evolução é definida por um modelo matemático como uma função ou equação diferencial.

Podem ser:

- contínuos ou discretos
- determinísticos ou estocásticos

VETORES DE ESTADOS

equações diferenciais do movimento:

$$a(t) = \frac{dv}{dt} = \frac{d^2x}{dt^2}$$

$$v(t) = \frac{dx}{dt}$$

Euler:

$$x(t + \Delta t) = x(t) + v(t)\,\Delta t$$

$$v(t + \Delta t) = v(t) + a(t)\,\Delta t$$

assim, o movimento 1D da partícula pode ser representado por um vetor de estados:

$$\vec{s} = (x, v)$$

o vetor de estados:

$$\vec{s} = (x(t), v(t))$$

define a posição e a velocidade da partícula no instante de tempo t.

- a implementação do modelo de movimento da partícula com o vetor de estados em uma modelagem por sistemas dinâmicos usando o algoritmo de Euler pode ser definida como:

$$\vec{s}(t + \Delta t) = \vec{s}(t) + \vec{r}(t)\Delta t$$

em que $\vec{s}$ é o vetor de estados e $\vec{r}$ é o vetor de taxas de variação.

JUPITER NOTEBOOK

Exercício: $\frac{dx}{dt} = 2t + 1$

A partícula se movimenta segundo a equação:

$$v(t) = \frac{dx}{dt} = 2t + 1$$
$$x(0) = 0$$

Solução analítica:

$$x(t) = t^2 + t + c$$

Como $x(0) = 0 \implies c = 0$

Solução de Euler:

$$x(t + \Delta t) = x(t) + (2t + 1)\Delta t$$
$$v(t + \Delta t) = 2(t + \Delta t) + 1 = (2t + 1) + 2\Delta t = v(t) + 2\Delta t$$

Exercício: Resolva na célula abaixo antes de olhar a solução

Dado que temos a equação analítica da velocidade, vamos usá-la na atualização da coordenada da velocidade no vetor de estados. O programa abaixo implementa a solução desse problema **com** a modelagem por sistemas dinâmicos e vetores de estado.

Exercício: $\frac{d^2x}{dt^2} = 6t$

Escreva a solução para a EDO:

$\frac{d^2x}{dt^2} = 6t$

SEMANA 6

MOVIMENTO BIDIMENSIONAL

Se adotarmos coordenadas cartesianas, a posição de uma partícula em movimento no plano será descrita pelo par de funções:

$$(x(t),\ y(t))$$

O movimento 2D da partícula pode ser representado pelo vetor de estados

$$\vec{s} = (\vec{x}, \vec{v}) = ([x_1, x_2], [v_1, v_2])$$

JUPITER NOTEBOOK

Exercício: $\frac{d^2\vec{x}}{dt^2} = \vec{a}(t)$

Escreva a solução para as EDOs:

$$\frac{d^2x_0}{dt^2} = \sin(k_0\pi t)$$
$$\frac{d^2x_1}{dt^2} = \cos(k_1\pi t)$$

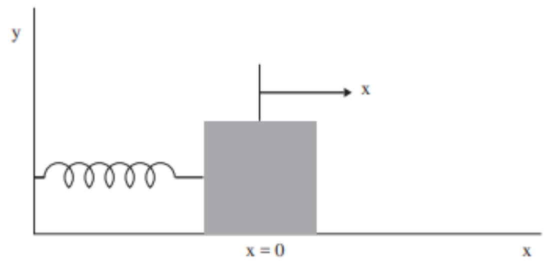
MAC0209 - modelagem e simulação

anotações para P2

semana 7

OSCILAÇÕES

Se um objeto sofre movimento periódico entre dois limites ao longo do mesmo caminho, chamamos o movimento de oscilatório.



bloco de massa m

a força no bloco na posição x é proporcional a x: $F = -Kx$

a constante K é uma medida da rigidez da mola.

A equação de movimento de Newton para o bloco pode ser escrita como:

$$(*) \quad \frac{d^2x}{dt^2} = -w_0^2 x$$

onde a frequência angular w_0 é definida por

$$w_0^2 = \frac{K}{m}$$

(*) é chamado de movimento harmônico simples e pode ser resolvido analiticamente em termos de seno e cosseno:

$$x(t) = A\cos(w_0 t + \delta)$$

onde A e δ são constantes e o cosseno é em radianos.

Como o cosseno é uma função periódica com período 2π, sabemos que $x(t)$ também é periódica.

T é definido como o menor tempo para o qual o movimento se repete, isto é:

$$x(t+T) = x(t)$$

Como $w_0 T$ corresponde a um ciclo, temos

$$T = \frac{2\pi}{w_0} = \frac{2\pi}{\sqrt{K/m}}$$

A frequência ν do movimento é o número de ciclos por segundo e é dada por $\nu = \frac{1}{T}$

Embora a posição e a velocidade do oscilador estejam mudando continuamente, a energia total E permanece constante e é dada por:

$$E = \frac{1}{2}mv^2 + \frac{1}{2}Kx^2 = \frac{1}{2}KA^2$$

MOVIMENTO CIRCULAR UNIFORME

- a trajetória é um círculo
- o módulo da velocidade instantânea é constante
- a partícula descreve arcos de círculo iguais em tempos iguais

Temos assim um movimento periódico, em que o período corresponde ao tempo levado para descrever uma volta completa, o que define um "relógio"

- a posição instantânea P da partícula fica definida pelo ângulo θ entre o vetor deslocamento $r = OP$ correspondente e o eixo Ox de um sistema cartesiano com origem no centro.

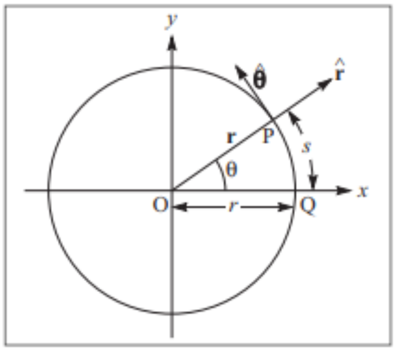


Figura 3.27 Movimento circular.

- o arco s correspondente ao ângulo θ sobre o círculo é dado por $s = r\theta$, onde θ é medido em radianos.

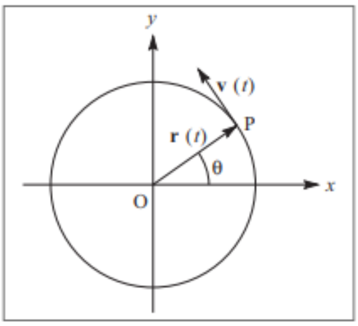


Figura 3.28 Velocidade instantânea.

- a velocidade instantânea é dada por $v = v\hat{\theta}$
- temos ainda: $v = ds/dt$
- o tempo T do movimento é o tempo para dar uma volta completa, ou seja, $T = 2\pi r / |v|$
- a frequência $f = 1/T$
- lei horária em termos do ângulo θ descrito em função do tempo:

$\theta = \theta_0 + \omega(t - t_0)$ onde $\omega = v/r$ chama-se velocidade angular

JUPITER NOTEBOOK: SEMANA 7

Simple Harmonic Oscilator - SHO

$\frac{d^2x}{dt^2} = -\omega^2 x$

$\omega = \sqrt{\frac{k}{m}}$

ou

$\frac{dv}{dt} = a(t) = -\omega^2 x(t)$

$\frac{dx}{dt} = v(t)$

e a formulação de Euler:

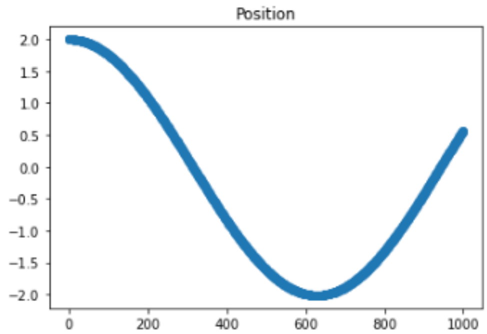
$$x(t + \Delta t) = x(t) + \Delta t\, v(t)$$
$$v(t + \Delta t) = v(t) - \Delta t\, \omega^2 x$$

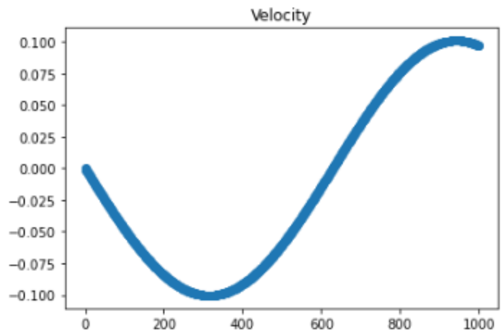
Podemos definir o vetor de estados e o vetor de taxas de variação:

$$\vec{S} = [\vec{S}[0], \vec{S}[1], \vec{S}[2]] = [x, v, t]$$
$$\vec{R} = [\vec{S}[1], -\omega^2 \vec{S}[0], 1]$$

Resolução: Compare sua solução

```
# d2x / dt2 = - omega^2 x

import math
import numpy as np
import matplotlib.pyplot as pyplot

def initState(x,v,t):
    S = np.array([x,v,t])
    return(S)

def nextState(S,dt):
    Sn = S + dt * rate(S,dt)
    return(Sn)

def rate(S,dt):
    omega = 0.05
    R = np.array([S[1], -omega*omega*S[0], 1])
    return(R)


def main():
    t=0
    tf = 100
    dt=0.1
    x=2
    v=0
    S = initState(x,v,dt)
    vve=[]
    vxe=[]
    while (S[2]<tf):
        vve.append(S[1])
        vxe.append(S[0])
        S = nextState(S,dt)


    pyplot.figure(0)
    pyplot.plot(vxe,label='Euler',linestyle='',marker='o')
    pyplot.title('Position')
    pyplot.show(block=False)

    pyplot.figure(1)
    pyplot.plot(vve,label='Euler',linestyle='',marker='o')
    pyplot.title('Velocity')
    pyplot.show()
```





SEMANAS 8 E 9

CAOS

- modelos determinísticos não lineares
- fenômenos naturais são intrisicamente não lineares
- padrões climáticos e o movimento turbulento de fluídos são exemplos cotidianos
- equação de diferença unidimensional: $x_{n+1} = 4 x x_n (1 - x_n)$

no qual x_n é a razão da população na n-ésima geração para uma população de referência.

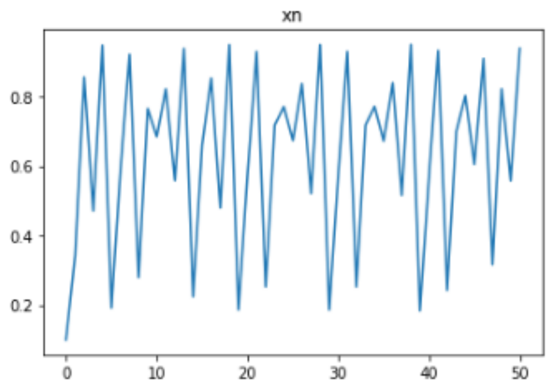
- um sistema é dito caótico quando apresenta sensibilidade às condições iniciais – trajetórias divergindo exponencialmente
- **PERIÓDICO × CAÓTICO**

Quando um sistema apresenta comportamento periódico sua trajetória no espaço de fases se mostra fechada, sendo o número de voltas necessárias para retornar ao ponto inicial correspondente à periodicidade do sistema. Já no sistema caótico, apesar das trajetórias no espaço de fase se manterem próximas e formarem um padrão, elas nunca se fecham, ou seja, o comportamento caótico é aperiódico.

JUPITER NOTEBOOK: SEMANA 8 E 9

mac0209-movimento-SHO.ipynb

```
import math
import matplotlib.pyplot as pyplot

xn = 0.1
geracoes = 50
sd = [xn]
r = 0.95   # tentar 0.1, 0.7, 0.95

for i in range(geracoes):
  xn = 4 * r * xn * (1 - xn)
  sd.append(xn)


pyplot.figure(0)
pyplot.plot(sd)
pyplot.title('xn')
pyplot.show(block=False)
```



```
!python -m pip install pynamical
!python -m pip uninstall matplotlib -y
!python -m pip install matplotlib==3.1.3
```

```
[ ] !python -m pip install imageio
```

```
[ ] from pynamical import logistic_map, simulate, bifurcation_plot
    import numpy as np

    import matplotlib.pyplot as plt
    plt.rcParams.update({'font.size': 22})
    SUB = str.maketrans("0123456789", "₀₁₂₃₄₅₆₇₈₉")
```

```
[ ] pops = simulate(model=logistic_map, num_gens=100, rate_min=0, rate_max=4, num_rates=1000, num_discard=100)
    bifurcation_plot(pops)
```

```
[ ] fig = bifurcation_plot(pops)
    #plt.axvline(x=3.1, ymin=0, ymax=1)
```

Aqui vamos analizar a primeira geração (x_1)onde a população inicial, $x_0 \in [0, 1]$ varia entre 0 e 1; a taxa de crecimento $r = 0.5$ é fixa.

```
[ ] x = np.linspace(0,1, 100)
    plt.scatter(x, logistic_map(x, 0.5))
    plt.title('Logistic map, r = 0.5')
    plt.ylabel('Population, x1'.translate(SUB))
    plt.xlabel('Initial population, x0'.translate(SUB))
```

Aqui vamos analizar a primeira geração onde a população inicial, $x_0 = 0.5$, e a taxa de crescimento $r \in [0, 4]$ varia entre 0 e 4.

```
[ ] r = np.linspace(0,4, 100)
    plt.scatter(r, logistic_map(.5, r))
    plt.title('Logistic map, x0'.translate(SUB)+' = 0.5')
    plt.xlabel('Growth Rate, r')
    plt.ylabel('Population, x1')
```

Aqui podemos ver que alguns pares de população inicial e taxas de crescimento atingem pontos de estabilidade após um certo número de gerações. Aqui rodamos cada caso por 100 gerações, isso é, de x_0 até x_{99}.

+ Código + Texto

```
xini0 = 0.3
xini1 = 0.5
xini2 = 0.8
r0 = 0
r1 = 0.5
r2 = 1
s0 = [xini0]
s1 = [xini1]
s2 = [xini2]
for _ in range(100):
  s0.append(logistic_map(s0[-1], r0))
  s1.append(logistic_map(s1[-1], r1))
  s2.append(logistic_map(s2[-1], r2))
fig, axs = plt.subplots(1, 3, figsize=(15,3))
axs[0].scatter(list(range(len(s0))), s0)
axs[0].set_title(f"r = 0, "+"x0".translate(SUB)+f" = {xini0}")

axs[1].scatter(list(range(len(s1))), s1)
axs[1].set_title(f"r = 0.5, "+"x0".translate(SUB)+f" = {xini1}")

axs[2].scatter(list(range(len(s2))), s2)
axs[2].set_title(f"r = 1, "+"x0".translate(SUB)+f" = {xini2}")
plt.tight_layout()

for ax in axs.flat:
    ax.set(xlabel='Geração')
    ax.set(ylabel='População')
    #ax[0].xaxis.set_xlabel('Geração')
```

No caso abaixo podemos ver que x convege para dois valores, podemos ver que isso coincide com oque observamos no "bifurcation map" onde para o valor de $r = 3.429$ esperamos que x (a população) convirja para dois valores.

+ Código + Texto

```
xini = 0.5
r = 3.429
x = [logistic_map(xini, r)]
for _ in range(1000):
    x.append(logistic_map(x[-1], r))
plt.figure(figsize=(15,3))
plt.scatter(list(range(len(x))), x)
plt.title("r = 3.429");
plt.xlabel('Geração')
plt.ylabel('População')
print(f"Últimos valores de população:")
print(f"x[996] = {x[996]}")
print(f"x[997] = {x[997]}")
print(f"x[998] = {x[998]}")
print(f"x[999] = {x[999]}")
```

```
[ ] xini = 0.5
r = 3.429
x = [logistic_map(xini, r)]
for _ in range(10000):
    x.append(logistic_map(x[-1], r))

plt.figure(figsize=(15,3))
plt.scatter(list(range(len(x[9900:]))), x[9900:])
plt.title("r = 3.429");
plt.xlabel('Geração')
plt.ylabel('População')
print(f"Últimos valores de população:")
print(f"x[9996] = {x[9996]}")
print(f"x[9997] = {x[9997]}")
print(f"x[9998] = {x[9998]}")
print(f"x[9999] = {x[9999]}")
```

Novamente de acordo com o "bifurcation map" para um $r = 3.8$ observamos que não é claro se x converge para algum conjunto de valores ou não.

```
[ ] xini = 0.5
r = 3.8
s = [logistic_map(xini, r)]
for _ in range(1000):
    s.append(logistic_map(s[-1], r))
plt.figure(figsize=(15,3))
plt.scatter(list(range(len(s))), s)
plt.ylabel('População')
plt.title("r = 3.8");
plt.xlabel('Geração')
plt.ylabel('População (x)')
```

Exercícios

Consulte a descrição dos exercícios nos slides da aula sobre Caos.

Mapa logístico

1. Gere um gráfico mostrando a evolução de um par de valores para população inicial e taxa de crescimento para 20 gerações que seja similar ao gráfico da figura a) no quadro branco.
    1. Gere um diagrama de bifurcação para o mapa logístico
    2. Gere um gráfico de cobweb (figura c) no quadro branco) para o mapa logístico
    3. Dica: Consulte a página do [pynamical]
2. Agora usando o matplotlib.pyplot.scatter gere o diagrama de bifurcação apenas para um subconjunto de gerações de x_{min} até x_{max} depois convergência ao ponto fixo, eg. x_{9000} até x_{10000}. Por exemplo, para $r = 3.429$ após 9996 iterações notamos que x_n onde $n \in [9996, \dots]$, basicamente assume os valores $0.846809528636908l$ e 0.44482068425314786, ou seja, para este r o mapa de bifurcação teria 2 pontos no eixo vertical.

```
[ ]
```

```
[ ] !python -m pip install pynamical==0.3.1
    !python -m pip install imageio
```

```
from pynamical import logistic_map, simulate, bifurcation_plot
import pynamical
import numpy as np

import matplotlib.pyplot as plt
import matplotlib
plt.rcParams.update({'font.size': 22})

print(f"pynamical version: {pynamical.__version__}")
print(f"matplotlib version: {matplotlib.__version__}")
print(f"numpy version: {np.__version__}")
```

+ Código + Texto

Apenas para recapitular, aqui temos o diagrama de bifurcação, para o mapa logístico, gerado pelo pacote pynamical.

```
[ ] pops = simulate(model=logistic_map, num_gens=100, rate_min=0, rate_max=4, num_rates=1000, num_discard=100)
    bifurcation_plot(pops)
```

```
[ ] xini = 0.3
    r = 3.429
    x = [xini]
    for _ in range(100):
        x.append(logistic_map(x[-1], r))

    plt.figure(figsize=(15,3))
    plt.plot(list(range(0,len(x))), x[0:])
    plt.title("r = 3.429");
    plt.xlabel('Geração')
    plt.ylabel('População')
    print(f"Últimos valores de população:")
```

Ao usarmos uma taxa de crescimento $r = 3.429$ observamos um comportamento periódico (com 2 valores) da população x ao longo das iterações do mapa logístico após uma certa iteração.

```
xini = 0.5
r = 3.429
x = [logistic_map(xini, r)]
for _ in range(10000):
    x.append(logistic_map(x[-1], r))

plt.figure(figsize=(15,3))
plt.scatter(list(range(9900,len(x))), x[9900:])
plt.title("r = 3.429");
plt.xlabel('Geração')
plt.ylabel('População')
print(f"Últimos valores de população:")
print(f"x[9996] = {x[9996]}")
print(f"x[9997] = {x[9997]}")
print(f"x[9998] = {x[9998]}")
print(f"x[9999] = {x[9999]}")
```

Exercícios

Consulte a descrição dos exercícios nos slides da aula sobre Caos.

Sistema de EDOs acopladas

Para solucionar as equações de Lorenz (nos exercícios) podemos usar o método de Euler também. No caso precisaremos resolver um sistema de equações da forma:

$\frac{dx}{dt} = xy - x$ $\frac{dy}{dt} = y - xy + sin^2(\omega t)$

- As equações acima foram baseadas nesta apresentação da Profa. Dra. Sandra Amato.

Abaixo temos uma possivel implementação do método de Euler para resolver esse sistema de EDOs acopladas:

```python
# funcoes genericas que podem ser re-usadas em outros problemas

import math
import matplotlib.pyplot as pyplot
import numpy as np
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D


# funcoes base para implementar o Euler.
# Deve-se implementar a funcao rates, que depende de cada modelo.

def initStateVector(s):
    return np.array(s)

def updateStateVectorEuler(s,dt):
    return s + rates(s,dt)

# State Vector Trajectories store state space evolution. Uses list to init empty.

def initSVTrajectory():
    return []

# append s a svt
def updateSVTrajectory(svt,s):
    svt.append(s)
    return svt

def extractSVTrajectory(svt,i): # returns the trajectory as numpy array
    foo = np.array(svt)
    return foo[:,i]

def plotEuler(vxe, vtime):
    fig, ax = pyplot.subplots()
    pyplot.plot(vtime, vxe, label='Euler',linestyle='',marker='o')
    pyplot.title('Posição')
    ax.set_xlabel('Tempo (segundos)')
    ax.set_ylabel('Posição (metros)')
    pyplot.show(block=False)

def erroTrajetorias(v1,v2,tipoErro):
    if (tipoErro == 0): # erro com sinal
        return(np.array(v1) - np.array(v2))
    elif (tipoErro == 1): # erro quadratico
        return((np.array(v1) - np.array(v2))**2)
    elif (tipoErro == 2): # erro em modulo
        return(fabs((np.array(v1) - np.array(v2))))


def easyPlot(v,title):
    pyplot.figure()
    pyplot.plot(v)
    pyplot.title(title)
    pyplot.show()

def easyPlot2D(x,y,title):
    pyplot.figure()
    pyplot.plot(x,y,'*')
    pyplot.title(title)
    pyplot.show()

def easyPlot3D(x,y,z,title,xl,yl,zl):
    mpl.rcParams['legend.fontsize'] = 10
    fig = pyplot.figure()
    ax = fig.gca(projection='3d')
    ax.plot(x, y, z, label=title)
    ax.set_xlabel(xl)
    ax.set_ylabel(yl)
    ax.set_zlabel(zl)
    ax.legend()
    pyplot.show()
```

```python
# Euler:
def rates(s,dt):
    omega = 0.115
    x1 = dt*(s[0]*s[1]-s[0]) # +x0
    y1 = dt*(s[1]-s[0]*s[1] + np.sin(omega*s[2])**2) # +y0
    r0 = x1
    r1 = y1
    r2 = dt
    return np.array([r0,r1,r2])

def main():
    t=0
    tf = 10
    dt = 0.1
    x0 = 1.2
    y0 = 3.1

    # state vector: [position x, position y, time t]
    stateVectorEuler = initStateVector([x0,y0,t])


    svtEuler = initSVTrajectory()

    while (stateVectorEuler[2] < tf):
        svtvEuler        = updateSVTrajectory(svtEuler,list(stateVectorEuler))
        stateVectorEuler = updateStateVectorEuler(stateVectorEuler,dt)
        #print(stateVectorEuler)
        #break

    vx = extractSVTrajectory(svtEuler,0)
    vy = extractSVTrajectory(svtEuler,1)

    vtime = extractSVTrajectory(svtEuler,2)

    easyPlot2D(vtime, vx, 't, x')
    easyPlot2D(vtime, vy, 't, y')
    easyPlot2D(vx, vy, 'x, y')



main()
```

Mapa logístico / Diagrama de bifurcação / Cobweb

1. Gere um gráfico de cobweb (figura c) no quadro branco) para o mapa logístico. Dica: Consulte a página do pynamical
2. Agora usando o `matplotlib.pyplot.scatter` gere o diagrama de bifurcação apenas para um subconjunto de gerações de x_{min} até x_{max} depois convergência ao ponto fixo, eg. x_{9000} até x_{10000}. Por exemplo, para $r = 3.429$ após 9996 iterações notamos que x_n onde $n \in [9996, \ldots]$, basicamente assume os valores 0.8468095286369081 e 0.44482068425314786, ou seja, para este r o mapa de bifurcação teria 2 pontos no eixo vertical.

```
[ ]
```

Equações de Lorenz

1. Usando o método de Euler resolva numéricamente as equações de Lorenz:
   - $\frac{dx}{dt} = \sigma(y-x)$
   - $\frac{dy}{dt} = x(\rho - z) - z$
   - $\frac{dz}{dt} = xy - \beta z$
2. Gere os 4 tipos de gráficos (ilustrados no quadro branco à direita a) b) c) e d)):
   - a) $x(t), y(t), z(t)$
   - b) $x \times y$, $x \times z$, $y \times z$
   - c) $x \times y \times z$
   - d) x, $x \times t$, x, $z \times t$, y, $z \times t$

Como gabarito, vamos escolher alguns valores para as constantes $\sigma = 10, \rho = 28$ e $\beta = 8/3$.

como condições de contorno: $x_0 = 0, y_0 = 1$ e $z_0 = 0$

e vamos iterar do tempo $t = 0$ até $t = 50$ com um passo $dt = 0.01$.

Para esta situação obtemos os gráficos abaixo para $x \times t$ e para $z \times x$:

Mapa de Lorenz, dois tipos de gráficos:

```
- a) x(n), y(n), x X y
- b) Mapa de bifurcações como o mapa logístico
```

SEMANAS 10 E 11

MÉTODOS ALEATÓRIOS

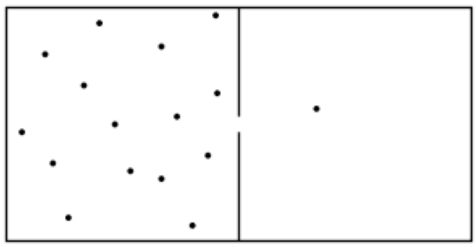
- Processos aleatórios são introduzidos no contexto de vários sistemas físicos simples, incluindo passeios aleatórios em uma rede, polímeros e difusão - reações químicas controladas.
- como o acaso pode gerar resultados estaticamente previsíveis, por exemplo: apostar muitas vezes no resultado de um jogo para o qual a probabilidade de ganhar é inferior a 50%, pode eventualmente perder dinheiro.

EXEMPLO DA CAIXA

Este é um exemplo que ilustra a tendência de sistemas de muitas partículas a evoluir para um estado bem definido.

Imagine uma caixa fechada que é dividida em duas partes de igual volume.

N partículas



Sabemos que depois de algum tempo, o número médio de partículas em cada metade da caixa se tornará $N/2$, e dizemos que o sistema atingiu o equilíbrio.

COMO SIMULAR? (modelo determinístico)

Uma maneira é dar a cada partícula uma velocidade inicial e posição e adotar um modelo determinístico do movimento das partículas. Por exemplo, poderíamos supor que cada partícula se move em linha reta até atingir uma parede da caixa ou outra partícula e sofrer uma colisão elástica.

MODELO PROBABILÍSTICO BASEADO EM UM PROCESSO ALEATÓRIO

As suposições básicas deste modelo são que o movimento das partículas é aleatório e as partículas não interagem umas com as outras. Portanto, a probabilidade por unidade de tempo de que uma partícula atravessa o buraco na partição é o mesmo para todas as N partículas. Assumimos que o tamanho do buraco é tal que apenas uma partícula pode passar por vez. Primeiro, modelamos o movimento de uma partícula passando pelo buraco escolhendo uma das N partículas aleatoriamente e movendo-a para o outro lado. Usaremos arrays para especificar a posição de cada partícula. A ferramenta que precisamos para simular esse processo aleatório é um gerador de números aleatórios

JUPITER NOTEBOOK: SEMANA 10 E 11

mac0209_randomMethods_CamaraParticulas.ipynb

Exercício com a câmara de particulas da Figura 7.1 dos slides

Retome o experimento da câmara da Figura 7.1 explicada em aula. Você deve implementar a simulação da evolução do sistema usando algumas variações definidas abaixo. Em todos os exercícios, você deve assumir que existem n partículas que são sorteadas usando as distribuições definidas abaixo. O sistema dinâmico é composto pelo conjunto de M partículas. O estado de cada partícula é a câmara que ela se encontra. Por exemplo, no caso da figura, 0 para a câmara da esquerda e 1 para a câmara da direita. Assim, o vetor de estados do sistema pode ser representado por um vetor binário de tamanho M.

As simulações devem produzir as seguintes visualizações gráficas:

- Acumule o estado das partículas ao longo da simulação em uma matriz E de tamanho $M \times N$, em que M é o número de partículas e N é o número de iterações da simulação. Cada coluna representa o estado do sistema dinâmico na i-ésima iteração. Visualize a matriz E como uma imagem.
- Plote os gráficos $n_0(i)$ e $n_1(i)$ partículas em cada câmara 0 e 1 de i iterações da simulação.

Distribuição uniforme

Assuma que as partículas são sorteadas usando uma distribuição uniforme sobre todas.

```
[ ]
```

Distribuição normal

Assuma que as partículas são sorteadas usando uma distribuição normal com média $M/2$ e desvio padrão arbitrário σ sobre todas. Faça testes com σ pequeno, médio e grande.

```
[ ]
```

Distribuição normal com controle na porta

Assuma que as partículas são sorteadas usando uma distribuição normal com média $M/2$ e desvio padrão arbitrário σ sobre todas. Além disso, a partícula só muda de câmara se seu índice (posição no vetor de estados) j verificar $j < M/k$, em que $0 < k < M$ é uma constante arbitrária. Faça testes com σ pequeno, médio e grande e $k = 2, 3, 4$.

```
[ ]
```