Cálculo

Uma Abordagem Computacional

Gustavo Mirapalheta

2022-02-02

Sumário

Capítulo 1

Introdução

Repetitia matter scientia est. A repetição é a mãe da ciência. Desde o seus mais incipientes primórdios, a ciência teve em seu cerne a idéia da repetição das experiências. Esta repetição foi responsável pelo correto acúmulo de conhecimento e pela confiança crescente que as pessoas em geral, cientistas ou não, depositaram nos seus ensinamentos. Com a matemática não poderia ser diferente. É a repetição, as vezes até a exaustão, que torna o aluno confiante em suas capacidades e por consequência capaz de aplicar corretamente seu conhecimento na resolução de novos problemas.

Durante muito tempo, a repetição foi limitada a habilidade algébrica mínima que era requerida dos estudantes. Esta acentuada curva de aprendizado tornou o estudo da matemática e de suas aplicações em uma tarefa árdua. Minimizar os efeitos desta curva tornaria mais eficiente o processo de ensino de ciência e engenharia como um todo. Até recentemente poucos recursos existiam para auxiliar em tal tarefa, e os mesmos quando existiam eram de natureza muito básica (réguas de cálculo, calculadores mecânicos, etc...). No entanto, a partir da década de 1970 surgiram calculadoras eletrônicas que aceleraram em muito a parte repetitiva da resolução de exercícios numéricos, tornando o aluno capaz de verificar por conta própria a exatidão de seus resultados. Com a acelerada disseminação dos computadores pessoais a partir da década de 1990, outros usos foram encontrados para os recursos computacionais, inclusive no âmbito da resolução algébrica de problemas.

Ao entrarmos na terceira década do século XXI existem disponíveis uma ampla gama de softwares, muitos deles de uso livre, que liberam o aluno do fardo do cálculo repetitivo, seja ele numérico, seja ele algébrico. O professor agora pode concentrar sua energia na elaboração de problemas que tenham como foco o pensar, a imaginação, sem se preocupar com as necessidades de cálculo, seja ele numérico seja ele algébrico. O esforço inicial em realizar cálculos numéricos ou algébricos é agora substituído pela necessidade de conhecer a fundo tais

ferramentas. Sendo assim, projetar os livros didáticos, tendo por base a premissa da utilização de tais softwares certamente trará grandes benefícios aos alunos em termos de rapidez do aprendizado e aos professores em termos do alcance que poderá ser dado ao ensino de matemática e ciência, em um dado período de tempo.

Esta foi a premissa que norteou a escrita deste texto. A idéia aqui é ensinar as diversas aplicações da matemática superior à resolução de problemas quantitativos, desde o início utilizando ferramentas de resolução numérica e/ou algébrica. A metodologia utilizada é concentrada em exercícios com um mínimo de teoria que é apresentada as vezes ao longo da própria resolução de exemplos. Espera-se com isto tornar o aprendizado mais intuitivo, dinâmico e eficaz, liberando mais tempo para o ensino de problemas que façam a diferença no desenvolvimento das capacidades de modelagem e resolução quantitativa.

Sabendo que a utilização de recursos computacionais será chave para o sucesso da tarefa que nos propomos a enfrentar, precisamos escolher uma ferramenta preferencial. Existem muitos softwares no mercado com ampla gama de recursos, mas nos últimos anos uma ferramenta se consolidou, tanto na academia quanto na indústria, seja para resolução numérica, seja para resolução algébrica de problemas: `Python`. Sendo assim, o curso se baseará fortemente nesta linguagem e nos pacotes `numpy` e `sympy` para resolução de problemas. Além de permitir a rápida absorção dos conceitos através da abordagem computacional, o uso destas ferramentas tornou-se recentemente um requisito de mercado na formação dos alunos. Se o aluno pouco ou nada absorver dos conceitos teóricos da matemática superior, o treino no ambiente `Python` será certamente um benefício ímpar quando o mesmo buscar uma colocação profissional no mercado de trabalho.

Sabe-se que sem a tecnologia moderna a sociedade do bem estar não seria possível. Esta tecnologia tornou-se realidade porque nossos antepassados alteraram a sua forma de ver o mundo, passando a fazer ciência. E a ciência moderna, desde os seus primórdios na renascença européia no século XVI foi construída em bases matemáticas. A matemática é, portanto, essencial para o mundo moderno. Maneiras que tragam mais e mais alunos para o apaixonante mundo da ciência e dos desafios tecnológicos do século XXI, serão desta forma muito úteis para o desenvolvimento de toda a sociedade.

Gustavo Corrêa Mirapalheta

São Paulo, 31 de Março de 2022

Capítulo 2

Introdução À Linguagem Python

A linguagem Python será extensamente utilizada neste livro para o ensino de matemática. Sendo assim nosso trabalho começa pelo entendimento básico da mesma e do ambiente de programação que será utilizado, o Google Colab. Este capítulo também pode ser utilizado de forma independente em um curso de linguagem de programação.

2.1 O Ambiente Google Colab

Antes de começar nosso trabalho com a linguagem Python é importante definir e configurar o ambiente no qual a mesma será utilizada. No caso dos exemplos e exercícios deste texto recomenda-se o uso do ambiente Google Colab.

O ambiente em nuvem do Google está disponível em: `http://colab.research.google.com`. Para acesso ao ambiente crie uma conta no Gmail e utilize o usuário da mesma para entrar no colab. O ambiente Google Colab utiliza o conceito de Jupyter Python Notebooks. Um Python Notebook é um arquivo dividido em células. Cada célula é um bloco de comandos (um pequeno programa) Python que pode ser executado com `Ctrl+Enter`

Para iniciar a programação no Colab deve-se seguir o procedimento descrito abaixo:

1. Criar uma conta no Gmail. Caso já tenha, acessar a mesma.
2. Acessar o sistema Google Colab no site `http://colab.research.google.com`
3. Criar um Jupyter Notebook
4. Alterar o nome do mesmo para `aula01.ipynb`

5. Inserir uma célula de texto no mesmo, digitando a frase: `Anotacoes de aula de` e o seu nome.
6. Inserir uma célula de texto do tipo `Heading` no arquivo com a frase: `Primeira Aula de Python`. Para que o python interprete a mesma como uma célula de título é necessário preceder o texto com um hashtag `#`
7. Reposicionar a célula de título para acima da célula de texto.
8. Inserir abaixo da célula de título uma célula de `Subheading` no arquivo com a frase: `Acesso ao Ambiente Google Colab`. Para que o python interprete a mesma como uma célula de sub-título é necessário precer o texto com duas hashtags `##`
9. Inserir abaixo da célula de texto uma célula de código.
10. Inserir na mesma o comando python de impressão: `print` com a frase `Bom dia. Bem-vindo a sua primeira aula de Python`. Em seguida executar o código pressionando `Ctrl+Enter`. O texto deve ser colocado entre parênteses e entre aspas duplas tal como pode ser visto a seguir.

```
print("Bom dia. Bem-vindo a sua primeira aula de Python")
```

```
Bom dia. Bem-vindo a sua primeira aula de Python
```

2.2 Expressões Matemáticas Básicas

O Python pode ser utilizado como uma calculadora, no cálculo do resultado de expressões aritméticas. Os resultados podem ser apresentados através do uso da função `print`

A adição, a subtração e a multiplicação são executadas com os operadores usuais:

```
print(2+2)
```

```
4
```

```
print(5-3)
```

```
2
```

```
print(3*5)
```

```
15
```

Ao contrário da maioria dos ambientes, onde a exponenciação é obtida com ^, no Python ela é executada com a dupla estrela (`**`), tal como pode ser visto a seguir:

```
print(2**3)
```

```
8
```

A parte inteira de uma divisão pode ser obtida com `//`

```
b = 22 // 8
print(b)
```

```
2
```

O resto de uma divisão é obtido através do operador $\%$

```
c = 22 % 8
print(c)
```

```
6
```

Para alterar a precedencia dos operadores utilizamos parenteses:

```
d = (5 - 1) * ( (7 + 1) / (3 - 1) )
print(d)
```

```
16.0
```

2.2.1 Exercícios

Calcule através do Python:

1. O resto da divisão de 5 por 2.
2. 5^4, $(-5)^4$, $-(5^4)$, 5^{-4}, $\frac{5^4}{5^3}$, $(\frac{5}{3})^{-2}$, $64^{-\frac{3}{4}}$
3. $5432.\sqrt{23}$. Dica: Eleve a 0.5 para obter a raiz quadrada.
4. A média dos números 730, 629 e 647.
5. Represente o valor $4,445.10^8$ em notação científica. Obs: no Python a notação científica é representada pela letra `e` minúscula.
6. A parte inteira de -22.3. Dica: Utilize a função `int`.
7. Arredonde -22.3 para zero casas decimais. Dica: Utilize a função `round`.
8. O valor absoluto de -32. Dica utilize a função `abs`.
9. Calcule $\pi.ln(\sqrt{42})$. Dica: Use as funções: `pi`, `log` e `sqrt` do pacote `numpy` carregando o mesmo através do comando: `import numpy as np`.

2.3 Variáveis

Ao invés de utilizarmos os valores diretamente como números em uma expressão os mesmos podem ser armazenados em uma variável. Em seguida esta variável pode ser utilizada em uma expressão em substituição ao número original. Isto tem a vantagem de permitir a alteração dos valores utilizados no cálculo sem ser necessária a reescrita da expressão.

```
a = 22/8
print(a)
```

```
2.75
```

Regras para designação de nomes a variáveis:

1. Deve ser apenas uma palavra (i.e. sem espaços)
2. Deve conter apenas letras, números e o sublinhado
3. Não pode começar com um número

2.3.1 Números inteiros e de ponto flutuante

Os tipos mais comuns de variável são os inteiros e os floats (números de ponto flutuante, isto é decimais com vírgula). Em Python o separador de decimal é o ponto. O tipo da variável pode ser confirmado através da função `type()`.

```
a = 5
print(type(a))
```

```
<class 'int'>
```

```
b = 3.14
print(type(b))
```

```
<class 'float'>
```

2.3.1.1 Exercícios

Calcule o resultado das seguintes expressões através do uso de variáveis na linguagem Python:

1. Se $b = 3$ e $c = 4$ e $a = \sqrt{b^2 + 2.b.c + c^2}$, qual o valor de a?
2. Na expressão anterior, qual o tipo das variáveis a, b e c?
3. Qual será o valor da variável `a` se fizermos primeiro `a=1` e depois `a=a+1`?

2.3.2 Strings

As variáveis podem conter valores de texto. Neste caso o tipo da variável chama-se "string"

```
a = "Isto eh um texto"
print(a)
```

```
Isto eh um texto
```

Duas ou mais strings podem ser concatenadas através da sua "adição"

```
a = "Eduardo"
b = "Luiza"
print(a + b)
```

```
EduardoLuiza
```

Além disso elas podem também ser repetidas através de sua "multiplicação"

```
c = "Thiago"
print(3*c)
```

```
ThiagoThiagoThiago
```

2.3.2.1 Exercícios

Execute as seguintes operações através do uso de variáveis do tipo `string` na linguagem Python.

1. Armazene a string `Hello World` em uma variável.
2. Junte as strings `Eduardo` e `Maria` em uma única string.
3. Repita a string `Carlos` cinco vezes.
4. Supondo que `escola = 'Harvard'`, qual o resultado de `print(escola[2])`?
5. Determine o número de caracteres que compõe a string "Eu sei programar em Python". Dica: Utilize a função `len()`.
6. A seguinte frase, na linguagem Python, apresentará uma mensagem de erro: `teslaQuote = 'I don't care that they stole my idea. I care that they don't have any of their own'`. Como ajustá-la para não termos mais um erro?
7. Supondo `a = 'Carro'`, `b = 'Carro Esportivo'`, qual o resultado de `a < b` e `len(a)<len(b)`?

2.3.3 Booleanos

Tipos de dados booleanos: Os valores que uma variável de tipo Booleano pode assumir são `True` ou `False` (em geral pensamos em `True` como `1` e `False` como `0`). Atentem para o seguinte: Apesar de ser possível utilizar `True` como `1` e `False` como `0` os valores que o Python vai armazenar na variável serão `True` ou `False`, e estes valores NÃO são texto. São booleanos. Booleanos surgem como o resultado de operações que avaliam a validade ou não de uma expressão. Por exemplo, a expressão $3 < 4$ dará como resultado True.

```
print(3 < 4)
```

```
True
```

As operações típicas para duas variáveis booleanas são `and` (&), `or` (|) e `not` (!). Estas são também conhecidas como operações lógicas.

Um `and` (&) entre duas condições terá como resultado True apenas se as duas condições forem True. Por exemplo, a operação `(3<4) & (10<5)` dará como resultado `True`.

```
print((3 < 4) & ( 10 < 15))
```

```
True
```

Por outro lado, um `or` (|) terá como resultado True se pelo menos uma das condições for True. No exemplo a seguir, a operação `(3 < 4) | (10 > 17)` dará como resultado `True`.

```
print((3 < 4) | (10 > 17))
```

```
True
```

Um `not` (`!`) por sua vez inverte o valor do booleano (`True` passa para `False` e `False` passa para `True`).

Os testes mais comuns, os quais quando efetuados entre duas variáveis quaisquer, dão como resultado um booleano são os de: igualdade `==`, maior ou igual `>=`, menor ou igual `<=` ou diferente de `!=`

2.3.3.1 Exercícios

Calcule os seguintes resultados através do uso de variáveis booleanas na linguagem Python:

1. `42 == '42'`, `42 == 42.0`, `42.0 == 0042.000`
2. Suponto `C=41`, calcule: a) `C == 40`, b) `C != 40 and C < 41`, c) `C != 40 or C < 41`, d) `not C == 40`, `not C > 40`, e) `C <= 41`

3. Calcule: a) `not False`, b) `True and False`, c) `False or True`, d) `False or False or False`, e) `True and True and False`, f) `False == 0`, g) `True == 0`, h) `True == 1`

4. Faça `a=1/947*947.0` e `b=1`. Em seguida calcule `a==b`. O resultado obtido foi aquele que você esperava? Este exercício mostra que a comparação de igualdade entre `floats` pode levar a erro devido ao arredondamento executado pela máquina ao armazenar um valor com vírgula. Para comparar dois floats deve-se determinar se a diferença absoluta entre eles é menor que um valor de tolerância pequeno e arbitrário escolhido pelo próprio usuário.

2.3.4 Conversão de variáveis

Uma variável numérica pode ser convertida em texto e uma string que só contenha algarismos pode ser convertida em um número. Para converter um número em uma string utilize a função `str()`. Para converter uma string que só contenha números em um inteiro utilize `int()`. Caso os números contenham o ponto decimal, a variável poderá ser convertida para um número através de `float()`.

```
a = "4"
print(a, type(a))
```

```
4 <class 'str'>
```

```
b = int(a)
print(b, type(b))
```

```
4 <class 'int'>
```

```
a = "3.14"
print(a, type(a))
```

```
3.14 <class 'str'>
```

```
b = float(a)
print(b, type(b))
```

```
3.14 <class 'float'>
```

2.4 Programas

Um programa é um conjunto de comandos que o computador executa de forma sequêncial, passo a passo. Um exemplo pode ser visto a seguir, onde temos um programa que se apresenta e informa o número de caracteres que compõe um determinado nome.

```
print("Alo Mundo")
```

```
Alo Mundo
```

```
meuNome = "Gustavo"

print("Foi bom te conhecer " + meuNome)
```

```
Foi bom te conhecer Gustavo
```

```
print("Teu nome tem " + str(len(meuNome)) + " caracteres")
```

```
Teu nome tem 7 caracteres
```

A execução de um programa é estritamente sequencial. Sendo assim a ordem das instruções é fundamental para a correta execução do código. No exemplo a seguir, foi incluido propositalmente um erro. Determine o mesmo, altere o código e execute o programa. Qual o resultado obtido?

```
try:
  C=A+B
  A=2
  B=3
  print(C)
except:
  print("Variáveis A e B não definidas")
```

```
Variáveis A e B não definidas
```

2.4.1 Interação com o Usuário

O programa pode interargir com o usuário através da função `input`. O código abaixo, caso seja executado, irá pedir ao usuário para inserir um valor, armazenando o mesmo na variável `a` e imprimindo uma mensagem de resposta.

```
a = input("Insira um valor")
print("O valor inserido foi: ", a)
```

2.4.2 Gerenciamento de Erros

Outro recurso interessante é o gerenciamento de erros. Para evitar que o programa pare, caso ocorra um erro, pode-se utilizar o par `try` e `except` (i.e.: manejo de exceções). Observe o uso do par a seguir:

```
a = 4
b = 0

try:
  c = a/b
except:
  c = "Erro. Divisão por Zero"

print(c)
```

```
Erro. Divisão por Zero
```

Um outro exemplo de utilização dos recursos de captura e desvio de erro do par **try: e except:** seria a extração dos dígitos de uma string. Por exemplo extrair os números 2 e 3 da string "2 cats and 3 dogs." Para tal utilizamos `try` para tentar converter cada elemento para inteiro e inserir o resultado na lista. Quando der erro, usamos `except: pass` para sair do erro e continuar o loop.

```
a = "2 cats and 3 dogs"
lista = []
for i in list(a):
  try:
    n = int(i)
    lista.append(n)
  except:
    pass

print(lista)
```

```
[2, 3]
```

2.4.3 Exercícios

Desenvolver os seguintes programas em Python:

1. Escrever um programa que pede para o usuário inserir três números e em seguida imprime a média dos três.
2. Escrever um programa que pergunta o valor gasto em um restaurante, o percentual de serviço e a gorjeta. O programa em seguida deverá imprimir em quatro linhas, o total da refeição, a taxa de serviço, a gorjeta e o total da conta.
3. Escrever um programa que pede para o usuário inserir: a) o nome, b) o sobrenome, c) número de horas trabalhadas em consultoria, d) o valor por hora pago pelo cliente, e) o percentual do valor hora que o funcionário

recebe como salário, f) o número de dependentes. O programa deverá então calcular e imprimir o nome completo do funcionário e o seu salário total. Considere que cada dependente acrescenta 145 reais ao salário e que o salário fixo do mesmo é de 2.800 reais.

4. Escrever um programa que peça ao usuário para inserir a quantidade, o preço sem impostos e o percentual de impostos (no estilo "por dentro") de cada um de dois tipos de peças. O programa deverá imprimir o preço total de venda com impostos da cada um dos tipos de peças.

5. Escrever um programa que pede para o usuário inserir o número de vendas efetuadas no mês, o valor médio destas vendas, o percentual de comissão, o valor da comissão fixa por venda e o salário fixo. O programa deverá então calcular o salário bruto do funcionário e imprimir o resultado.

6. Escrever um programa que escolhe um número ao acaso entre 1 e 20. Utilize para tal a função `np.random.random.randint(1, 20)` da `numpy`. O usuário terá então 7 tentativas para adivinhar o número. O programa deve pedir ao usuário para inserir uma tentativa e caso ela seja maior que o número ele deverá indicar `Estimativa muito alta`, caso contrário deverá indicar `Estimativa muito alta`. Caso o usuário acerte, o programa deverá indicar `Estimativa correta` e em seguida informar quantas tentativas foram necessárias para o usuário escolher o número certo. Caso o usuário não acerte o número em até 7 tentativas o programa deverá parar o jogo e informar ao usuário qual era o número correto.

2.5 Desvio condicional

A sequência das operações em um programa pode ser modificada através de um desvio condicional (também chamado de decisão), o qual é implementado pelos comandos `if else`.

No exemplo abaixo, temos uma comparação com o valor da variável `name`. A partir do resultado desta comparação será executado ou não um segmento do código. Outras decisões (`ifs`) podem ser misturados entre si, tal como neste exemplo.

```
name = "Mary"
password = "swordfish"

if name == "Mary":
  print("Hello Mary")
  if password == "swordfish":
    print("Access granted")
  else:
    print("Wrong Password")
```

```
Hello Mary
Access granted
```

Neste próximo exemplo é apresentada uma variante do par `if else`, o `elif` que quer dizer `else if`. Se a comparação no `if` principal tiver como resultado um `False`, o programa segue para o `elif` onde outra comparação (decisão) será tomada, até chegarmos no `else` final.

```
name = "Alice"
age = 9

if name == "Alice":
  print('Hi, Alice.')
elif age < 12:
  print('You are not Alice, kiddo.')
elif age > 2000:
  print('Unlike you, Alice is not an undea, immortal vampire.')
elif age > 100:
  print('You are not Alice, grannie')
```

```
Hi, Alice.
```

2.5.1 Exercícios

Desenvolva os programas a seguir através do uso de desvios condicionais na linguagem Python:

1. Supondo que a senha de acesso a um sistema seja "GrandeJogo", peça para o usuário digitar uma senha. Se a senha digitada estiver correta imprima "Acesso autorizado", caso contrário imprima "Acesso negado".

2. Criar um programa que recebe dois números `a` e `b`, e executa as seguintes ações: a) imprime o maior entre os dois, b) caso `b` seja maior que `a + 100` imprime `b é muito maior que a`, c) caso ambos sejam pares ele deverá imprimir `Ambos são pares`, caso contrário deverá imprimir `Pelo menos um numero é impar` e d) se os dois números são múltiplos um do outro deverá imprimir as mensagens `Os numeros são múltiplos entre si` ou `Os numeros não são múltiplos entre si`.

3. Criar um programa que recebe um valor inteiro e três valores reais, `a`, `b` e `c`. Se o inteiro for negativo o programa deverá imprimir os três valores em ordem decrescente. Se o inteiro for maior que 0 o programa deverá imprimir os três valores em ordem crescente. Se o inteiro for igual a zero o programa deverá imprmir o maior valor no meio dos outros dois, começando a impressão pelo menor valor. Além disso, caso a diferença entre `a` e `b` seja menor que `c` em termos absolutos o programa deverá inserir `Números mais`

`próximos entre si do que c`, caso contrário o programa deverá inserir `Números mais distantes entre si do que c`.

4. Crie um programa que pergunta ao usuário se ele quer converter Celsius para Farenheit ou o contrário. O usuário após indicar a conversão desejada deve inserir um número e obter como resposta a temperatura convertida para a nova escala.

5. Criar um programa que recebe um número e imprime `Maior de idade` se a for maior ou igual a 18, imprima `Ja pode votar` se `a>=16` ou `Menor de idade sem poder votar` caso contrário.

6. Peça para o usuário digitar seu peso e sua altura. Em seguida calcule e imprima o valor e a categoria do IMC, onde $IMC = peso/altura^2$, além da faixa de peso saudável. A categoria do IMC é definida pelas seguintes faixas: a) < 16: Magreza grave, b) 16 a < 17: Magreza moderada, c) 17 a $< 18{,}5$: Magreza leve, d) 18,5 a < 25: Saudável, e) 25 a < 30: Sobrepeso, f) 30 a < 35: Obesidade Grau I, g) 35 a < 40: Obesidade Grau II (severa), h) $>= 40$: Obesidade Grau III (mórbida)

7. Um ano é bissexto, isto é, tem 366 dias, se for divisível por 4. Existem, no entanto, exceções: se for divisível por 100 então não é bissexto. Entretanto, se for divisível por 400 é bissexto. Escreva um programa que determine se um ano é bissexto ou não.

8. Escreva um programa que pede para o usuário inserir a hora de início e de fim de um jogo com as horas e os minutos separados por dois pontos. O programa deverá então calcular o tempo de duração do jogo em horas e minutos. O jogo pode começar em um dia e terminar no dia seguinte. O tempo máximo de duração é de 2 horas e 30 minutos.

´

2.6 Repetição

2.6.1 Loops for

O tipo mais comum de alteração na sequência de execução de um programa é o loop. A forma mais simples de implementar um loop é através do comando `for`. Neste caso uma parte do código será repetida tantas vezes quantas se desejar (isto é, quantas vezes for indicado na definição do loop). Vide a seguir.

```
for i in range(12, 18, 2):
  print(i)
```

```
12
14
16
```

2.6.2 Loops while

Outro tipo de controle de fluxo é a repetição condicional de uma instrução. Uma forma de implementar um loop condicional é através do comando `while`. O programa sai do loop quando a condição de teste do `while` deixar de ser `True`.

```
spam = 0
while spam < 5:
  print('Hello, world.')
  spam = spam + 1
```

```
Hello, world.
Hello, world.
Hello, world.
Hello, world.
Hello, world.
```

2.6.3 Saida: break

A saida de um loop pode ser forçada através do comando `break`

```
while True:
  print("Please type your name.")
  name = "your name"
  # name = input()
  if name == "your name":
    break
```

```
Please type your name.
```

```
print('Thank you!')
```

```
Thank you!
```

2.6.4 Retorno: continue

Assim como a saida pode ser forçada com `break` o programa pode ser levado a executar a próxima operação do loop através de `continue`.

```
while True:
  print('Who are you?')
  name = "Joe"
  #name = input()
  if name != "Joe":
    continue
  print('Hello, Joe. What is the password? (It is a fish.)')
```

```
  #password = input()
  password = "swordfish"
  if password == "swordfish":
    break
```

```
Who are you?
Hello, Joe. What is the password? (It is a fish.)
```

```
print("Access granted")
```

```
Access granted
```

2.6.5 Finalização: exit

Finalização forçada de um programa (esteja ele ou não em um loop). A finalização é feita com a função `exit` do pacote `sys`. Observe a forma de utilizar a função e carregar o pacote no exemplo a seguir:

```
import sys

while True:
  print("Type exit to exit")
  response = "exit"
  # response = input()
  if response == "exit":
    sys.exit()
  print("You typed " + response + ".")
```

2.6.6 Exercícios

Resolver os exercícios a seguir utilizando os recursos de repetição da linguagem Python.

1. Imprimir todos os números : a) de 0 a 9 (inclusive), de 2 em 2, b) em ordem decrescente, de 99 a 0, de 2 em 2, c) da sequência: 8, 11, 14, 17, 20, . . . , 83, 86, 89, d) da sequência 100, 98, 96, . . . , 4, 2, e e) que pertencem à serie Fibonacci entre 0 e 50.

2. Determinar os números: a) divisíveis por 7 e múltiplos de 5 existentes entre 1500 e 2700, b) entre 1.000 e 3.000 os quais quando divididos por 11 dão 7 como resultado, c) entre 10 e 100 que não sejam divisíveis por 7. Ao encontrar um número divisível por 7, o programa deve parar o processamento. Dica: use `break`.

3. Escrever um programa que recebe um número n, inteiro e positivo e retorna: a) o n-ésimo termo da série: -1 1 7 9 15 17 23 25 31 33 39 etc..., b) a soma

dos números de 1 até 10 (inclusive), c) a fatorial de n, d) a tabuada de multiplicar (de 1 a 10) para o mesmo.

4. Escreva um programa que recebe dois valores e retorna todos os ímpares entre o menor e o maior deles.

5. Escreva um programa que pede ao usuário para inserir uma string e uma letra. Utilize um loop while para determinar se a letra está ou não presente na string. Caso esteja, imprima a posição da primeira ocorrência da letra na string. Refaça o programa utilizando um loop for e a função break.

6. Escreva um programa que armazena a senha 'brazil2100' em uma variável. O usuário ao executar o programa deverá receber o pedido de inserir a senha. Caso ele faça a inserção correta em uma das três primeiras tentativas o programa deverá imprimir "Acesso autorizado". Caso contrário o programa deverá imprimir "Acesso negado".

7. Escreva um programa que irá jogar "Papel", "Pedra", "Tesoura"com o usuário. Quem ganhar 3 rodadas em sequência será o vencedor do jogo.

8. Escreva um programa que implemente o seguinte algorítmo para o cálculo da raiz quadrada de um número. Suponha que você queira calcular a raiz quadrada de 20. Inicie com 10. Calcule $(10 + 20/10)/2 = 6$. Use o resultado anterior para calcular $(6 + 20/6)/2 = 14/3$. Aplique o processo à fração $14/3$ e calcule uma nova estimativa para a raiz de 20. Pare o algoritmo quando a diferença entre duas estimativas for menor que 10^{-10}. O algoritmo no entanto deverá executar estes cálculos apenas com frações, sem o uso de decimais, inclusive a comparação das duas estimativas que irá interromper o cálculo.

2.7 Listas

Listas e tuplas são conjuntos de números. Listas são criadas com colchetes e tuplas com parênteses. A principal diferença é que o elementos de um tupla, uma vez criados não podem ser modificados. Neste texto vamos concentrar à atenção no estudo das listas.

```
a = (1,2,3,4)
print("Isto eh uma tupla: ", a)
```

```
Isto eh uma tupla:  (1, 2, 3, 4)
```

```
b = [1,2,3,4]
print("Isto eh uma lista: ", b)
```

```
Isto eh uma lista:  [1, 2, 3, 4]
```

2.7.1 Acesso aos elementos

O acesso aos elementos de uma lista é feito pelo seu índice, o qual deve ser colocado entre colchetes após o nome da lista. Vide a seguir. Observe que a numeração dos endereços em Python começa em 0.

```
# Primeiro elemento da lista
a = [1, 2, 3]
a[0]
```

```
1
```

```
# Terceiro valor da lista b
b = ["Albeto", "Bernardo", "Carlos", "Eugenio"]
b[2]
```

```
'Carlos'
```

```
# Novo valor no terceiro elemento da lista b
b[2] = "Thiago"
b[2]
```

```
'Thiago'
```

```
# Tipo de dado do terceiro elemento da lista c
c = ["Bom dia", 3.1415, True, None, 42]
type(c[2])
```

```
<class 'bool'>
```

Um elemento em uma determinada posição pode também ser obtido através do método `.index`

```
# Posição do nome Lucas na lista nomes
nomes = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
nomes.index("Lucas")+1
```

```
3
```

2.7.2 Listas de listas

Os elementos de uma lista podem ser outra lista. O acesso aos elementos da lista dentro da lista é feito com colchetes duplos, tal como pode ser visto a seguir:

```
lista = [["Carlos","Bernardo"],[10,20,30,40,50]]

print(lista[0])
```

```
['Carlos', 'Bernardo']
```

```
print(lista[1])
```

```
[10, 20, 30, 40, 50]
```

```
print(lista[0][1])
```

```
Bernardo
```

```
print(lista[1][4])
```

```
50
```

2.7.3 Concatenação

Para concatenar duas listas basta "soma-las" (na verdade uni-las), conforme pode ser visto a seguir:

```
[1, 2, 3] + ["A", "B", "C"]
```

```
[1, 2, 3, 'A', 'B', 'C']
```

2.7.4 Inserção

A inserção ao final de uma lista pode ser feita por concatenação (torna-se o elemento uma lista envolvendo o mesmo em colchetes e depois concatenamos esta lista na lista original)

```
lista = ["A","B","C"]
elemento = ["D"]
lista = lista + elemento
lista
```

```
['A', 'B', 'C', 'D']
```

Outra forma de executar a inserção de um elemento ao final de uma lista é através do método `.append`

```
lista = ["A","B","C"]
lista.append("D")
lista
```

```
['A', 'B', 'C', 'D']
```

A inserção em uma posição específica é feita por pelo médodo `insert`, descrito a seguir: `nomedalista.insert(posicao, valor)`. Suponha que desejamos ao final ter a seguinte lista de nomes: `nomes = ['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']`. Porém no momento a lista nomes é composta dos seguintes elementos: `nomes = ['Ricardo', 'Tiago', 'Tereza', 'Maria']`. Temos então de inserir `Lucas` e `Joana` na terceira e quartas posições da lista final. Lembre-se que em python a numeração começa em 0, logo a terceira posição é a posição com índice igual a 2.

```
nomes = ["Ricardo", "Tiago", "Tereza", "Maria"]
print(nomes)
```

```
['Ricardo', 'Tiago', 'Tereza', 'Maria']
```

```
nomes.insert(2,"Lucas")
print(nomes)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Tereza', 'Maria']
```

```
nomes.insert(3,"Joana")
print(nomes)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

2.7.5 Replicação

Uma lista pode ter seus elementos replicados `n` vezes através da "multiplicação" da lista por n. Observe um exemplo no código abaixo:

```
["X", "Y", "Z"] * 3
```

```
['X', 'Y', 'Z', 'X', 'Y', 'Z', 'X', 'Y', 'Z']
```

2.7.6 Eliminação

Já a eliminação de valores é feita pelo comando `del`.

```
nomes = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
print(nomes)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
del nomes[2]
print(nomes)
```

```
['Ricardo', 'Tiago', 'Joana', 'Tereza', 'Maria']
```

```
del nomes[2]
print(nomes)
```

```
['Ricardo', 'Tiago', 'Tereza', 'Maria']
```

2.7.7 Classificação

Por último a classificação dos elementos de uma lista pode ser feita pelo método `.sort`

```
# Lista de nomes classificada em ordem alfabetica
nomes = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
nomes.sort()
nomes
```

```
['Joana', 'Lucas', 'Maria', 'Ricardo', 'Tereza', 'Tiago']
```

2.7.8 Slices

O comprimento (número de elementos) da lista é obtido com a função `len`.

```
lista = ["A","B","C","D","E","F","G","H","I"]
n = len(lista)
print("A lista tem", n, "elementos")
```

```
A lista tem 9 elementos
```

Conforme já mencionado, pode-se obter um elemento específico de uma lista através de seu endereço. Devemos lembrar que os endereços em uma lista começam em 0 e vão até $n-1$ onde n é o número de elementos da lista.

```
lista = ["A","B","C","D","E","F","G","H","I"]
lista[0]
```

```
'A'
```

Para obter uma parte dos elementos de uma lista (um *slice* da lista) passamos uma faixa de endereços $i:j$. Isto trará como resultado os elementos de índice i até $j-1$. Por exemplo, para obter os elementos de índice 1 até 4 passamos `lista[1:5]`:

```
lista = ["A","B","C","D","E","F","G","H","I"]
lista[1:5]
```

```
['B', 'C', 'D', 'E']
```

Para obter do elemento de índice 1 até o de índice 5, porém de duas em duas posições fazemos:

```
lista = ["A","B","C","D","E","F","G","H","I"]
lista[1:6:2]
```

```
['B', 'D', 'F']
```

Caso um dos índices seja omitido ele será considerado o índice 0 ou o índice do último elemento ou se for o passo de aumento será considerado 1. Sendo assim, para obter do elemento de índice 0 (ínicio da lista) até o elemento de índice 4 podemos fazer:

```
lista = ["A","B","C","D","E","F","G","H","I"]
lista[0:5], lista[:5]
```

```
(['A', 'B', 'C', 'D', 'E'], ['A', 'B', 'C', 'D', 'E'])
```

Por outro lado, para obter do elemento de índice 3 até o último elemento da lista fazemos:

```
lista = ["A","B","C","D","E","F","G","H","I"]
lista[3:9], lista[3:]
```

```
(['D', 'E', 'F', 'G', 'H', 'I'], ['D', 'E', 'F', 'G', 'H', 'I'])
```

Uma das características mais exclusivas nas listas em Python é o sistema de numeração negativo. Para entender este sistema você tem que imaginar que cada elemento na lista tem dois endereços. Um deles é positivo, começando do primeiro elemento (elemento de índice 0) e se movendo da esquerda para a direita. O outro é negativo, começa pelo último elemento (elemento de índice -1) e se move da direita para a esquerda. Para entender suponha a lista a seguir:

```
lista = ["A","B","C","D","E","F"]
lista
```

```
['A', 'B', 'C', 'D', 'E', 'F']
```

Seus elementos terão os seguintes índices:

Elemento	A	B	C	D	E	F
Índice Positivo	0	1	2	3	4	5
Índice Negativo	-6	-5	-4	-3	-2	-1

Os índices devem ser entendidos de forma independente. O que interessa é o elemento inicial e o final em si e como queremos ir de um para o outro, ou seja da esquerda para a direita (com passo positivo) ou da direita para a esquerda (com passo negativo).

Por exemplo, podemos obter o slice com os elementos `[B, C, D]` das seguintes formas, todas elas obtidas com passo positivo:

Usando apenas os endereços positivos

```
lista = ["A","B","C","D","E","F"]
lista[1:4:1]
```

```
['B', 'C', 'D']
```

Usando apenas os endereços negativos

```
lista = ["A","B","C","D","E","F"]
lista[-5:-2:1]
```

```
['B', 'C', 'D']
```

Misturando os endereços, porém contando sempre de forma crescente

```
lista = ["A","B","C","D","E","F"]
lista[-5:4:1]
```

```
['B', 'C', 'D']
```

```
lista = ["A","B","C","D","E","F"]
lista[1:-2:1]
```

```
['B', 'C', 'D']
```

Ao contar de forma descrescente invertemos a sequência de elementos. Por exemplo, poderíamos ter invertido a ordem da lista fazendo:

```
lista = ["A","B","C","D","E","F"]
lista[5:-7:-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

Ou:

```
lista = ["A","B","C","D","E","F"]
lista[-1:-7:-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

```
lista = ["A","B","C","D","E","F"]
lista[-1::-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

```
lista = ["A","B","C","D","E","F"]
lista[::-1]
```

```
['F', 'E', 'D', 'C', 'B', 'A']
```

Ou então poderíamos ter obtido o slice [“D,”“C,”“B”] contando de forma descrescente e passando os seguintes endereços, inicial e final:

```
lista = ["A","B","C","D","E","F"]
lista[3:0:-1]
```

```
['D', 'C', 'B']
```

Como os endereços 0 e -1 representam elementos distintos, se quisermos ir do elemento de índice 3 (neste caso “D”) até o primeiro elemento (neste caso “A”) podemos proceder conforme abaixo:

```
lista = ["A","B","C","D","E","F"]
lista[3::-1]
```

```
['D', 'C', 'B', 'A']
```

Se quisermos colocar um número entre os dois pontos, não podemos colocar -1 para que a contagem pare um antes (no índice 0). Para isso temos que pensar que os índices negativos continuam para a direita. Como o primeiro elemento nesta lista tem índice negativo -6, devemos colocar -7 na segunda posição.

```
lista = ["A","B","C","D","E","F"]
lista[3:-7:-1]
```

```
['D', 'C', 'B', 'A']
```

Para não esquecer, as listas tem dois sistemas de numeração, independentes. Cada elemento tem dois índices, um começando em 0 a partir da esquerda (primeiro elemento) e outro começando em -1 a partir da direita (último elemento).

2.7.9 Loops

Para começar, observe a forma como são obtidos os elementos de uma lista quando ela é colocada em um loop do Python. Cada elemento é obtido como uma variável. Se quisermos concatenar o mesmo em outra lista ele deverá ser transformado em lista primeiro (colocando-se o mesmo entre colchetes). No caso do método `.append` isto não é necessário pois .append pressupõe que será passado uma variável individual para ser acrescentada em uma lista.

```
nomesdosalunos = ["Ricardo", "Tiago", "Lucas"]
nomesdasalunas = ["Joana", "Tereza", "Maria"]

nomes = []

for nome in nomesdosalunos:
  nomes = nomes + [nome] #Insercao por concatenacao

for nome in nomesdasalunas:
  nomes.append(nome) #Insercao pelo metodo .append()

print(nomes)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

2.7.10 Compreensão

A compreensão de uma lista (*list comprehension*) é uma forma de acessar seus elementos um a um, tal qual em um loop. Observe o exemplo a seguir:

```
nomes = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
[str(nome) for nome in nomes]
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
tipos = [str(nome) for nome in nomes]
for valor in tipos:
  print(valor)
```

```
Ricardo
Tiago
Lucas
Joana
Tereza
Maria
```

2.7.11 Enumeração

Outra forma é a enumeração de lista. A diferença neste caso está na possibilidade de se controlar simultaneamente o índice e o valor do elemento da lista que está sendo utilizado em um loop. Observe a seguir:

```
lista = ["A","B","C","D","E"]

for indice, valor in enumerate(lista):
  print(indice, valor)
```

```
0 A
1 B
2 C
3 D
4 E
```

2.7.12 Operador in

Para determinar se um elemento está ou não na lista podemos utilizar o operador `in` o qual retorna `True` ou `False` se o elemento estiver ou não na lista, respectivamente.

```
nomes = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
aluna = "Tereza"

print(aluna in nomes)
```

```
True
```

2.7.13 zips de Listas

Podemos também agrupar duas listas e percorre-las em paralelo através de um `zip`. Um `zip` é um agrupamento de objetos quaisquer, listas ou não. Abaixo pode-se ver o efeito em um loop ao criar um `zip` a partir de duas listas.

```
lista1 = [10,20,30,40,50,60]
lista2 = ["A","B","C","D","E","F"]

for i,j in zip(lista1,lista2):
  print(i, j)
```

```
10 A
20 B
30 C
40 D
```

```
50 E
60 F
```

2.7.14 sets

Por último temos os `sets` (conjuntos). Um set serve (por exemplo) para você encontrar os valores únicos em uma lista.

```
lista = [1,2,3,1,2,3,1,2,3,1,2,3]
conjunto = set(lista)
print(conjunto)
```

```
{1, 2, 3}
```

Ou então um `set` pode ser utilizado para determinar os elementos que pertençam a duas listas. Suponha por exemplo as listas x=[1,3,6,78,35,55] e y=[12,24,35,24,88,120,155,3]. O programa a seguir determina os elementos comuns entre `x` e `y`.

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = list(set(x))
y1 = list(set(y))

z = []
for i in x1:
  if i in y1:
    z.append(i)

print(z)
```

```
[3, 35]
```

O problema anterior poderia ser resolvido facilmente através do uso do operador de intersecção "&," como podemos ver a seguir:

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = set(x)
y1 = set(y)

z = x1 & y1 #Operador de interseccao. Funciona apenas com "sets"
print(z)
```

```
{35, 3}
```

Caso queira-se determinar os elementos que pertencem a qualquer uma das listas podemos utilizar o operador de união "|" (vide a seguir).

```
x=[1,3,6,78,35,55]
y=[12,24,35,24,88,120,155,3]

x1 = set(x)
y1 = set(y)

z = x1 | y1 #Operador de união. Funciona apenas com "sets"
print(z)
```

```
{1, 3, 35, 6, 88, 12, 78, 55, 24, 155, 120}
```

2.7.15 Designação

Se fizermos a = 4 e b = a, ao alterarmos o valor de a, não iremos alterar o valor de b. Isto se chama passagem por valor. Neste caso estamos lidando com varíaveis escalares (sem dimensão). Observe a seguir.

```
a = 4
b = a
print(b)
```

```
4
```

```
a = 5
print(b)
```

```
4
```

Com as listas ocorre algo diferente. Os valores dos seus elementos são passados por referência. Logo se criamos uma lista `nomes1 = ['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']` e depois fazemos outra lista (`nomes2` por exemplo) ser igual a `nomes1`, a alteração de um elemento em `nomes1` irá se refletir em `nomes2`. Isto se chama passagem por referência. Observe a seguir.

```
nomes1 = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
nomes2 = nomes1
print(nomes2)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
nomes1[0] = "Gustavo"
print(nomes2)
```

```
['Gustavo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

No entanto o nome da lista em sí é passado por valor! Isto significa que se criarmos uma lista `nomes1 = ['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']` e depois fizermos `nomes2 = nomes1`, se trocarmos a lista inteira, `nomes1` para outro conjunto de valor, `nomes2` não será alterada. Observe a seguir:

```
nomes1 = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]
nomes2 = nomes1
print(nomes2)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
nomes1 = ['André']
print(nomes2)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

Caso você queira criar listas "desconectadas" isto é, cujos valores foram passados por "valor," deverá utilizar a função `copy` do modulo de mesmo nome. Caso você esteja copiando uma lista que tenha outras listas, utilize a função `deepcopy` (também do módulo `copy`)

```
import copy as cp
nomes1 = ["Ricardo", "Tiago", "Lucas", "Joana", "Tereza", "Maria"]

nomes2 = cp.copy(nomes1)
print(nomes2)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

```
nomes[1] = "Cristina"
print(nomes2)
```

```
['Ricardo', 'Tiago', 'Lucas', 'Joana', 'Tereza', 'Maria']
```

Este tipo de passagem de dados em que os elementos de uma lista são passados por referência e o nome da lista por valor é chamado de passagem por designação.

2.7.16 Exercícios

Resolver os problemas a seguir através do uso de listas na linguagem Python:

1. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Determine quantos elementos de `a` são negativos

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
sum([i<0 for i in a])
```

```
5
```

2. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Crie uma lista com as posições dos elementos negativos

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
lista = []
for i, valor in enumerate(a):
  if valor<0:
    lista.append(i)
print(lista)
```

```
[0, 3, 4, 7, 9]
```

3. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Substitua todos os valores negativos por 1 na lista original e apresente a lista resultante.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
for i, valor in enumerate(a):
  if valor<0:
    a[i] = 1
print(a)
```

```
[1, 5, 6, 1, 1, 4, 7, 1, 9, 1]
```

4. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Determine o maior elemento na lista e a posição do mesmo.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
lista = []
for i, valor in enumerate(a):
  if valor == max(a):
    lista.append(i)
print(max(a), lista)
```

```
9 [8]
```

5. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. classifique os valores da mesma em ordem crescente e apresente a nova lista ordenada.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
a.sort()
print(a)
```

```
[-12, -10, -3, -2, -1, 4, 5, 6, 7, 9]
```

6. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Inverta a ordem de apresentação dos elementos da mesma e apresente o resultado.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
print(a[::-1])
```

```
[-2, 9, -12, 7, 4, -3, -1, 6, 5, -10]
```

7. Suponha a lista `a=[-10,5,6,-1,-3,4,7,-12,9,-2]`. Troque os elementos de posição par com os de posição impar. Observe que no python o primeiro elemento (portanto um elemento de posição ímpar) tem índice igual a zero (portanto um índice par)

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]

for indice, valor in enumerate(a[:-1]):
  if indice % 2 == 0:
    a1 = a[indice]
    a2 = a[indice+1]
    a[indice] = a2
    a[indice+1] = a1

print(a)
```

```
[5, -10, -1, 6, 4, -3, -12, 7, -2, 9]
```

8. Suponha as listas `a = [-10,5,6,-1,-3,4,7,-12,9,-2]` e `b = [-10,5,6,-1,-3,4,7,-12,9]`. Troque os elementos acima da posição mediana da lista com os elementos abaixo. Por exemplo se a lista tiver 10 elementos, o programa deverá trocar o primeiro com o sexto, o segundo com o sétimo e assim por diante.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]

n = len(a)

if n % 2 == 0:
  abaixo = a[:int(n/2)]
```

```
  meio = []
  acima = a[int(n/2):]
else:
  abaixo = a[:int(n/2)]
  meio = a[int(n/2)]
  acima = a[int(n/2)+1:]

a = acima + meio + abaixo
print(a)
```

```
[4, 7, -12, 9, -2, -10, 5, 6, -1, -3]
```

```
b = [-10,5,6,-1,-3,4,7,-12,9]

n = len(b)

if n % 2 == 0:
  abaixo = b[:int(n/2)]
  meio = []
  acima = b[int(n/2):]
else:
  abaixo = b[:int(n/2)]
  meio = [b[int(n/2)]]
  acima = b[int(n/2)+1:]

b = acima + meio + abaixo
print(b)
```

```
[4, 7, -12, 9, -3, -10, 5, 6, -1]
```

9. Suponha a lista `a = [-10,5,6,-1,-3,4,7,-12,9,-2]`. Multiplique todos os elementos dela pelo número 3 e apresente a nova lista.

```
a = [-10,5,6,-1,-3,4,7,-12,9,-2]
print([i*3 for i in a])
```

```
[-30, 15, 18, -3, -9, 12, 21, -36, 27, -6]
```

10. Escreva um programa que recebe uma lista com 20 valores. Estes valores deverão primeiro serem ordenados de forma crescente. Em seguida o programa irá receber mais 10 valores. Cada um destes valores deverá ser inserido na posição correta na lista de forma a mante-la ordenada com valores crescentes. Apresentar a lista resultante de 30 números ordenados de forma crescente ao final.

11. Escreva um programa que lê uma lista com 20 elementos e em seguida calcula para cada valor distinto da lista original a quantidade de vezes em que o mesmo aparece repetido.

2.8 Funções

Um dos mais importantes conceitos de programação em Python é o de função. Uma função permite encapsular uma parte do código evitando a necessidade de repetir os comandos todas vez que uma operação tenha de ser executada.

O usuário tanto pode utilizar as funções nativas do Python (como por exemplo a `print`) quanto criar suas próprias funções. Por exemplo a função a seguir recebe um número como parâmetro e retorna um valor de texto dizendo se o mesmo é maior ou menor que 5. Observe que a função não imprime o resultado, apenas repassa um valor de texto para o programa principal.

```
def maiorque5(x):
  if x>5:
    a = "O numero eh maior que 5"
  else:
    a = "O numero eh menor ou igual a 5"

  return(a)

c = 10
b = maiorque5(c)
print(b)
```

```
O numero eh maior que 5
```

2.8.1 Escopo

As funções criam outro importante conceito de programação no Python que é o Escopo da variável. O escopo é o local de validade do valor da variável, e pode ser local (apenas na função que a criou) ou global (válido para todas as funções do ambiente). No exemplo a seguir vamos criar uma variável que será global `x` e uma variável que será local `y`. Observe que `x` por ter sido criado no escopo global, "existe" no escopo que fica dentro da função.

```
def funcao1():
  y = 5
  print('Este é o valor de y no escopo local da funcao1:',y, "\n")
  print('Este é o valor de x no escopo local da funcao1:',x, "\n")
  return

x = 10
```

```
print('Este é o valor de x no escopo global:', x, "\n")
```

```
Este é o valor de x no escopo global: 10
```

```
funcao1()
```

```
Este é o valor de y no escopo local da funcao1: 5

Este é o valor de x no escopo local da funcao1: 10
```

No entanto se tentarmos imprimir fora de `funcao1` o valor de y (variável que foi criada dentro do escopo de `funcao1`), obteremos um erro. Observe abaixo:

```
def funcao1():
  y = 5
  print('Este é o valor de y no escopo local da funcao1:',y, "\n")
  return

funcao1()
```

```
Este é o valor de y no escopo local da funcao1: 5
```

```
try:
  print('Este é o valor de y no escopo global:', y, "\n")
except:
  print('y não existe no escopo global')
```

```
Este é o valor de y no escopo global: [12, 24, 35, 24, 88, 120, 155, 3]
```

Caso você queira que uma variável criada dentro de uma função seja acessível fora da mesma (no escopo `global`) a mesma deve ser definida como tal, dentro da função, tal como pode ser visto a seguir. Esta no entanto não é considerada uma boa prática de programação e deve ser evitada sempre que possível.

```
def funcao1():
  global y
  y = 5
  print('Este é o valor de y no escopo local da funcao1:',y, "\n")
  return

funcao1()
```

```
Este é o valor de y no escopo local da funcao1: 5
```

```
print('Este é o valor de y no escopo global:', y, "\n")
```

```
Este é o valor de y no escopo global: 5
```

O mais usual é a passagem de valores do escopo global para o escopo de uma função através do uso de argumentos na definição da função. Observe a seguir:

```
def funcao1(y):
  print('Este é o valor de y no escopo local da funcao1:',y, "\n")
  return

x = 10
funcao1(x)
```

```
Este é o valor de y no escopo local da funcao1: 10
```

Outro ponto importante é que se temos uma variável x no escopo global e criamos outra variável de mesmo nome no escopo local de uma função, a variável x no escopo global não será alterada.

```
def funcao1():
  x = 7
  print('Este é o valor de x no escopo local da funcao1:',x, "\n")
  return

x = 10
print('Este é o valor de x no escopo global, antes de executar funcao1:',x,
```

```
Este é o valor de x no escopo global, antes de executar funcao1: 10
```

```
funcao1()
```

```
Este é o valor de x no escopo local da funcao1: 7
```

```
print('Este é o valor de x no escopo global, depois de executar funcao1:',x,
```

```
Este é o valor de x no escopo global, depois de executar funcao1: 10
```

2.8.2 Designação, II

Lembre-se que com as listas ocorre um processo um pouco diferente, chamado passagem por designação. Recordando, na passagem por designação o nome da lista é passado por valor. Logo se criarmos uma lista, passarmos ela para uma função e lá dentro da função trocarmos o valor designado ao nome por completo isto não terá efeito no escopo global. Vide a seguir

```
lista = [1,2,3,4,5]

def funcao1(valor):
  print('valor no escopo local da funcao1:',valor)
  valor = 3
  print('valor no escopo local da funcao1:',valor)

print('lista no escopo global, antes de funcao1:',lista)
```

```
lista no escopo global, antes de funcao1: [1, 2, 3, 4, 5]
```

```
funcao1(lista)
```

```
valor no escopo local da funcao1: [1, 2, 3, 4, 5]
valor no escopo local da funcao1: 3
```

```
print('lista no escopo global, depois de funcao1:',lista)
```

```
lista no escopo global, depois de funcao1: [1, 2, 3, 4, 5]
```

Porém vamos recordar que os elementos de uma lista são passados por referência. Sendo assim, se ao invés de alterarmos o valor de toda a lista na função, alterarmos um de seus elementos, esta alteração se refletirá no escopo global. Observe que iremos alterar o valor de um dos elementos do argumento passado como parâmetro dentro da função e esta alteração irá se refletir no escopo global.

```
lista = [1,2,3,4,5]

def funcao1(valor):
  print('valor no escopo local da funcao1:',valor)
  valor[0] = -3
  print('valor no escopo local da funcao1:',valor)

print('lista no escopo global, antes de funcao1:',lista)
```

```
lista no escopo global, antes de funcao1: [1, 2, 3, 4, 5]
```

```
funcao1(lista)
```

```
valor no escopo local da funcao1: [1, 2, 3, 4, 5]
valor no escopo local da funcao1: [-3, 2, 3, 4, 5]
```

```
print('lista no escopo global, depois de funcao1:',lista)
```

```
lista no escopo global, depois de funcao1: [-3, 2, 3, 4, 5]
```

Como já foi dito anteriormente, esta combinação de passagem do objeto em si por referência e do nome dele por valor se chama passagem por designação.

2.8.3 Exercícios

Resolva os exercícios a seguir através do recurso de funções definidas pelo usuário da linguagem Python

1. Escreva uma função que recebe um número inteiro e retorna: a) o quadrado do mesmo, b) todos os cubos de 0 a n, c) o seu valor absoluto, d) a sua fatorial, inclusive do 0. Caso seja fornecido um número negativo, fracionário ou uma string deverá ser fornecida como resposta uma mensagem de erro específica para cada caso.
2. Escreva uma função que recebe dois números (a,b), retorna o maior deles e uma das mensagens a seguir: a) "a é o maior", b) "a é igual a b"ou c) "b é o maior"
3. Escreva uma função que converte a temperatura em Celsius para Fahrenheit (C = 9/5F + 32)
4. Escreva uma função que receba a área de um círculo e retorne o comprimento de sua circunferência
5. Escreva uma função que recebe um número e determina se o mesmo é igual quando lido da esquerda para a direita ou da direita para a esquerda.

2.9 Strings, II

O tratamento de variáveis com valores de texto (strings) é extremamente sofisticado na linguagem Python, a qual conta com inúmeras funções, conforme poderemos observar.

2.9.1 Conversão em Lista

O mais interessante recurso é a possibilidade de manipular uma string (`Python` por exemplo) como uma lista cujos elementos são cada um dos caracteres originais da string.

```
nome = "Gustavo"
print(nome)
```

```
Gustavo
```

```
lista = list(nome)
print(lista)
```

```
['G', 'u', 's', 't', 'a', 'v', 'o']
```

2.9.2 Reconversão em String

Da mesma forma, uma lista composta de caracteres pode ser transformada em uma única variável de texto. Para isso utilizamos o método `.join`.

```
nome = "Gustavo"
lista = list(nome)
print(lista)
```

```
['G', 'u', 's', 't', 'a', 'v', 'o']
```

```
novo_nome = "".join(lista)
print(novo_nome)
```

```
Gustavo
```

2.9.3 Detecção de Caixa

Os métodos `.islower` e `isupper` permitem determinar se todos os caracteres de uma string são formados por minúsculas ou maiúsculas.

```
nome = "gustavo"
print(nome.islower())
```

```
True
```

```
nome = "GUSTAVO"
print(nome.isupper())
```

```
True
```

2.9.4 Conversão de Caixa

De forma similar os métodos `.lower()\tthl{ e }.upper()` permitem converter todos os caracteres de uma string nos seus equivalentes em minúsculas ou maiúsculas.

```
nome = "gustavo"
print(nome.upper())
```

```
GUSTAVO
```

```
nome = "GUSTAVO"
print(nome.lower())
```

```
gustavo
```

2.9.5 Exercícios

Desenvolva funções de manipulação de strings em Python que:

1. Recebe uma string e troca todos os espaços para sublinhado. Use como string exemplo: `Inteligencia Artificial e Machine Learning`.

```
nome = "Inteligencia Artificial e Machine Learning"
nome = list(nome)
for indice, valor in enumerate(nome):
  if valor == " ":
    nome[indice] = "_"

nome = "".join(nome)
print(nome)
```

```
Inteligencia_Artificial_e_Machine_Learning
```

2. Recebe uma string e troca os pontos por vírgula. Exemplo: `32.054,23`. Resultado esperado: `32,054.23`

3. Recebe uma string e: a) remove os espaços e b) troca a letra `l` por `1`. Exemplo: `Gustavo Correa Mirapalheta`. Resultado esperado: `Gustavo CorreaMirapa1heta`

4. Recebe uma string e troca todos os caracteres presentes em `lista = [' ','.','-',';']` por "_"e passe a caixa das letras para caixa baixa. Use como string exemplo `Inteligencia.Artificial e-Machine;Learning`.

```
nome = 'Inteligencia.Artificial e-Machine;Learning'
lista = [" ",".","-",";"]

def troca(nome, lista):
  nome = list(nome.lower())
  for indice, valor in enumerate(nome):
    if valor in lista:
      nome[indice] = "_"
  nome = "".join(nome)
  return(nome)
```

```
troca(nome, lista)
```

```
'inteligencia_artificial_e_machine_learning'
```

5. Recebe uma string e troca todos os caracteres presentes em `lista1 = [' ','.','-',';']` pelos valores de mesma posição em `lista2 = ['_','_','_','_']` e passe a caixa dos mesmos para caixa baixa. Use como string exemplo `Inteligencia.Artificial e-Machine;Learning`.

```
nome = 'Inteligencia.Artificial e-Machine;Learning'
lista1 = [" ",".","-",";"]
lista2 = ["_","_","_","_"]

def troca(nome, lista1, lista2):
  nome = list(nome.lower())
  for indice, valor in enumerate(nome):
    if valor in lista1:
      nome[indice] = lista2[lista1.index(valor)]
  nome = "".join(nome)
  return(nome)

print(troca(nome, lista1, lista2))
```

```
inteligencia_artificial_e_machine_learning
```

6. Recebe uma string e troca todas as minúsculas por maiúsculas e vice-versa.
7. Recebe uma string e um número e retorna todos os caracteres que ocorrem em posições múltiplas do número.
8. Recebe uma string e um caracter e retorna todas as posições na string em que ocorre o caracter.
9. Recebe duas strings e determina quantos caracteres iguais na mesma posição existem em ambas.
10. Recebe duas strings e determina se elas diferem por apenas um caracter.

2.10 Dicionários

Um dicionário é uma estrutura na qual os dados aparecem em pares denominados "key-value" (ou "chave-valor"). Não existe uma sequência prédeterminada de valores nos dicionários (ao contrário das listas). No entanto podemos listas o conjunto de chaves, de valores e de pares de chaves e valores com os métodos `.keys`, `values` e `items` respectivamente.

```
# Criação do dicionário
a = {"Logica":8, "Financas":6, "Marketing":10, \
     "Matematica":7, "Eletiva":"Nenhuma"}
```

```
# Inserção de par chave valor
a["Imersao"]=True
```

```
# Valor para a chave "Logica"
a["Logica"]
```

```
8
```

```
# Lista de chaves do dicionario
for chave in list(a.keys()):
  print(chave)
```

```
Logica
Financas
Marketing
Matematica
Eletiva
Imersao
```

```
# Lista de valores do diconario
for valor in list(a.values()):
  print(valor)
```

```
8
6
10
7
Nenhuma
True
```

```
# Lista de pares chave,valor
for par in list(a.items()):
  print(par)
```

```
('Logica', 8)
('Financas', 6)
('Marketing', 10)
('Matematica', 7)
('Eletiva', 'Nenhuma')
('Imersao', True)
```

2.10.1 Procura de Valor

A recuperação de um valor a partir de sua chave em um dicionário pode ser feita pelo método `.get()`. O método `get` requer dois valores, a `key` de pesquisa no dicionário e o valor que deve ser retornado caso a `key` não seja encontrada.

```
a = {"Logica":8, "Financas":6, "Marketing":10, \
     "Matematica":7, "Eletiva":"Nenhuma"}
a["Imersao"]=True

a.get("Financas",False)
```

```
6
```

```
a.get("Microeconomia",False)
```

```
False
```

O método `setdefault` permite atribuir um valor para uma determinada chave, caso ela ainda não exista no dicionário

```
a = {"Logica":8, "Financas":6, "Marketing":10, \
     "Matematica":7, "Eletiva":"Nenhuma"}
a["Imersao"]=True

a.setdefault("Microeconomia", True)
```

```
True
```

```
a.get("Microeconomia", 0)
```

```
True
```

Um exemplo de aplicação do método setdefault() pode ser visto abaixo na contagem dos diferentes caracteres de um texto

```
message = 'It was a bright cold day in April, and '
message = message + 'the clocks were striking thirteen.'

count = {} #Cria um dicionario vazio

for character in message:
  a = count.setdefault(character, 0)
  count[character] = count[character] + 1

i = 1
for par in count.items():
```

```
  if i // 5 == i /5:
    print(par)
  else:
    print(par, end=" ")
  i = i+1
```

```
('I', 1) ('t', 6) (' ', 13) ('w', 2) ('a', 4)
('s', 3) ('b', 1) ('r', 5) ('i', 6) ('g', 2)
('h', 3) ('c', 3) ('o', 2) ('l', 3) ('d', 3)
('y', 1) ('n', 4) ('A', 1) ('p', 1) (',', 1)
('e', 5) ('k', 2) ('.', 1)
```

```
print("Célula executada ok")
```

```
Célula executada ok
```

2.10.2 Exercícios

Resolva os exercícios a seguir utilizando dicionários na lingugagem Python.

1. Escreva um dicionário com os seguintes pares chave-valor: a) `'banana': 4`, b) `'apple': 2`, c) `'orange': 1.5`, d) `'pear': 3`
2. Suponha o dicionário: `acoes= { 'BBDC4': 'Bradesco', 'ITUB4':'Itau Unibanco', 'VALE3':'Vale do Rio Doce'}`. Adicione ao dicionário BBAS3, Banco do Brasil e exclua do dicionário BBDC4, Bradesco.
3. Escreva um código para ter o valor máximo e mínimo de um dicionário. Supondo `myDict = {'x':500, 'w':5103 , 'y':5874, 'z': 560}`, o resultado esperado seria `{'Valor maximo': 5874, 'Valor minimo' : 500}`
4. Escreva o programa a seguir de duas formas: a) utilizando loops e b) utilizando a compreensão de dicionário. O programa deverá receber um número n e a partir do mesmo produzir um dicionário com todos os números de 1 até n (inclusive) como chaves e n ao quadrado como valores. Ao final o dicionário deverá ser impresso. Por exemplo se n for igual a 8 deverá ser impresso o seguinte dicionário: `{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}`. Dica: um dicionário vazio pode ser criado com `dicionario = {}`. A inserção de uma nova chave pode ser feita com `dicionario[chave]=valor`
5. Escreva um código para criar um dicionário com a contagem de cada letra em um texto. Suponha: texto = `Fundacao Getulio Vargas`. Resultado esperado: `{' ': 2, 'F': 1, 'G': 1, 'V': 1, 'a': 3, 'd': 1, 'e': 1, 'g': 1, 'i': 1, 'l': 1, 'n': 1, 'o': 2, 'r': 1, 's': 1, 't': 1, 'u': 1, 'ã': 1, 'ç': 1, 'ú': 1}`

6. Crie um dicionário com o nome dos meses do ano como chaves e o número de dias em cada mês como valores. A partir deste dicionário crie uma função que verifica apenas as três primeiras letras do mês fornecido como parâmetro e devolve o número de dias neste mês.

7. Um método simples de criptografia é o da substituição de letras. Por exemplo a ser substituido por e, b por a, etc... O único cuidado necessário neste caso é manter as substituições únicas. Crie um programa que recebe duas strings e determina se elas podem ser utilizads como base de um código de substituição. Por exemplo `casa` e `zero` não serviriam não saberíamos como transformar o a (para e ou o?) tanto na ida quanto na volta. Se elas puderem, deverão ser armazenadas em um dicionário como um par `chave:valor`.

8. Na identificação de padrões em DNA iniciamos com sequências básicas conhecidas tais como: `L = ['AATAATAC', 'CATAATCA', 'AAATTCTA', 'AATACTAT', 'ACATATTA']` e tentamos identificar em qual sequência básica uma amostra com valores conhecidos e desconhecidos se encaixaria, por exemplo AA. Neste caso esta amostra se encaixa na primeira e quarta sequências. Uma forma de determinar em quais sequências básicas uma amostra se encaixa é utilizar um dicionário onde as chaves são os índices com os valores conhecidos nas sequências básicas e os valores são os caracteres. Crie um programa que usa uma estrutura tipo dicionário para determinar em quais sequências básicas se encaixa uma amostra fornecida pelo usuário.

9. Crie um programa que converte números Romanos em números decimais. As chaves de básicas conversão são: `M=1000, D=500, C=100, L=50, X=10, V=5, I=1`. Além disso inclua combinações como `IV=4, XL=40`, etc... Utilize para formar os pares `chave:valor` uma estrutura de dicionários.

10. Crie um programa em que o usuário insere uma string e o computador lista todas as palavras que podem ser formadas com as letras desta string.

11. Monte um tabuleiro e um sistema de registro para um Jogo da Velha

```
# Tabuleiro Vazio

print("Estrutura de dados")
theBoard = {'top-L': " ", 'top-M': " ", 'top-R': " ",
            'mid-L': " ", 'mid-M': " ", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': " "}
print(theBoard)
print()

print("Tabuleiro vazio")
def printBoard(board):
  print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
  print('-+-+-')
  print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
```

```
  print('-+-+-')
  print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])

printBoard(theBoard)
print()

print("Tabuleiro com uma jogada registrada")
theBoard = {'top-L': "O", 'top-M': "O", 'top-R': "O",
            'mid-L': "X", 'mid-M': "X", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': "X"}

# Registro de Jogadas

theBoard = {'top-L': " ", 'top-M': " ", 'top-R': " ",
            'mid-L': " ", 'mid-M': " ", 'mid-R': " ",
            'low-L': " ", 'low-M': " ", 'low-R': " "}

def printBoard(board):
  print(board['top-L'] + '|' + board['top-M'] + '|' + board['top-R'])
  print('-+-+-')
  print(board['mid-L'] + '|' + board['mid-M'] + '|' + board['mid-R'])
  print('-+-+-')
  print(board['low-L'] + '|' + board['low-M'] + '|' + board['low-R'])

turn = "X"

for i in range(9):
  printBoard(theBoard)
  print("Turn for " + turn + '. Move on which space?')
  move = input()
  theBoard[move] = turn
  if turn == "X":
    turn = "O"
  else:
    turn = "X"

printBoard(theBoard)
```

2.11 Classes

Um dos mais sofisticados recursos de programação é o da Orientação a Objetos. Nele criamos uma `classe` (família) de objetos com características específicas. Durante a execução do programa instanciamos a classe e criamos objetos (variáveis) específicas as quais "nascem" com todas as propriedades definidas pela

classe.

Para criar uma classe devemos atentar para um certo número de regras essenciais: a) Comece definindo o nome da classe com o comando `class NomeDaClasse`, b) Todos os métodos que a classe irá conter serão definidos por `def NomeDoMetodo(self, parâmetros):`, c) O primeiro método sempre será definido por: `def __init__(self, parâmetros):`, d) Um parâmetro (x por exemplO) usado na criação da classe deve ser renomeado para `self.x = x`, de forma a ser utilizado pelos outro métodos da classe.

A seguir são apresentados exemplos de criação de classes.

1. Crie uma classe que tenha dois métodos: a) `getString`: para obter uma string e b) `printString`: para imprimir a string em caixa alta

```
class caixaAlta:
  def __init__(self, texto1="texto nulo"):
    self.texto1 = texto1

  def getString(self,texto2):
    self.texto1 = texto2

  def printString(self):
    print(self.texto1.upper())

teste = caixaAlta()
teste.printString()
```

```
TEXTO NULO
```

```
teste.getString("Isto eh um teste")
teste.printString()
```

```
ISTO EH UM TESTE
```

2. Crie uma classe de nome Retangulo, na qual, quando um objeto for instanciado, receberá valores de largura e comprimento. A classe deverá ter um método Area, o qual irá calcular a área do objeto.

```
class Retangulo:
  def __init__(self, larg, compr):
    self.larg = larg
    self.compr= compr

  def Area(self):
    area = self.larg * self.compr
    return(area)
```

```
obj = Retangulo(4,3)
print(obj.Area())
```

```
12
```

3. Crie uma classe, a qual ao ser criada recebe o número N. Quando for utilizado o método `.generate` deverá ser passado um número x. A classe então irá devolver uma lista com todos os números entre 0 e x (inclusive) que sejam divisíveis por N.

```
class divisivelPorN:
  def __init__(self, N):
    self.N = N

  def generate(self, x):
    lista = []
    for i in range(x+1):
      if i % self.N == 0:
        lista.append(i)
    return(lista)

g = divisivelPorN(7)
print(g.generate(50))
```

```
[0, 7, 14, 21, 28, 35, 42, 49]
```

2.11.1 Exercícios

1. Crie uma classe denominada `Investimento` com campos principal e juro. Deverá existir um método chamado `valorFuturo` que retornará o valor do investimento após n anos com juros compostos. Também deverá existir o método `__str__` para que a impressão do objeto retorne o valor de principal e juro do objeto.

2. Crie uma classe denominada `Product`. A classe deverá ter campos denominados `nome`, `montante` e `preco` os quais indicarão o nome do produto, o número de itens em estoque e o preço de lista do mesmo. Deverá existir o método `.procuraPreco` que receberá o número dos itens a serem adquiridos e retornará o custo de aquisição, quando forem adquiridos até 10 itens. Um desconto de 10

3. Escreva uma classe chamada `.GerenciaSenha`. A classe deve ter uma lista chamada `senhasAntigas` que conterá todas as senhas anteriores do usuário. O último item da lista é a senha atual do usuário. Deverá existir um método chamado `.obtemSenha` que retorne a senha atual e um método chamado `.defineSenha` que modificará a senha do usuário. O método

`.defineSenha` deverá alterar a senha apenas se a nova senha for diferente de todas as senhas anteriores do usuário. Por fim, deverá existir também um método chamado `.estaCorreto` que receberá uma string e retornará um booleano `True` ou `False`, dependendo se a string é igual à senha atual ou não.

4. Escreva uma classe chamada Tempo, cujo único campo é o tempo em segundos. Deve ter um método chamado `.converteParaMinutos` que recebe um número de segundos e retorna uma sequência de minutos e segundos no formato minutos:segundos. Também deverá existir um método chamado `.converteParaHoras` o qual receberá um número em segundos e retornará sequência de horas, minutos e segundos no formato horas:minutos:segundos.

5. Escreva uma classe chamada `.JogaPalavra`. Deverá ter um campo que contém uma lista de palavras. O usuário da classe deve passar a lista de palavras que deseja utilizar com a classe. Deverão existir os seguintes métodos:

   (a) `.palavrasEextensao` - retorna uma lista de todas as palavras com o número de caracteres especificado em extensão.

   (b) `.comecaCom(x)` - retorna uma lista de todas as palavras que começam com x.

   (c) `.terminaCom(x)` - retorna uma lista de todas as palavras que terminam com x.

   (d) `.Palindromos()` - retorna uma lista de todos os palindromos existentes na lista de instanciação da classe.

   (e) `.somente(L)` - retorna uma lista das palavras que contêm apenas as letras em L

   (f) `.sem(L)` - retorna uma lista das palavras que não contêm nenhuma letra em L

6. Crie uma classe para simular jogadas de um jogo de cartas com as seguintes regras: cada jogador recebe metade das cartas de um baralho escolhidas ao acaso. A cada jogada cada um dos jogadores apresenta a carta na parte mais alta de sua pilha de cartas. Aquele que tiver a carta maior recebe as duas cartas e as coloca na parte de baixo de sua pilha. Se houver empate, as duas cartas serão eliminadas do jogo. O jogo termina quando um jogador ficar sem cartas.

7. Escreva uma classe chamada `PedraPapelTesoura` que implemente a lógica do jogo de mesmo nome. Nele o usuário joga contra o computador por um certo número de rodadas. A classe deverá ter um campo para quantas rodadas o jogo terá, o número da rodada atual e o número de vitórias que cada jogador tem. Deverão existir métodos para se obter a escolha do computador e para encontrar o vencedor de uma rodada.

2.12 Recursividade

A recursividade torna possível uma função chamar a sí mesma. Este recurso proporciona soluções elegantes para uma vasta gama de problemas.

A seguir temos o cálculo da fatorial de um número resolvido através de recursividade. Para implementar a recursividade, a função retorna 1 caso n seja 1. Caso n seja maior que 1 a função deverá calcular e retornar a variavel resultado através da fórmula: `resultado = n * fatorial(n-1)`

```
def fatorial(n):
  if n == 1:
    return(1)
  else:
    resultado = n*fatorial(n-1)
    return(resultado)

fatorial(5)
```

```
120
```

2.12.1 Exercícios

Resolver os exercícios a seguir utilizando recursividade. Alguns dos exercícios requerem o uso de arrays `numpy`. Em todos eles pede-se desenvolver uma função para:

1. Determinar se um número m é primo.
2. Calcular X^Y
3. Calcular o termo de ordem n da série Fibonacci (1,1,2,3,5,8,13,21,34,...). Não tente calcular termos acima de ordem 20, pois o processo poderá ser muito demorado.
4. Ordenar os elementos de um array unidimensional.
5. Localizar um valor x em um array bidimensional, onde o valor x deverá ser procurado no intervalo de posições i, k
6. Localizar x no intervalo de endereços de k até n no array unidimensional v, retornando a posição do mesmo.
7. Inverter um array unidimensional.
8. Obter o menor valor no intervalo de endereços de k até l de um array unidimensional
9. Somar os k últimos elementos de um array unidimensional, retornando a soma dos mesmos.

10. Receber uma matriz quadrada $[n, n]$ e um número k. A função deve retornar a matriz original com os valores da parte com endereços maiores ou iguais a k transpostos.

11. Implementar a chamada função de Ackerman. Pede-se testar esta função com valores pequenos de m e n. Esta função é definida conforme a seguir: a) $n + 1$ se $m = 0$, b) $A(m, n) = A(n - 1, 1)$ se $m \neq 0$ e $n = 0$, c) $A(m - 1, A(m, n - 1)$ se $m \neq 0$ e $n \neq 0$

2.13 Exercícios em Geral

1. Escreva um programa que, a partir de uma sequência de números separados por vírgula, vindos do teclado como uma única string, gere uma lista. Suponha que a sequência de entrada seja `34,67,55,33,12,98`. Dica: Para transformar uma string em uma lista, separando os elementos por vírgula utilize `lista = variavel.split(sep=',')`.

2. Utilizando tanto o recurso de compreensão de lista quanto a função `map`, crie um programa que calcule e imprima a parte inteira do resultado da seguinte formula: $Q = \sqrt{\frac{2.C.D}{H}}$, onde C é 50 e H é 30. D é a variável cujo valor deverá ser passado para o seu programa através de uma sequência de números separados por vírgula. Para este exercício suponha que D = `100,150,180`. Neste caso o resultado será `18,22,24`.

```
# Resolução por compreensão de lista
def calculo(D):
  C = 50
  H = 30
  x = int((2*C*D/H)**(0.5))
  return(x)

def separa(D):
  Di = D.split(',')
  Di = [int(i) for i in Di]
  return(Di)

def junta(D):
  cont = str(D[0])
  for i in D[1:]:
    cont = cont + ',' + str(i)
  return(cont)

D = separa('100,150,180')
R = [calculo(i) for i in D]
J = junta(R)
print(J)
```

```
18,22,24
```

```
# Resolução por map
def calculo(D):
  C = 50
  H = 30
  x = int((2*C*D/H)**(0.5))
  return(x)

R0 = '100,150,180'
R1 = R0.split(',')
R2 = map(int,R1)
R3 = map(calculo,R2)
R4 = map(str,R3)
R5 = ','.join(R4)
print(R5)
```

```
18,22,24
```

3. Escreva um programa que receba como entrada uma sequência de palavras separadas por espaço e retorne como saida as palavras separadas por espaço, sem repetição e classificadas em ordem alfabética. Para este exercício suponha que a sequência de entrada seja: `hello world and practice makes perfect and hello world again`. Dica: o comando `set(lista)` retorna um conjunto com os valores únicos na lista. Para tornar o resultado de set uma outra lista faça `list(set(lista))`. Utilize o método `.split(',')` na string para separar a string pela vírgula, gerando uma lista. Utilize `.sort()` na lista para classificar seus elementos.

4. Utilizando tanto funções regulares (criadas com `def`) quanto funções `lambda` escreva um programa que receba uma sequência de quatro números binários, separados por vírgula e retorne uma sequência também separada por vírgula, apenas daqueles que forem divisíveis por 5. Suponha para este exercício que a sequência de entrada é: `0100,0011,1010,1001`. Dica: a conversão de uma string de números para números inteiros na base 2, pode ser feita com `int(string,2)`. Lembre-se também que `map(função, lista)` aplica a função em cada elemento de lista e retorna o resultado como um objeto do tipo `map`. Para tornar o resultado de map uma lista faça: `list(map(função, lista))`. Uma função `lambda` é uma pequena função implementada em uma única linha. Por exemplo: `f = lambda x : x**2` implementa uma função de nome `f` que recebe um número `x` e retorna o quadrado de `x`.

5. Utilizando as funções lambda, filter e map crie um programa que irá calcular o quadrado de cada número impar em uma sequência de números

separados por vírgula. Use a sequência de números de 0 a 9 como valores de entrada para este exercício.

```
ehimpar = lambda x : x%2!=0

impares = list(filter(ehimpar, range(10)))

quad = lambda x : x**2

quadrados = list(map(quad, impares))

print(quadrados)
```

```
[1, 9, 25, 49, 81]
```

6. Escreva um programa que receberá uma sequência de strings que representam Depósitos (D) ou Saques (W) e os respectivos valores de uma conta corrente. Supondo que o saldo inicial da conta seja 0, calcule o saldo final da conta. Para este exercício suponha que os valores de entrada sejam: `['D 100', 'W 200', 'D 300', 'D 300', 'W 200', 'D 100']`

7. Crie um programa para verificar a validade de um conjunto de senhas. O programa deverá aceitar uma sequência de senhas separadas por vírgula e imprimir como resposta apenas aquelas que atendem ao conjunto de critérios apresentados a seguir: a) pelo menos: uma letra minúscula, um número, uma letra maiúscula e um caracter entre `[$#]`, b) comprimento mínimo: 6 caracteres e c) comprimento máximo: 12 caracteres. Para teste, utilize como valores de entrada a string: 'ABd12341,a F1#,2w3E*,2We3345'. Neste conjunto, a única senha que atende a todos os critérios é `ABd12341`

Solução: o programa é apresentado a seguir. Para implementa-lo foi utilizado o pacote `re` que disponibiliza o uso de `regex` (regular expressions) em Python.

```
def testasenha(x):
  import re
  criterio1 = not not re.search('[a-z]',x)
  criterio2 = not not re.search('[0-9]',x)
  criterio3 = not not re.search('[A-Z]',x)
  criterio4 = not not re.search("[$#@]",x)

  n = len(x)
  criterio5 = n ≥ 6
  criterio6 = n ≤ 12

  validade = criterio1 and criterio2 and criterio3
  validade = validade and criterio4 and criterio5
  validade = validade and criterio6
```

```
  return(validade)

senhas = 'ABd1234@1,a F1#,2w3E*,2We3345'
senhas = senhas.split(',')
resultado = filter(testasenha, senhas)
print(list(resultado))
```

```
['ABd1234@1']
```

8. Utilizando a) apenas loops, b) a função `sorted(lista)` e c) a função `sorted(lista)` e funções `lambda`, crie um programa para classificar as tuplas formadas pelas variáveis (name, age, score) em ordem ascendente, onde name é uma string, age e score são números. Utilize como valores exemplos, as strings: a) `Tom,19,80`, b) `John,20,90`, c) `Jony,17,91`, d) `Jony,17,93` e e) `Json,21,85`.

Solução apenas com loops

```
# Lista original
lista = ['Tom,19,80', 'John,20,90', 'Jony,17,91']
lista = lista + ['Jony,17,93', 'Json,21,85']
for valor in lista:
  print(valor)
```

```
Tom,19,80
John,20,90
Jony,17,91
Jony,17,93
Json,21,85
```

```
# Maior primeiro elemento
def maiorL1(lista):
  L = 0
  for valor in lista:
    x = valor.split(',')
    n = len(x[0])
    if n>L:
      L=n
  return(L)

L = maiorL1(lista); L
```

```
4
```

```
# Nova lista
def insere1C(lista, C=" ", L=0):
  for index, valor in enumerate(lista):
    x = valor.split(',')
    n = len(x[0])
    if L>n:
      x[0] = x[0] + (L-n)*C
    novo_valor = ','.join(x)
    lista.pop(index)
    lista.append(novo_valor)
  return(lista)

lista2 = insere1C(lista, " ", L=4);
for valor in lista2:
  print(valor)
```

```
John,20,90
Jony,17,93
Tom ,19,80
Json,21,85
Jony,17,91
```

```
# Lista Ordenada
lista2.sort()
for valor in lista2:
  print(valor)
```

```
John,20,90
Jony,17,91
Jony,17,93
Json,21,85
Tom ,19,80
```

Solução com `sorted(lista)`

```
#Lista Original
lista1 = ['Tom,19,80', 'John,20,90', 'Jony,17,93']
lista1 = lista1 + ['Jony,17,91', 'Json,21,85']

for valor in lista1:
  print(valor)
```

```
Tom,19,80
John,20,90
Jony,17,93
```

```
Jony,17,91
Json,21,85
```

```
# Lista com elementos divididos
lista2 = [valor.split(',') for valor in lista1]
for valor in lista2:
  print(valor)
```

```
['Tom', '19', '80']
['John', '20', '90']
['Jony', '17', '93']
['Jony', '17', '91']
['Json', '21', '85']
```

```
# Lista classificada
def gera_chaves(valor):
  chave1 = valor[0]
  chave2 = valor[1]
  chave3 = valor[2]
  return(chave1, chave2, chave3)

lista3 = sorted(lista2, key = gera_chaves)
for valor in lista3:
  print(valor)
```

```
['John', '20', '90']
['Jony', '17', '91']
['Jony', '17', '93']
['Json', '21', '85']
['Tom', '19', '80']
```

Solução com `sorted(lista)` e funções `lambda`

```
#Lista Original
lista1 = ['Tom,19,80', 'John,20,90', 'Jony,17,93']
lista1 = lista1 + ['Jony,17,91', 'Json,21,85']
for valor in lista1:
  print(valor)
```

```
Tom,19,80
John,20,90
Jony,17,93
Jony,17,91
Json,21,85
```

```
# Lista classificada
lista2 = sorted(lista1, key = lambda x:(x[0],x[1],x[2]) )
for valor in lista2:
  print(valor)
```

```
John,20,90
Jony,17,93
Jony,17,91
Json,21,85
Tom,19,80
```

9. Um robô se movimenta em um plano começando pelo ponto $(0, 0)$ e seguindo pelas direções ACIMA, ABAIXO, ESQUERDA e DIREITA, na quantidade de passos indicados por um número ao lado da direção. Crie um programa que recebe uma sequência arbitrária de movimentos e calcula a distância final entre o ponto (0,0) e a posição de parada do robô. Caso o resultado seja um número com vírgula, apresente o inteiro mais próximo. Para este exemplo, considere a sequência:`['ACIMA 5', 'ABAIXO 3', 'ESQUERDA 3', 'DIREITA 2']`

10. Escreva um programa para calcular a frequência das palavras em um texto. Os resultados devem ser apresentados em ordem alfabética. Utilize como entrada a string: `Iniciante em Python ou escolhendo entre o Python 2 e o Python 3? Leia livros sobre Python 2 e Python 3 em ambos os casos.`. Dica: Para contar o número de vezes que um caracter aparece em uma lista de caracteres utilize o método `lista.count(caracter)`

```
texto = "Iniciante em Python ou escolhendo entre o Python 2 "
texto = texto + "e o Python 3? Leia livros sobre Python 2 e "
texto = texto + "Python 3 em ambos os casos."

lista = list(texto)
caracteres = sorted(list(set(texto)))
dicionario = {}

a = [dicionario.update({i:lista.count(i)}) for i in caracteres]
i = 0

for entrada in dicionario.items():
  i = i + 1
  if i//5 == i/5:
    print(entrada, "\n")
  else:
    print(entrada, end=" ")
```

```
(' ', 24) ('.', 1) ('2', 2) ('3', 2) ('?', 1)

('I', 1) ('L', 1) ('P', 5) ('a', 4) ('b', 2)

('c', 3) ('d', 1) ('e', 11) ('h', 6) ('i', 4)

('l', 2) ('m', 3) ('n', 9) ('o', 15) ('r', 3)

('s', 7) ('t', 7) ('u', 1) ('v', 1) ('y', 5)
```

11. Dada a lista `[12,24,35,24,88,120,155,88,120,155]`, escreva um programa que remova todos os valores duplicados e imprima a lista resultante, com os valores na ordem original. Dica: para inverter a ordem dos elementos em uma lista utilize a função `reversed(lista)`

```
x = [12,24,35,24,88,120,155,88,120,155,88,-10]

def limpadup(lista):
  lista2 = list(reversed(lista))

  for valor in lista2:
    cont_val = lista2.count(valor)
    if cont_val > 1:
      lista2.remove(valor)

  lista3 = list(reversed(lista2))
  return(lista3)

print(limpadup(x))
```

```
[12, 24, 35, 88, 120, 155, -10]
```

12. Dado um inteiro N, imprima um alfabeto Rangoli de tamanho N. Os alfabetos Rangoli são uma forma de arte indiana, baseada na criação de parâmetros. Dois exemplos de alfabetos Rangoli são apresentados a seguir:

De tamanho 3

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 =  9

def linha(n):
  l = 4*n-3
  x =     ["a","b","c","d","e","f","g","h","i","j","k","l","m"]
  x = x + ["n","o","p","q","r","s","t","u","v","y","x","w","z"]
```

```
  lista2 = x[:n][::-1]

  def desenha(i):
    lista = lista2[:i]
    if i > 1:
      lista = lista + list(reversed(lista2[:i-1]))
    texto1 = '-'.join(lista)
    n_ = int((l-len(texto1))/2)
    texto2 = '-'*n_
    print(texto2+texto1+texto2)

  for i in range(1,n+1):
    desenha(i)

  for i in range(n-1,0,-1):
    desenha(i)

linha(3)
```

```
----c----
--c-b-c--
c-b-a-b-c
--c-b-c--
----c----
```

De tamanho 5

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 =  9

def linha(n):
  l = 4*n-3
  x =     ["a","b","c","d","e","f","g","h","i","j","k","l","m"]
  x = x + ["n","o","p","q","r","s","t","u","v","y","x","w","z"]
  lista2 = x[:n][::-1]

  def desenha(i):
    lista = lista2[:i]
    if i > 1:
      lista = lista + list(reversed(lista2[:i-1]))
    texto1 = '-'.join(lista)
    n_ = int((l-len(texto1))/2)
    texto2 = '-'*n_
    print(texto2+texto1+texto2)
```

```
  for i in range(1,n+1):
    desenha(i)

  for i in range(n-1,0,-1):
    desenha(i)

linha(5)
```

```
--------e--------
------e-d-e------
----e-d-c-d-e----
--e-d-c-b-c-d-e--
e-d-c-b-a-b-c-d-e
--e-d-c-b-c-d-e--
----e-d-c-d-e----
------e-d-e------
--------e--------
```

```
# n + (n-1)*3 = 4n-3
# 5 + (5-1)*3 = 17
# 3 + (3-1)*3 =  9

def linha(n):
  l = 4*n-3
  x =     ["a","b","c","d","e","f","g","h","i","j","k","l","m"]
  x = x + ["n","o","p","q","r","s","t","u","v","y","x","w","z"]
  lista2 = x[:n][::-1]

  def desenha(i):
    lista = lista2[:i]
    if i > 1:
      lista = lista + list(reversed(lista2[:i-1]))
    texto1 = '-'.join(lista)
    n_ = int((l-len(texto1))/2)
    texto2 = '-'*n_
    print(texto2+texto1+texto2)

  for i in range(1,n+1):
    desenha(i)

  for i in range(n-1,0,-1):
    desenha(i)

linha(7)
```

```
------------g------------
----------g-f-g----------
--------g-f-e-f-g--------
------g-f-e-d-e-f-g------
----g-f-e-d-c-d-e-f-g----
--g-f-e-d-c-b-c-d-e-f-g--
g-f-e-d-c-b-a-b-c-d-e-f-g
--g-f-e-d-c-b-c-d-e-f-g--
----g-f-e-d-c-d-e-f-g----
------g-f-e-d-e-f-g------
--------g-f-e-f-g--------
----------g-f-g----------
------------g------------
```

13. Utilizando apenas o gerador de números aleatórios da `numpy`, `np.randint`, escreva um programa para gerar um número aleatório entre 10 e 150 (inclusive), o qual deve ser divisível por 5 e 7.

```
import numpy as np

lista = []
for i in range(10,150+1,1):
  if ((i%5==0) and (i%7==0)):
    lista.append(i)

n = len(lista)
index = np.random.randint(n)
print(lista[index])
```

```
35
```

14. Utilizando compreensão de lista e o uso da função `np.random.choice` da `numpy` escreva um programa para gerar um número aleatório entre 10 e 150 (inclusive), o qual deve ser divisível por 5 e 7.

```
import numpy as np

def testadiv(x):
  result = ((x%5==0) and (x%7==0))
  return(result)

lista = [i for i in range(10,150+1) if testadiv(i)]
print('A lista de elementos eh = ',lista)
```

```
A lista de elementos eh =  [35, 70, 105, 140]
```

```
sorteio = np.random.choice(lista)
print('A escolha aleatoria foi = ', sorteio)
```

```
A escolha aleatoria foi =  105
```

15. Utilizando compreensão de lista e o uso da função `np.random.choice` da `numpy` escreva um programa, em uma única linha, para gerar um número aleatório entre 10 e 150 (inclusive), o qual deve ser divisível por 5 e 7.

```
print(np.random.choice([i for i in range(10,150+1) if i % 35 == 0]))
```

```
105
```

16. Um número é chamado número perfeito se for igual à soma de todos os seus divisores, não incluindo o número em si. Por exemplo, 6 é um número perfeito porque os divisores de 6 são 1, 2, 3, 6 e $6 = 1 + 2 + 3$. Como outro exemplo, 28 é um número perfeito porque seus divisores são 1, 2, 4, 7, 14, 28 e $28 = 1 + 2 + 4 + 7 + 14$. No entanto, 15 não é um número perfeito porque seus divisores são 1, 3, 5, 15 e 15 $6 = 1 + 3 + 5$. Escreva um programa que encontre todos os quatro números perfeitos que são inferiores a 10000.

17. Um número palidrômico é aquele que é o mesmo se lido da esquerda para a direita e da direita para a esquerda. Se somarmos um número ao seu reverso as vezes é possível gerar um palíndromo. Por exemplo 241 + 142 = 383. O processo pode ser repetido até atingirmos o resultado desejado, por exemplo 84 + 48 = 132. 132 + 231 = 363. Escreva um programa que irá gerar números palindrômicos pela aplicação do processo aqui descrito em até vinte repetições do mesmo.

18. Escreva um programa para determinar com quantos zeros termina a fatorial de 1000.

19. Implemente uma busca de palavra em um dicionário através do método de pesquisa em árvore. Neste método você recebe uma palavra a ser procurada em uma lista e uma lista de palavras em ordem alfabética. Comece determinando se a palavra procurada é a palavra do meio da lista. Se for, retorne ok. Caso não seja, determine se a palavra procurada está à esquerda ou à direita da palavra do meio da lista. Suponha que ela está na metade esquerda. Em seguida procure a palavra na metade para parte em que ela estará e compare com a palavra que você tem. Caso seja a palavra procura retorne ok. Caso contrário repita o processo até ficar com duas palavras em sequência na lista original. Caso nenhuma delas seja a palavra que você está procurando retorne não encontrado.

20. Escreva uma função que recebe um número inteiro até a casa dos trilhões e retorna o número por extenso (1243 deve retornar um mil, duzentos e quarenta e três)

21. Crie uma função que recebe uma lista de números e um número. Em seguida ela deverá determinar o número na lista que, sendo menor que o número passado como argumento, está o mais próximo dele.

Capítulo 3

Álgebra Linear

A Álgebra Linear é a parte da matemática que trata das matrizes e dos vetores. Recentemente sua importância no ensino e estudo de matemática foi ampliada, uma vez que os computadores utilizam preferencialmente estruturas matriciais para realizar cálculos de alta complexidade. No caso da linguagem Python o pacote `numpy` proporciona uma ampla gama de recursos, tornando a mesma uma excelente ferramenta de ensino. A seguir é apresentada a teoria e exercícios de Álgebra Linear utilizando-se amplamente recursos da `numpy`.

3.1 Matrizes

Uma matriz é uma tabela de números. Para nossos propósitos uma matriz é um array `numpy` com apenas dois eixos. Cada eixo pode conter n dimensões. Por exemplo a matriz a seguir é um objeto de 2 eixos, sendo o primeiro eixo (das linhas) composto de 2 dimensões e o segundo eixo (das colunas) composto de 3 dimensões. Por motivos que serão explicados ao longo do texto, deve-se evitar o uso do termo "Dimensão" no lugar de "Eixo."

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \end{bmatrix}$$

A seguir vemos como criar a matriz acima como um array `numpy`:

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A
```

```
array([[1, 2, 3],
       [2, 1, 4]])
```

3.2 Matriz Transposta

A transposta de uma matriz é obtida trocando-se os valores das suas linhas pelos de suas colunas e é indicada como A^T. Sendo assim a transposta de uma matriz $[3, 2]$ será uma matriz $[2, 3]$. No python ela pode ser calculada pelo método `.T`.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.T
```

```
array([[1, 2],
       [2, 1],
       [3, 4]])
```

3.3 Soma Matricial

Para que duas matrizes A e B possam ser somadas elas devem ter o mesmo tamanho. A soma será efetuada somando-se os elementos de mesma posição em cada matriz. No código a seguir somamos a matriz A consigo mesma.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A + A
```

```
array([[2, 4, 6],
       [4, 2, 8]])
```

3.4 Produto Matricial

O produto matricial $A.B$ é calculado multiplicando-se o valor de cada linha de A pelo valor de cada coluna em B. Sendo assim, para que possa ser calculado, a primeira matriz deverá ter um número de colunas igual ao de linhas da segunda matriz. Além disso, a matriz resultante terá o número de linhas da primeira matriz e o número de colunas da segunda. A seguir é calculada a matriz $A^T.A$ a quals será uma matriz $[3, 3]$, pois A^T é uma matriz $[3, 2]$ e A uma matriz $[2, 3]$. Na `numpy` o produto de duas matrizes é calculado através do método `.dot`

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.T.dot(A)
```

```
array([[ 5,  4, 11],
       [ 4,  5, 10],
       [11, 10, 25]])
```

Uma observação importante. O produto matricial não é comutativo. Isto pode ser facilmente percebido calculando-se os produtos matriciais $A^T.A$ e $A.A^T$ os quais produzem resultados distintos.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4]])
A.dot(A.T)
```

```
array([[14, 16],
       [16, 21]])
```

3.5 Matriz Identidade

Na multiplicação temos o número 1 como o elemento neutro. Isto quer dizer que se multiplicamos qualquer número pelo número 1, obteremos como resultado o próprio número original. Na álgebra linear existe o conceito de Matriz Identidade. A matriz identidade é tal que $A.I = I.A = A$.

Para matrizes $[3, 3]$ a matriz identidade tem a forma:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para gerar uma matriz identidade utiliza-se a função `np.eye`, indicando como parâmetro o tamanho desejado. Abaixo geramos uma matriz identidade $[3, 3]$

```
import numpy as np
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Abaixo mostramos que multiplicar uma matriz qualquer pela matriz identidade produz uma matriz igual a matriz original.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4],[3,4,1]])
I3 = np.eye(3)
A.dot(I3)
```

```
array([[1., 2., 3.],
       [2., 1., 4.],
       [3., 4., 1.]])
```

3.6 Matriz Simétrica

Uma matriz é considerada simétrica quando ela for igual a sua transposta. Prove que a matriz K_4 (abaixo) é uma matriz simétrica.

$$K_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Para confirmar que as matrizes são iguais vamos criar uma matriz de booleanos (através de `==`) e em seguida verificar se todos os elementos de mesma posição são iguais (através do método `.all`)

```
import numpy as np
K4 = np.array([[2,-1,0,0],[-1,2,-1,0],[0,-1,2,-1],[0,0,-1,2]])
(K4 == K4.T).all()
```

```
True
```

3.7 Inversão de Matrizes

Na multiplicação de números, se multiplicamos um número (3 por exemplo) pelo seu inverso ($\frac{1}{3}$, também representado por 3^{-1} obtemos como resultado o elemento neutro, o número 1. Na álgebra de matrizes temos um conceito similar, para matrizes quadradas. Dizemos que uma matriz A^{-1} é a matriz inversa da matriz A se $A^{-1}.A = A.A^{-1} = I$, ou seja: se multiplicamos uma matriz pela sua inversa obteremos como resultado a matriz identidade. Na `numpy` a inversão de matrizes é obtida com a função `np.linalg.inv`.

```
import numpy as np
A = np.array([[1,2,3],[2,1,4],[3,4,1]])
A
```

```
array([[1, 2, 3],
       [2, 1, 4],
       [3, 4, 1]])
```

```
Ainv = np.linalg.inv(A)
Ainv
```

```
array([[-0.75,  0.5 ,  0.25],
       [ 0.5 , -0.4 ,  0.1 ],
       [ 0.25,  0.1 , -0.15]])
```

Confirmamos que a matriz `Ainv` é a inversa da matriz `A` calculando o produto das duas e observando que o mesmo é igual a matriz identidade $[2, 2]$ (dentro de uma margem de erro numérico apropriada)

```
np.round(A.dot(Ainv),2)
```

```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0., -0.,  1.]])
```

3.8 Resolução de sistemas lineares

Dado um sistema linear determinado, $A.\vec{x} = \vec{B}$ pode-se obter o vetor $\vec{x}$ das soluções x_1, x_2, x_3, $...x_n$ para este sistema executando-se a operação $\vec{x} = A^{-1}.B$ conforme pode ser visto na listagem a seguir. Como exemplo, vamos resolver o sistema de equações 3.1:

$$\begin{array}{rrrcl} x & +2y & -z & = & 2 \\ 2x & -y & +3z & = & 9 \\ 3x & +3y & -2z & = & 3 \end{array} \tag{3.1}$$

através da `numpy`

```
import numpy as np
A = np.array([[1, 2, -1],[2, -1, 3],[3, 3, -2]])
A
```

```
array([[ 1,  2, -1],
       [ 2, -1,  3],
       [ 3,  3, -2]])
```

```
B = np.array([[2, 9, 3]]).T
B
```

```
array([[2],
       [9],
       [3]])
```

```
x = np.linalg.inv(A).dot(B)
x
```

```
array([[1.],
       [2.],
       [3.]])
```

Confirmamos que a solução encontrada está correta mostrando que $A.\vec{x} = B$

```
A.dot(x)
```

```
array([[2.],
       [9.],
       [3.]])
```

Para resolver sistemas lineares indeterminados a maneira mais prática é utilizar a `sympy`. Podemos com este pacote resolver sistemas indeterminados homogêneos e não-homogêneos. Por exemplo, o sistema de equações 3.2, o qual contém mais variáveis que equações:

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= -10 \\ x_1 - 3x_2 + 8x_3 &= +85 \end{aligned} \tag{3.2}$$

Para tal procedemos conforme a seguir:

```
import sympy as sp
x1, x2, x3 = sp.symbols("x1 x2 x3")
Eq1 = 2*x1 + 3*x2 -5*x3 + 10
Eq2 =   x1 - 3*x2 +8*x3 - 85
sp.solve([Eq1, Eq2],[x1, x2])
```

```
{x1: 25 - x3, x2: 7*x3/3 - 20}
```

3.9 Número complexos

A vantangem principal das funções da `numpy` é que elas permitem a inserção e o cálculo de matrizes com números complexos. Tal recurso é essencial quando estamos trabalhando com exercícios de Computação Quântica. O uso de números complexos na linguagem Python (e principalmente na `numpy`) é feito de forma transparente, sendo os números complexos tratados como um real qualquer.

Por exemplo o número imaginário $j = \sqrt{-1}$ é representado em Python com `1j`. Um real com representação nos números complexos é criado somando-se ao número propriamente dito o "valor" `0j`. Este valor será do tipo `complex`.

```
a = -1 + 0j
a, type(a)
```

```
((-1+0j), <class 'complex'>)
```

Abaixo calculamos $\sqrt{-1} = j$, além de j^3 e j^{15}

```
import numpy as np
np.sqrt(-1+0j), (1j)**3, (1j)**15
```

```
(1j, (-0-1j), (-0-1j))
```

As partes real, imaginária, o conjugado, módulo e ângulo de um número complexo são obtidos através dos métodos `.real`, `.imag`, `.conjugate` e das funções `np.abs` e `np.angle`. A seguir calcula-se tais valores para o número $c = 3 + 4j$

```
import numpy as np
c = 3+4j
c, c.real, c.imag, c.conjugate(), np.abs(c), np.angle(c)
```

```
((3+4j), 3.0, 4.0, (3-4j), 5.0, 0.9272952180016122)
```

3.10 Matriz Hermitiana

A matriz hermitiana é obtida transpondo os elementos da matriz original e transformando os valores em seus complexos conjugados (isto é invertendo o sinal da parte imaginária). No código abaixo criamos uma matriz A com valores complexos e multiplicamos ela pela sua matriz hermitiana correspondente.

```
import numpy as np
A = np.array([[1+1j, 1-2j],[2+1j, 2-1j]])
A
```

```
array([[1.+1.j, 1.-2.j],
       [2.+1.j, 2.-1.j]])
```

```
Ah = A.conj().T
Ah
```

```
array([[1.-1.j, 2.-1.j],
       [1.+2.j, 2.+1.j]])
```

```
A.dot(Ah)
```

```
array([[ 7.+0.j,  7.-2.j],
       [ 7.+2.j, 10.+0.j]])
```

3.11 Raízes de Um Complexo

Por exemplo, as raízes cúbicas de $c = (1+j)$ podem ser expressas como a solução da equação $x^3 = 1 + j$. Em outras palavras podemos pensar que queremos encontrar as raízes do polinômio $x^3 + 0.x^2 + 0.x - (1 + j)$. Na numpy, as raízes de um polinômio podem ser calculadas a partir dos seus coeficientes pela função `np.roots`.

```
import numpy as np
# Criamos uma lista com os coeficientes do polinômio
coef = [1,0,0,-(1+1j)]

# Passamos esta lista em seguida para np.roots
np.round(np.roots(coef),3)
```

```
array([-0.794+0.794j,  -0.291-1.084j,   1.084+0.291j])
```

O Python também possui um módulo de matemática simbólica (denominado `sympy`). Quando estamos interessados em analisar um problema do ponto de vista algébrico este módulo proporciona recursos interessantes. No exemplo a seguir vamos provar que, dados dois números complexos c_1 e c_2, temos a seguinte relação para seus módulos: $|c_1||c_2| \geq |c_1.c_2|$

```
import sympy as sp
a, b, c, d = sp.symbols("a b c d", real=True)
c1 = a + b*1j
c2 = c + d*1j
sp.Abs(c1)*sp.Abs(c2)
```

```
sqrt(a**2 + 1.0*b**2)*sqrt(c**2 + 1.0*d**2)
```

A expressão acima é claramente igual a expressão abaixo.

```
sp.Abs(c1*c2)
```

```
sqrt(a**2 + 1.0*b**2)*sqrt(c**2 + 1.0*d**2)
```

3.12 Matrizes Unitárias

Uma matriz quadrada composta de números complexos é unitária se sua transposta conjugada U^* é também sua inversa. A seguir vamos provar que a matriz

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{j}{\sqrt{2}} & \frac{j}{\sqrt{2}} & 0 \\ 0 & 0 & j \end{bmatrix}$$

é unitária.

```
import numpy as np
U = np.array([[1/np.sqrt(2), 1/np.sqrt(2),0],
              [-1j/np.sqrt(2),1j/np.sqrt(2),0],
              [0,0,1j]])
Ud = U.conj().T
np.round(Ud,3)
```

```
array([[ 0.707-0.j   , -0.   +0.707j,  0.   -0.j   ],
       [ 0.707-0.j   ,  0.   -0.707j,  0.   -0.j   ],
       [ 0.   -0.j   ,  0.   -0.j   ,  0.   -1.j   ]])
```

3.13 Produto Hadamard

O produto Hadarmd de duas matrizes é calculado fazendo-se a multiplicação elemento a elemento. Pela `numpy` esta operação é feita pelo operador `*` A seguir calculamos o produto Hadamard da matriz

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ -\frac{j}{\sqrt{2}} & \frac{j}{\sqrt{2}} & 0 \\ 0 & 0 & j \end{bmatrix}$$

consigo mesma.

```
import numpy as np
U = np.array([[1/np.sqrt(2), 1/np.sqrt(2),0],
              [-1j/np.sqrt(2),1j/np.sqrt(2),0],
              [0,0,1j]])
Uhad = U*U
np.round(Uhad,3)
```

```
array([[ 0.5+0.j,  0.5+0.j,  0. +0.j],
       [-0.5+0.j, -0.5+0.j,  0. +0.j],
       [ 0. +0.j,  0. +0.j, -1. +0.j]])
```

3.14 Produto Tensorial

O produto tensorial na `numpy` é calculado pela função `np.kron`. A seguir calculamos o produto tensorial das matrizes

$$A = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \text{ e } C = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \tag{3.3}$$

```
import numpy as np
a = np.array([[1],[2]])
c = np.array([[3],[4]])
np.kron(a,c)
```

```
array([[3],
       [4],
       [6],
       [8]])
```

E a seguir calculamos o produto tensorial das matrizes

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \text{ e } Y = \begin{bmatrix} 0 & -j \\ j & 0 \end{bmatrix} \tag{3.4}$$

```
import numpy as np
X = np.array([[0,1],[1,0]])
Y = np.array([[0,-1j],[1j,0]])
np.kron(X,Y)
```

```
array([[0.+0.j, 0.-0.j, 0.+0.j, 0.-1.j],
       [0.+0.j, 0.+0.j, 0.+1.j, 0.+0.j],
       [0.+0.j, 0.-1.j, 0.+0.j, 0.-0.j],
       [0.+1.j, 0.+0.j, 0.+0.j, 0.+0.j]])
```

3.15 Autovalores e Autovetores

Podemos dividir a Álgebra Linear em duas grandes áreas : os problemas de sistemas de equações e os problemas de autovalores e autovetores. A idéia básica de um autovalor e um autovetor pode parecer abstrata a primeira vista, no entanto poucas idéias tiveram uma quantidade de aplicações na Física, Engenharia, Estatística e na própria matemática como o conceito de autovalor e autovetor.

Na sua concepção a idéia de um autovalor e de um autovetor provém da solução do seguinte problema : sabendo que o produto de uma matriz quadrada $\mathbf{A}$ por um vetor $\vec{\mathbf{X}}$ resulta em um outro vetor $\vec{\mathbf{Z}}$, encontrar o vetor $\vec{\mathbf{X}}$ tal que o produto $\mathbf{A}.\vec{\mathbf{X}}$ seja paralelo ao próprio $\vec{\mathbf{X}}$, o que em outras palavras significa : $\mathbf{A}.\vec{\mathbf{X}} = \lambda.\vec{\mathbf{X}}$. Ao vetor que satisfizer tal relação, da-se o nome de *autovetor* da matriz $\mathbf{A}$ e o fator de proporcionalidade λ chamamos de *autovalor* da matriz $\mathbf{A}$, associado ao autovetor $\vec{\mathbf{X}}$

3.15.1 Representação Geométrica

Antes de resolver o problema e apresentar a metodologia de cálculo vamos apresentar a representação geométrica de um autovetor. Para tanto devemos lembrar que a multiplicação de uma matriz $\mathbf{A}$ por um vetor $\vec{\mathbf{X}}$ produz outro vetor $\vec{\mathbf{Y}}$. Podemos portanto representar esta operação em uma tabela por uma multiplicação matricial. Tornando ambos os vetores (de entrada e saida) uma função do ângulo θ formado pelas componentes do vetor de entrada podemos desenhar a circularização do vetor de entrada e do vetor de saida. Ao passo que o vetor de entrada irá descrever uma circunferência, o vetor de saida descreverá uma elipse.

```
import numpy as np
A = np.array([[1., 1.],[0.,1.]]); A
```

```
array([[1., 1.],
       [0., 1.]])
```

```
xi = lambda theta: [np.cos(theta*np.pi/180), np.sin(theta*np.pi/180)]
xi(45)
```

```
[0.7071067811865476, 0.7071067811865475]
```

```
xo = lambda theta:list(A.dot(xi(theta)))
xo(45)
```

```
[1.414213562373095, 0.7071067811865475]
```

```
import pandas as pd
thetas = np.linspace(0,360,361)
xis = np.array(list(map(xi, thetas)))
xis[:3]
```

```
array([[1.        , 0.        ],
       [0.9998477 , 0.01745241],
       [0.99939083, 0.0348995 ]])
```

```
import pandas as pd
thetas = np.linspace(0,360,361)
xos = np.array(list(map(xo, thetas)))
xos[:3]
```

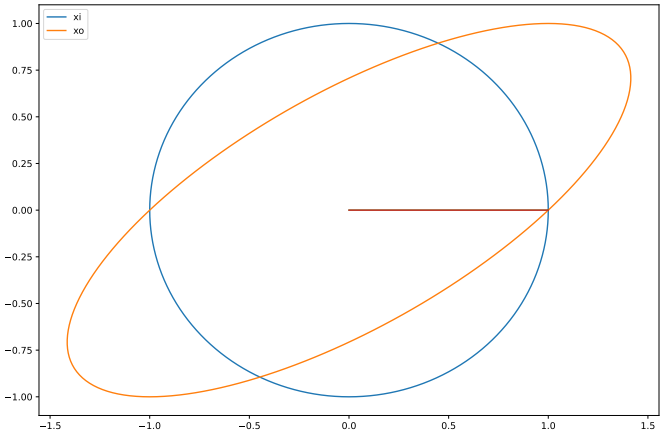
```
array([[1.        , 0.        ],
       [1.0173001 , 0.01745241],
       [1.03429032, 0.0348995 ]])
```

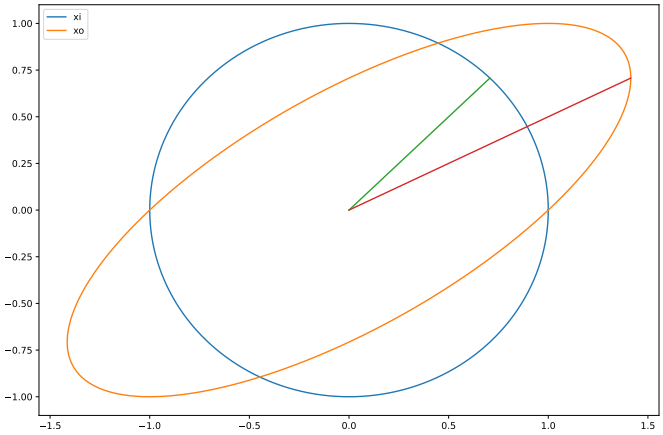
```
dados = pd.DataFrame({'theta': thetas})
dados['xi_x'] = xis[:,0]
dados['xi_y'] = xis[:,1]
dados['xo_x'] = xos[:,0]
dados['xo_y'] = xos[:,1]
dados.head()
```

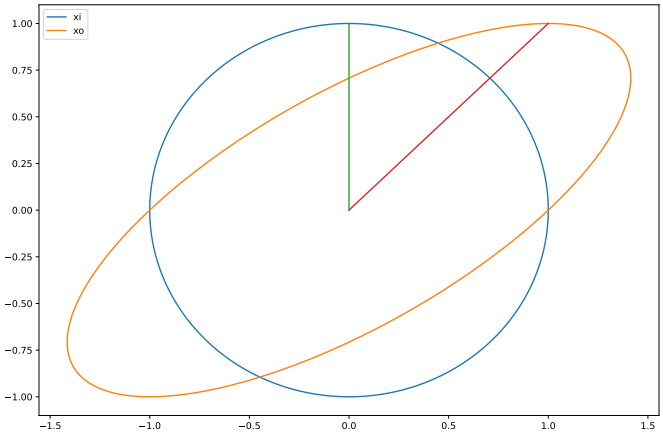
	theta	xi_x	xi_y	xo_x	xo_y
0	0.0	1.000000	0.000000	1.000000	0.000000
1	1.0	0.999848	0.017452	1.017300	0.017452
2	2.0	0.999391	0.034899	1.034290	0.034899
3	3.0	0.998630	0.052336	1.050965	0.052336
4	4.0	0.997564	0.069756	1.067321	0.069756

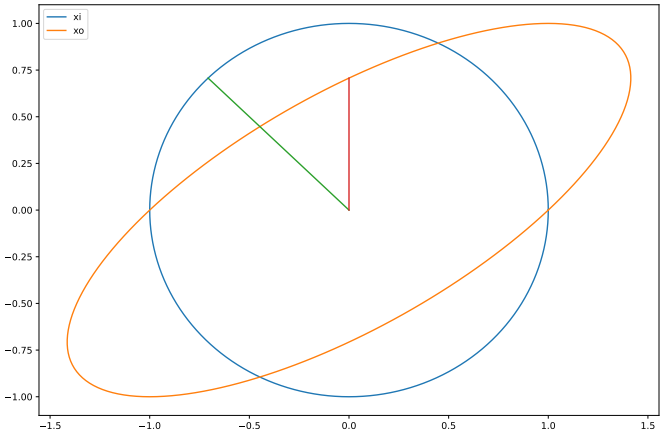
Observe agora o efeito de variar theta, na representação dos vetores de entrada e de saida. Veja que a medida que theta aumenta, primeiro o vetor de saida é paralelo, depois fica para trás, depois diminui o atraso e por fim volta a ficar paralelo ao vetor de entrada. Os valores de theta para os quais os vetores estão alinhadados, são os valores de x_i que são os autovetores da matriz A. A relação de tamanho entre os dois vetores $(\frac{x_o}{x_i})$ é o autovalor.

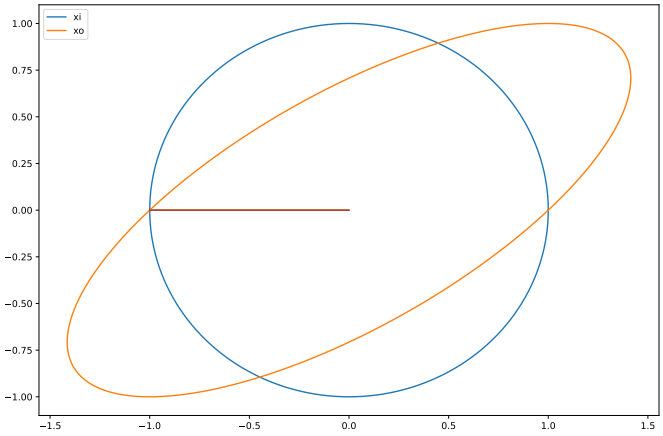
```
from matplotlib import pyplot as plt
for th in np.linspace(0,360,9):
  g = plt.plot(dados['xi_x'], dados['xi_y'], label = 'xi')
  g = plt.plot(dados['xo_x'], dados['xo_y'], label = 'xo')
  g = plt.plot([0, dados['xi_x'][th]], [0, dados['xi_y'][th]])
  g = plt.plot([0, dados['xo_x'][th]], [0, dados['xo_y'][th]])
  g = plt.legend();
  plt.show()
```

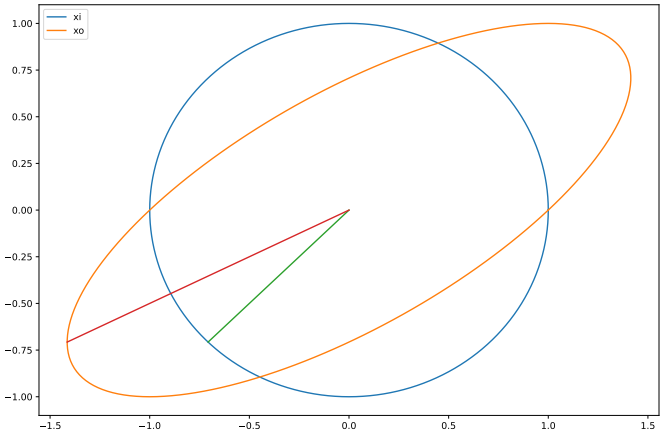

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

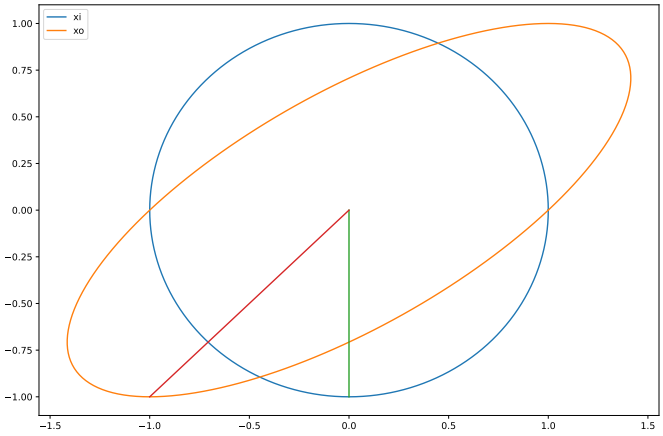

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

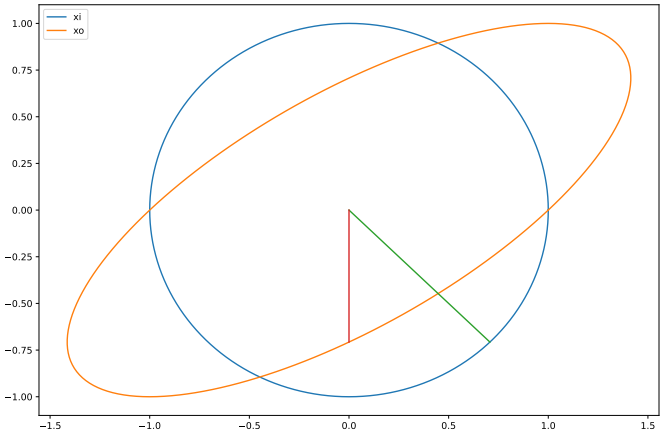

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

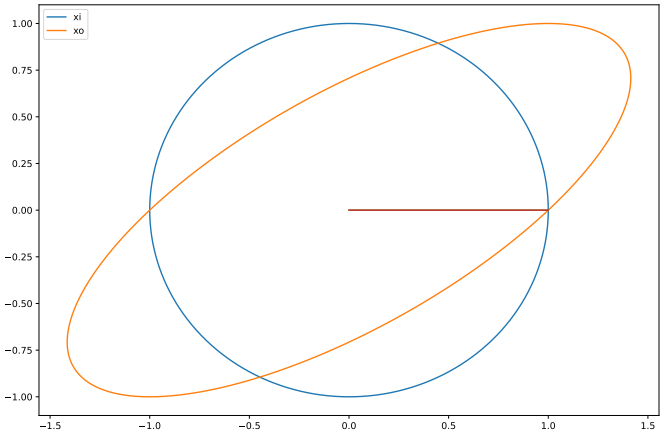

xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5


xi
xo
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5



3.15.2 Metodologia de Cálculo

A seção 3.15.1 descreveu o significado geométrico de um autovalor e um autovetor. Nesta etapa vamos apresentar a metodologia de cálculo. O problema é dividido em duas partes : primeiro são calculados os autovalores e a partir de cada autovalor é calculado o autovetor associado.

Do ponto de vista algébrico a resolução é obtida da seguinte forma :

$$\begin{aligned} \mathbf{A}.\vec{\mathbf{X}} &= \lambda.\vec{\mathbf{X}} \\ \mathbf{A}.\vec{\mathbf{X}} - \lambda.\vec{\mathbf{X}} &= \vec{0} \\ \mathbf{A}.\vec{\mathbf{X}} - \lambda.\mathbf{I}.\vec{\mathbf{X}} &= \vec{0} \\ [\mathbf{A} - \lambda.\mathbf{I}]\,.\vec{\mathbf{X}} &= \vec{0} \end{aligned} \tag{3.5}$$

Da equação 3.5 percebe-se que os autovalores provêm do cálculo do determinante da matriz

$$[\mathbf{A} - \lambda.\mathbf{I}] \tag{3.6}$$

A equação 3.6 será uma equação polinomial de grau igual a ordem da matriz quadrada $\mathbf{A}$ e terá portanto o número de raízes igual a ordem da mesma. Após o cálculo dos autovalores cada autovalor deve ser substituido em 3.5, o que irá gerar cada um, um sistema indeterminado. A solução (as soluções na verdade) de cada sistema irão compor cada autovetor associado da matriz $\mathbf{A}$. Esta técnica irá ficar mais clara no exemplo (resolvido de forma literal) apresentado na seção 3.15.3

3.15.3 Exemplo de Cálculo

Seja a matriz $\mathbf{A} = \begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix}$. Vamos calcular os autovalores e autovetores de $\mathbf{A}$ replicando a sequência de passos descritos na equação 3.5. Para tanto fazemos :

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{3.7}$$

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \lambda . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.8}$$

$$\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \lambda . \mathbf{I} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.9}$$

$$\left[\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \lambda . \mathbf{I} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.10}$$

$$\left[\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \lambda . \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.11}$$

$$\left[\begin{bmatrix} 1 & -5 \\ -5 & 1 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \right] . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \vec{0} \tag{3.12}$$

$$\begin{bmatrix} 1-\lambda & -5 \\ -5 & 1-\lambda \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.13}$$

Agora para calcular λ precisamos resolver a equação 3.14, ou seja, calcular λ de forma que o determinante seja 0.

$$\begin{vmatrix} 1-\lambda & -5 \\ -5 & 1-\lambda \end{vmatrix} = 0 \tag{3.14}$$

$$(1-\lambda).(1-\lambda) - 25 = 0 \tag{3.15}$$

$$(1-\lambda)^2 = 25 \tag{3.16}$$

$$1 - \lambda = \pm 5 \tag{3.17}$$

$$\lambda_1 = -4 \lambda_2 = 6 \tag{3.18}$$

Resumindo, para o cálculo dos autovalores de uma matriz quadrada **A** basta diminuir λ na diagonal principal e igualar a 0 o determinante da matriz resultante. Outro detalhe que precisa ser mencionado é que a equação 3.15 na maioria das vezes precisa ser expandida para formar um polinômio e então ser resolvida a equação polinomial resultante. Neste caso a expansão tomaria a seguinte forma:

$$\lambda^2 - 2\,\lambda - 24 = 0 \tag{3.19}$$

A equação 3.19 é denominada *equação característica* da matriz **A**. Agora devemos retornar a equação 3.11 e substituir cada valor de λ, resolvendo os sistemas resultantes e calculando os autovetores correspondentes. Neste caso para $\lambda_1 = -4$ teremos :

$$\begin{bmatrix} 1-(-4) & -5 \\ -5 & 1-(-4) \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.20}$$

$$\begin{bmatrix} 5 & -5 \\ -5 & 5 \end{bmatrix} . \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.21}$$

$$\begin{aligned} 5x_1 - 5x_2 &= 0 \\ -5x_1 + 5x_2 &= 0 \end{aligned} \tag{3.22}$$

O sistema representado nas equações 3.22 é indeterminado (o que era esperado uma vez que calculamos o valor de λ de forma que o determinante da matriz de coeficientes do sistema tivesse valor igual a 0). Desta forma iremos calcular uma relação entre x_1 e x_2 o que nos fornecerá a reta, isto é, a direção, na qual o autovetor esta localizado. Neste caso teremos :

$$\begin{aligned} 5x_1 - 5x_2 &= 0 \\ 5x_1 &= 5x_2 \\ x_1 &= x_2 \end{aligned} \tag{3.23}$$

Isto faz com que o autovetor tenha a forma mostrada na equação 3.24

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} .x_1 \tag{3.24}$$

Considerando que x_1 é arbitrário podemos representa-lo pela letra k. O autovetor fica então representado por

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} .k \tag{3.25}$$

Dentre todos os possíveis autovetores da matriz **A**, associados ao autovalor $\lambda_1 = -4$ é comum utilizar um autovetor unitário, isto é, com módulo igual a 1. Este processo é chamado de normalização do vetor. Sendo assim, normalizando o vetor representado na equação 3.25 teremos :

$$\sqrt{k^2 + k^2} = 1\sqrt{2}.k^2 = 1k = \frac{1}{\sqrt{2}} \tag{3.26}$$

Sendo assim o primeiro par de autovalor λ_1 e autovetor normalizado e_1 da matriz **A** será :

$$e_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ com } \lambda_1 = -4 \tag{3.27}$$

Repetindo a sequência de cálculos apresentada entre 3.14 e 3.27 para o autovalor $\lambda_2 = 6$ teremos o segundo par de autovalor e autovetor normalizado da matriz **A** o qual será :

$$e_2 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ com } \lambda_2 = 6 \tag{3.28}$$

3.16 Cálculo via numpy

O cálculo de autovalores e autovetores é feito pela função `numpy.linalg.eig`. Como era de se esperar, esta função retorna os autovalores tanto reais quanto complexos. Para simplificar, suponha a matriz A (abaixo), a qual por ser simétrica só contém autovalores reais.

```
import numpy as np
A = np.array([[1,2,3],[2,4,1],[3,1,5]]); A
```

```
array([[1, 2, 3],
       [2, 4, 1],
       [3, 1, 5]])
```

A função `numpy.linalg.eig` retorna tanto os autovalores quanto os autovetores associados. Os autovalores podem ser vistos a seguir.

```
vals, vecs = np.linalg.eig(A);
vals
```

```
array([-0.97344343,  7.58959802,  3.38384541])
```

E abaixo temos os autovetores dispostos nas colunas do array.

```
vecs
```

```
array([[-0.87716982,  0.47994164, -0.015137  ],
       [ 0.2733656 ,  0.47320619, -0.83746472],
       [ 0.39477127,  0.73873671,  0.54628172]])
```

Observe o efeito de multiplicar a matriz A por um dos autovetores e em seguida dividir cada componente pelo autovalor associado. O vetor resultante é igual ao vetor inicial.

```
A.dot(vecs[:,0])/vals[0]
```

```
array([-0.87716982,  0.2733656 ,  0.39477127])
```

3.17 Exercícios

1. Executar as seguintes operações com números complexos, fornecendo o resultado tanto na forma retangular quanto na forma polar:
   (a) $(2-4j)+(3+4j)$
   (b) $(2-4j).(3+5j).(3-2j).(-2-3j)^*$
   (c) $\left(\frac{3-2j}{3+2j}\right)'$
   (d) $\left(100\sqrt{2}\,(1-j)\right)^{1/4}$
   (e) $\frac{-8+3\,j}{-2+4\,j}$
2. Calcular o valor das funções de variável complexa $f(z)$ nos pontos $z = a+b\,j$ indicados
   (a) z^2+j em $z = 1+j$
   (b) $|z|^2+j$ em $z = 1+j$
   (c) $f^*(z)$ se $f(z) = z^2+2\,z\,z^*+j$ em $z = 1+2\,j$
   (d) $f(z) = 1-z+|z|^2$ em $z = 1+2\,j$
3. Desenhar o gráfico de $f(z) = z+\frac{1}{z}$ para z variando de $0+0\,j$ até $10+10\,j$
4. Calcule as operações matriciais apresentadas abaixo:
   (a) Seja $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 4 \end{bmatrix}$ e $\mathbf{B} = \begin{bmatrix} 2 & 3 & 0 \\ -1 & 2 & 5 \end{bmatrix}$, calcule $\mathbf{A}+\mathbf{B}$ e $\mathbf{A}-\mathbf{B}$
   (b) Seja $\mathbf{A} = \begin{bmatrix} 1 & -2 \\ 2 & 3 \end{bmatrix}$, $k_1 = 5$ e $k_2 = -3+2j$, calcule $k_1\,A$ e $k_2\,A$.

(c) Produzir uma Matriz Identidade 3x3

(d) Seja a matriz $\mathbf{A} = \begin{bmatrix} 1+2j & j \\ 3 & 2-3j \end{bmatrix}$, calcule $\mathbf{A}^{*T}$ isto é a transposta do conjugado de $\mathbf{A}$

5. A transformação $L(\overrightarrow{u}) = \begin{bmatrix} -5u_1 + 4u_2 + 5 \\ +3u_1 - 6u_2 \end{bmatrix}$ é linear? Prove sua afirmação testando ambas as condições de linearidade.

6. Verifique se a transformação definida por $L([u_1, u_2, u_3]) = [u_1 + u_2 - u_3, u_1 + 2u_3, u_1 - 2u_2 + 4]$ é ou não linear, testando AMBAS as condições de linearidade. Esta transformação possui matriz canônica? Se sim, apresente a mesma, se não explique porquê.

7. Verifique se uma transformação que tem como resultados os abaixo, poderá ou não ser definida como linear, explicando o porquê de sua resposta

   (a) L(u) = [9 ; 8], L[v] = [5 ; 4] e L[u+v] = [14 ; 12]

   (b) L(u) = [9 ; 8], L[3u] = [27 ; 34]

8. Calcule o valor de K para que o sistema abaixo seja impossível.

$$\begin{gathered} X + 3Y + 4Z = -1 \\ +Y + KZ = -3 \\ 2X + 2Z = -2 \end{gathered}$$

9. Encontre o maior valor de X que torna o determinante da matriz abaixo igual a 0

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ x & 1 & 2 & 3 \\ x^2 & 1 & 4 & 9 \\ x^3 & 1 & 8 & 27 \end{bmatrix}$$

10. Seja X1, X2, X3 as soluções do sistema representado pelas matrizes A e B abaixo, onde X1, X2 e X3 estão dispostos em ordem CRESCENTE. Qual o valor da expressão X1+X2-X3?

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 3 & 2 \end{bmatrix}, B = \begin{bmatrix} 3 \\ 8 \\ 11 \end{bmatrix}$$

11. Calcule os autovalores e os autovetores da matriz $A = \begin{bmatrix} 3 & -1 \\ -1 & 5 \end{bmatrix}$(utilize 2 casas depois da virgula como precisão numérica, caso necessário)

12. A matriz abaixo pode representar uma matriz de covariâncias? Justifique sua resposta.

$$\begin{bmatrix} 0,2 & 0,1 & 0,3 \\ 0,2 & 0,1 & 0,3 \\ 0,3 & 0,4 & 0,5 \end{bmatrix}$$

13. Encontre a equação que determina os autovalores, (também chamada e equação característica), os autovalores e os autovetores normalizados (isto é, com módulo igual a 1) da seguinte matriz:

$$\begin{bmatrix} -1 & 2 \\ 2 & 3 \end{bmatrix}$$

14. Considere o conjunto de autovalores e autovetores normalizados associados apresentados na tabela 3.1. Determine quantos fatores serão necessários para representar pelo menos 90% da matriz original e apresente esta matriz a partir da fórmula da análise espectral.

Tabela 3.1: Tabela de Autovalores e Autovetores

Autovalor	Autovetor_1	Autovetor_2	Autovetor_3	Autovetor_4
70.778687	-0.053753	0.857547	-0.405648	-0.311726
8.673544	-0.009352	0.361164	0.049184	0.931157
1.802103	-0.314252	0.328895	0.872815	-0.176826
0.265425	0.947771	0.161251	0.266878	-0.067122

15. Considere o conjunto de autovalores e autovetores normalizados associados disponíveis na tabela 3.2. Determine quantos fatores serão necessários para representar pelo menos 90% da matriz original e apresente esta matriz a partir da fórmula da análise espectral.

Tabela 3.2: Tabela de Autovalores e Autovetores

Autovalor	Autovetor_1	Autovetor_2	Autovetor_3	Autovetor_4	Autovetor_5
21.71	0.76	0.01	-0.18	-0.21	-0.59
9.62	0.44	-0.37	-0.13	-0.36	0.73
5.35	0.46	0.26	0.67	0.47	0.21
4.15	0.03	0.89	-0.23	-0.30	0.24
0.97	0.16	0.03	-0.66	0.72	0.15

16. Um laticínio vai mistura dois tipos de leite: um que tem 1% de gordura e outro que tem 6%. Quantos litros de cada tipo deverão ser misturados para que se obtenham 1 000 litros de leite com 3

17. Um combustível para automóveis tem 10% de álcool e o restante de gasolina. Outro combustível tem 4% de álcool e o restante de gasolina. Quanto devemos juntar de cada um desses combustíveis para obter 90 litros de combustível que tenha 6% de álcool e o restante de gasolina?

18. Um modelo macroeconômico simples consiste de três equações: a) Consumo $C = \alpha_0 + \alpha_1 Y + \alpha_2 T$, b) Investimento $I = \beta_0 + \beta_1 Y + \beta_2 K$ e Renda

$Y = C + I + G$ onde T são os impostos, K o estoque de Capital e G os gastos do governo. Esse modelo poderia ser escrito em forma matricial conforme descrito na 3.29. Dê uma condição necessária para que tal sistema tenha solução única. Resp: por exemplo a matriz de coeficientes $[3,3]$ da esquerda deve ser inversível, isto é ter determinante diferente de zero.

$$\begin{bmatrix} 1 & 0 & -\alpha_1 \\ 0 & 1 & -\beta_1 \\ -1 & -1 & 1 \end{bmatrix} . \begin{bmatrix} C \\ I \\ Y \end{bmatrix} = \begin{bmatrix} \alpha_0 + \alpha_2 T \\ \beta_0 + \beta_2 K \\ G \end{bmatrix} \tag{3.29}$$

19. De um exemplo onde a combinação linear de três vetores não nulos do R^4 produza o vetor nulo. Confirme o resultado criando com estes três vetores uma matriz A e multiplicando a mesma por um vetor x apropriado para gerar o vetor nulo. Indique também as dimensões de A, x e do vetor nulo.

```
import numpy as np
import sympy as sp
x1 = np.array([1,1,1,1]).reshape(-1,1)
x2 = np.array([2,3,4,5]).reshape(-1,1)
x3 = np.array([3,4,5,6]).reshape(-1,1)
A = np.concatenate([x1,x2,x3],axis=1)
A
```

```
array([[1, 2, 3],
       [1, 3, 4],
       [1, 4, 5],
       [1, 5, 6]])
```

```
x = np.array([1,1,-1]).reshape(-1,1)
x
```

```
array([[ 1],
       [ 1],
       [-1]])
```

```
R = A.dot(x)
R
```

```
array([[0],
       [0],
       [0],
       [0]])
```

20. Suponha uma matriz com vetores no R^4 onde existam duas colunas em que cada uma seja combinação linear de outras duas colunas. Com base nesta afirmação encontre duas soluções distintas x e y para Az=0 (isto é vetores x e y tais que Ax=0 e Ay=0).

21. Suponha uma matriz de uns (3x3). Encontre dois vetores independentes que sejam solução de Az=0. Existe um 3o vetor independente? Não! O espaço das colunas de A tem dimensão 1. A dimensão de A é 3. Logo o espaço nulo de A tem dimensão 2. Tendo dimensão 2 temos apenas dois vetores independentes que podem ser solução de A.z = 0.

22. Os vetores $v = (1, 1, 0)$ e $w = (0, 1, 1)$ formam um plano no R^3. Encontre um vetor z perpendicular a este plano. Solução: se este vetor for chamado de z teremos $(c\vec{v} + d\vec{w}).z = 0$. Logo $c = -z_1$, $d = -z_3$ e $c + d = -z_2$. Logo o vetor z perpendicular ao plano formado por v e w terá o formato $z_1 + z_3 = z_2$. Um exemplo seria $z = (1, 2, 1)$

Capítulo 4

Álgebra Linear via Python: Numpy

Vamos agora conhecer melhor a `numpy`, a biblioteca de manipulação de arrays do Python. Entenda um array `numpy` como uma matrix armazenada na memória do computador. Primeiro vamos configurar o ambiente, conforme descrito nos comandos a seguir:

```
import numpy as np
# Número de casas decimais para impressão de resultados
np.set_printoptions(precision=3)

# Se será suprimida ou mantida a notação científica
np.set_printoptions(suppress=False)

# No máximo de elementos que serão impressos
np.set_printoptions(threshold=1000)

# Versão da numpy em uso
print(np.__version__)
```

```
1.21.2
```

Para começar criamos um array numpy similar a matriz descrita abaixo:

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \end{bmatrix} \tag{4.1}$$

Podemos criar este array explicitamente indicando cada número que o compõe:

```
import numpy as np
a = np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]])
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

Uma vez que os números encontram-se em ordem crescente, de 0 a até 11 podemos também criar o mesmo de forma automática com `np.arange` (o que irá gerar um array `numpy` unidimensional também chamado de vetor) e em seguida reformata-lo (através do método `.reshape`):

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

4.1 Eixos e Dimensões

Uma primeira observação importante. O array:

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a
```

```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

tem 2 eixos, o primeiro eixo tem 3 dimensões e o segundo eixo tem 4 dimensões. Dizemos portanto que ele é `2D [3,4]`. Leia o D maiúsculo sempre como **eixo** e evite ler o D como "dimensão." A palavra dimensão especifica quanto endereços temos em cada eixo. Um outro exemplo: um array 3D [5,1,7] é um objeto de três eixos, o primeiro eixo com cinco dimensões, o segundo eixo com uma dimensão e o terceiro eixo com sete dimensões.

Usar a denominação "eixo" e evitar o uso da palavra "dimensão" para o D maiúsculo acaba com a ambiguidade entre vetores em programação e vetores em matemática. Em matemática um vetor é uma matriz com n linhas e apenas uma coluna. Sendo assim em matemática, um vetor é um objeto com dois eixos (2D).

Em programação um vetor é um objeto com apenas um eixo (1D portanto) e n dimensões. O leitor deve ficar atento para o contexto e lembrar que "vetor" em matemática é um objeto diferente de "vetor" em ciência da computação.

Dito isto, a indexação, isto é a numeração dos endereços em cada eixo nos arrays `numpy`, funciona de maneira similar a das listas. De certa forma, os arrays `numpy` podem ser entendidos como "listas com maiores funcionalidades de cálculo." A mais importante delas é a possibilidade de se executar uma operação matemática diretamente em cada um dos elementos do array sem o uso de loops ou compreensão de lista. Por exemplo, suponha que você queira somar 3 em cada um dos elementos do array mais acima. Para isto basta somar 3 ao array, conforme pode ser visto a seguir:

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a + 3
```

```
array([[ 3,  4,  5,  6],
       [ 7,  8,  9, 10],
       [11, 12, 13, 14]])
```

Ou por exemplo determinar qual elemento é igual a 5. Para executar tal operação basta comparar o array a 5 utilizando `==`. Isto irá gerar uma sequência de `True` e `False` com o valor `True` nas posições em que o valor do elemento é 5.

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
a == 5
```

```
array([[False, False, False, False],
       [False,  True, False, False],
       [False, False, False, False]])
```

A respeito de operações com booleanos, os operadores not:`~`, and:`&` e ou:`|` podem ser aplicados normalmente nos arrays, como por exemplo a seguir quando calculamos $(not(A == 5))\ and\ (not(A == 6))$.

```
import numpy as np
a = np.arange(0,12).reshape(3,4)
(~(a==5))&(~(a==6))
```

```
array([[ True,  True,  True,  True],
       [ True, False, False,  True],
       [ True,  True,  True,  True]])
```

Outros métodos importantes são `.any(axis=número)` e `.all(axis=número)`. Quando aplicados em um array de boleanos eles verificam se pelo menos um

(qualquer, any em inglês) ou todos (all em inglês) os elementos de um array (de elementos lógicos) ao longo de um eixo contém o valor `True`. Observe a seguir. Criamos o array de lógicos acima e salvamos ele na variável `b`. Se aplicarmos em `b` o método {.all(axis=0)} receberemos como resposta um array de um eixo (1D) com 4 posições (isto é quatro dimensões) uma para cada coluna de `b`. O valor deste resultado será `True` se todos os elementos da coluna forem `True`.

```
import numpy as np
a = np.arange(12).reshape(3,4)
b = (~(a==5))&(~(a==6))
b.all(axis=0)
```

```
array([ True, False, False,  True])
```

Ou se desejarmos avaliar quais linhas tem quaisquer dos seus elementos iguais a `True`, aplicaremos o método `.any(axis=1)`, obtendo como resultado um array de um eixo (1D) de três dimensões, uma para cada linha, conforme pode ser visto a seguir.

```
import numpy as np
a = np.arange(12).reshape(3,4)
b = (~(a==5))&(~(a==6))
b.any(axis=1)
```

```
array([ True,  True,  True])
```

Outra observação importante, em especial para usuários acostumados à lógica de cálculo do R ou do Excel. No Python quando passamos o parâmetro `axis=0` queremos dizer que o cálculo será feito **ao longo* do eixo 0, portanto os resultados serão obtidos por coluna! Isto é o contrário do que ocorre no R onde devemos especificar o eixo em que queremos os resultados! Portanto lembrem-se, no Python especificamos o eixo ao longo do qual a operação será executada e não o eixo ao longo do qual queremos os resultados. Em especial usuários experientes de R demoram um bom tempo para criar o "instinto" necessário para saber quando especificar `axis=0` ou `axis=1`, porque para eles a lógica funciona de forma invertida.

Como tudo em computação existem vantagens e desvantagens nas duas abordagens. A lógica do R (especificar o eixo no qual queremos "ver" os resultados) é mais "visual" porém padece de uma falha. Ela funciona bem apenas para duas dimensões quando conseguimos "ver" onde os resultados vão aparecer (ao longo do eixo das linhas ou ao longo do eixo das colunas). Mas como "ver" o resultado em 3, 4, 5 dimensões? Suponha um array de 4 eixos com 10 dimensões em cada eixo. Cada elemento dele seria especificado por exemplo como: `a[1,5,3,4]`. Como visualizar o eixo ao longo do qual os resultados vão aparecer? O que significa pedir para os resultados serem apresentados "ao longo" do terceiro eixo (`axis=2`)?

Agora pense do jeito que o Python "pensa." Se você disser `a.all(axis=2)` (terceiro eixo) ele vai (neste exemplo) analisar para cada trio de valores (nos outros 3 eixos) se todos os elementos `a[a0,a1,0,a3]` até `a[a0,a1,9,a3]` são `True` e irá te devolver como resposta um novo array agora de três eixos. Complicado? Sim, mas pelo menos deste jeito você consegue manipular objetos de mais de dois eixos, enquanto que do outro jeito, se o número de eixos é maior que dois, "who knows?"

Continuando nosso estudo de recursos da `numpy`, é interessante mencionar as seguintes funções, as quais serão muito úteis na resolução dos exercícios deste capítulo:

1. O método `.ravel()` "desmonta"um array para o formato 1D (com um eixo apenas). Isto permite localizar em um array com mais eixos o elemento de n-ésima posição.

2. Números aleatórios podem ser gerados pelas funções `np.random.randint(menor, maior, quantidade)`, `np.random.uniform(menor, maior, quantidade)` e `np.random.normal(média, desvio, quantidade)`, onde o parâmetro `quantidade` é o número de amostras desejadas. O resultado será um array 1D (de um eixo apenas) o qual poderá ser reformatado através do método `.reshape`. A título de exemplo, para criar um array de dois eixos (2D), de dimensões `[4,5]` com formado por 20 amostras de uma distribuição de números inteiros entre 1 e 6 fazemos: `np.random.randint(1,7,20).reshape(4,5)`

3. Para determinar os elementos comuns a dois arrays (de mesma quantidade de eixos e formatos) podemos usar uma combinação da função `set` (a qual fornece os elementos únicos de um objeto) e o operador lógico `and` o qual no Python é `&` fazendo: `set(x) & set(y)`. Ou então podemos utilizar a função `np.intersect1D(x,y)`. Para determinar todos os elementos distintos nos dois arrays (sua união) fazemos: `set(list(set(x)) + list(set(y)))`

4. A posição de um elemento (a partir do seu valor, por exemplo o maior valor em um array de um eixo (1D)) pode ser determinada da seguinte forma: `z[z==np.max(z)]`. Observe que `z==np.max(z)` vai compararar cada valor de `z` com o maior valor do conjunto. Isto vai gerar uma sequência de `True` e `False`. Quando aplicamos esta sequência de booleanos nos endereços do array `z` obtemos apenas os valores para os quais temos `True` na sequência de booleanos.

5. Quando queremos obter a posição de um elemento em um array de mais de um eixo (2D por exemplo) utilizamos a função `np.unravel_index`. Suponha que você queira determinar a posição do 10o elemento, mas em um array de dois eixos, no formato `[5,5]`. Para isso faríamos `np.unravel_index(9, (5,5))` e obteríamos como resposta `(1,4)` isto é 2a linha, quarta coluna.

6. A lógica de procura da posição de um valor em um array apresentada em detalhes nos ítens 4 e 5 é também implementada de forma direta através

da função `np.where`

7. Criar um array 3D (de 3 eixos) de dimensões `[3,4,5]` significa na prática criar 3 matrizes diferentes, cada uma delas de dimensões `4,5`. Para observar tal efeito execute `np.arange(60).reshape(3,4,5)`

4.2 Arrays Estruturados

Um array estruturado é um array no qual cada posição do mesmo é outro array. Por exemplo observe abaixo a criação de um tipo de dado (dtype) que descreve uma cor como quatro "unsigned" bytes (RGBA)

```
import numpy as np
color = np.dtype([("r", np.ubyte, (1,)),
                  ("g", np.ubyte, (1,)),
                  ("b", np.ubyte, (1,)),
                  ("a", np.ubyte, (1,))])
print(color)
```

```
[('r', 'u1', (1,)), ('g', 'u1', (1,)), ('b', 'u1', (1,)), ('a', 'u1', (1,))]
```

4.2.1 1o Exemplo

Como um primeiro exemplo, suponha que desejamos criar um array estruturado, no qual as coordenadas x e y cobrem a área $[0,1],[0,1]$. Como foi dito um array estruturado é aquele no qual cada elemento do array é um outro array, tupla ou lista. O array a seguir é do tipo não-estruturado.

```
z = np.zeros([3,3], dtype=float)
print(z)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

O array a seguir é do tipo estruturado, porém cada elemento tem uma estrutura interna única.

```
z = np.zeros([3,3], dtype=[("x",float)])
print(z)
```

```
[[(0.,) (0.,) (0.,)]
 [(0.,) (0.,) (0.,)]
 [(0.,) (0.,) (0.,)]]
```

Podemos carregar a estrutura interna do array através do seu label específico.

```
print(z["x"])
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

```
z["x"] = 1
print(z)
```

```
[[(1.,) (1.,) (1.,)]
 [(1.,) (1.,) (1.,)]
 [(1.,) (1.,) (1.,)]]
```

Vamos agora criar um array com duas estruturas internas. Observe que neste momento estamos criando um array de pares de 0s. O array do 1o elemento de cada par é acessado como `x`, o tipo de dado é float. O array do 2o elemento de cada par é acessado como `y`, o tipo de dado também é float.

```
z = np.zeros([3,3], dtype=[("x",float),("y",float)])
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

Vamos agora modificar as estruturas internas do array. Primeiro a estrutura cujo label chama-se `x`.

```
z["x"]=1
print(z)
```

```
[[(1., 0.) (1., 0.) (1., 0.)]
 [(1., 0.) (1., 0.) (1., 0.)]
 [(1., 0.) (1., 0.) (1., 0.)]]
```

E agora a estrutura com o label `y`

```
z["y"]=2
print(z)
```

```
[[(1., 2.) (1., 2.) (1., 2.)]
 [(1., 2.) (1., 2.) (1., 2.)]
 [(1., 2.) (1., 2.) (1., 2.)]]
```

Observe que cada elemento do array `z` é uma estrutura com dois valores. Por exemplo `z[0,0]`

```
print(z[0,0])
```

```
(1., 2.)
```

Por sua vez o elemento `(0,0)` da estrutura `x` do array `z` é um valor individual e igual a `1`.

```
print(z["x"][0,0])
```

```
1.0
```

Podemos então inserir os elementos desejados em cada um dos "sub-arrays." Primeiro criamos o array estruturado com dois sub-arrays "paralelos."

```
z = np.zeros([3,3], dtype = [("x",float),("y",float)])
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

Criamos um array 1D, o qual para fins de trabalho com arrays 2D é considerado um array linha.

```
x = np.linspace(0,2,3)
print(x)
```

```
[0. 1. 2.]
```

O sub-array `z['x']` é um array 2D, $(5, 5)$. Para fins de inserção dos valores o array 1D `x` é considerado $[1, 5]$ e será alinhado no topo do array `z['x']`. Em seguida os valores serão igualados linha a linha.

```
z["x"] = x
print(z)
```

```
[[(0., 0.) (1., 0.) (2., 0.)]
 [(0., 0.) (1., 0.) (2., 0.)]
 [(0., 0.) (1., 0.) (2., 0.)]]
```

```
print()
```

Alteramos a forma do array x para um array 2D, $[3, 1]$. Para fins de comparação com o array `z['y']`, que é $[3, 3]$, ele será alinhado "em pé" ao lado do array `z['y']` e em seguida terá os valores igualados coluna por coluna.

```
z["y"] = x.reshape(3,1)
print(z)
```

```
[[(0., 0.) (1., 0.) (2., 0.)]
 [(0., 1.) (1., 1.) (2., 1.)]
 [(0., 2.) (1., 2.) (2., 2.)]]
```

A forma a seguir "empilha" arrays 1D através de `np.vstack()`.

```
a = np.linspace(0,2,3)
print(a)
```

```
[0. 1. 2.]
```

```
print(np.vstack([a,a,a]))
```

```
[[0. 1. 2.]
 [0. 1. 2.]
 [0. 1. 2.]]
```

```
print(np.linspace(0,2,3).reshape(3,1))
```

```
[[0.]
 [1.]
 [2.]]
```

```
print(np.vstack([a,a,a]).transpose())
```

```
[[0. 0. 0.]
 [1. 1. 1.]
 [2. 2. 2.]]
```

```
a = np.linspace(0,1,3)
print(a)
```

```
[0.  0.5 1. ]
```

```
print()
```

```
z = np.zeros( [3,3] , dtype = [("x",float), \
                               ("y",float)] )
print(z)
```

```
[[(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]
 [(0., 0.) (0., 0.) (0., 0.)]]
```

```
print()
```

A seguir, por questões de espaço de impressão vamos "empilhar" apenas três cópias do array 1D `a`.

```
z["x"], z["y"] = [np.vstack([a,a,a]), \
                  np.vstack([a,a,a]).transpose()]
print(z)
```

```
[[(0. , 0. ) (0.5, 0. ) (1. , 0. )]
 [(0. , 0.5) (0.5, 0.5) (1. , 0.5)]
 [(0. , 1. ) (0.5, 1. ) (1. , 1. )]]
```

Esta é a forma mais eficiente. Ela gera automaticamente dois arrays independentes e transpostos entre si. Um com elementos iguais nas linhas e outro com elementos iguais nas colunas. Podemos utilizar este sistema para não ter que atribuir valores transpondo `np.linspace` explicitamente.

```
z = np.meshgrid(np.linspace(0,1,3),np.linspace(0,1,3))
print(z[0], "\n\n", z[1])
```

```
[[0.  0.5 1. ]
 [0.  0.5 1. ]
 [0.  0.5 1. ]]

 [[0.  0.  0. ]
 [0.5 0.5 0.5]
 [1.  1.  1. ]]
```

Vamos então passar cada um dos resultados de `np.meshgrid` para cada um dos sub arrays que formam o array estruturado `z`.

```
z = np.zeros((3,3), [("x",float),("y",float)])
z["x"], z["y"] = np.meshgrid(np.linspace(0,1,3),
                             np.linspace(0,1,3))
print(z)
```

```
[[(0. , 0. ) (0.5, 0. ) (1. , 0. )]
 [(0. , 0.5) (0.5, 0.5) (1. , 0.5)]
 [(0. , 1. ) (0.5, 1. ) (1. , 1. )]]
```

4.2.2 2o Exemplo

Neste exemplo vamos criar um array estruturado 5x1 representando uma posição (x, y) e uma cor (r, g, b).

Primeiro criamos o array com a representação de x e y

```
posicao = [("x","float"),("y","float")]
print(posicao)
```

```
[('x', 'float'), ('y', 'float')]
```

Depois o array com a representação da cor

```
cor = [("r","float"),("g","float"),("b","float")]
print(cor)
```

```
[('r', 'float'), ('g', 'float'), ('b', 'float')]
```

E em seguida o array `z`, `[3,1]`, com o `datatype=posicao,cor`

```
z = np.zeros([3,1], dtype=[("p",posicao),("c",cor)])
print(z)
```

```
[[((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]]
```

Ou de forma direta:

```
Z = np.zeros([5,1], [ ("position", \
                       [ ("x", float, 1),
                         ("y", float, 1) ] ),
                      ("color",
                       [ ("r", float, 1),
                         ("g", float, 1),
                         ("b", float, 1) ] ) ] )
print(Z)
```

```
[[((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]
 [((0., 0.), (0., 0., 0.))]]
```

Vamos interpretar o resultado anterior. Começando por um array de 5 elementos.

```
z = np.zeros(5)
print(z)
```

```
[0. 0. 0. 0. 0.]
```

Um array de 5 elementos, estruturado em 2 sub-arrays de 2 elementos. O primeiro se chama position e o segundo color. Foram adotados tipos numéricos diferentes para facilitar a visualização.

```
z = np.zeros(5, dtype=[("position",np.int64), \
                       ("color",float)])
print(z)
```

```
[(0, 0.) (0, 0.) (0, 0.) (0, 0.) (0, 0.)]
```

Acessando o primeiro sub-array:

```
print(z["position"])
```

```
[0 0 0 0 0]
```

Acessando o segundo sub-array:

```
print(z["color"])
```

```
[0. 0. 0. 0. 0.]
```

Continuando a partir da célula anterior. Vamos agora aumentar a complexidade do primeiro sub-array. Cada elemento do primeiro array será composto de 2 elementos

```
import numpy as np
z = np.zeros(5, dtype=[("position", \
                        [("x",float), \
                         ("y",float)]), \
                         ("color",float)])
print(z)
```

```
[((0., 0.), 0.) ((0., 0.), 0.) ((0., 0.), 0.) ((0., 0.), 0.)
 ((0., 0.), 0.)]
```

```
print(z["position"])
```

```
[(0., 0.) (0., 0.) (0., 0.) (0., 0.) (0., 0.)]
```

Vamos agora aumentar a complexidade do segundo sub-array. Cada elemento do mesmo será composto de três elementos

```
z = np.zeros(5, dtype=[("position",[("x",float),
                                    ("y",float)]),
                       ("color",[("r",float),
                                 ("g",float),
                                 ("b",float)])])
print(z)
```

```
[((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.))]
```

Vamos incluir o terceiro argumento em cada dtype

```
z = np.zeros(5, dtype=[("position",[("x",float,1),
                                    ("y",float,1)]),
                       ("color",[("r",float,1),
                                 ("g",float,1),
                                 ("b",float,1)])])
print(z)
```

```
[((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.)) ((0., 0.), (0., 0., 0.))
 ((0., 0.), (0., 0., 0.))]
```

4.2.3 3o Exemplo

Neste exemplo vamos criar um grid de cálculo para $z = x^2 + y^2$ e obter todos os valores de `z` no domínio plano $x = [-2, 2]$ e $y = [-2, 2]$ divididos em 5 pontos.

Primeiro vamos gerar um grid "zerado"

```
w = np.zeros([5,5])
print(w)
```

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

Vamos em seguida produzir o eixo horizontal, com 11 pontos entre -5 e 5

```
x = np.linspace(-2,2,5).reshape(1,-1)
print(x)
```

```
[[-2. -1.  0.  1.  2.]]
```

E agora utilizando o recurso de broadcast façamos `xg = x - w` e vejamos o resultado

```
xg = x - w
print(xg)
```

```
[[-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]]
```

Façamos algo similar para `y`, porém tendo o cuidado de torna-lo primeiro um array vertical.

```
y = np.linspace(-2,2,5).reshape(-1,1)
print(y)
```

```
[[-2.]
 [-1.]
 [ 0.]
 [ 1.]
 [ 2.]]
```

Observe agora o efeito de fazer `yg = y - w`

```
yg = y - w
print(yg)
```

```
[[-2. -2. -2. -2. -2.]
 [-1. -1. -1. -1. -1.]
 [ 0.  0.  0.  0.  0.]
 [ 1.  1.  1.  1.  1.]
 [ 2.  2.  2.  2.  2.]]
```

Vamos agora criar o grid de pares de pontos `xg,yg` através de um array estruturado de nome `g`

```
g = np.zeros((5,5), [("x",float),("y",float)])
g["x"]=xg
g["y"]=yg
print(g)
```

```
[[(-2., -2.) (-1., -2.) ( 0., -2.) ( 1., -2.) ( 2., -2.)]
 [(-2., -1.) (-1., -1.) ( 0., -1.) ( 1., -1.) ( 2., -1.)]
 [(-2.,  0.) (-1.,  0.) ( 0.,  0.) ( 1.,  0.) ( 2.,
0.)]
```

```
 [(-2.,  1.) (-1.,  1.) ( 0.,  1.) ( 1.,  1.) ( 2.,
1.)]
 [(-2.,  2.) (-1.,  2.) ( 0.,  2.) ( 1.,  2.) ( 2.,
2.)]]
```

A partir do array estruturado `g` calculamos a função $z = x^2 + y^2$

```
z = g["x"]**2 + g["y"]**2
print(z)
```

```
[[8. 5. 4. 5. 8.]
 [5. 2. 1. 2. 5.]
 [4. 1. 0. 1. 4.]
 [5. 2. 1. 2. 5.]
 [8. 5. 4. 5. 8.]]
```

4.3 Generators e Iterators

`Generators` e `Iterators` em Python são similares aos loops `while` e {for} com a diferença que eles permitem que o código saia do loop e ao retornar encontre o mesmo no ponto em que foi abandonado da última vez. Da mesma forma que com os loops `while` um `Generator` mantém seu estado através de variáveis locais e de forma similar aos loops `for` or `Iterators` mantém seu estado de forma automática (por uma variável definida junto com o `Iterator`). Vejamos alguns exemplos a seguir.

4.3.1 1o Exemplo

Como um primeiro exemplo vamos criar um generator que produza 10 números inteiros. Utilizaremos o mesmo juntamente com a função `np.fromiter` para produzir arrays com tamanho, 1, 2, 3, 10. Veremos também o que acontecerá nos casos em que o tamanho indicado no parâmetro de contagem for igual tanto a -1 e 11.

```
def generate():
  for x in range(10):
    yield(x)

z = np.fromiter(generate(), dtype="int64", count=1)
print(z)
```

```
[0]
```

```
z = np.fromiter(generate(), dtype="int64", count=2)
print(z)
```

```
[0 1]
```

```
z = np.fromiter(generate(), dtype="int64", count=3)
print(z)
```

```
[0 1 2]
```

```
z = np.fromiter(generate(), dtype="int64", count=10)
print(z)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
z = np.fromiter(generate(), dtype="int64", count=-1)
print(z)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

```
try:
  z = np.fromiter(generate(), dtype="int64", count=11)
  print(z)
except:
  print("Generator produziu um erro por estar fora da faixa")
```

```
Generator produziu um erro por estar fora da faixa
```

4.3.2 2o Exemplo

Neste exemplo vamos utilizar um iterator para calcular a soma de dois arrays com formatos `[1,3]` e `3,1`.

Observe a solução a seguir:

```
A = np.arange(3).reshape(3,1)
B = np.arange(3).reshape(1,3)
it = np.nditer([A,B,None])
for x,y,z in it:
  z[...] = x + y

print(it.operands[2])
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]]
```

Para entender o código apresentado, procuramos no help da `numpy` a definição de Iterator onde obtemos o seguinte exemplo:

```
a = np.arange(6).reshape(2,3)
print(a)
```

```
[[0 1 2]
 [3 4 5]]
```

```
print()
```

```
for x in np.nditer(a):
  print(x, end=' ')
```

```
0 1 2 3 4 5
```

Como pode ser visto a função `np.nditer()` percorre elemento por elemento do array. Analisemos então o código fornecido acima. Para melhor identificar A e B vamos designar valores distintos

```
A = np.arange(3).reshape(3,1)
print(A)
```

```
[[0]
 [1]
 [2]]
```

```
B = np.arange(10,13).reshape(1,3)
print(B)
```

```
[[10 11 12]]
```

```
for x in np.nditer(A):
  print(x,' ')
```

```
0
1
2
```

```
for x in np.nditer(B):
  print(x,' ')
```

```
10
11
12
```

Veja que o efeito de `np.nditer([A,B])` é igual ao de um loop duplo

```
for x in np.nditer([A,B]):
  print(x,' ')
```

```
(array(0), array(10))
(array(0), array(11))
(array(0), array(12))
(array(1), array(10))
(array(1), array(11))
(array(1), array(12))
(array(2), array(10))
(array(2), array(11))
(array(2), array(12))
```

```
for x in np.nditer([A,B]):
  print(x[0], ' ')
```

```
0
0
0
1
1
1
2
2
2
```

```
for x in np.nditer([A,B]):
  print(x[1],' ')
```

```
10
11
12
10
11
12
10
11
12
```

Qual a função de None no iterador original? Veremos que ele serve para criar uma variável não inicializada

```
for x in np.nditer([A,B,None]):
  print(x,' ','shape =',np.array(x).shape)
```

```
(array(0), array(10), array(0))   shape = (3,)
(array(0), array(11), array(4602678819172646912))
shape = (3,)
(array(0), array(12), array(4607182418800017408))
shape = (3,)
(array(1), array(10), array(0))   shape = (3,)
(array(1), array(11), array(4602678819172646912))
shape = (3,)
(array(1), array(12), array(4607182418800017408))
shape = (3,)
(array(2), array(10), array(0))   shape = (3,)
(array(2), array(11), array(4602678819172646912))
shape = (3,)
(array(2), array(12), array(4607182418800017408))
shape = (3,)
```

Agora, relembrando, o comando abaixo cria o array 2D, A com o formato `[3,1]`

```
A = np.arange(3).reshape(3,1)
print(A)
```

```
[[0]
 [1]
 [2]]
```

Cria o array 2D, com o formato `B[1,3]`

```
B = np.arange(3).reshape(1,3)
print(B)
```

```
[[0 1 2]]
```

Gera um objeto de iteração. Este objeto por vez gera 3 valores `A[i], B[i], None`

```
it = np.nditer([A,B,None])
```

Antes do loop. O 3o elemento de `[A,B,None]` é inicializado com o valor que estiver na memória no momento.

```
print(it.operands[0])
```

```
[[0]
 [1]
 [2]]
```

```
print()
```

```
print(it.operands[1])
```

```
[[0  1  2]]
```

```
print()
```

```
print(it.operands[2])
```

```
[[                   0                    0                    0
0]
 [4602678819172646912 4602678819172646912 4602678819172646912]
 [4607182418800017408 4607182418800017408 4607182418800017408]]
```

```
print()
```

O loop é executado

```
for x,y,z in it:
  z[...] = x + y
```

Depois do loop

```
print(it.operands[0])
```

```
[[0]
 [1]
 [2]]
```

```
print()
```

```
print(it.operands[1])
```

```
[[0  1  2]]
```

```
print()
```

```
print(it.operands[2])
```

```
[[0  1  2]
 [1  2  3]
 [2  3  4]]
```

Falta entender o operador `...` também conhecido por elipsis. Vamos voltar ao código. Observe o que ocorre ao reinicializarmos os valores de `it` ao final do código e depois reexecutamos o mesmo

```
A = np.arange(0,4,dtype=np.int32).reshape(4,1)
B = np.arange(0,4,dtype=np.int32).reshape(1,4)

it = np.nditer([A,B,None])

print(it.operands[2])
```

```
[[-1680603523 -1076628655           0           0]
 [-1009901430  1072548873           0           0]
 [    -7696581 -1077501381           0           0]
 [ 1255642279 -1078907160           0           0]]
```

```
print()
```

```
for x,y,z in it:
  z[...] = 3*x + y

print(it.operands[2])
```

```
[[ 0  1  2  3]
 [ 3  4  5  6]
 [ 6  7  8  9]
 [ 9 10 11 12]]
```

```
print()
```

```
it = np.nditer([A,B,np.zeros([4,4])])
```

Em resumo os valores de `it.operands[2]` são inicializados com o valor que estiver disponível na memória, ou seja, qualquer valor. Normalmente eles são inicializados com os últimos valores designados para aquela posição de memória

Observe agora o efeito de retirar a elipsis de `z`

```
A = np.arange(0,4,dtype=np.int32).reshape(4,1)
B = np.arange(0,4,dtype=np.int32).reshape(1,4)

it = np.nditer([A,B,None])

print(it.operands[2])
```

```
[[ 7  0 11  0]
 [15  0 19  0]
 [24  0 28  0]
 [32  0 36  0]]
```

```
print()
```

```
for x,y,z in it:
  z = 3*x + y

print(it.operands[2])
```

```
[[ 7  0 11  0]
 [15  0 19  0]
 [24  0 28  0]
 [32  0 36  0]]
```

```
print()
```

```
it = np.nditer([A,B,np.zeros([4,4])])
```

Em resumo os valores são perdidos

4.3.3 3o Exemplo

Vamos agora ver como implementar o recurso equivalente a `enumerate` em arrays `numpy`.

Primeiro vamos recordar um exemplo de enumerate. Enumerate torna mais fácil a sincronização de índices e valores em um loop que percorra todos os elementos de um array.

```
lista = ["Argelia","Brasil","China", \
         "Espanha","Finlandia"]

for indice, valor in enumerate(lista):
  print(indice, valor)
```

```
0 Argelia
1 Brasil
2 China
3 Espanha
4 Finlandia
```

Supondo que lista seja um numpy array podemos fazer:

```
lista = np.array(["Argelia","Brasil","China", \
                  "Espanha","Finlandia"])

for indice, valor in np.ndenumerate(lista):
  print(indice, valor)
```

```
(0,) Argelia
(1,) Brasil
(2,) China
(3,) Espanha
(4,) Finlandia
```

Observe que o índice veio no formato da dimensão de um numpy array. Vamos observar o que irá ocorrer se o numpy array for do tipo 2D. Por conveniência vamos utilizar um array com dados numéricos apenas.

```
z = np.arange(16).reshape(4,4)
print(z)
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
```

Vemos pelo exemplo abaixo que enumerate "desce" apenas um nível na estrutura de dados (seja uma lista, seja um numpy array) e enumera os respectivos índices e valores

```
z = np.arange(16).reshape(4,4)
for indice, valor in enumerate(z):
  print(indice, valor)
```

```
0 [0 1 2 3]
1 [4 5 6 7]
2 [ 8  9 10 11]
3 [12 13 14 15]
```

Poderíamos ter usado um loop aninhado

```
z = np.arange(16).reshape(4,4)
for indice1, valor1 in enumerate(z):
  for indice2, valor2 in enumerate(valor1):
    print(indice1, indice2, valor2)
```

```
0 0 0
0 1 1
0 2 2
0 3 3
1 0 4
1 1 5
1 2 6
1 3 7
2 0 8
2 1 9
2 2 10
2 3 11
3 0 12
3 1 13
3 2 14
3 3 15
```

Ou np.nditer(z):

```
z = np.arange(16).reshape(4,4)
for indice, valor in enumerate(np.nditer(z)):
  print(indice, valor)
```

```
0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
10 10
11 11
12 12
13 13
14 14
15 15
```

Em ambos os casos para recuperar o endereço no formato original teríamos de lançar mão de np.unravel_index.

```
z = np.arange(16).reshape(4,4)
for indice, valor in enumerate(np.nditer(z)):
  indice2d = np.unravel_index(indice,shape=[4,4])
```

```
  print(indice2d, valor)
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(0, 3) 3
(1, 0) 4
(1, 1) 5
(1, 2) 6
(1, 3) 7
(2, 0) 8
(2, 1) 9
(2, 2) 10
(2, 3) 11
(3, 0) 12
(3, 1) 13
(3, 2) 14
(3, 3) 15
```

Ou então simplesmente np.ndenumerate(z), o qual tem a vantagem de preservar o endereço original, sem a necessidade de reconversão

```
z = np.arange(16).reshape(4,4)
for indice, valor in np.ndenumerate(z):
  print(indice, valor)
```

```
(0, 0) 0
(0, 1) 1
(0, 2) 2
(0, 3) 3
(1, 0) 4
(1, 1) 5
(1, 2) 6
(1, 3) 7
(2, 0) 8
(2, 1) 9
(2, 2) 10
(2, 3) 11
(3, 0) 12
(3, 1) 13
(3, 2) 14
(3, 3) 15
```

4.4 Exercícios

1. Crie um array de dois eixos (2D), com dimensões `[5,6]` formado pelos números de 0 a 29. Nas linhas de índice par os números deverão estar dispostos em ordem crescente e nas linhas de índice ímpar deverão estar dispostos em ordem decrescente. Chame este array de `x`. Em seguida transforme este array em um array de apenas um eixo (1D) chamando este segundo array de `y` e determine:

   (a) a partir de `x`:

      i. todos os elementos do array

      ii. o valor na posição 12

      iii. o último valor

      iv. o primeiro valor

      v. o penúltimo valor

      vi. os elementos da quarta até a décima posição

      vii. todos os elementos a partir da posição 7 situados posições com índices pares

      viii. todos os elementos em ordem reversa.

      ix. a posição do elemento cujo valor é 9

      x. a posição do elemento de menor valor

      xi. a posição de todos os elementos pares. Use o operador de resto `%`

      xii. como trocar todos os pares por -1

   (b) a partir de `y`:

      i. o valor na 1a linha e 2a coluna

      ii. o valor na 3a linha e última posição da 3a coluna

      iii. como alterar o valor na posição [0,0] para 12

      iv. todos os valores na última linha

      v. todos os valores na última coluna

      vi. a soma dos valores na segunda coluna

      vii. a soma dos valores na primeira e terceira linhas

      viii. todos os valores entre a segunda e terceira linhas e da segunda até a última coluna

      ix. a posição 2D (em dois eixos, linha e coluna) do maior elemento

2. Crie dois arrays de números aleatórios. O primeiro deverá ser 1D e conter 10 valores entre -1 e 1 obtidos a partir de uma distribuição uniforme. Chame este array de `x`. O segundo deverá ter dois eixos (2D) de dimensões `[4,6]` e conter 24 valores obtidos a partir de uma distribuição normal com média 10 e desvio padrão 2. Chame este array de `y`. A partir destes dois arrays determine:

   (a) Para o array `x`:

      i. A posição e o valor do elemento em `x` que esteja mais próximo do valor `c=0,5` em termos absolutos. Dica: use `np.where`

      ii. A posição e o valor de todos os elementos menores que 0

      iii. As posições dos cinco maiores valores. Dica: use o método `.arg sort()`

      iv. Arredonde os valores negativos para menos (-1.1 torna-se -2) e os valores positivos para mais (1.1 torna-se 2). Dica: use `np.ceil` e `np.floor`

      v. Determine o resultado das operações `np.tile(x,2)` e `np.repeat(x,2)`

   (b) Para o array `y`:

      i. A posição em dois eixos (2D) e o valor do elemento em `y` que esteja mais próximo do valor `c=0,5` em termos absolutos. Dica: use `np.where`

      ii. Se alguma das colunas de `y` é composta apenas por valores menores que 10.

      iii. Quantos elementos serão obtidos pelo slice `y[[1,2,3],[1,2,3]]`? Resp: 3 (TRÊS!). Os elementos nas posições `y[1,1]`, `y[2,2]` e `y[3,3]`. Isto mostra que o slice `z[1:4,1:4]` é um pouco mais complexo que duas listas de números. O slice `z[1:4,1:4]` é na verdade todas às combinações de endereços de linha `[1,2,3]` com todas as respectivas combinações de endereços de coluna `[1,2,3]`.

      iv. Como trocar todos os elementos menores que 10 por 9 e todos os elementos maiores que 10 por 11? Dica: `np.where(condição, valor se verdadeiro, valor se falso)`

      v. Os valores da última coluna em ordem ascendente, alterando a mesma com tais valores.

      vi. Os valores da segunda linha em ordem descendente, alterando a mesma com tais valores

vii. A média de cada coluna, da última até segunda coluna (isto é em ordem invertida)

viii. A soma de cada linha, da segunda até a última linha

ix. Como classificar as linhas de acordo com os valores na última coluna. Dica: use o método `.argsort()`

x. As posições dos cinco maiores valores. Dica: use o método `.argsort()`

xi. Como colocar o array `y` ao lado e sobre de uma cópia de si mesmo. Dica: use `np.r_` e hlnp.c_

xii. Como padronizar (diminuir da média e dividir pelo desvio padrão) os valores de `y` por coluna

3. Crie um array 2d 5x5 com os seguintes valores
   (a) 1 nas bordas e 0 na parte interna. Atribua os valores no centro do array em uma única instrução
   (b) 0 nas bordas e a parte interna preenchida com números inteiros aleatórios entre 1 e 9. Dica: utilize a função `np.pad`
   (c) Um número aleatório entre 1 e 9 na posição central do array. Envolva o mesmo com 1s e envolva este array ([3,3]) com zeros.
4. A partir do array 1D `[1,2,3,4,5]` crie:
   (a) Utilizando a função `np.vstack` crie um array 2D `[3,5]` em que o array 1D esteja "empilhado"na horizontal.
   (b) Utilizando a função `np.hstack` crie um array 2D `[5,4]` em que o array 1D esteja colocado lado a lado na vertical.
   (c) Repita os exercícios anteriores utilizando a função `np.concatenate`. Obs: `np.concatenate` requer arrays de dois eixos (2D).
5. Crie uma matriz 4x5, no qual os valores da linha deverão estar entre 0 e 1 e serem igualmente espaçados entre si.
6. A partir de uma matriz 5x5 de zeros, crie uma matriz com os valores 1,2,3,...: (Dica: use `np.diag()`)
   (a) na diagonal
   (b) abaixo da diagonal
   (c) acima da diagonal
   (d) na primeira diagonal à esquerda da principal
   (e) na segunda diagonal à esquerda da principal

7. Crie uma matriz 8x8, com e sem o uso de loops, cujos valores serão dados pelas seguintes operações (i é número da linha e j o número da coluna) ou pelos formatos indicados:
   (a) i+j
   (b) o resto de (i+j) / 2
   (c) o padrão de um tabuleiro de xadrez (branco=0, preto=1)
   (d) cinco linhas iguais, com os valores da sequência 0,2,4,...
   (e) cinco colunas iguais, com os valores da sequência 1,3,5,...
   (f) com os valores de cada linha iguais a i + 3
   (g) com os valores de cada coluna iguais a j - 3
8. Divida o array `z = np.arange(10)` em três arrays. O array do meio deverá ir da posição 3 até a posição 6. Use `np.split`.
9. Transforme o array `x = [0,1,2,3,4]` em um array 2D `[n,1]` e o array `y = x + 0.5` em um array `[1,n]`. Calcule a diferença de cada par de valores na matriz `[n,n]` associada utilizando: a) `np.vstack` e `np.hstack`, b) `np.concatenate` e c) as regras de broadcast de valores da `numpy`.

4.5 Exercícios II

1. Suponha 2 conjuntos de pontos P0,P1 descrevendo linhas (2d) e um conjunto de pontos P. Como calcular a distância de cada ponto j (P[j]) a cada linha i (P0[i],P1[i])?

```
# Author: Italmassov Kuanysh

def distance(P0, P1, p):
    T = P1 - P0
    L = (T**2).sum(axis=1)
    U = -((P0[:,0]-p[...,0])*T[:,0] + \
          (P0[:,1]-p[...,1])*T[:,1]) / L
    U = U.reshape(len(U),1)
    D = P0 + U*T - p
    return np.sqrt((D**2).sum(axis=1))


P0 = np.random.uniform(-10, 10, (5,2))
P1 = np.random.uniform(-10,10,(5,2))
p = np.random.uniform(-10, 10, (5,2))
np.array([distance(P0,P1,p_i) for p_i in p])
```

```
array([[ 5.524,  2.627,  7.145,  0.693,  6.753],
       [ 4.83 , 12.377, 15.479,  3.126, 14.04 ],
       [ 5.634,  2.814, 15.765,  6.836, 15.187],
       [ 1.872,  3.443,  0.587,  7.602,  0.041],
       [ 2.203,  3.78 , 12.298,  2.687, 11.513]])
```

2. Crie um programa capaz de adicionar 1 a cada elemento de um vetor, indexado por um segundo vetor. Leve em consideração o caso de índices repetidos.

```
# Author: Brett Olsen

Z = np.ones(10)
I = np.random.randint(0,len(Z),20)
Z += np.bincount(I, minlength=len(Z))
print(Z)

# Another solution
# Author: Bartosz Telenczuk
```

```
[2. 2. 1. 4. 2. 3. 6. 3. 4. 3.]
```

```
np.add.at(Z, I, 1)
print(Z)
```

```
[ 3.  3.  1.  7.  3.  5. 11.  5.  7.  5.]
```

3. Como acumular os elementos de um vetor (x) para um array (F) tomando por base uma lista de índices (I)?

```
# Author: Alan G Isaac

X = [1,2,3,4,5,6]
I = [1,3,9,3,4,1]
F = np.bincount(I,X)
print(F)
```

```
[0. 7. 0. 6. 5. 0. 0. 0. 0. 3.]
```

4. Suponha uma imagem `(w,h,3)` do tipo `(dtype=ubyte)`. Calcule o número de cores únicas da mesma. Dica: Calcule primeiro o resultado de 256*256. Caso contrário a numpy tornará o dtype F igual a "uint16"e isto provocará um overfolw na sequência.

```
# Author: Nadav Horesh

w,h = 16,16
I = np.random.randint(0,2,(h,w,3)).astype(np.ubyte)
F = I[...,0]*(256*256) + I[...,1]*256 +I[...,2]
n = len(np.unique(F))
print(n)
```

```
8
```

5. Suponha um array tetradimensional. Crie um programa para obter a soma ao longo dos dois últimos eixos de uma única vez.

Solução através da passagem de uma tupla de eixos (válida a partir da numpy 1.7.0).

```
A = np.random.randint(0,10,(3,4,3,4))
sum = A.sum(axis=(-2,-1))
print(sum)
```

```
[[48 52 33 54]
 [57 47 54 41]
 [52 62 59 51]]
```

Solução através da condensação (flattening) das duas últimas dimensões em uma única dimensão. Esta solução é útil para funções que ainda não aceitam tuplas como argumento para a criação e manuseio de eixos.

```
sum = A.reshape(A.shape[:-2] + (-1,)).sum(axis=-1)
print(sum)
```

```
[[48 52 33 54]
 [57 47 54 41]
 [52 62 59 51]]
```

6. Suponha um vetor D 1D. Calcule a média de subconjuntos do vetor utilizando um vetor S do mesmo tamanho que descreve os índices de cada subconjunto.

Solução apenas com numpy arrays

```
# Author: Jaime Fernandez del Rio

D = np.random.uniform(0,1,100)
S = np.random.randint(0,10,100)
D_sums = np.bincount(S, weights=D)
D_counts = np.bincount(S)
```

```
D_means = D_sums / D_counts
print(np.round(D_means,2))
```

```
[0.33 0.55 0.46 0.4  0.34 0.56 0.47 0.52 0.57 0.53]
```

Solução através de dataframes panda

```
import pandas as pd
print(pd.Series(D).groupby(S).mean())
```

```
0    0.329000
1    0.545204
2    0.463267
3    0.399964
4    0.342278
5    0.563987
6    0.468011
7    0.522587
8    0.572582
9    0.533442
dtype: float64
```

7. Como obter a diagonal de um produto matricial?

```
# Author: Mathieu Blondel

A = np.random.uniform(0,1,(5,5))
B = np.random.uniform(0,1,(5,5))
```

Versão lenta

```
np.diag(np.dot(A, B))
```

```
array([1.647, 1.208, 1.334, 0.702, 1.052])
```

Versão com velocidade regular

```
np.sum(A * B.T, axis=1)
```

```
array([1.647, 1.208, 1.334, 0.702, 1.052])
```

Versão mais rápida

```
np.einsum("ij,ji->i", A, B)
```

```
array([1.647, 1.208, 1.334, 0.702, 1.052])
```

8. Suponha o vetor [1, 2, 3, 4, 5]. Crie um novo vetor com 3 zeros em sequência intercalados entre cada valor do vetor.

```
# Author: Warren Weckesser

Z = np.array([1,2,3,4,5])
nz = 3
Z0 = np.zeros(len(Z) + (len(Z)-1)*(nz))
Z0[::nz+1] = Z
print(Z0)
```

```
[1. 0. 0. 0. 2. 0. 0. 0. 3. 0. 0. 0. 4. 0. 0. 0. 5.]
```

9. Crie um programa capaz de trocar duas linhas de posição entre si em um array.

```
# Author: Eelco Hoogendoorn

A = np.arange(25).reshape(5,5)
A[[0,1]] = A[[1,0]]
print(A)
```

```
[[ 5  6  7  8  9]
 [ 0  1  2  3  4]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
```

10. Suponha um conjunto de 5 trios de valores descrevendo 5 triângulos (os quais estão conectados pelos seus vértices. Encontre o conjunto de segmentos de linha únicos que compõe o conjunto dos triângulos.

```
# Author: Nicolas P. Rougier

faces = np.random.randint(0,100,(5,3))
F = np.roll(faces.repeat(2,axis=1),-1,axis=1)
F = F.reshape(len(F)*3,2)
F = np.sort(F,axis=1)
G = F.view( dtype=[("p0",F.dtype),("p1",F.dtype)] )
G = np.unique(G)
print(G)
```

```
[( 8, 16) ( 8, 90) (11, 12) (11, 32) (12, 32) (16, 24) (16, 66) (16, 90)
 (24, 66) (31, 32) (31, 37) (32, 37) (42, 75) (42, 95) (75, 95)]
```

4.6 Exercícios III

1. Crie variáveis dummy para cada valor único do array a seguir. Em inglês esta operação chama-se cálculo dos one-hot encodings.

```
np.random.seed(101)
arr = np.random.randint(1,4, size=6)
arr
#> array([2, 3, 2, 2, 2, 1])
# Solucao:
```

```
array([2, 3, 2, 2, 2, 1])
```

```
def one_hot_encodings(arr):
  uniqs = np.unique(arr)
  out = np.zeros((arr.shape[0], uniqs.shape[0]))
  for i, k in enumerate(arr):
    out[i, k-1] = 1
  return out

print(one_hot_encodings(arr))
```

```
[[0. 1. 0.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]]
```

```
print("")

# Method 2:
```

```
print((arr[:, None] == np.unique(arr)).view(np.int8))
```

```
[[0 1 0]
 [0 0 1]
 [0 1 0]
 [0 1 0]
 [0 1 0]
 [1 0 0]]
```

2. Dado um array numérico, calcule o rank do mesmo.

```
np.random.seed(10)
a = np.random.randint(20, size=10)
print('Array: ', a)
# Solucao
```

```
Array:  [ 9  4 15  0 17 16 17  8  9  0]
```

```
print(a.argsort().argsort())
```

```
[4 2 6 0 8 7 9 3 5 1]
```

```
print('Array: ', a)
```

```
Array:  [ 9  4 15  0 17 16 17  8  9  0]
```

3. Crie um array de rank do mesmo formato que um dado array numérico a.

```
np.random.seed(10)
a = np.random.randint(20, size=[2,5])
print(a)
# Solucao
```

```
[[ 9  4 15  0 17]
 [16 17  8  9  0]]
```

```
print(a.ravel().argsort().argsort().reshape(a.shape))
```

```
[[4 2 6 0 8]
 [7 9 3 5 1]]
```

4. Calcule o valor máximo em cada linha de um array.

```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a
# Solucao 1
```

```
array([[9, 9, 4],
       [8, 8, 1],
       [5, 3, 6],
       [3, 3, 3],
       [2, 1, 9]])
```

```
print(np.amax(a, axis=1))
# Solucao 2
```

```
[9 8 6 3 9]
```

```
print(np.apply_along_axis(np.max, arr=a, axis=1))
```

```
[9 8 6 3 9]
```

5. Calcule o min-by-max para cada linha de um array 2d

```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
a
# Solucao
```

```
array([[9, 9, 4],
       [8, 8, 1],
       [5, 3, 6],
       [3, 3, 3],
       [2, 1, 9]])
```

```
f = lambda x: np.min(x)/np.max(x)
print(np.apply_along_axis(f, arr=a, axis=1))
```

```
[0.444 0.125 0.5   1.    0.111]
```

6. Encontre as entradas duplicadas em um dado array numpy e marque as mesmas como True. A primeira ocorrência de cada valor deve ser marcada como False.

```

np.random.seed(100)
a = np.random.randint(0, 5, 10)
## Solucao
out = np.full(a.shape[0], True)

# Encontrando as posições dos elementos únicos.
unique_positions = np.unique(a, return_index=True)[1]

# marcação das posições como False
out[unique_positions] = False
print(out)
```

```
[False  True False  True False False  True  True
True  True]
```

7. Delete todos os valores nan de um array 1D

```
a = np.array([1,2,3,np.nan,5,6,7,np.nan])
print(a[~np.isnan(a)])
```

```
[1. 2. 3. 5. 6. 7.]
```

8. Encontre todos os máximos locais em um array 1D. Define-se máximo local em um array, valores que estejam cercados por números menores à direita e à esquerda.

```
a = np.array([1, 3, 7, 1, 2, 6, 0, 1])
doublediff = np.diff(np.sign(np.diff(a)))
peak_locations = np.where(doublediff == -2)[0] + 1
print(peak_locations)
```

```
[2 5]
```

9. Calcule a média móvel de tamanho 3 para um array 1D

```
# Solucao
# Source: https://stackoverflow.com/questions/14313510/
#         how-to-calculate-moving-average-using-numpy

def moving_average(a, n=3) :
  ret = np.cumsum(a, dtype=float)
  ret[n:] = ret[n:] - ret[:-n]
  return ret[n - 1:] / n

np.random.seed(100)
Z = np.random.randint(10, size=10)
print('array: ', Z)
```

```
array:  [8 8 3 7 7 0 4 2 5 2]
```

```
# metodo 1
moving_average(Z, n=3).round(1)
```

```
array([6.3, 6. , 5.7, 4.7, 3.7, 2. , 3.7, 3. ])
```

```
# metodo 2: # Fonte: AlanLRH!
# np.ones(3)/3 gives equal weights.
# Use np.ones(4)/4 for window size 4.
z1 = np.ones(3)/3
print('moving average:', np.convolve(Z, z1, mode="valid"))
```

```
moving average: [6.333 6.    5.667 4.667 3.667 2.
3.667 3.   ]
```

10. Dado um array com uma sequência de datas não consecutivas, torne o mesmo contínuo inserindo as datas faltantes no array.

```
dates = np.arange(np.datetime64('2018-02-01'),
                  np.datetime64('2018-02-25'), 2)
print(dates)
```

```
['2018-02-01' '2018-02-03' '2018-02-05' '2018-02-07' '2018-02-09'
 '2018-02-11' '2018-02-13' '2018-02-15' '2018-02-17' '2018-02-19'
 '2018-02-21' '2018-02-23']
```

```
# Solucao ---------------
filled_in = np.array([np.arange(date, (date+d))
            for date, d in zip(dates, np.diff(dates))])

filled_in = filled_in.reshape(-1)
```

```
# incluindo o ultimo dia
output = np.hstack([filled_in, dates[-1]])
print(output)
```

```
['2018-02-01' '2018-02-02' '2018-02-03' '2018-02-04' '2018-02-05'
 '2018-02-06' '2018-02-07' '2018-02-08' '2018-02-09' '2018-02-10'
 '2018-02-11' '2018-02-12' '2018-02-13' '2018-02-14' '2018-02-15'
 '2018-02-16' '2018-02-17' '2018-02-18' '2018-02-19' '2018-02-20'
 '2018-02-21' '2018-02-22' '2018-02-23']
```

```
# Versao com loops
out = []
for date, d in zip(dates, np.diff(dates)):
  out.append(np.arange(date, (date+d)))

filled_in = np.array(out).reshape(-1)

# incluindo o ultimo dia
output = np.hstack([filled_in, dates[-1]])
print(output)
```

```
['2018-02-01' '2018-02-02' '2018-02-03' '2018-02-04' '2018-02-05'
 '2018-02-06' '2018-02-07' '2018-02-08' '2018-02-09' '2018-02-10'
 '2018-02-11' '2018-02-12' '2018-02-13' '2018-02-14' '2018-02-15'
```

```
 '2018-02-16' '2018-02-17' '2018-02-18' '2018-02-19' '2018-02-20'
 '2018-02-21' '2018-02-22' '2018-02-23']
```

11. Dado um array 1D, gere uma matriz 2D utilizando o conceito de strides, com um comprimento de janela de 4 e strides de 2. O resultado final deverá ficar parecido com `[ [0,1,2,3], [2,3,4,5], [4,5,6,7]..]`

```
def gen_strides(a, stride_len=5, window_len=5):
  n_strides = ((a.size-window_len)//stride_len) + 1
  sL = stride_len
  return np.array([a[s:(s+window_len)]
                   for s in np.arange(0, n_strides*sL,sL)])

z = np.arange(15)
print(gen_strides(z, stride_len=2, window_len=4))
```

```
[[ 0  1  2  3]
 [ 2  3  4  5]
 [ 4  5  6  7]
 [ 6  7  8  9]
 [ 8  9 10 11]
 [10 11 12 13]]
```

12. Calcule através de bootstrapping, intervalos de confiança de 95

```
# Author: Jessica B. Hamrick

X = np.random.randn(100) # random 1D array
N = 1000 # number of bootstrap samples
idx = np.random.randint(0, X.size, (N, X.size))
means = X[idx].mean(axis=1)
confint = np.percentile(means, [2.5, 97.5])
print(confint)
```

```
[-0.268  0.131]
```

13. Calcule as contagens de valores únicos em um array 2D, ao longo de suas linhas.

```
np.random.seed(100)
arr = np.random.randint(1,11,size=(6, 10))
print(arr)
```

```
[[ 9  9  4  8  8  1  5  3  6  3]
 [ 3  3  2  1  9  5  1 10  7  3]
 [ 5  2  6  4  5  5  4  8  2  2]
```

```
 [ 8  8  1  3 10 10  4  3  6  9]
 [ 2  1  8  7  3  1  9  3  6  2]
 [ 9  2  6  5  3  9  4  6  1 10]]
```

Solução

```
# Solucao
def counts_of_all_values_rowwise(arr2d):
  # Unique values and its counts row wise
  temp = [np.unique(row, return_counts=True) for row in arr2d]
  num_counts_array = temp
  # Counts of all values row wise
  unicos = np.unique(arr2d)
  return([[int(b[a==i]) if i in a else 0 for i in unicos]
                              for a, b in num_counts_array])
# Print
print(np.arange(1,11))
```

```
[ 1  2  3  4  5  6  7  8  9 10]
```

```
print(np.array(counts_of_all_values_rowwise(arr)))
```

```
[[1 0 2 1 1 1 0 2 2 0]
 [2 1 3 0 1 0 1 0 1 1]
 [0 3 0 2 3 1 0 1 0 0]
 [1 0 2 1 0 1 0 2 1 2]
 [2 2 2 0 0 1 1 1 1 0]
 [1 1 1 1 1 2 0 0 2 1]]
```

Example 2

```
arr = np.array([np.array(list('bill clinton')),
                np.array(list("narendramodi")),
                np.array(list("jjayalalitha"))])
print(np.unique(arr))
```

```
[' ' 'a' 'b' 'c' 'd' 'e' 'h' 'i' 'j' 'l' 'm' 'n' 'o' 'r' 't' 'y']
```

```
print(np.array(counts_of_all_values_rowwise(arr)))
```

```
[[1 0 1 1 0 0 0 2 0 3 0 2 1 0 1 0]
 [0 2 0 0 2 1 0 1 0 0 1 2 1 2 0 0]
 [0 4 0 0 0 0 1 1 2 2 0 0 0 0 1 1]]
```

14. Suponha que uma barra de madeira seja cortada em dois pontos ao acaso. Qual a probabilidade dos três pedaços da barra resultantes produzirem um triângulo?

```
import numpy as np

def triangulo():
  L1 = np.random.uniform()
  x2 = np.random.uniform(L1,1)
  L2 = x2 - L1
  L3 = 1 - x2

  d = False
  if L1 < L2 + L3:
    if L2 < L1 + L3:
      if L3 < L1 + L2:
        if L1 > np.abs(L2 - L3):
          if L2 > np.abs(L1 - L3):
            if L3 > np.abs(L1 - L2):
              d = True
  return(d)

def prob(n):
  soma = 0
  for i in range(n):
    soma = soma + triangulo()
  resultado = soma/n
  return(resultado)

prob(100000)
```

```
0.19278
```

15. Dado um array c que seja um *bincount*, como criar um array A tal que np.bincount(A) == C?

```
# Author: Jaime Fernandez del Rio

C = np.bincount([1,1,2,3,4,4,6])
A = np.repeat(np.arange(len(C)), C)
print(A)
```

```
[1 1 2 3 4 4 6]
```

16. Como calcular médias a partir de uma janela de dados móvel que percorre um array?

```
# Author: Jaime Fernandez del Rio

def moving_average(a, n=3) :
    ret = np.cumsum(a, dtype=float)
    ret[n:] = ret[n:] - ret[:-n]
    return ret[n - 1:] / n
Z = np.arange(10)
print(moving_average(Z, n=3))
```

```
[1. 2. 3. 4. 5. 6. 7. 8.]
```

17. Suponha um array z 1D. A partir dele crie um array 2D cuja primeira linha seja `(Z[0],Z[1],Z[2])` e cada linha subsequente seja deslocada de uma unidade. A última linha neste caso deverá ser `(Z[-3],Z[-2],Z[-1])`.

```
# Author: Joe Kington / Erik Rigtorp
from numpy.lib import stride_tricks

def rolling(a, window):
    shape = (a.size - window + 1, window)
    strides = (a.itemsize, a.itemsize)
    return stride_tricks.as_strided(a, shape=shape, \
                                    strides=strides)
Z = rolling(np.arange(10), 3)
print(Z)
```

```
[[0 1 2]
 [1 2 3]
 [2 3 4]
 [3 4 5]
 [4 5 6]
 [5 6 7]
 [6 7 8]
 [7 8 9]]
```

18. Como negar um booleano. Como trocar o sinal de um dado tipo float sem cópia (isto é `inplace`)?

```
# Author: Nathaniel J. Smith

Z = np.random.randint(0,2,5)
print(np.logical_not(Z, out=Z))
```

```
[0 0 0 0 1]
```

```
Z = np.random.uniform(-1.0,1.0,5)
print(np.round(np.negative(Z, out=Z),4))
```

```
[ 0.458  0.074 -0.983  0.093  0.791]
```

19. Suponha um array arbitrário. Crie uma função que vai extrair um subconjunto do mesmo com um formato pré-definido e centralizado em um elemento específico. Acrescente valores fixos com 'fill' sempre que necessário).

```
# Author: Nicolas Rougier

Z = np.random.randint(0,10,(5,5))
shape = (2,2)
fill  = 0
position = (1,1)

R = np.ones(shape, dtype=Z.dtype)*fill
P  = np.array(list(position)).astype(int)
Rs = np.array(list(R.shape)).astype(int)
Zs = np.array(list(Z.shape)).astype(int)

R_start = np.zeros((len(shape),)).astype(int)
R_stop  = np.array(list(shape)).astype(int)
Z_start = (P-Rs//2)
Z_stop  = (P+Rs//2)+Rs%2

R_start = (R_start - np.minimum(Z_start,0)).tolist()
Z_start = (np.maximum(Z_start,0)).tolist()

zmax = np.maximum(Z_stop-Zs,0)
R_stop = np.maximum(R_start, (R_stop - zmax))
R_stop = R_stop.tolist()
Z_stop = (np.minimum(Z_stop,Zs)).tolist()

Rzip = zip(R_start,R_stop)
Zzip = zip(Z_start,Z_stop)
r = [slice(start,stop) for start,stop in Rzip]
z = [slice(start,stop) for start,stop in Zzip]
R[r] = Z[z]
print(Z)
```

```
[[4 9 2 1 6]
 [4 2 3 7 9]
 [1 0 0 2 3]
```

```
 [4 6 0 3 2]
 [9 7 1 6 9]]
```

```
print(R)
```

```
[[4 9]
 [4 2]]
```

20. Suponha um array `Z = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]`. Crie um array `R = [[1,2,3,4], [2,3,4,5], [3,4,5,6], ..., [11,12,13,14]]`.

```
# Author: Stefan van der Walt

Z = np.arange(1,15,dtype=np.uint32)
R = stride_tricks.as_strided(Z,(11,4),(4,4))
print(R)
```

```
[[ 1  2  3  4]
 [ 2  3  4  5]
 [ 3  4  5  6]
 [ 4  5  6  7]
 [ 5  6  7  8]
 [ 6  7  8  9]
 [ 7  8  9 10]
 [ 8  9 10 11]
 [ 9 10 11 12]
 [10 11 12 13]
 [11 12 13 14]]
```

21. Obtenha os primeiros blocos 3x3 contínuos de uma matriz aleatória 5x5.

```
# Author: Chris Barker

Z = np.random.randint(0,5,(5,5))
n = 3
i = 1 + (Z.shape[0]-3)
j = 1 + (Z.shape[1]-3)
Z1 = Z.strides + Z.strides
s = (i, j, n, n)
C = stride_tricks.as_strided(Z, shape=s, strides=Z1)
print(C[0])
```

```
[[[0 4 4]
  [0 0 1]
  [1 0 3]]
```

```
 [[4 4 1]
  [0 1 3]
  [0 3 2]]

 [[4 1 2]
  [1 3 4]
  [3 2 2]]]
```

22. Crie uma subclasse dos arrays 2D tal que Z[i,j] == Z[j,i]. A funcionalidade só precisará funcionar para arrays 2D com atribuição de valores através de índices.

```
# Author: Eric O. Lebigot

class Symetric(np.ndarray):
    def __setitem__(self, index, value):
        i,j = index
        super(Symetric, self).__setitem__((i,j), value)
        super(Symetric, self).__setitem__((j,i), value)

def symetric(Z):
    Z1 = np.asarray(Z + Z.T - np.diag(Z.diagonal()))
    return Z1.view(Symetric)

S = symetric(np.random.randint(0,10,(5,5)))
S[2,3] = 42
print(S)
```

```
[[ 9 15 17  6 12]
 [15  5  2  6  9]
 [17  2  6 42  9]
 [ 6  6 42  5 10]
 [12  9  9 10  6]]
```

23. Suponha um conjunto de matrizes com shape (n,n) e um conjunto de p vetores com shape (n,1). Como calcular a soma dos p produtos de matrizes de uma única vez, de tal forma que o resultado tenha shape (n,1).

A função a seguir irá funcionar uma vez que: - M é (p,n,n) - V é (p,n,1) - Portanto, calculando a soma ao longo dos eixos emparelhados 0 e 0 (de M e V de forma independente) e dos eixos 2 e 1, termina-se com um vetor (n,1).

```
# Author: Stefan van der Walt

p, n = 5, 10
M = np.ones((p,n,n))
V = np.ones((p,n,1))
```

```
S = np.tensordot(M, V, axes=[[0, 2], [0, 1]])
print(S)
```

```
[[50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]
 [50.]]
```

24. Suponha um array 16x16. Como obter a soma dos blocos 4x4 que compõe o array?

```
# Author: Robert Kern

Z = np.ones((16,16))
k = 4

Z0 = np.arange(0, Z.shape[0], k)
Z1 = np.arange(0, Z.shape[1], k)
S = np.add.reduceat(np.add.reduceat(Z, Z0, axis=0),
                                       Z1, axis=1)
print(S)
```

```
[[16. 16. 16. 16.]
 [16. 16. 16. 16.]
 [16. 16. 16. 16.]
 [16. 16. 16. 16.]]
```

25. Implementar o Game of Life utilizando apenas numpy arrays.

```
# Author: Nicolas Rougier

def iterate(Z):
    # Count neighbours
    N = (Z[0:-2,0:-2] + Z[0:-2,1:-1] + Z[0:-2,2:] +
         Z[1:-1,0:-2]                + Z[1:-1,2:] +
         Z[2:  ,0:-2] + Z[2:  ,1:-1] + Z[2:  ,2:])

    # Apply rules
    birth = (N==3) & (Z[1:-1,1:-1]==0)
    survive = ((N==2) | (N==3)) & (Z[1:-1,1:-1]==1)
```

```
    Z[...] = 0
    Z[1:-1,1:-1][birth | survive] = 1
    return Z

Z = np.random.randint(0,2,(50,50))
for i in range(100): Z = iterate(Z)
print(Z)
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

26. Obtenha os n maiores valores de um array.

```
Z = np.arange(10000)
np.random.shuffle(Z)
n = 5
```

Forma lenta

```
print (Z[np.argsort(Z)[-n:]])
```

```
[9995 9996 9997 9998 9999]
```

Forma rápida

```
print (Z[np.argpartition(-Z,n)[:n]])
```

```
[9999 9998 9997 9996 9995]
```

27. Dado um número arbitrário de vetores, calcule o produto cartesiano dos mesmos, isto é a combinação de todos os vetores dois a dois.

```
# Author: Stefan Van der Walt

def cartesian(arrays):
    arrays = [np.asarray(a) for a in arrays]
    shape = (len(x) for x in arrays)

    ix = np.indices(shape, dtype=int)
    ix = ix.reshape(len(arrays), -1).T

    for n, arr in enumerate(arrays):
```

```
        ix[:, n] = arrays[n][ix[:, n]]

    return ix

print (cartesian(([1, 2, 3], [4, 5], [6, 7])))
```

```
[[1 4 6]
 [1 4 7]
 [1 5 6]
 [1 5 7]
 [2 4 6]
 [2 4 7]
 [2 5 6]
 [2 5 7]
 [3 4 6]
 [3 4 7]
 [3 5 6]
 [3 5 7]]
```

28. Como criar um array de "records"(um record array) a partir de um array comum?

```
Z = np.array([("Hello", 2.5, 3),
              ("World", 3.6, 2)])
R = np.core.records.fromarrays(Z.T,
                    names='col1, col2, col3',
                    formats = 'S8, f8, i8')
print(R)
```

```
[(b'Hello', 2.5, 3) (b'World', 3.6, 2)]
```

29. Suponha dois arrays A e B de dimensões (8,3) e (2,2). Obtenha as linhas de A que contém elementos presentes em cada linha de B, independente da sua ordem em B.

```
# Author: Gabe Schwartz

A = np.random.randint(0,5,(8,3))
B = np.random.randint(0,5,(2,2))

C = (A[..., np.newaxis, np.newaxis] == B)
rows = np.where(C.any((3,1)).all(1))[0]
print(rows)
```

```
[0 1 2 3 4]
```

30. Suponha uma matriz 10x3. Obtenha as linhas que contenham apenas valores únicos.(por ex.[2,2,3])

    Solução para arrays de todos os tipos (incluindo arrays de strings e record arrays).

```
# Author: Robert Kern

Z = np.random.randint(0,5,(10,3))
print(Z)
```

```
[[2 2 1]
 [3 1 2]
 [1 3 3]
 [0 3 3]
 [1 4 1]
 [3 0 0]
 [1 1 3]
 [4 1 2]
 [4 1 2]
 [4 0 4]]
```

```
E = np.all(Z[:,1:] == Z[:,:-1], axis=1)
U = Z[~E]
print(U)
```

```
[[2 2 1]
 [3 1 2]
 [1 3 3]
 [0 3 3]
 [1 4 1]
 [3 0 0]
 [1 1 3]
 [4 1 2]
 [4 1 2]
 [4 0 4]]
```

Solução para arrays numéricos. Esta solução funciona para qualquer número de colunas em z

```
U = Z[Z.max(axis=1) != Z.min(axis=1),:]
print(U)
```

```
[[2 2 1]
 [3 1 2]
 [1 3 3]
 [0 3 3]
```

```
 [1 4 1]
 [3 0 0]
 [1 1 3]
 [4 1 2]
 [4 1 2]
 [4 0 4]]
```

31. Converta um vetor de inteiros em uma representação por matriz binária.

```
# Author: Warren Weckesser

I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128])
B = ((I.reshape(-1,1) & (2**np.arange(8))) != 0)
B = B.astype(int)
print(B[:,::-1])
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

```
# Author: Daniel T. McDonald

I = np.array([0, 1, 2, 3, 15, 16, 32, 64, 128])
I = np.array(I, dtype=np.uint8)
print(np.unpackbits(I[:, np.newaxis], axis=1))
```

```
[[0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 1 1]
 [0 0 0 0 1 1 1 1]
 [0 0 0 1 0 0 0 0]
 [0 0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
```

32. Dado um array bidimensional, como extrair linhas únicas do mesmo?

```
# Author: Jaime Fernandez del Rio

```

```
Z = np.random.randint(0,2,(6,3))
Z1 = np.ascontiguousarray(Z)
Z2 = Z.dtype.itemsize * Z.shape[1]
T = Z1.view(np.dtype((np.void, Z2)))
_, idx = np.unique(T, return_index=True)
uZ = Z[idx]
print(uZ)
```

```
[[1 0 0]
 [1 0 1]
 [1 1 0]
 [1 1 1]]
```

33. Suponha dois vetores A & B, escreva a soma de einstein (ou soma na notação de tensores) equivalente para os métodos numpy: .inner(), .outer(), .sum(), e .mul().

    Leia a referência bibliográfica: http://ajcr.net/Basic-guide-to-einsum/

```
# Author: Alex Riley

A = np.random.uniform(1,2,5)
B = np.random.uniform(1,2,5)

# np.sum(A)
print(np.einsum('i->', A))

# A * B
```

```
7.7761001287961315
```

```
print(np.einsum('i,i->i', A, B))

# np.inner(A, B)
```

```
[1.604  1.716  2.305  2.75   1.267]
```

```
print(np.einsum('i,i', A, B))

# np.outer(A, B)
```

```
9.642733771091653
```

```
print(np.einsum('i,j->ij', A, B))
```

```
[[1.604 1.683 1.886 2.766 1.675]
 [1.636 1.716 1.923 2.821 1.708]
 [1.961 2.057 2.305 3.381 2.047]
 [1.595 1.673 1.875 2.75  1.665]
 [1.214 1.274 1.427 2.093 1.267]]
```

34. A partir do caminho descrito por dois vetores (X,Y), como amostrar o mesmo utilizando apenas amostras equidistantes?

```
# Author: Bas Swinckels

phi = np.arange(0, 10*np.pi, 0.1)
a = 1
x = a*phi*np.cos(phi)
y = a*phi*np.sin(phi)

# segment lengths
dr = (np.diff(x)**2 + np.diff(y)**2)**.5
r = np.zeros_like(x)

# integrate path
r[1:] = np.cumsum(dr)

# regular spaced path
r_int = np.linspace(0, r.max(), 200)

# integrate path
x_int = np.interp(r_int, r, x)
y_int = np.interp(r_int, r, y)
```

35. Dado um inteiro n e um array 2D x, selecione de x as linhas que possam ser interpretadas como obtidas de uma distribuição multinomial com n graus de liberdade, isto é, as linhas que apenas contenham inteiros e as quais somam n.

```
# Author: Evgeni Burovski

X = np.asarray([[1.0, 0.0, 3.0, 8.0],
                [2.0, 0.0, 1.0, 1.0],
                [1.5, 2.5, 1.0, 0.0]])
n = 4
M = np.logical_and.reduce(np.mod(X, 1) == 0, axis=-1)
M &= (X.sum(axis=-1) == n)
print(X[M])
```

```
[[2. 0. 1. 1.]]
```

36. Crie um vetor aleatório de dois eixos (2D) com dimensões `[5,2]`, formado por valores inteiros de -9 a 9. Suponha que cada linha do vetor sejam as coordenadas x,y de um ponto. Calcule a matriz com as distâncias entre todos os pares de pontos.

```
np.random.seed(100)
z = np.random.randint(-9,9,(5,2))
x1 = z[:,0].reshape(-1,1); x2 = x1.T
y1 = z[:,1].reshape(-1,1); y2 = y1.T
print( np.sqrt((x1-x2)**2 + (y1-y2)**2) )
```

```
[[ 0.     12.042 10.63   6.083 12.53 ]
 [12.042  0.     10.296 13.928  5.099]
 [10.63  10.296  0.     16.125 14.56 ]
 [ 6.083 13.928 16.125  0.     12.   ]
 [12.53   5.099 14.56  12.     0.   ]]
```

37. A partir da amostra do dataset iris.target obtida com: `np.sort(np.random.choice(iris.target, size=15)` conte os números de linha por variável categórica. Para os usuários do R: estamos implementando contagem por group_by.

```
from sklearn.datasets import load_iris
iris = load_iris()
np.random.seed(100)
amostra = np.random.choice(iris.target, size=15)
species_small = np.sort(amostra)
sp_sm = species_small
print(sp_sm)
```

```
[0 0 0 0 0 1 1 1 1 1 1 1 2 2 2]
```

```
print([i for val in np.unique(sp_sm)
         for i, grp in enumerate(sp_sm[sp_sm==val])])
```

```
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 5, 6, 0, 1, 2]
```

38. Crie group ids a partir de uma variável categórica, utilizando o dataset species_small (abaixo) como exemplo.

```
species = iris.target
species_small = np.sort(np.random.choice(species, size=15))
species_small
```

```
array([0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

```
# Solucao:
unicos = np.unique(species_small)
output = [np.argwhere(unicos == s).tolist()[0][0] \
    for val in unicos \
        for s in species_small[species_small==val]]
```

```
# Solucao: For Loop version
output = []
uniqs = np.unique(species_small)
# uniq values in group
for val in uniqs:
  # each element in group
  for s in species_small[species_small==val]:
    # groupid
    groupid = np.argwhere(uniqs == s).tolist()[0][0]
    output.append(groupid)

print(output)
```

```
[0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2]
```

39. Calcule a média de uma coluna numérica agrupada por uma coluna categórica em um array 2D

```
iris_2d = iris.data
iris_rsp = iris.target.reshape(-1,1)
iris_2d = np.concatenate([iris_2d, iris_rsp], axis=1)
names = ("sepallength", "sepalwidth", "petallength", \
         "petalwidth", "species")

# Solucao
# sepalwidth
numeric_column = iris_2d[:, 1].astype("float")
# species
grouping_column = iris_2d[:, 4]
```

```
# Solucao usando compreensao de lista
gp_col = grouping_column
gp_unq = np.unique(grouping_column)
# gp_val eh~ abreviacao de group_val
sol = [[gp_val, numeric_column[gp_col==gp_val].mean()]
                                for gp_val in gp_unq]
print(np.round(sol,3))
```

```
[[0.    3.428]
 [1.    2.77 ]
 [2.    2.974]]
```

```
# Solucao usando loop for
output = []
gp_col = grouping_column
for group_val in np.unique(grouping_column):
  media = numeric_column[gp_col==group_val].mean()
  output.append([group_val, media])

print(np.round(output,3))
```

```
[[0.    3.428]
 [1.    2.77 ]
 [2.    2.974]]
```

40. Crie uma amostragem aleatória do dataset iris.data indicando quais linhas do mesmo deverão ser escolhidas, de forma que a quantidade de amostras da espécie setosa seja o dobro da quantidade das espécies versicolor e virginica.

```
np.random.seed(100)
np.set_printoptions(precision=3)
probs = np.r_[np.linspace(0, 0.500, num=50),
              np.linspace(0.501, .750, num=50),
              np.linspace(.751, 1.0, num=50)]
print(np.round(probs[:10]),2)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.] 2
```

```
index = np.searchsorted(probs, np.random.random(150))
print(index[:10])
```

```
[ 59  28  42 119   1  12  84 115  14  65]
```

```
species_out = iris.target[index]
print(species_out[:10])
```

```
[1 0 0 2 0 0 1 2 0 1]
```

```
print(np.unique(species_out, return_counts=True))
```

```
(array([0, 1, 2]), array([77, 37, 36]))
```

41. Importe uma image disponível em uma URL e converta a mesma para um numpy array. Na solução é utilizada uma URL da wikipedia.

```
from io import BytesIO
from PIL import Image
import PIL, requests

# Importação dos dados a partir da internet
URL = 'https://upload.wikimedia.org/wikipedia/"
URL = URL + 'commons/8/8b/Denali_Mt_McKinley.jpg'
response = requests.get(URL)

# Leitura dos dados como uma imagem
I = Image.open(BytesIO(response.content))

# Ajuste de tamanho
I = I.resize([150,150])

# Conversão para um numpy array
arr = np.asarray(I)

# Pode-se também fazer a conversão inversa:
#         (array -> imagem) e apresentar a mesma.
im = PIL.Image.fromarray(np.uint8(arr))
#Image.Image.show(im)
```

Capítulo 5

Funções

Neste capítulo serão exploradas as capacidades de cálculo básicas e gráficas do Python, através dos pacotes `numpy` e `sympy` e `matplotlib`. O cálculo de funções envolverá exemplos práticos e será apresentado a plotagem básica de gráficos através da linguagem `Python`.

5.1 Modelos Matemáticos

Um modelo matemático é um descrição numérica de um fenômeno real. Exemplos de modelos matemáticos seriam: a) a descrição da quantidade de pessoas na população de um país ao longo do tempo, b) a quantidade de produtos demandados por um grupo de clientes, de produtos ofertados por um conjunto de empresas e o valor do preço pelo qual ambos os grupos estarão dispostos a executar uma transação comercial e c) a taxa de retorno em um investimento a partir da qual estamos dispostos a investir no mesmo.

O processo de criação de um modelo envolve as etapas de: a) determinação das variáveis a serem estudadas, b) obtenção de valores reais para estas variáveis, c) proposição de um relacionamento entre elas, d) execução do modelo, e) análise de resultados e f) apresentação dos resultados obtidos. Neste texto estaremos interessados principalmente na formulação de modelos e na sua execução tanto de forma algébrica (manual) quanto numérica (pelo computador).

5.2 Representação Algébrica de Uma Função

Funções são expressões nas quais sobre o valor de uma variável (por exemplo x) são executadas operações e em seguida é obtido o valor de uma outra variável (por exemplo y) dizendo-se portanto que y é uma função de x e representando esta relação genericamente por $y = f(x)$.

O conceito de modelo matemático está intimamente relacionado ao de função. Em ambos os casos executamos operações matemáticas em um conjunto de valores dados de forma a obter outro conjunto de valores. Os elementos que constituem uma função $y = f(x)$ são:

1. Uma variável de entrada (também chamada de variável independente) geralmente representada por x
2. Um conjunto de operações matemáticas a serem executadas na variável de entrada x, o qual ficam representados de forma implícita no símbolo $f()$
3. Uma variável de saida (ou de resposta, também chamada de variável dependente) geralmente representada por y
4. O conjunto dos valores possíveis para a variável de entrada x é chamado de domínio da função e o conjunto dos valores possíveis para a variável de saida y é chamado de imagem da função.

5.3 Função Constante

A primeira função que iremos analisar é a função constante, por exemplo $y = f(x) = 4$. Observe que neste caso, para qualquer valor escolhido para x o valor da função não se altera. A representação gráfica da mesma é uma reta horizontal (isto é paralela ao eixo dos x, vide 5.1)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-10,10,20)
y = x - x + 4
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D object at 0x7f1695b33be0>]
```

```
plt.show()
```

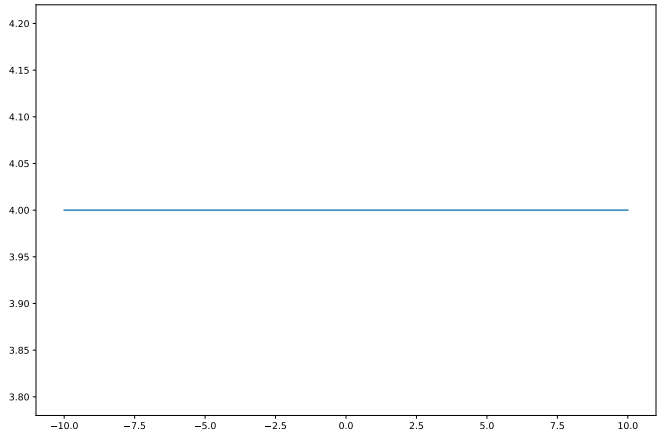


Figura 5.1: Representação Gráfica da Função Constante

5.4 Função de Primeiro Grau

A próxima função a ser analisada é a função de primeiro grau. Esta função (na verdade esta família de funções) é toda a relação entre x e y que possa ser representada por $y = m.x + p$ onde m e p são dois números dados. m é denominado o coeficiente angular da função e p seu intercepto. Como a representação gráfica deste tipo de função é sempre uma linha reta, m e p são também chamados de coeficiente angular e intercepto da reta $y = mx + p$.

Se m for positivo a reta associada terá inclinação para cima. Por exemplo $y = 3x + 5$, cuja representação gráfica pode ser vista na 5.2:

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = 3*x+5
plt.plot(x,y);
plt.show();
```

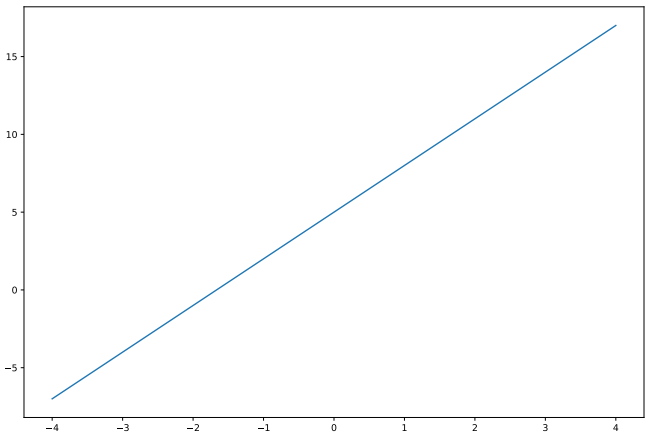


Figura 5.2: Representação Gráfica da Função y = 3x+5

Se m for negativo a reta terá inclinação para baixo. Por exemplo $y = -3x + 5$ (representação gráfica na 5.3)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = -3*x+5
plt.plot(x,y);
plt.show();
```

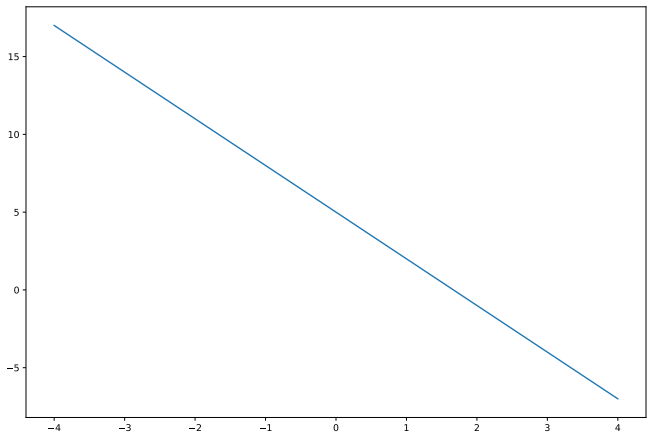


Figura 5.3: Representação Gráfica da Função y = -3x+5

Observe que se $m = 0$ a função de primeiro grau se reduz a função constante, onde o valor da constante será dado pelo valor do intercepto, como por exemplo em $y = 0.x + 4$, cujo gráfico pode ser visto na 5.4.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-4,4,100)
y = 0*x+4
plt.plot(x,y);
plt.show();
```

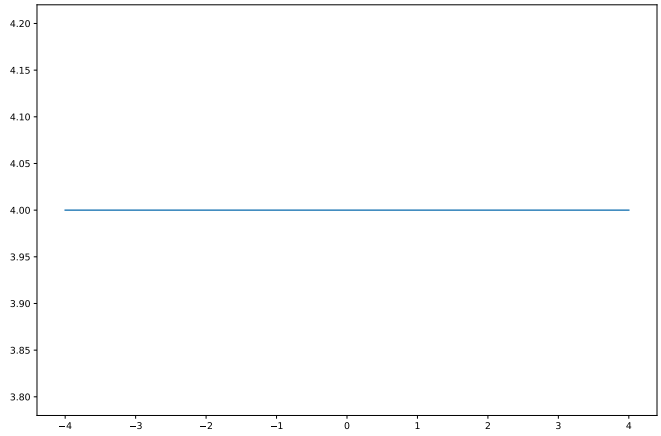


Figura 5.4: Representação Gráfica da Função y = 0.x+4

Como medida de comparação a seguir são apresentados dois conjuntos cada um com três retas de diferentes coeficientes angulares, todas com o intercepto p igual a 0 (zero). O primeiro conjunto (5.5) com três valores positivos para m e o segundo (5.6) com três valores negativos de m. Por eles percebe-se que quanto maior for o valor absoluto de m mais inclinada será a reta correspondente.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y1 = 0.5*x
y2 = 1*x
y3 = 2*x
plt.plot(x, y1);
plt.plot(x, y2);
plt.plot(x, y3);
plt.show();
```

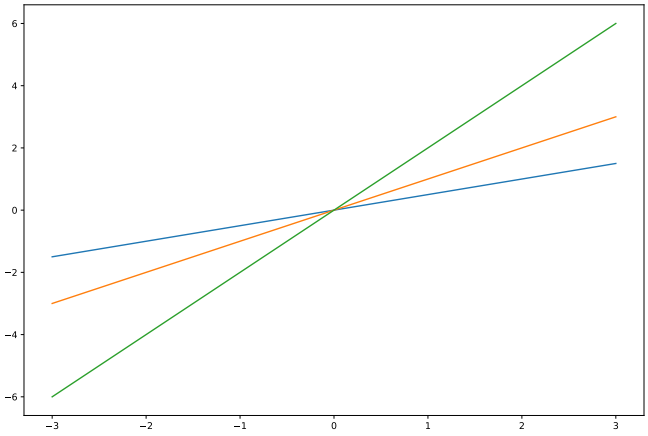


Figura 5.5: Três Retas com diferentes valores de m, todos positivos (0.5, 1, 2)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y1 = -0.5*x
y2 = -1*x
y3 = -2*x
plt.plot(x, y1);
plt.plot(x, y2);
plt.plot(x, y3);
plt.show();
```

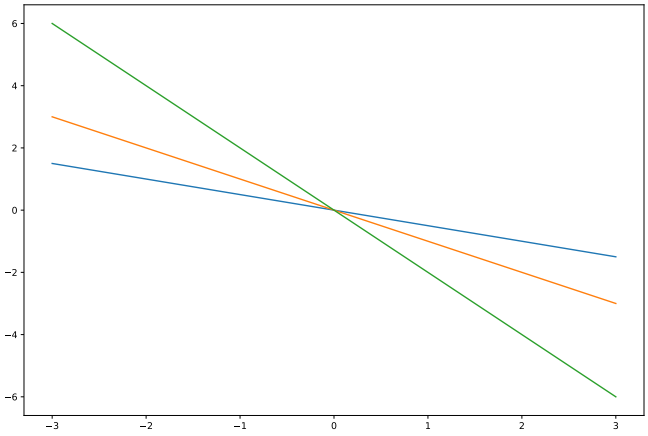


Figura 5.6: Três Retas com diferentes valores de m, todos negativos (-0.5, -1, -2)

Dados dois pontos pertencentes a mesma função de primeiro grau (x_0, y_0) e (x_1, y_1) pode-se calcular tanto o coeficiente angular m quanto o intercepto p. Para calcular o coeficiente angular m fazemos: $m = \frac{y_1 - y_0}{x_1 - x_0}$ e $p = f(0)$.

5.5 Aplicação em Economia: Função Oferta de Um Produto

Em Economia estamos interessados em determinar a relação entre a quantidade que os produtores estarão dispostos a fornecer a partir do preço unitário que eles poderão obter pelo seu produto. A partir desta informação, considere o exemplo a seguir:

1. Encontre a função de primeiro grau, a qual relaciona a quantidade ofertada pelos produtores e o preço que eles poderão obter, sabendo que quando a quantidade ofertada foi de 10 unidades o preço obtido foi de 5 unidades monetárias e quando a quantidade ofertada foi de 15 unidades o preço obtido foi de 10 unidades monetárias.

Resolução: Supomos que a quantidade seja y e o preço seja x. Sendo assim, a partir do enunciado sabemos que $y(5) = 10$ e $y(10) = 15$. Sendo assim, $m = \frac{15-10}{10-5} = 1$. Sabendo que $y = 1.x + p$ calculamos $p = 10 - 5 = 5$. Logo $y = x + 5$

5.6 Aplicação em Recursos Humanos: Salário de Um Operário

O salário é a remuneração recebida por uma pessoa pelo seu trabalho. Esta remuneração pode ser fixa (independente de qualquer outro fator) ou variável quando é dependente de um terceiro fator, por exemplo o número de horas-extras trabalhadas ao longo do mês. Com base nestas informações resolva o exemplo a seguir:

1. Um operário tem seu salário composto de duas partes: uma fixa e outra variável. A parte variável é diretamente proporcional ao número de horas-extras trabalhadas. Sabendo que no mês em que o operário trabalhou 12 horas-extras a remuneração foi de R$840,00 e no mês em que foram trabalhadas 20 horas-extras a remuneração foi de R$1000,00, determine a função linear que relacionada o número de horas-extras com o salário total do operário, o salário fixo do mesmo e o valor por hora-extra trabalhada.

Resolução: $m = \frac{f(20)-f(12)}{20-12} = \frac{1000-840}{20-12} = 20$. Sendo assim $y = m.x + p$, logo $p = y - m.x = 840 - 20.12 = 600$. Com isto chegamos à conclusão que $y = 20.x + 600$. O salário fixo portanto é de R$600,00 e cada hora-extra é remunerada com R$20,00 reais.

5.7 Função de Segundo Grau

Uma função do segundo grau é toda relação entre y e x que seja da forma: $y = a.x^2 + b.x + c$ onde $a \neq 0$. O gráfico de tal função é uma parábola. Se $a > 0$ a parábola tem concavidade para cima, tal como pode ser visto a seguir para $y = x^2 + x + 1$ na 5.7

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y = x**2 + x + 1
plt.plot(x,y);
plt.show();
```

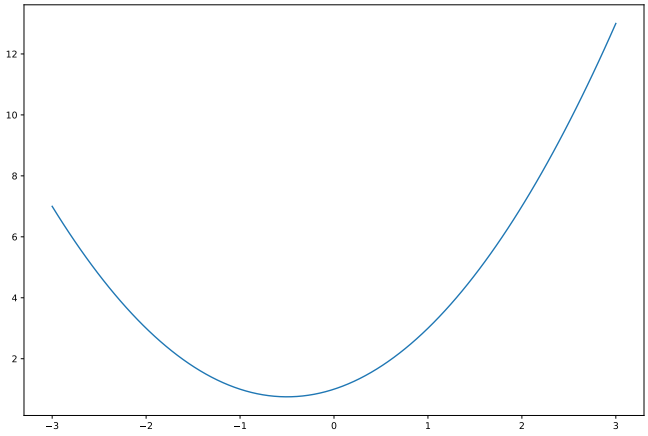


Figura 5.7: Gráfico de y = x2 + x + 1

Caso o valor de a seja menor que zero a parábola terá concavidade para baixo. Na 5.8 podemos ver o gráfico para $y = -x^2 + x + 1$.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-3,3,100)
y = -x**2 + x + 1
plt.plot(x,y);
plt.show();
```

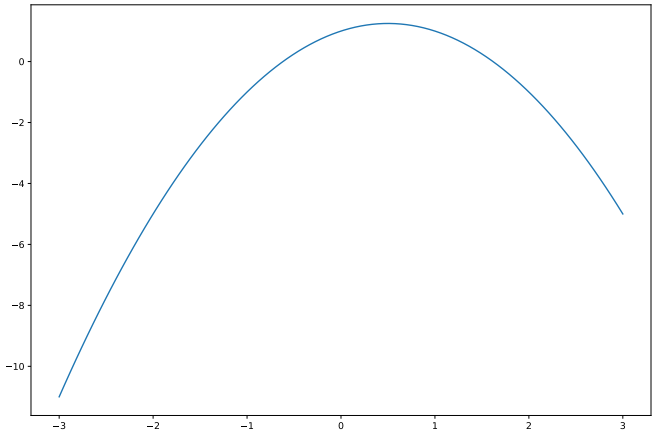


Figura 5.8: Gráfico de y = x2 + x + 1

Os valores de x que tornam $y = 0$ são chamados de raízes de função de segundo grau. Se $y = a.x^2 + b.x + c$ as raízes podem ser calculadas pela fórmula de Báskara $x_r = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$. Dependendo do valor de $\Delta = b^2 - 4ac$ as raízes podem ser ambas reais e distintas (quando $\Delta > 0$), reais e iguais (quando $\Delta = 0$) ou complexas conjugadas (quando $\Delta < 0$).

A seguir calculamos as raízes para $y = x^2 - 5x + 6$ através da fórmula de Báskara.

```
import numpy as np
a = 1; b = -5; c = 6;
d = b**2 -4*a*c
x1 = (-b + d)/(2*a)
x2 = (-b - d)/(2*a)
x1, x2
```

```
(3.0, 2.0)
```

Por último, uma parábola tem um ponto extremo que pode ser um máximo de y (se $a < 0$) ou um mínimo de y (se $a > 0$). Partindo da expressão da parábola $y = ax^2 + bx + c$ este ponto é dado por $(\frac{-b}{2a}, \frac{-\Delta}{4a})$

5.8 Aplicação em Marketing: Vendas Passadas e Futuras de Um Produto

Considere o seguinte exemplo:

1. Uma firma de materiais para escritório determinou que o numero de telefones celulares vendidos x anos a partir do ano atual é dado, aproximadamente, pela função $f(x) = 40 + 3x + 0,5\,x^2$, onde $x = 0$ corresponde ao ano corrente. Com base nestes dados determine:

   (a) O que representa $f(0)$

   (b) A quantidade de aparelhos celulares que se espera vender em seis anos.

   (c) Os valores vendidos de dois anos atrás até o valor atual, além dos valores esperados a serem vendidos em dois, quatro e cinco anos no futuro.

   (d) Apresentar estes resultados em forma gráfica

O valor da função em $x = 0$ representa o número de aparelhos vendidos no ano atual. Para obter-se o número de aparelhos telefônicos vendidos seis anos no futuro deve-se calcular $f(6)$, cujo valor neste caso é igual a: $f(6) = 40 + 3 * 6 + 0,5 * (6)^2$. Tal cálculo é feito de maneira imediata a seguir:

```
def f(x):
  f = 40 + 3*x + 0.5*(x**2)
  return(f)

f(6)
```

```
76.0
```

Para desenhar o gráfico de uma função é necessário calcular os seus valores em distintos pontos do seu domínio. Isto requer a execução em sequência de uma série de operações. O `Python` possui o recurso de executar uma operação de forma repetida para vários valores de uma variável de entrada x através da biblioteca *numpy*. A seguir utiliza-se este recurso para calcular o valor da função $f(x) = x^3 - 5x^2 + 10x - 7$, nos pontos $f(-2)$, $f(-1)$, $f(0)$, $f(2)$, $f(4)$ e $f(5)$.

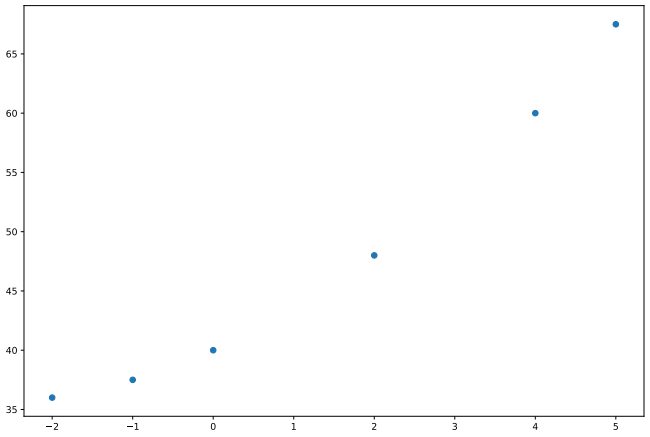
```
import numpy as np
x = np.array([-2, -1, 0, 2, 4, 5])
y = 40 + 3*x + 0.5*(x**2)
f(x)
```

```
array([36. , 37.5, 40. , 48. , 60. , 67.5])
```

Uma vez que os pares de pontos x e y tenham sido calculados é possível gerar um gráfico a partir dos mesmos conforme pode ser visto a seguir.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.array([-2, -1, 0, 2, 4, 5])
```

```
y = 40 + 3*x + 0.5*(x**2)
plt.scatter(x,y);
plt.show();
```



5.9 Aplicação em Administração em Geral: Ponto de Equilíbrio e Lucro Máximo

As funções de segundo grau são úteis pois servem para descrever o comportamento de uma função que ora aumenta ora diminui a partir de uma variável independente. Este é um comportamento típico das funções receita e custo de um produto. Devido a tal comportamento surge o conceito de ponto de equilíbrio (break-even point em inglês), o qual é definido como a quantidade que deverá ser comercializada para que o lucro total da empresa seja zero. Este conceito é utilizado para resolver o exemplo a seguir:

1. Para comercialização de um determinado produto um varejista observou que a receita oriunda da venda de q unidades do mesmo é dada pela função $R = -3q^2 + 120q$ e o custo associado com sua produção é dado por $C = +2q^2 + 20q + 375$. Com base nestas infomações pede-se:

   (a) Esboce os gráficos da receita e custo sobre o mesmo sistema de eixos, determinando e indicando os break-even points.

   (b) Indique no gráfico do item anterior as quantidades para as quais o lucro é positivo

   (c) Obtenha a função lucro e esboce o gráfico, indicando os principais pontos.

(d) Qual a quantidade de relógios a ser comercializada para que o lucro seja máximo? Qual o lucro máximo?

(e) Para quais quantidade scomercializadas o lucro é positivo? Compare com os resultados obtidos para o ponto de equilíbrio

5.10 Aplicação em Finanças: Preços de Uma Ação Negociada em Bolsa

Muitas vezes não temos uma expressão algébrica precisa para representar os pares de valores x e y de uma função (tal como no exemplo anterior, onde $y = 40 + 3x + 0.5x^2$). Neste caso podemos lançar mão de uma tabela com os pares de pontos x e y. Um exemplo pode ser visto a seguir:

1. Suponha que o preço de uma ação ao longo do tempo possa ser representado pela função: $y = f(t)$ onde t são os meses do ano corrente, de Janeiro ($t = 0$) à Dezembro ($t = 11$). Os valores da função y neste caso estão representados na tabela a seguir. Com base nestes valores pede-se:

   (a) Apresente a representação gráfica da função.

   (b) Supondo que o mês atual seja $t = 2$, qual o valor da ação três meses no futuro?

   (c) Qual o mês no qual a ação teve seu valor máximo?

   (d) Qual o mês no qual a ação teve seu valor mínimo?

   (e) Qual o mês com a menor valorização percentual? (em relação ao mês anterior)

   (f) Qual o mês com a maior valorização percentual? (em relação ao mês anterior)

   (g) Qual o valor médio da ação ao longo do ano?

   (h) Suponha que um investidor tenha comprado 100 ações da empresa por um total de R$370.00. Em qual mês, aproximadamente você acha que ele executou esta operação de compra?

```
x = [0,1,2,3,4,5,6,7,8,9,10,11]
y = [5.0, 6.5,  7.8, 3.5, 4.8, 5.3]
y = y+[8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
```

Tabela 5.1: Representação de Uma Função em Forma de Tabela

x	y
0	5.0
1	6.5

x	y
2	7.8
3	3.5
4	4.8
5	5.3
6	8.1
7	9.5
8	10.1
9	7.5
10	6.5
11	9.9

Resolução:

A representação gráfica da função pode ser vista na figura 5.9:

```
import numpy as np
from matplotlib import pyplot as plt
t = [0,1,2,3,4,5,6,7,8,9,10,11]
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
plt.scatter(t,p);
plt.plot(t,p);
plt.show()
```

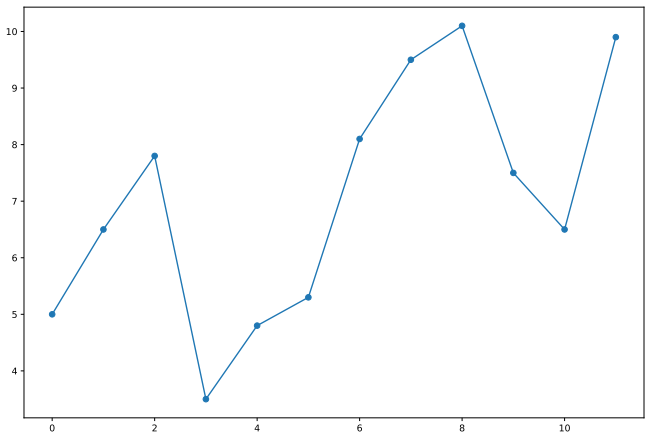


Figura 5.9: Representacao Grafica, Preco de Uma Acao Negociada em Bolsa ao Longo do Tempo

Se o mês atual é $t = 2$, o mês $t = 2 + 3 = 5$ estará três meses no futuro. Para $t = 5$ vemos pela tabela que $p = 5.3$. Podemos obter este valor diretamente do array p através de `p[5]` conforme a seguir:

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p[5]
```

```
5.3
```

Para obter os meses nos quais a ação teve seus valores máximo e mínimo utilizados os métodos `.argmax` e `argmin` (de arrays `numpy`), os quais fornecem os meses $t = 8$ (o que seria Setembro) e $t = 3$ (o que seria Abril):

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)
p.argmax(), p.argmin()
```

```
(8, 3)
```

Para obter os meses com as maiores e menores valorizações percentuais vamos primeiro calcular a variação percentual do preço da ação (a partir do mês $t = 1$) em um outro array `numpy` o qual chamaremos de d. Em seguida obtemos os valores de `.argmax` e `argmin` de `d`.

```
import numpy as np
from matplotlib import pyplot as plt
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)

p1 = p[1:] # Do 2o ao último mês
p0 = p[:-1] # Do 1o ao penúltimo mês
d = p1/p0-1
d.argmax()+1, d.argmin()+1
```

```
(6, 3)
```

A média do valor da ação pode ser obtida com `np.mean`:

```
import numpy as np
from matplotlib import pyplot as plt
```

```
p = [5.0, 6.5, 7.8, 3.5, 4.8, 5.3]
p = p + [8.1, 9.5, 10.1, 7.5, 6.5, 9.9]
p = np.array(p)
np.mean(p)
```

```
7.041666666666667
```

Por último, um investidor que tenha comprado 100 ações por um total de R$370.00, comprou cada ação ao preço unitário de R$3.70. Pela tabela percebe-se que o mês mais provável para tal aquisição é $t = 3$, o que seria Abril.

5.11 Aplicação em Física: Níveis de Energia em Um Átomo

No exemplo a seguir recomenda-se o uso de notação científica. No Python a notação científica é indicada pela letra e minúscula. Sendo assim `4.445e8` no python é igual a $4,445.10^8$. Os níveis de energia n em um átomo de hidrogênio são dados pela expressão $E_n = -\frac{m_e e^4}{8\epsilon_0^2 h^2} \cdot \frac{1}{n^2}$ onde $m_e = 9,1094.10^{-31}$ kg é a massa do elétron, $e = 1,6022.10^{-19}$ C é a carga do elétron, $\epsilon_0 = 8,8542.10^{-12}\ C^2\ s^2\ kg^{-1}\ m^{-3}$ é a permissividade elétrica do vácuo, e $h = 6,6261.10^{-34} Js$. A energia liberada quando um elétron se move de um nível n_i para um nível n_f é dada por $\Delta E = -\frac{m_e e^4}{8\epsilon_0^2 h^2} \cdot \left(\frac{1}{n_i^2} - \frac{1}{n_f^2}\right)$. Pede-se calcular os 20 primeiros níveis de energia E_n e a energia liberada quando um elétron se move de 1 a 5 níveis acima do atual.

5.12 Aplicação em Ecologia: Nível de Água em Um Reservatório

O volume final de um reservatório de água é igual ao seu volume inicial mais a quantidade de chuva (em metros cúbicos). Sabendo que o volume inicial de um reservatório é de `4.445e8` metros cúbicos e uma tempestade de verão trouxe para o mesmo um total de `5e6` metros cúbicos para o mesmo, pede-se determinar o volume final bruto da água no reservatório. Uma vez calculado este valor, deve-se realizar os ajustes descritos a seguir para a determinação do volume final líquido da água no reservatório:

1. Diminuir o valor da variável ‘chuva‘ em 10
2. Aumenter o volume do reservatório em 5
3. Diminuir o volume em 5
4. Diminuir ‘2.5e5‘ metros cúbicos devido à água que será canalizada para regiões áridas.
5. Resposta final: 447.627.500,0 metros cúbicos

5.13 Exercícios

1. Uma corretora de valores cobra uma comissão de $1,2\%$ nas operações de compra e venda de ações na faixa de \$0 a \$5000 reais. Para compras excedendo \$5000 a firma cobra 1% do total da compra mais \$10 reais. Seja x o valor negociado em ações (em reais) e por $f(x)$ a comissão cobrada em função de x. Com base nestes dados pede-se:

   (a) Montar uma tabela onde seja calculado o valor da comissão cobrada a partir do valor negociado em ações.

   (b) Calcular $f(2.000)$ e $f(10.000)$

2. Suponha que a corretora do exercício 5.13 decida manter as taxas de comissão fixas em $1,4\%$, para transações até \$6.000 e para compras de mais de 6.000 cobrar $1,5\%$ mais 15 por operação. Com base nestes dados pede-se:

   (a) Montar uma planilha onde seja calculado o valor da comissão cobrada a partir do valor do ouro comprado

   (b) Calcular $f(100)$ e $f(500)$

3. Uma pesquisa de mercado indicou que a relação entre o preço do açucar (em dólares por quilograma) e a quantidade demandada (em milhares de toneladas) em um determinado país estão relacionados pela função: $p(x) = -0,25 * x + 50$. Sabendo que o custo de produção $C(x)$ é dado pela expressão $C(x) = 2,25\,x^2 + 3\,x + 70$ pede-se:

   (a) Montar em planilha a expressão do lucro esperado em função do preço praticado, isto é $L(p)$ e do lucro esperado em função da quantidade demandada, isto é $L(x)$.

   (b) Determinar o preço p e a quantidade x que tornam a produção de açucar lucrativa.

4. Suponha que o custo total de produção de um certo produto seja dado pela função $C(x) = x^3 - 20\,x^2 + 400\,x + 300$, onde x é o número de unidades do produto fabricadas. Com base nestes dados pede-se:

   (a) Determinar o custo de produção de 10 unidades do produto

   (b) Determinar o custo de produção apenas da 10^a unidade do produto

5. Sabe-se que a quantidade média de poluentes no ar de uma determinada cidade é dada (em partes por milhão) pela expressão $c(a) = 0,5\,a + 2$ onde a é a quantidade de automóveis em circulação (em milhares de automóveis). Presumindo que a quantidade de automóveis em circulação t anos a partir da data atual obedeça a relação $a(t) = 15 + 0,15\,t^2$ pede-se:

   (a) Determinar a quantidade c de poluentes no ar, a partir do tempo t, isto é $c(t)$.

(b) Determinar quando a quantidade de poluentes no ar irá atingir $8,5$ partes por milhão.

6. Suponha que o número de ligações telefônicas necessárias (em centenas de chamadas) para atingir $x\%$ das pessoas em uma determinada cidade, seja dado pela função $Nt(x) = \frac{500\,x}{400-x}$. Com base nesta afirmação pede-se:

   (a) Determinar os valores de x para os quais a função "faz sentido", isto é o Domínio da função $Nt(x)$.

   (b) Quantas ligações telefônicas serão necessárias para atingir 50% da população?

   (c) Para atingir 100% da população, quantas ligações serão necessárias?

   (d) Qual o percentual da população da cidade que terá sido atingido quando tiverem sido efetuadas 14.000 chamadas telefônicas?

7. O **custo médio de fabricação** é definido como o custo de produção de x unidades de um produto (medido em unidades monetárias) dividido pela própria quantidade fabricada x. Supondo que um produto tenha um custo de fabricação dado pela expressão $C(x) = 180 + 3\,x$ pede-se determinar:

   (a) O custo médio de fabricação de 35 unidades

   (b) O custo médio de fabricação de 85 unidades

   (c) Para qual valor tenderá o custo médio a medida que a quantidade fabricada x aumentar?

8. O imposto de renda é um valor percentual cobrado a partir da renda mensal média do indivíduo. Suponha a seguinte composição :

   (a) Indivíduos com renda até $R\$1.499,15$ estão isentos

   (b) Indivíduos com renda de $R\$1.499,16$ até $R\$2.246,75$ pagam $7,5\%$ e descontam deste total $R\$112,43$

   (c) Indivíduos com renda de $R\$2.246,76$ até $R\$2.995,70$ pagam 15% e descontam deste total $R\$280,94$

   (d) Indivíduos com renda de $R\$2.995,71$ até $R\$3.743,19$ pagam $22,5\%$ e descontam deste total $R\$505,62$

   (e) Indivíduos com renda acima de $R\$3.743,19$ pagam 27% e descontam deste total $R\$692,78$

   Com base nestes dados pede-se :

   (a) Calcular o valor a ser pago de imposto de renda para uma pessoa com renda mensal média de $R\$3.700,00$ e o valor líquido a ser recebido

   (b) Calcular o valor a ser pago de imposto de renda para uma pessoa com renda mensal média de $R\$3.800,00$ e o valor líquido a ser recebido

(c) Com base no cálculo acima, é válida a afirmação de que por ter trocado de faixa salarial, um aumento de salário na verdade se transformou em um valor líquido a ser recebido menor?

(d) Montar em uma planilha uma função para calcular o valor a ser pago de imposto de renda e o valor líquido a ser recebido por uma pessoa tendo por base o valor mensal médio de seus recebimentos tributáveis.

9. Em uma cidade a tarifa de consumo de água obedece as regras apresentadas na tabela 5.2. Com base nos dados apresentados pede-se:

   (a) O custo total e o custo médio para quem consome 9, 18, 27 e 65 m^3

   (b) Montar em planilha uma fórmula para o cálculo do custo total e do custo médio no consumo de água.

Tabela 5.2: Tabela de Consumo de Água

Consumo M3	Tarifa
1	1
2	2
3	3
4	4
5	5

10. Um processo de fabricação segue a função $f(x) = -x^3 + 6\,x^2 + 15x$, onde x é o número de horas de processo e $f(x)$ é a quantidade de produto fabricado, medida em dezenas de quilogramas. Com base nestes dados, pede-se:

    (a) A quantidade de produto fabricada duas horas após o início do processo.

    (b) A quantidade de produto fabricada entre a 2^a e a 3^a hora.

11. Após a utilização de um terreno por uma empresa é necessária a limpeza do mesmo, através da eliminação de um determinado percentual do total de poluentes liberados no meio ambiente. Suponha que o custo da limpeza seja dado pela função $C(x) = \frac{250\,x}{300-x}$ onde x é o percentual a ser eliminado e $C(x)$ o custo de limpeza em centenas de milhares de reais. Com base nestes dados pede-se:

    (a) Qual a região de validade da função $C(x)$, isto é, qual o domínio de $C(x)$

    (b) Qual o custo de limpeza de 40% dos poluentes?

    (c) Qual o custo de limpeza dos demais 40%?

    (d) Qual o percentual que poderá ser limpo com $R\$1M$ de reais?

12. Supondo que o número de habitantes de uma certa região seja dado pela função $Np(a) = 39\,a^(0,33)$, onde a é a área em quilômetros quadrados, pede-se:

    (a) Qual a quantidade de habitantes que pode-se esperar em uma região de $10\,km^2$?

    (b) Ao triplicar-se a área, por qual valor é multiplicada a população?

    (c) Para se encontrar 10.000 pessoas, qual o tamanho da região necessário?

13. Suponha que a função demanda de uma commoditie seja dada pela expressão $Q(p) = \frac{5,437}{p^2}$ onde p é o preço por quilograma e $Q(p)$ a quantidade demandada por semana. Supondo que o preço por quilograma varie ao longo do tempo de acordo com a função $P(t) = 0,03\,t^2 + 0,25\,t + 10,9$, com o tempo t medido em semanas, determine:

    (a) A demanda semanal em função do tempo

    (b) A quantidade que será demandada em 8 semanas

    (c) Se e quando a demanda atingirá 27,356 quilogramas

14. Um cabo deverá ser estendido sobre um rio e paralelo ao mesmo, para levar energia elétrica desde um ponto de distribuição até uma fábrica. Suponha que o rio tem 800 metros de largura e a fábrica encontra-se a 4.500 metros do ponto de distribuição. O custo de estender o cabo sobre a água é de $\$5,5$ reais por metro e sobre a terra de $\$3,8$ reais por metro. Chamando de x a distância ao longo do rio, do ponto de distribuição até o ponto onde o cabo chegará na outra margem, expresse o custo de extensão total do cabo em função de x.

15. O pagamento semanal de uma vendedora depende do volume de vendas. Se ela vender x unidades de bens, então recebe y = 5x + 60 reais. Forneça uma interpretação para o coeficiente angular e para a interseção com o eixo dos y desta reta.

16. Em uma economia a função Consumo C(y), é dada por C(y) = 100 + 0,8y. Com base neste dado calcule a renda necessária para que ocorra um investimento (poupança) de 20 unidades monetárias.

17. Uma companhia de gás irá pagar para um proprietário de terra R$ 5.000,00 pelo direito de perfurar a terra para encontrar gás natural, e R$ 0,10 para cada mil metros cúbicos de gás extraído. Expresse, como função da quantidade de gás extraído, o total que o proprietário irá receber.

18. O custo para se remover o percentual de um poluente é dado pela função : $f(x) = 50x/(105 - x)$, para $0 <= x <= 100$. Qual o domínio da função? Calcule o custo para remover 0,70 do total de poluente

19. Os custos fixos de uma empresa de fabricação de tortas são o pagamento de 5 empregados a R$ 600, contas de água, luz e telefone no valor total

de R$ 500 e aluguel no valor de R$ 1,5 mil; o custo de matéria-prima e embalagem na produção da torta é de R$ 5 por torta. Com base nestas informações pede-se:

(a) Determine a função custo total C(q).

(b) Qual o preço de venda para garantir lucro de 20% em mil unidades vendidas?

(c) Suponha que o preço de venda seja R$ 15. Determine a função receita total R(q) e o ponto de equilíbrio do processo.

(d) Assumindo imposto de renda de 20% sobre o lucro total, qual a função L(q) que representa o lucro líquido?

(e) Sabendo que a empresa deseja obter lucro líquido mensal de R$ 10 mil e que no último mês foram vendidas 5 mil tortas, a meta de lucro foi atingida?

20. Uma empresa produz espátulas lâminas de raspagem com um custo unitário de R$ 2,50 cada, além de arcar com custos fixos para pagar a conta de água (R$ 50), luz (R$ 150), aluguel do galpão (R$ 250), contador (R$ 100) e ajuda de custo de cinco funcionários (R$ 90 cada) a cada mês.

    (a) Qual preço de venda equilibra os custos, considerando um mínimo de 100 espátulas vendidas por mês?

    (b) Qual o lucro mensal quando há uma demanda de 200 espátulas ao mês?

    (c) Um funcionário acidentou-se e foi demitido. O TRT-SP deu ganho de causa ao funcionário e condenou a empresa a pagar R$ 2,5 mil. Quantas espátulas precisam ser vendidas para que o lucro mensal possa cobrir esta despesa inesperada?

21. Uma empresa implementou uma política de fidelização para pedidos de grandes compradores. Fixando até o máximo de 200 produtos por pedido, o comprador pode escolher entre uma redução de R$ 0,50 no preço por peça para pedidos acima de 100 peças ou por um preço de compra unitário que varia segundo o número de peças vendidas, reduzindo 0,005 por unidade vendida (isto é, duas unidades saem R$ 2,49 cada e dez unidades saem R$ 2,45 cada). Qual a melhor estratégia, sob o ponto de vista do comprador, para pedidos de 50 peças, 100 peças, 125 peças, 150 peças e 200 peças?

22. Uma empresa de cursos online varia seus preços por hora/aula conforme a demanda por aulas. A estratégia atual, baseada na procura da semana anterior, fixa o preço em R$ 100/hora quando a demanda da semana é até 8 horas/ semana e há diminuição de R$ 10 quando a necessidade de demanda aumenta em 4 horas/aula (isto é, por exemplo, R$ 90/hora para 12 horas/semana).

(a) Qual a receita máxima que pode ser obtida, considerando um limite de 20 horas/semana de trabalho e custo zero de captação dos pedidos?

(b) Sabendo que os principais custos mensais do negócio são as contas de telefone celular (R$ 250) e fixo (R$ 50), a hospedagem do site (R$ 50), a impressão de folhas e cartões (R$ 100), as tarifas bancárias (R$ 50) e que há, ainda, receita adicional mensal de R$ 200 em publicidade, determine as funções custo, receita e lucro total semanal, encontre o ponto de equilíbrio e o número aproximado de horas/aula semanais que resulte em R$ 1 mil de lucro semanal.

23. O custo médio de produção de sucos com uma fruta é de R$ 1,25, com duas frutas é R$ 1,50 e com três frutas é R$ 2,50. Os salgados têm custo unitário de R$ 1,50. Além dos custos com matéria-prima, já informados, há custos com 8 funcionários (R$ 1,5 mil cada), aluguel no hall central (R$ 9 mil) e publicidade em jornais, revistas e sites especializados (R$ 9 mil). Estudos anteriores indicam que a empresa vende 20

    (a) A função custo total C(q).

    (b) A função receita total, R(q), admitindo que os produtos sejam vendidos pelo dobro do preço de custo.

    (c) A função lucro total L(q).

    (d) O ponto de equilíbrio das vendas, isto é, o número mínimo de sucos que seu Armando precisa vender para não ter prejuízo.

    (e) Quantos sucos devem ser vendidos para que a empresa tenha lucro mensal mínimo de R$ 12 mil, já descontados os impostos e taxas de cartão.

24. Crie uma função Python que calcule o valor da conta de água a partir do volume total (em metros cúbicos) consumidos no mês. Procure a fórmula de cálculo da conta de água no site da concessionária ou em uma conta de água da sua região.

25. Crie uma função Python que calcule o valor da conta de luz a partir do consumo total de energia elétrica. Procure a metodologia de cálculo no site da concessionária de energia elétrica de sua região.

26. Uma empresa paga a seus vendedores um salário fixo mais uma comissão sobre o total das vendas efetuadas. Até R$ 10.000,00 a comissão é de 3

27. Crie uma função Python que recebe o total do salário mensal de um funcionário e retorna o valor líquido a ser recebido descontando: 1,8

Capítulo 6

Geometria Analítica

Vamos agora extender os conceitos apresentados no capítulo sobre Funções, aprofundando as formas de plotagem de gráficos e revisando conceitos de geometria analítica. A plotagem dos gráficos irá desde as idéias básicas da Geometria Analítica (como pontos e coordenadas) até gráficos em três dimensões.

6.1 Pontos e Coordenadas

A idéia básica da Geometria Analítica esta centrada na capacidade de expressar o posicionamento de pontos, retas, figuras planas e espaciais em geral em termos de um sistema de coordenadas. Este sistema de coordenadas no plano é formado por duas retas perpendiculares entre sí, sendo o seu cruzamento denominado como origem. Um ponto é então definido pelas distâncias do mesmo até estas retas. As demais figuras geométricas são definidas pelas distâncias de cada um de seus pontos até tais retas, as quais são denominadas de *eixos coordenados*. O eixo horizontal é chamado de eixo dos X e o eixo vertical de eixo dos Y.

1. Desenhe os pontos do plano com o seguinte conjunto de coordenadas: (-2; 3), (4; 2) and (-4;-1).

Como pode ser visto na tabela 6.1 e na figura 6.1, os pontos podem ser desenhados diretamente através de arrays `numpy` e da biblioteca `matplotlib`.

```
import numpy as np
x = [-2, 4, -4]; y = [3, 2, 1]

import pandas as pd
df = pd.DataFrame({"x":x, "y":y});

from matplotlib import pyplot as plt
graf = plt.scatter(x, y);
```

```
plt.show();
```

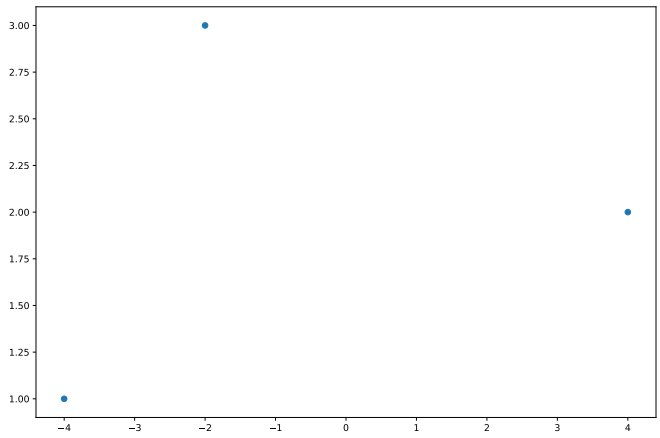


Figura 6.1: Pontos no plano com coordenadas (-2; 3), (4; 2) and (-4;-1)

Tabela 6.1: Tabela para Desenho de Pontos no Plano

x	y
-2	3
4	2
-4	1

6.2 Distância entre Pontos

A distância entre dois pontos de coordenadas (x_1, y_1) e (x_2, y_2) no plano é calculada pela fórmula: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

1. Calcule a distância entre os pontos $A = (1, 2)$ e $B = (4, 6)$

```
import numpy as np
A = np.array([[1],[2]])
B = np.array([[4],[6]])
d = np.sqrt((A[0,0]-B[0,0])**2 + (A[1,0]-B[1,0])**2)
d
```

```
5.0
```

6.3 Regiões no Plano xy

Regiões no Plano XY é o conjunto de pontos que satisfaz uma determinada condição. A seguir são apresentados exemplos de regiões no Plano XY.

1. Desenhe no plano XY as regiões definidas por:
   (a) $(x, y)|x >= 0$
   (b) $(x, y)|y = 1$
   (c) $(x, y)||y| < 1$

6.4 Triângulos

A criação de figuras geométricas formadas por segmentos de retas (polígonos) também pode ser feita no Python, bastando para isso utilizar o método `.plot` ao invés do método `.scatter`.

2. Desenhe os triângulo formado pelos pontos (0;0), (4;0) e (4;3).

Como pode ser facilmente percebido pelas células abaixo, a ligação de pontos com linhas retas produz o triângulo desejado (vide figura 6.2). Deve-se salientar que o ponto (0;0) foi fornecido duas vezes (no início e no final dos arrays `numpy`), para que o triângulo fosse "fechado."

Pode-se criar a sequência de valores `x` e `y` de forma separada como mostrado no código que gerou a figura 6.2:

```
import numpy as np
x = [0,4,4,0]; y = [0,0,3,0]

from matplotlib import pyplot as plt
graf = plt.plot(x,y);
plt.show();
```

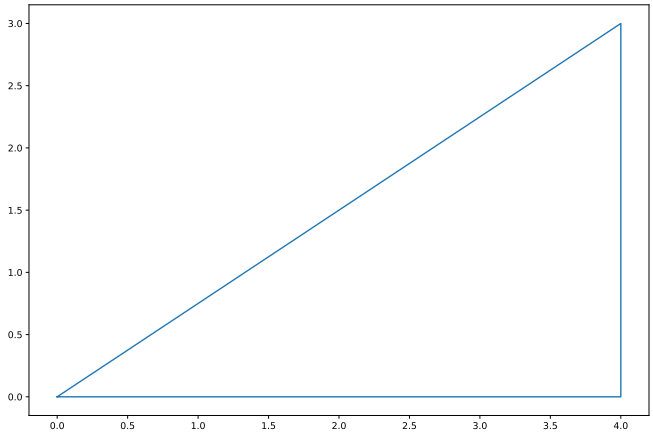


Figura 6.2: Triângulo formado pelos pontos (0;0),(4;0) e (4;3), a partir de arrays numpy 1D

Ou criar os pontos em separado e "empilha-los" formando um array 2D. As colunas `x` e `y` correspondentes são então geradas a partir da primeira e segunda colunas deste array 2D, conforme mostrado na figura 6.3.

```
import numpy as np
from matplotlib import pyplot as plt
x1 = np.array([0,0]); x2 = np.array([4,0]); x3 = np.array([4,3])
ptos = np.array([x1,x2,x3,x1]); x = ptos[:,0]; y = ptos[:,1]
graf = plt.plot(x,y);
plt.show(block=graf);
```

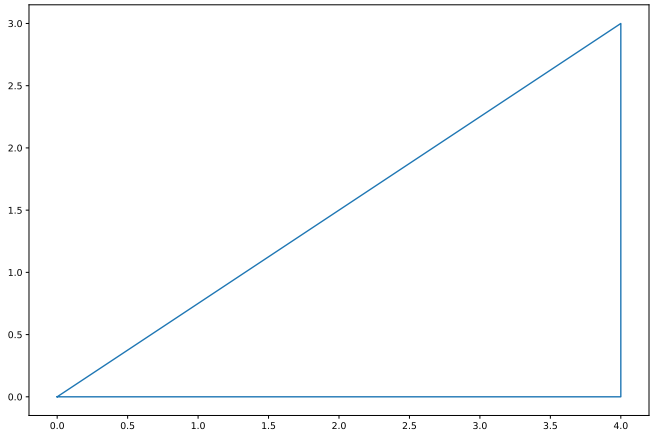


Figura 6.3: Triângulo formado pelos pontos (0;0),(4;0) e (4;3), a partir de arrays numpy 2D

A terceira forma de gerar o gráfico é criar um data frame pandas onde cada coluna é composta de um array `numpy` e mostrar os valores a partir das colunas deste data frame. Esta forma pode ser vista na figura 6.4, cujo desenho foi criado a partir dos dados na tabela 6.2

```
import numpy as np
x = [0,4,4,0]; y = [0,0,3,0]

import pandas as pd
df = pd.DataFrame({"x":x, "y":y})

from matplotlib import pyplot as plt
graf = plt.plot(df['x'],df['y']);
plt.show();
```

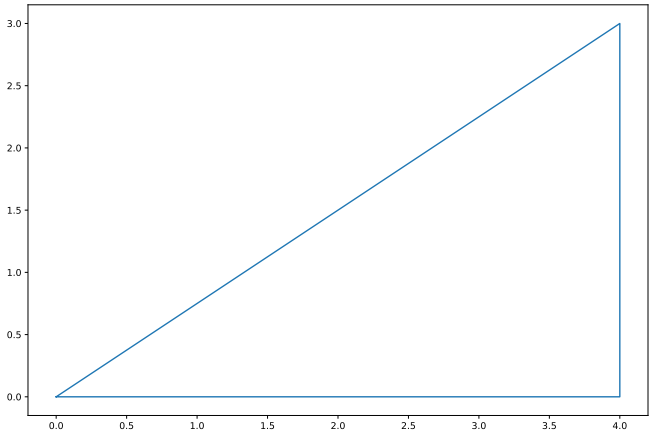


Figura 6.4: Triângulo formado pelos pontos (0;0),(4;0) e (4;3), a partir de um pandas dataframe

Tabela 6.2: Tabela para desenho do triângulo com os pontos (0;0),(4;0) e (4;3)

x	y
0	0
4	0
4	3
0	0

3. A partir do triângulo formado pelos pontos (0;0),(4;0) e (4;3), desenhe as medianas do mesmo, mostrando que elas se encontram em um único ponto (também chamado de "baricentro").

O ponto médio de um segmento é determinado pela média das coordenadas dos pontos localizados nas extremidades do mesmo. Aplicando esta regra para cada um dos lados do triângulo obteremos três pontos médios. Ligando estes pontos às extremidades opostas obtemos as medianas (figura 6.5).

```
import numpy as np
import sympy as sp
x, y = sp.symbols("x y")

from matplotlib import pyplot as plt
import matplotlib as mpl
```

```
A = np.array([0,0]); B = np.array([4,0]); C = np.array([4,3])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

reta = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = reta(A,B); g = reta(B,C); g = reta(C,A)
g = reta(A,BC); g = reta(B,AC); g = reta(C,AB)
plt.show()
```

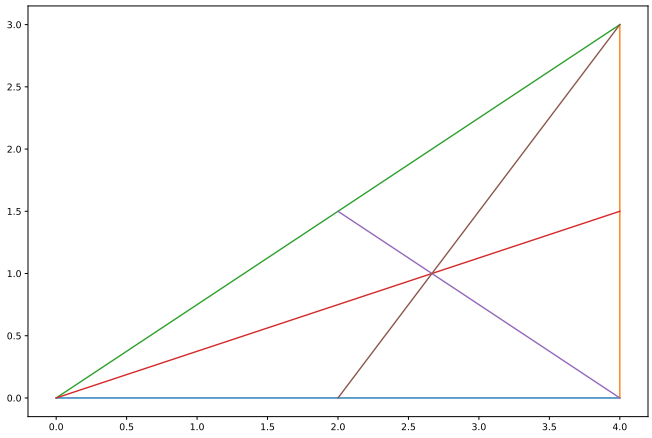


Figura 6.5: Ponto de Encontro das Medianas de Um Triangulo

6.5 Circunferências

Circunferências são curvas cujos pontos estão a uma distância constante de um ponto denominado centro. As curvas em geral podem ser expressas na forma retangular, (o que no caso da circunferência seria $R^2 = x^2 + y^2$), ou na forma polar na qual as coordenadas x e y são substituidas por uma distância até a origem, normalmente representada pela letra grega ρ (rho) e pelo ângulo formado pela reta anterior e o eixo horizontal, normalmente representada pela letra grega theta θ. A relação entre as coordenadas retângulares x e y e as coordenadas polares ρ e θ é dada por:

$$x = \rho * \cos(\theta) \tag{6.1}$$

$$y = \rho * \sin(\theta) \tag{6.2}$$

As coordenadas polares são úteis na criação de gráficos os quais a função assume dois valores distintos para o mesmo valor de x, os quais seriam de representação difícil no formato $y = f(x)$. O exemplo mais simples neste caso é o desenho de uma circunferência.

4. Desenhar a circunferência com centro (2,3) e raio $r = 4$.

Primeiro devemos montar as componentes x e y em função das coordenadas polares ρ e θ. Para este caso teremos:

$$x = 2 + 4 * \cos(\theta) \tag{6.3}$$
$$y = 3 + 4 * \sin(\theta) \tag{6.4}$$

Onde θ irá variar de 0 a 360 graus.

A figura formada pelo gráfico de dispersão descrito em 6.5 pode ser vista na figura 6.6:

```
import numpy as np
from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use("TkAgg")

r = 4
theta = np.linspace(0,360,361)
x = r*np.cos(theta)
y = r*np.sin(theta)
g = plt.scatter(x,y, s=1);
plt.show();
```

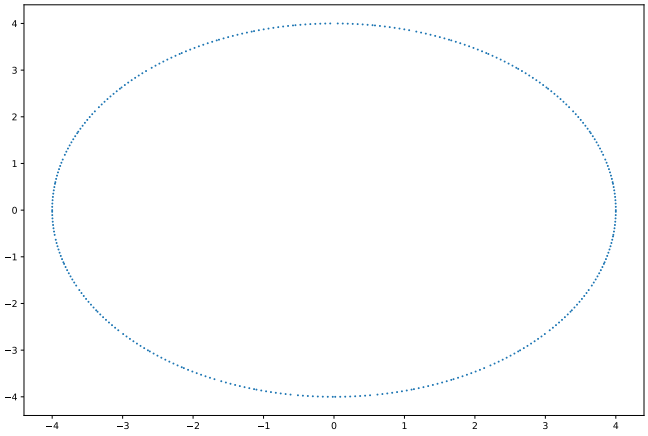


Figura 6.6: Circunferência com centro em (2,3) e raio igual a 5

5. A partir dos pontos (3;0) e (0;4) desenhar :
   (a) Um triângulo equilátero, com comprimento do lado igual a disância entre os pontos (3;0) e (0;4).
   (b) As alturas de cada lado, mostrando que elas se encontram em um único ponto denominado ortocentro.
   (c) A circunferência inscrita ao triângulo e que passa pelos cruzamentos das alturas com os lados do triângulo.

A distância d entre os pontos (3;0) e (0;4) é dada por $d = \sqrt{(3-0)^2 + (0-4)^2} = 5$. Queremos portanto construir um triângulo equilátero de lado igual a cinco, com vértices nos pontos (3;0) e (0;4).

Precisamos agora determinar as coordenadas (x, y) do terceiro vértice do triângulo. Sabendo que o triângulo é equilátero, este ponto estará a uma distância igual a 5 tanto do ponto (3;0) quanto do ponto (0;4). Sendo assim podemos resolver o seguinte sistema de equações:

$$(x-0)^2 + (y-4)^2 = 5^2 \tag{6.5}$$
$$(x-3)^2 + (y-0)^2 = 5^2 \tag{6.6}$$

```
import sympy as sp
x, y = sp.symbols("x y")
eq1 = (x-0)**2 + (y-4)**2 - 5**2
eq2 = (x-3)**2 + (y-0)**2 - 5**2
```

```
sols = sp.solve([eq1, eq2],[x,y])
print(sols[0])
```

```
(3/2 - 2*sqrt(3), 2 - 3*sqrt(3)/2)
```

```
print(sols[1])
```

```
(3/2 + 2*sqrt(3), 2 + 3*sqrt(3)/2)
```

Temos duas soluções, uma para cada lado que desenharmos o triângulo. Adotando a solução no primeiro quadrante, teremos $(x; y) = (\frac{3}{2} + 2\sqrt{3}; 2 + \frac{3\sqrt{3}}{2})$. Uma vez que o triângulo é equilátero, as alturas e medianas coincidem. Podemos então utilizar um esquema similar ao adotado no desenho da figura 6.5 para desenhar as medianas. O triângulo resultante e suas medianas pode ser visto na figura 6.7.

```
import numpy as np
A = np.array([0,4]); B = np.array([3,0]); C = np.array(sols[1])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

reta = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = reta(A,B); g = reta(B,C); g = reta(C,A)
g = reta(A,BC); g = reta(B,AC); g = reta(C,AB)
plt.show();
```

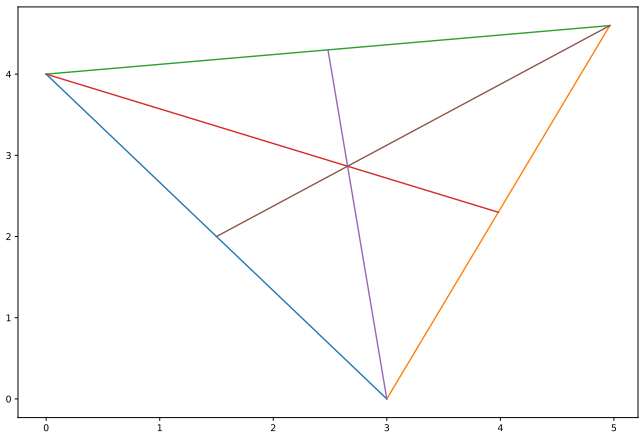


Figura 6.7: Triângulo Equilátero com vértices em (0;4) e (3;0)

Para desenhar a circunferência inscrita precisamos dos pontos AB, BC, CA. Com estes três pontos, podemos encontrar a equação da circunferência inscrita e em

seguida desenhar seu gráfico. Observe a seguir que primeiro determinamos o centro da circunferência e o seu raio, resolvendo um sistema de equações. De posse do centro e do raio da circunferência vamos desenhar o seu gráfico em duas etapas. Primeiro calculamos as coordenadas de uma circunferência com centro na origem. Vamos fazer isso utilizando coordenadas polares e convertendo os resultados para coordenadas retangulares. Com as coordenadas x e y de uma circunferência centrada na origem, vamos acrescentar os valores das coordenadas do centro da circunferência tanto a x quanto a y e assim obter a sequência de pontos a serem desenhados.

```
import numpy as np
A = np.array([0,4]); B = np.array([3,0]); C = np.array(sols[1])
BC = (B + C)/2; AC = (A + C)/2; AB = (A + B)/2

reta = lambda x,y : plt.plot([x[0],y[0]],[x[1],y[1]])
g = reta(A,B); g = reta(B,C); g = reta(C,A)
g = reta(A,BC); g = reta(B,AC); g = reta(C,AB)

import sympy as sp
x, y, R = sp.symbols("x y R", real=True)

Eq1 = (x-AB[0])**2 + (y-AB[1])**2 - R**2
Eq2 = (x-BC[0])**2 + (y-BC[1])**2 - R**2
Eq3 = (x-AC[0])**2 + (y-AC[1])**2 - R**2
xc,yc,Rc = sp.solve([Eq1, Eq2, Eq3],[x,y,R])[1]
theta = np.linspace(0,2*np.pi,100)
x1 = Rc*np.cos(theta) + xc
y1 = Rc*np.sin(theta) + yc
g = plt.plot(x1, y1);
plt.show();
```

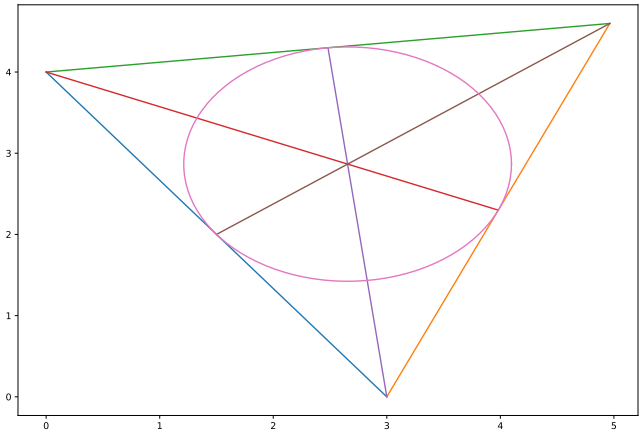


Figura 6.8: Circunferência Inscrita no Triângulo Equilátero

6.6 Trigonometria

Para além do estudo das propriedades dos triângulos existem as funções cuja variável independente é um valor de ângulo. Os ângulos são medidos neste caso em um círculo de raio = 1 chamado de círculo trigonométrico. As unidades de medida dos ângulos podem ser graus (neste caso um giro completo no círculo trigonométrico equivale a 360o) ou radianos (quando um giro completo equivale a 2π radianos).

Sendo assim um radiano equivale aproximadamente $\frac{180}{\pi} \approx 57,3^o$. Observe que para converter de graus para radianos utilizamos a relação: $graus = \frac{180}{\pi}.radianos$ e para converter de radianos para graus fazemos: $radianos = \frac{\pi}{180}.graus$

Dado um triângulo retângulo ABC (vide 6.9), com o ângulo θ formado pelo encontro dos lados AB e AC, onde AB é denominado cateto adjacente ao ângulo θ, BC é o cateto oposto ao ângulo θ e AC é a hipotenusa, as funções trigonométricas mais importantes são definidas da seguinte forma:

```
import numpy as np
from matplotlib import pyplot as plt
plt.plot([0,4],[0,0], color='blue'); #AB
```

```
[<matplotlib.lines.Line2D object at 0x7f16926e32b0>]
```

```
plt.plot([4,4],[0,3], color='green'); #BC
```

```
[<matplotlib.lines.Line2D object at 0x7f16926e3a60>]
```

```
plt.plot([0,4],[0,3], color='red'); #AC
```

```
[<matplotlib.lines.Line2D object at 0x7f16926e3e20>]
```

```
plt.show();
```

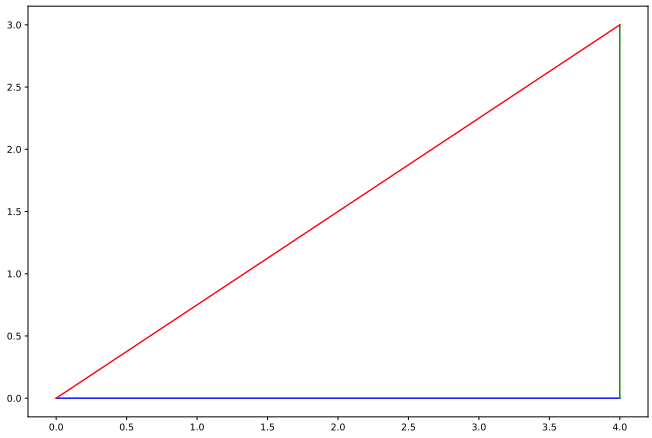


Figura 6.9: Triângulo Retângulo ABC, com catetos AB (azul) e BC (verde) e hipotenusa AC (vermelho)

1. $cos(\theta) = \frac{azul}{vermelho}$
2. $sin(\theta) = \frac{verde}{vermelho}$
3. $tg(\theta) = \frac{verde}{azul}$

Com estas funções podemos definir suas inversas:

1. $sec(\theta) = \frac{1}{cos(\theta)}$
2. $cossec(\theta) = \frac{1}{sin(\theta)}$
3. $cotg(\theta) = \frac{1}{tg(\theta)}$

E as relações básicas entre as funções trigonométricas:

1. $sin(\theta)^2 + cos(\theta)^2 = 1$
2. $1 + tg(\theta)^2 = sec(\theta)^2$

3. $1 + cotg(\theta)^2 = cossec(\theta)^2$

Em seguida temos as relações para o seno e cosseno da soma de dois angulos α e β:

1. $sin(\alpha + \beta) = sin(\alpha).cos(\beta) + sin(\beta).cos(\alpha)$
2. $cos(\alpha + \beta) = cos(\alpha).cos(\beta) - sin(\beta).sin(\alpha)$

Por último podemos juntar a relação básica entre seno e cosseno e a soma dos cossenos para obter duas relações muito úteis (vide a seguir):

1. $cos^2(\alpha) + sin^2(\alpha) = 1$
2. $cos^2(\alpha) - sin^2(\alpha) = cos(2\alpha)$
3. A soma das relações acima nos dá: $cos^2(\alpha) = \frac{1+cos(2\alpha)}{2}$
4. A diferença das relações acima nos dá: $sin^2(\alpha) = \frac{1-cos(2\alpha)}{2}$

6.7 Funções Trigonométricas

As funções $sin(x)$, $cos(x)$ e $tg(x)$ apresentam os gráficos conforme a seguir: (seno na 6.10, cosseno na 6.11 e tangente na 6.12). Todas elas tem um periodo de 2π radianos.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,+np.pi,1000)
y = np.sin(x)
plt.plot(x,y);
plt.show();
```

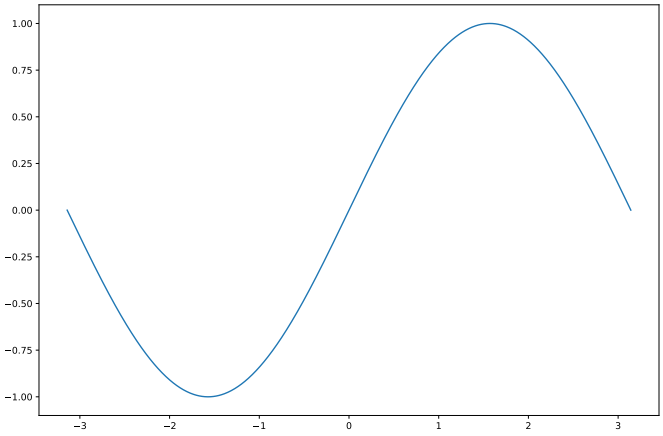


Figura 6.10: Gráfico de f(x)=sin(x)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,np.pi,1000)
y = np.cos(x)
plt.plot(x,y);
plt.show();
```

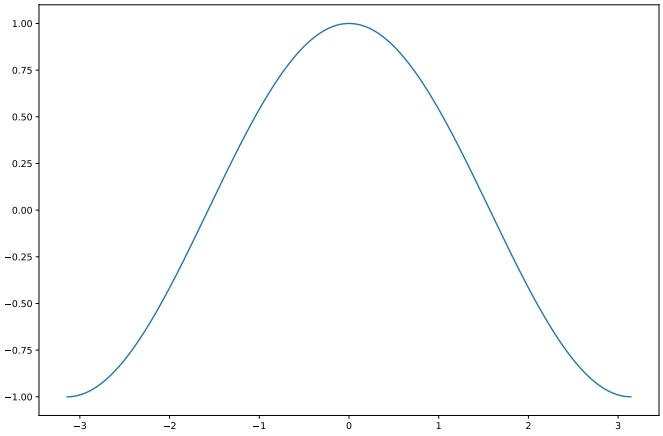


Figura 6.11: Gráfico de f(x)=cos(x)

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(-np.pi,+np.pi,1000)
y = np.tan(x)
plt.scatter(x,y);
plt.show();
```

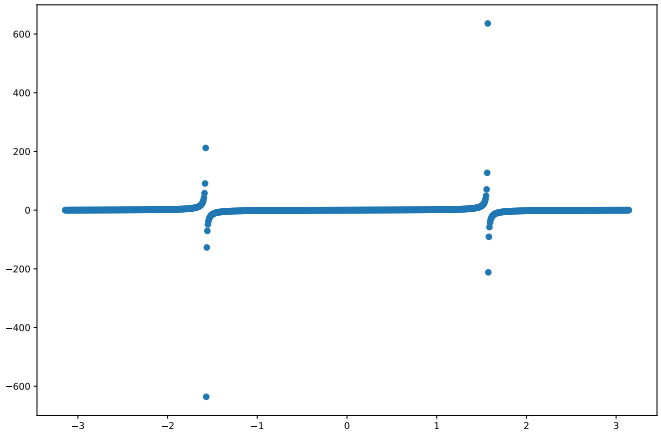


Figura 6.12: Gráfico de f(x)=tg(x)

6.8 Curvas em Coordenadas Polares

No exercício apresentado na 6.8 utilizamos o recurso das coordenadas polares. Este recurso transforma um gráfico que seria difícil de desenhar em coordenadas retangulares (ainda mais pelo fato do mesmo ser composto de duas ou mais funções de x e y, quando expressas de forma algébrica e não uma só) em uma função de variável única a partir da variável θ, o ângulo formado pela linha que vai do ponto em questão até a origem. Nesta seção iremos desenhar algumas destas curvas e incorporar recursos gráficos que permitam a determinação de cada ponto da curva.

5. Desenhar para θ variando de 0^o até 360^o, a espiral de Arquimedes dada pela expressão $r = \frac{\theta}{2}$

Na figura 6.13 pode ser visto o gráfico correspondente. Deve-se atentar para o fato do argumento (i.e. o ângulo) ser sempre passado para as funções correspondentes em radianos, por isso a multiplicação por π e a divisão por 180.

```
import numpy as np
theta = np.linspace(0,360,361)*2*np.pi/180
r = theta/2
x = r*np.cos(theta)
y = r*np.sin(theta)

from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use('TkAgg')
```

```
g = plt.plot(x,y);
plt.show();
```

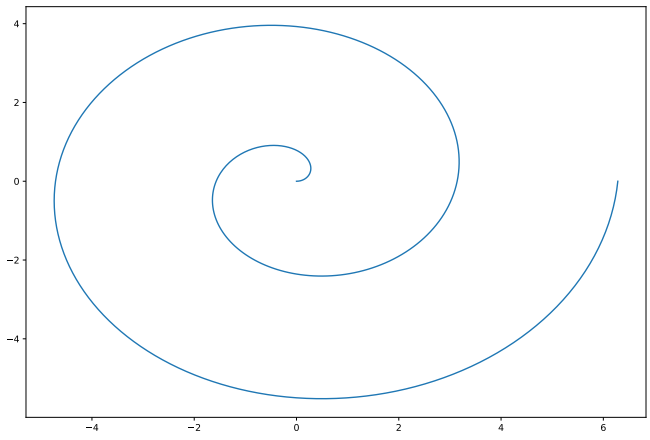


Figura 6.13: Desenho da espiral de Arquimedes

6. Desenhar a função $r = 10\sin(3\theta)$, também conhecida por *rosa de três pétalas*.

Vide figura 6.14

```
import numpy as np
theta = np.linspace(0,360,361)*np.pi/180
r = 10*np.sin(3*theta)
x = r*np.cos(theta)
y = r*np.sin(theta)

from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use('TkAgg')
g = plt.plot(x,y);
plt.show();
```

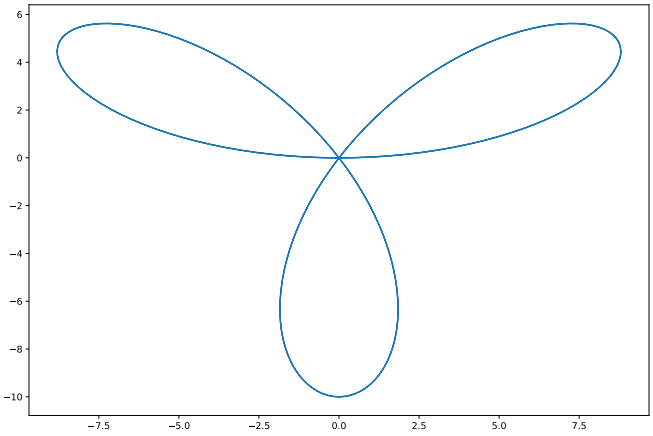


Figura 6.14: Desenho da Rosa de Três Pétalas

7. Desenhar a figura de *Lissajous* dada parametricamente pelas expressões :

$$\begin{aligned} x &= sin(2 * t) \\ y &= sin(3 * t) \end{aligned} \tag{6.7}$$

As figura 6.15 apresenta o gráfico associado. Observe que a indicação do ponto da curva foi feita criando-se uma nova série de dados, com apenas dois pontos, um deles fixo em (0;0) e outro que é calculado de acordo com o parâmetro t, o qual pode ser modificado livremente.

```
import numpy as np
t = np.linspace(0,360,361)*np.pi/180
x = np.sin(2*t)
y = np.sin(3*t)

from matplotlib import pyplot as plt
import matplotlib as mpl
mpl.use('TkAgg')
g = plt.plot(x,y);
plt.show();
```

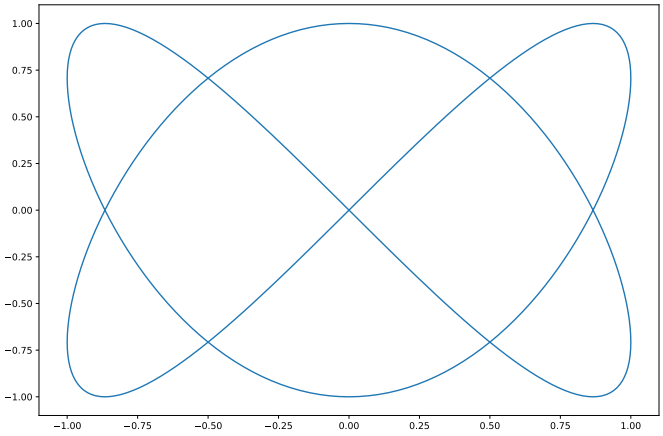


Figura 6.15: Desenho da figura de Lissajous

8. Desenhar o gráfico do caminho percorrido por um satélite que orbita um planeta com período orbital de 1 mês, planeta este que orbita uma estrela com período de 1 ano (ou de 12 meses). Supor as órbitas circulares e que o raio da órbita do satélite é 20% do raio da órbita do planeta.

Temos aqui um movimento circular composto de duas rotações. Sendo assim, a cada instante a posição do satélite é composta por um círculo de raio igual a 1 mais um círculo com raio igual a 0,1. Sendo assim podemos decompor as coordenadas x e y da órbita planetária como:

$$\begin{aligned} x &= x_p + x_s \\ y &= y_p + y_s \end{aligned} \tag{6.8}$$

Ao aplicarmos as fórmulas de movimento circular em coordenadas polares, estas expressões transformam-se em :

$$\begin{aligned} x &= 10\cos(2\pi.1.t) + 1\cos(2\pi.12.t) \\ y &= 10\sin(2\pi.1.t) + 1\sin(2\pi.12.t) \end{aligned} \tag{6.9}$$

As expressões 6.9 podem ser calculadas nas colunas de um data frame. A atenção que deve ser dada diz respeito ao parâmetro t. Ele deve variar de 0 a 1 para que o argumento do cos e do sin varie de 0 a 2π. Para criar uma curva bem suave,

utilizaremos 1.001 pontos entre 0 e 1 para o parâmetro t. O gráfico e a tabela correspondente podem ser vistos na 6.16 e na 6.3.

```
import numpy as np
t = np.linspace(0,1,1001)
xp = 10*np.cos(2*np.pi*t*1)
yp = 10*np.sin(2*np.pi*t*1)
xm = 2*np.cos(2*np.pi*t*12)
ym = 2*np.sin(2*np.pi*t*12)
x = xp + xm; y = yp + ym
df = pd.DataFrame({"x":x, "y":y})
g = plt.plot(x, y);
plt.show();
```

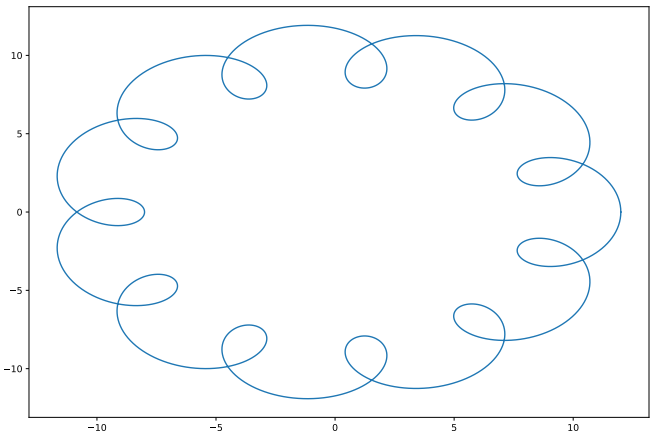


Figura 6.16: Gráfico da órbita de um satélite de um planeta, visão a partir da estrela

Tabela 6.3: Tabela de cálculo dos pontos da órbita de um satélite

x	y
12.00000	0.0000000
11.99412	0.2134851
11.97651	0.4261116
11.94728	0.6370259
11.90657	0.8453841
11.85462	1.0503567

6.9 Gráficos em 3D

A forma mais prática de desenho de superfícies em 3D é através da `sympy`. Para tal, podemos utilizar uma função de duas variáveis independentes, cada uma com seu intervalo de cálculo. Vejamos um exemplo:

9. Desenhar o gráfico da função $z = x.y$ no intervalor $x \in [0, 10]$ e $y \in [0, 10]$.

```
import numpy as np
import sympy as sp
from sympy.plotting import plot3d

x, y = sp.symbols("x y")
g = plot3d(x*y, (x,0,10), (y,0,10));
```

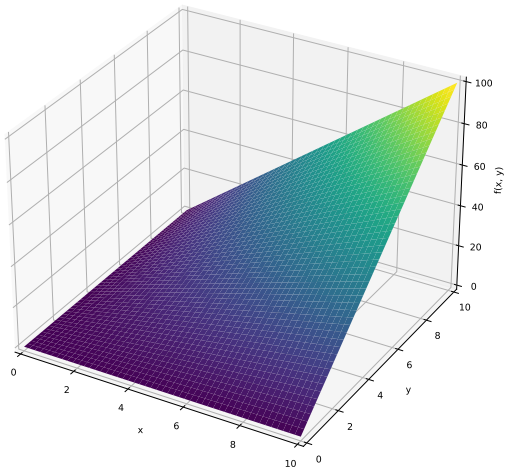


Figura 6.17: Gráfico de z = x.y

Outro gráfico interessante de ser gerado é o das linhas de contorno. Ele nada mais é que uma visão de cima, com as alturas pintadas de cores distintas. É a forma padrão dos mapas em um Atlas de geografia. Tal gráfico para a função $z = x.y$ pode ser visto na figura 6.18. Observe que primeiro criamos os limites de plotagem em cada eixo (x e y). Em seguida geramos o grid de desenho e depois a função Z a partir dos valores do grid. Com os valores dos grids para X, Y e Z, utilizamos a função `plt.contourf` para obter o efeito desejado.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0,10,101)
y = np.linspace(0,10,101)
```

```
X,Y = np.meshgrid(x,y)
Z = X*Y
g = plt.contourf(X, Y, Z);
plt.show()
```

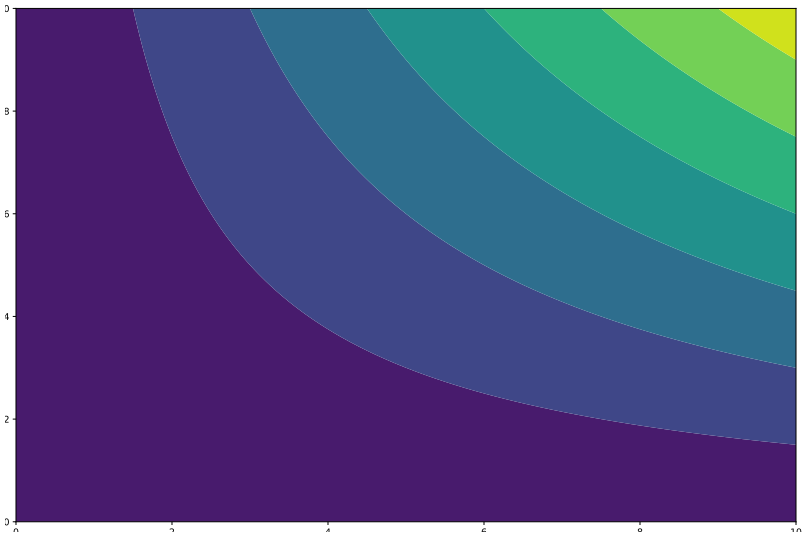


Figura 6.18: Gráfico de Contorno para z = x.y

10. Desenhar o gráfico da função $z(r) = \frac{sin(r)}{r}$ onde $r = \sqrt{x^2 + y^2}$ no intervalo $x \in [-6,28; 6,32]$ e $y \in [-6,26; -6,32]$.

```
import sympy as sp
from sympy.plotting import plot3d
x, y = sp.symbols("x y")
r = sp.sqrt(x**2 + y**2)
z = sp.sin(r)/r
g = plot3d(z, (x,-6.28, 6.32), (y,-6.28, 6.32));
```

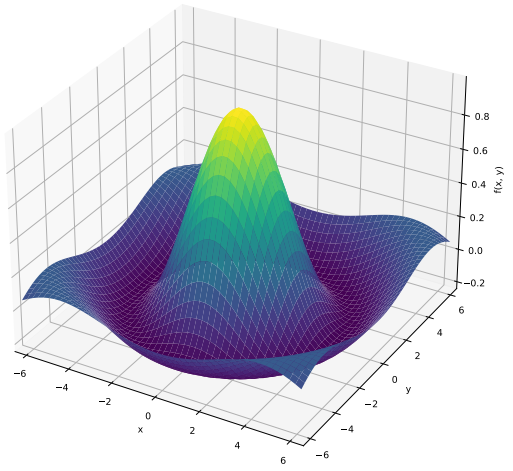


Figura 6.19: Gráfico de $z(r) = \frac{sin(r)}{r}$ onde $r = \sqrt{x^2 + y^2}$

11. Desenhar o gráfico da função $f(t) = 3e^{-4t}\cos(5t) - 2e^{-3t}\sin(2t) + \frac{t^2}{t+1}$ no intervalor $x \in [0, 4]$

```
import sympy as sp
from sympy.plotting import plot
t = sp.symbols("t")
f=3*sp.exp(-4*t)*sp.cos(5*t)-2*sp.exp(-3*t)*sp.sin(2*t)+t**2/(t+1)
g = plot(f, (t,0,4))
```

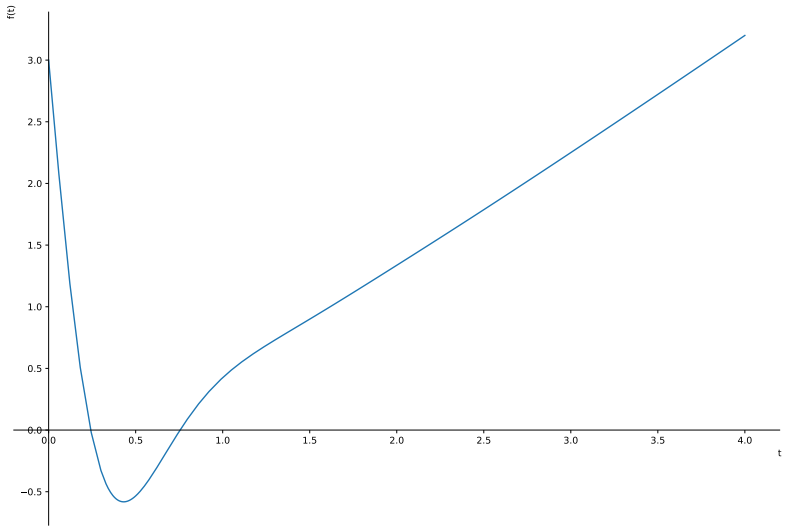


Figura 6.20: Gráfico da função $f(t) = 3 * e^{-4t}\cos(5t) - 2e^{-3t}\sin(2t) + \frac{t^2}{t+1}$ no intervalor $x \in (0, 4)$

12. Desenhar o gráfico da função $z = -2x^3 + x + 3y^2 - 1$ no intervalor $x \in [-10, 10]$ e $y \in [-10, 10]$.

```
import sympy as sp
from sympy.plotting import plot3d
x, y = sp.symbols("x y")
z=-2*x**3+x+3*y**2-1
g = plot3d(z, (x,-10,10), (y,-10,10));
```

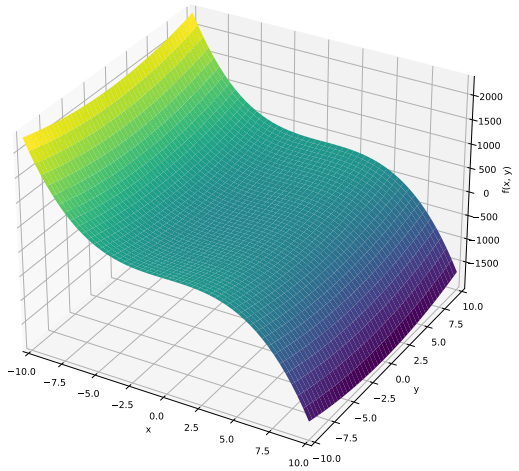


Figura 6.21: $z = -2x^3 + x + 3y^2 - 1$ no intervalor $x \in (-10, 10)$ e $y \in (-10, 10)$

13. Desenhar o gráfico da função $V(h, r) = 1/3\pi r^2 h$ no intervalor $r \in [0, 4]$ e $h \in [0, 6]$.

```
import sympy as sp
from sympy.plotting import plot3d
h, r = sp.symbols("h r")
V=1/3*sp.pi*r**2*h
g = plot3d(V, (r,0,4), (h,0,6))
```

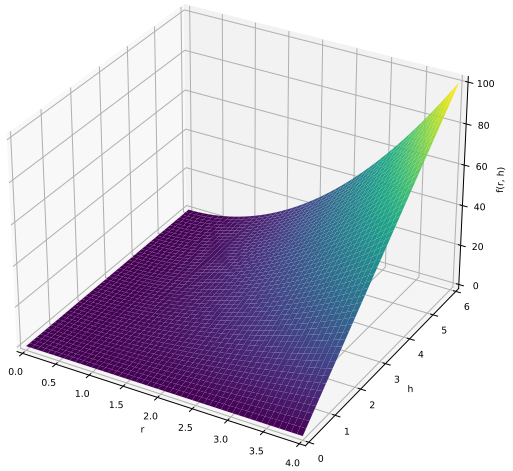


Figura 6.22: gráfico da função $V(h, r) = 1/3\pi r^2 h$ no intervalor $r \in (0, 4)$ e $h \in (0, 6)$

6.10 Exercícios

1. Plote o gráfico das seguintes funções
   (a) $y = 5$
   (b) $x = 2$
   (c) $y = |x| + 2$
   (d) $xy = 10$
2. Plote a reta que atende às seguintes condições:
   (a) Passa pelo ponto (3,7) com inclinação $\frac{2}{3}$
   (b) Passa pelo ponto (-4,8) com inclinação $-\frac{1}{2}$
   (c) Passa pelo ponto (2,5) e é paralela ao eixo dos x
   (d) Passa pelo ponto (2,5) e é paralela ao eixo dos y
   (e) Passa pela origem e é paralela à reta $y = 3x + 2$
   (f) Passa pela origem e é perpendicular à reta $y = 3x + 2$
3. Plote a circunferência de raio $R = 4$ que passa pelos pontos $(1, 1)$ e $(2, 3)$.
4. Plote a curva definida pela expressão $x^2 + 4y^2 = 4$
5. Desenhar um hexágono regular com centro na origem e lado igual a 1.

6. A partir de um hexágono regular com centro na origem e lado igual a 1, desenhar uma circunferência circunscrita ao mesmo.

7. Desenhar os gráficos das seguintes funções dadas em coordenadas retangulares :

   (a) $y = \frac{2x}{x^2+1}$

   (b) $y = \sin(x) - \frac{1}{3}\sin(3x)$

   (c) $y = e^{x^2}$

   (d) $y = \ln(\cos(x))$

8. Desenhar os gráficos das seguintes funções dadas em coordenadas polares :

   (a) $r = \frac{1}{\sin(\theta)}$

   (b) $r = a(1 + cos(\theta))$ para a igual a 1,2 e 3

   (c) $r = \sec(\frac{\theta}{2})^2$

   (d) $r = 10\cos(4\theta)$

9. Desenhar os gráficos das seguintes funções : (em todos os casos considerar x e $y \in [-10, 10]$ e h igual a 0,2)

   (a) $z = x^2 + y^2$

   (b) $z = x + y$

   (c) $z = \frac{x-y}{x+y}$

   (d) $z = \sin(x)\cos(y)$

10. Desenhar as curvas de nível das seguintes funções : (em todos os casos considerar x e $y \in [-2, 2]$ e h igual a 0,1)

    (a) $z = x^2 + y^2 - 2x^2y + 25$

    (b) $z = x\sin(y) + y\sin(x)$

    (c) $z = \frac{1}{xy}$

11. Descreva o domínio de f(x) = 8x / ((x-1).(x-2))

12. Converta de graus para radianos: a) 270o, b)-45o

13. Converta de radianos para graus: a) $3\pi/2$ b) $-\pi/4$

14. Determine o comprimento do arco de circunferência com 45o e um raio de 10 cm

15. Demonstre que $\frac{2tg(\theta)}{1+tg^2(\theta)} = sen(2\theta)$

16. Crie uma função Python que recebe as coordenadas x e y de dois pontos e em seguida calcula e retorna a distância euclidiana e a distância manhatan entre ambos. OBS: A distância euclidiana é a distância "direta"entre os pontos. A distância manhatan é calculada caminhando pelas "laterais"do triângulo (como ocorreria em uma cidade).

17. Sabendo que a área de um triângulo pode ser calculada pela fõrmula $A = \frac{p}{2}\left[\left(\frac{p}{2}-a\right)\left(\frac{p}{2}-b\right)\left(\frac{p}{2}-c\right)\right]$, onde a, b e c são o comprimento de cada um dos lados e p o perímetro, crie uma função Python que recebe o comprimento de cada um dos lados e retorna a área do triângulo correspondente. Suponha que o usuário irá fornecer valores tais que $a < b + c$ onde a é o maior valor. Esta condição irá garantir que os segmentos de reta formam um triângulo.

18. Criar uma função Python que recebe as coordenadas cartesianas de três pontos A, B e C. A partir das distâncias entre cada par dos três pontos a função deverá retornar uma das seguintes mensagens: "Os pontos formam um triâgulo"ou "Os pontos não formam um triângulo".

19. Criar uma função Python que recebe três comprimentos, a, b e c e retorna se os comprimentos formam ou não um triângulo e caso formem, qual o tipo de triângulo representado. Para determinar estas condições, suponha que a é o maior dos três valores inseridos. Nestas condições:

    (a) Se $a > b + c$ então não formam triângulo algum

    (b) Se $a^2 = b^2 + c^2$ então formam um triângulo retângulo

    (c) Se $a^2 > b^2 + c^2$ então formam um triângulo obtuso

    (d) Se $a^2 < b^2 + c^2$ então formam um triângulo agudo

    (e) Se $a = b$ e $b = c$ então formam um triângulo equilátero

    (f) Se $a = b$ ou $b = c$ ou $a = c$ e $a \neq b$ ou $a \neq c$ então formam triângulo isósceles.

Capítulo 7

Funções Racionais e Polinômios

As funções que envolvem apenas potências inteiras de uma variável x recebem o nome de funcções polinomiais. O estudo de sua variação e do cálculo de suas raízes, isto é dos valores de x tais que $f(x) = 0$, ocupam um lugar central na Álgebra.

7.1 Teorema Fundamental da Álgebra

O teorema fundamental da álgebra diz que se $f(x)$ é um polinômio de grau n, com n>0, então a função $f(x)$ terá n raízes. Estas raízes podem ser reais ou complexas (da forma $a + b.i$, onde $i = \sqrt{(}-1)$ também conhecido como número imaginário. Caso existam raízes complexas elas aparecerão sempre aos pares, complexos conjugados. Isto quer dizer que se $f(a + b.i) = 0$ então $f(a - bi)$ também será igual a 0.

7.2 Teorema da Fatoração Linear

Se $f(x)$ é um polinômio de grau n, onde $n > 0$, então $f(x)$ tem precisamente n fatores lineares tais que: $f(x) = a_n(x - x_1)(x - x_2)...(x - x_n)$ onde x_1, x_2, ..., x_n são números complexos. Devemos lembrar que todo número real é também pertencente ao conjunto dos complexos, porém com sua parte imaginária $b.i$ nula.

7.3 Raízes de Uma Função Polinomial

Para encontrar os zeros (raízes de uma função, seja ela polinomial ou não) devemos fazer $f(x) = 0$ e resolver a equação resultante para x. Caso $x = x_0$ seja uma das raízes de $f(x)$, se calcularmos o quociente polinomial $f(x)/(x - x_0)$ o resultado deverá ter resto igual a 0.

Além disso, caso o polinômio tenha raízes inteiras, sabendo que $f(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$ e que portanto $f(x) = a_n(x-x_1)(x-x_2)...(x-x_n)$, as raízes inteiras deverão ser divisores do termo independente. Isto pode ser comprovado igualando-se as duas representações do polinômio e comparando os termos de mesma potência em x. Se procedermos desta forma veremos que o termo independente a_0 deverá ser igual a $a_0 = a_n.(-x_1).(-x_2)...(-x_n)$

1. Como exemplo inicial calculemos as raízes de $f(x) = x^2 - 6x + 9$.

Os divisores de são ± 1, ± 3, e ± 9. Testando as possíveis raízes inteiras de $f(x)$ vemos (vide a seguir) que $x = 3$ é uma raiz dupla e $f(x)$ pode ser fatorado em $f(x) = (x - 3)(x - 3)$

```
import numpy as np
f = lambda x: x**2 -6*x + 9
f(-9), f(-3), f(-1), f(1), f(3), f(9)
```

```
(144, 36, 16, 4, 0, 36)
```

```
import sympy as sp
x = sp.symbols("x")
f = (x-3)*(x-3)
f.expand()
```

```
x**2 - 6*x + 9
```

A raiz poderia ter sido calculada diretamente através da `sympy`, utilizando-se a função `solve` conforme pode ser visto a seguir:

```
import sympy as sp
x = sp.symbols("x")
f = x**2 - 6*x + 9
sp.solve(f)
```

```
[3]
```

2. Calcule as raízes de $f(x) = x^4 - x^3 + x^2 - 3x - 6$

Sabendo que $a_0 = a_n.(-x_1)(-x_2)...(-x_n)$ teremos neste caso, se existirem raízes inteiras $-6 = (-x - 1).(-x_2)(-x_3)(-x_4)$. Os divisores de 6 são ± 1, ± 2, ± 3 e ± 6. Testando (vide a seguir)

```
import numpy as np
f = lambda x: x**4 -x**3 +x**2 -3*x -6
x = np.array([-6,-3,-2,-1,1,2,3,6])
y = f(x); y
```

```
array([1560,  120,   28,    0,   -8,    0,   48, 1092])
```

Logo temos como raízes reais $x = -1$ e $x = 2$. Dividindo $f(x)$ primeiro por $x+1$ obteremos:

$$\frac{x^4 - x^3 + x^2 - 3x - 6}{x+1} = x^3 - 2x^2 + 3x - 6$$

O que pode ser confirmado fazendo-se:

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
f1 = x + 1
g = f/f1
g.simplify()
```

```
x**3 - 2*x**2 + 3*x - 6
```

Dividindo o polinômio resultante $g(x) = x^3 - 2x^2 + 3x - 6$ por $x - 2$ obtemos:

$$\frac{x^3 - 2x^2 + 3x - 6}{x-2} = x^2 + 3$$

O que também pode ser confirmado fazendo-se:

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
f1 = x + 1
f2 = x - 2
g = f/f1
h = g/f2
h.simplify()
```

```
x**2 + 3
```

Este último polinômio $h(x) = x^2 + 3$ possui duas raízes complexas conjugadas iguais a $x = \sqrt{3}i$ e $x = -\sqrt{3}i$.

Poderíamos ter calculado todas as raízes de $f(x)$ diretamente através da função `solve`, o que é apresentado a seguir e confirma nosso cálculo anterior.

```
import sympy as sp
x = sp.symbols("x")
f = x**4 -x**3 +x**2 -3*x -6
sp.solve(f,x)
```

```
[-1, 2, -sqrt(3)*I, sqrt(3)*I]
```

7.4 Funções Racionais

Uma função racional é dada por $y = f(x) = \frac{P(x)}{Q(x)}$ onde $P(x)$ e $Q(x)$ são funções polinomiais. A função racional $f(x)$ terá pontos de descontinuidade onde $Q(x) = 0$.

7.5 Assíntotas Horizontais e Verticais

$y = L$ é dita uma assíntota horizontal da curva $y = f(x)$ se $\lim_{x\to+\infty} f(x) = L$ ou $\lim_{x\to-\infty} f(x) = L$

$x = L$ é dita uma assíntota vertical da curva $y = f(x)$ se $\lim_{x\to L} f(x) = +\infty$ ou $\lim_{x\to L} f(x) = -\infty$

7.6 Aplicação em Marketing: Efeito da Propaganda na Receita de Um Produto

1. Suponha que a receita R de um produto a partir do valor investido em propaganda x seja dada pela função $R(x) = \frac{100x+300}{x+10}$. Esboce o gráfico desta função, aponte os valores de x que formam o domínio da mesma e indique quais valores tem sentido real, caso x seja considerado o valor investido em propaganda.

Para esboçar o gráfico da função observamos que ela tem um zero em $x = -3$ e o ponto $x = -10$ não pertence ao seu domínio, pois neste caso $x + 10 = 0$. Do ponto de vista prático, apenas valores não nulos de x representam possíveis montantes para o gasto com propaganda. Além disso se calcularmos o valor da função para grandes valores tanto negativos quanto positivos de x veremos que a função se aproxima de 100. Sendo assim, esta função possui uma assíntota horizontal em $x = 100$ e uma assíntota vertical em $x = -10$.

Podemos plotar o gráfico da função de duas formas através do Python, pela `numpy` e pela `sympy`. Ambas as formas podem ser vistas a seguir nas 7.1 e 7.2.

```
import numpy as np
from matplotlib import pyplot as plt

```

```
x1 = np.linspace(-100.1,-12,1111)
y1 = (100*x1+300)/(x1+10)
plt.plot(x1,y1);

x2 = np.linspace(-8,100,1111)
y2 = (100*x2+300)/(x2+10)
plt.plot(x2,y2);

plt.show();
```

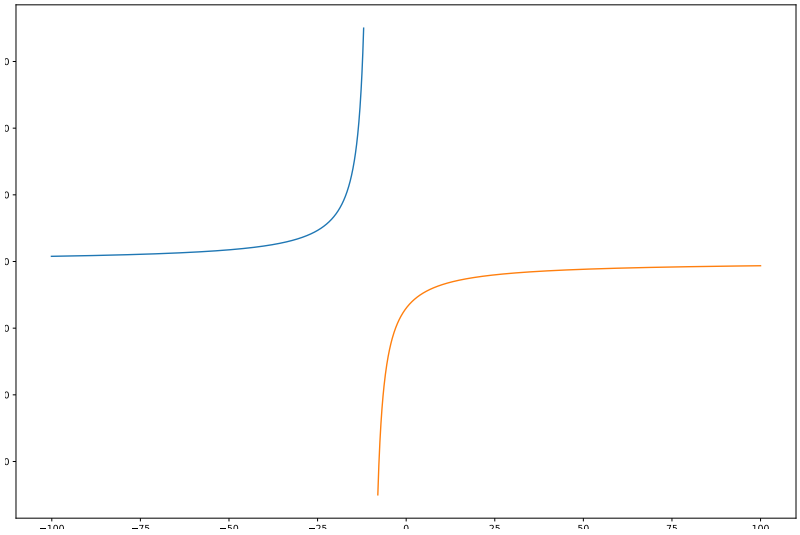


Figura 7.1: Esboço de f(x) através da numpy

```
import sympy as sp
x = sp.symbols("x")
f = (100*x+300)/(x+10)
graf1 = sp.plot(f, (x,-100,-12), show=False);
graf2 = sp.plot(f, (x,-8,100), show=False);
graf1.append(graf2[0]);
graf1.show()
```

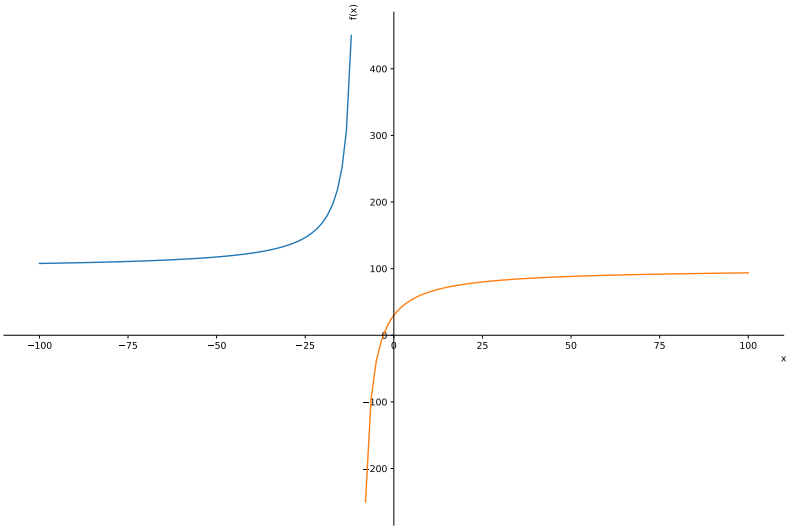


Figura 7.2: Esboço de f(x) através da sympy

7.7 Exercícios

Conforme visto nos exemplos, tanto a `numpy` quanto a `sympy` permitem a execução de operações com polinômios e funções racionais. Abaixo são apresentados exercícios que podem ser resolvidos tanto utilizando-se a `numpy` quanto a `sympy`.

1. Avalie a função $f(x) = \frac{500x+2000}{x+50}$ determinando:
   (a) Pontos de descontinuidade no domínio de $f(x)$
   (b) Pontos de zero de $f(x)$
   (c) Assíntotas horizontais
   (d) Assíntotas verticais
   (e) Supondo que $f(x)$ represente a capacidade de transporte de um item a partir da quantidade de produto disponível em estoque, qual o significado de $f(0)$.
   (f) Se $f(x)$ for a receita que pode ser obtida pela fabricação de um produto x o que significa $\lim_{x\to\infty} f(x)$
2. Suponha que a função oferta de um produto seja dada pela função $f(x) = \frac{2000x+400}{x+40}$ onde f é a quantidade em toneladas e x o preço unitário em Reais por tonelada. Com base nestes dados faça os exercícios a seguir:
   (a) Plote o gráfico de $f(x)$, determinando: a) pontos de zero, b) de descontinuidade no domínio da função, c) assíntotas horizontais e d)

assíntotas verticais

(b) O que significa $f(0)$ do ponto de vista matemático e do ponto de vista gerencial neste caso?

(c) O que significa $\lim_{x\to\infty} f(x)$ do ponto de vista tanto matemático quanto gerencial neste caso?

3. Suponha que o custo de produção de uma empresa seja dado pela função $f(x) = 4x^2 + 12x + 100$ onde x é a quantidade de produto fabricado (em milhares de unidades). A partir desta função determine:

   (a) O custo total para se produzir 1000, 2000, 4000, e 20000 unidades

   (b) O custo médio de produção por unidade fabricada para cada um dos volumes de produção acima

   (c) Suponha que o custo médio seja descrito por uma função $g(x)$ onde x é a quantidade fabricada. Qual a expressão algébrica para $g(x)$

   (d) Qual o significado gerencial de $f(0)$ e de $\lim_{x\to\infty} f(x)$

   (e) Qual o significado gerencial de $\lim_{x\to 0} g(x)$ e de $\lim_{x\to\infty} g(x)$

   (f) Esboce o gráfico de $f(x)$

   (g) Esboce o gráfico de $g(x)$

4. Expanda e simplifique as expressões polinomiais a seguir:

   (a) $4(x + 6) + 3(2x - 5)$

   (b) $(x + 5)(x + 2)$

   (c) $(\sqrt{x} + \sqrt{y})(\sqrt{x} - \sqrt{y})$

   (d) $(3x + 2)^3$

   (e) $(x + 3)^2$

5. Fatore as expressões polinomiais a seguir:

   (a) $8x^2 - 50$

   (b) $x^2 + \frac{5}{2}x - 6$

   (c) $x^3 - 3x^2 - 4x + 12$

   (d) $x^4 + 125x$

   (e) $3x^{\frac{5}{2}} - 9x^{\frac{3}{2}} + 6x^{\frac{1}{2}}$

   (f) $x^5y^2 - 4x^2y^2$

6. Simplifique as expressões polinomiais a seguir:

   (a) $\frac{x^2+5x+4}{x^2+2x+1}$

(b) $\frac{x^2}{x^2-16} - \frac{x+1}{x+4}$

(c) $\frac{2x^2-x-1}{x^2-81} \cdot \frac{x+9}{2x+1}$

(d) $\frac{\frac{x}{y}-\frac{y}{x}}{\frac{1}{x}-\frac{1}{y}}$

7. Avalie o valor do polinômio $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$ no ponto $x = 1$

8. Avalie o valor do polinômio $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$ no ponto $x = 1 + j$

9. Calcule o quociente da divisão do polinômico $p(x) = x^7 - 3x^5 + 5x^3 + 7x + 9$ por $2x^6 - 8x^5 + 4x^2 + 10x + 12$

10. Calcule o produto escalar do polinômio $x^4 + 2x^3 + 3x^2 + 4x + 5$ por $-2x^4 + 6x^3 - 3x^2 + 8x + 7$

11. Calcule o deslocamento do polinômio $x^2 - 6x + 20$ para $x + 5$

12. Desenhe o gráfico da parte real, imaginária e o módulo de $Z(\omega)$ em função da frequência ω para $Z(\omega) = 10 + \frac{10^4 - j10^6/\omega}{10 + j(0.1\omega - 10^5/\omega)}$ com ω variando de 0 até $2.200 rad/s$

13. Calcular as raízes do polinômio: $p(x) = x^5 - 7\,x^4 + 16\,x^2 + 25\,x + 52$, tanto pela forma literal quanto pela forma numérica.

14. Determinar os coeficientes do polinômio formados pelas raízes: [-1, -2, -3, 4+5j, 4-5j] através da 'sympy'.

Capítulo 8

Funções Exponencial e Logarítmo

Duas funções de grande importância no estudo de Administração são as funções Exponencial e Logarítmo as quais são estudadas a seguir.

8.1 A Função Exponencial

Um função exponencial é dada pela expressão $y = f(x) = b.a^x$, com $a > 0$, $b \neq 0$ e $a \neq 1$.

Caso $a > 1$ a exponencial é dita crescente (8.1 e 8.2).

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 2
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

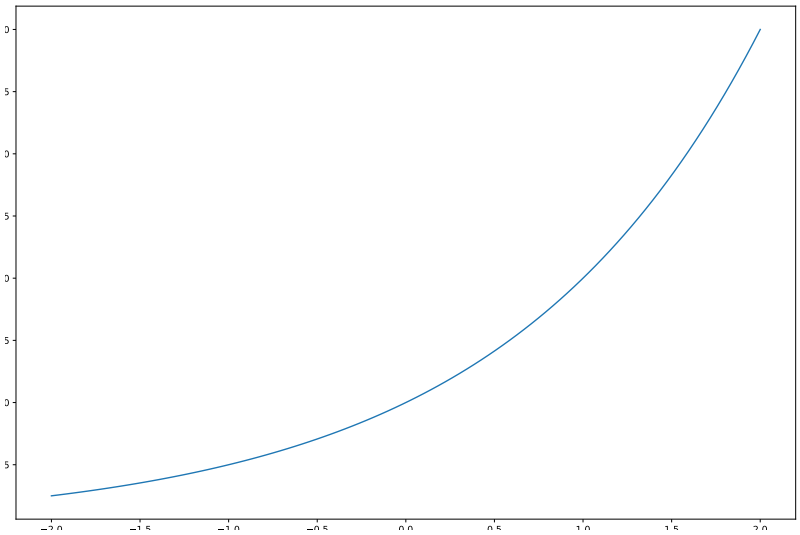


Figura 8.1: Exponencial crescente com b=5, a=2

```
import sympy as sp
x = sp.symbols("x")
b = 5
a = 2
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

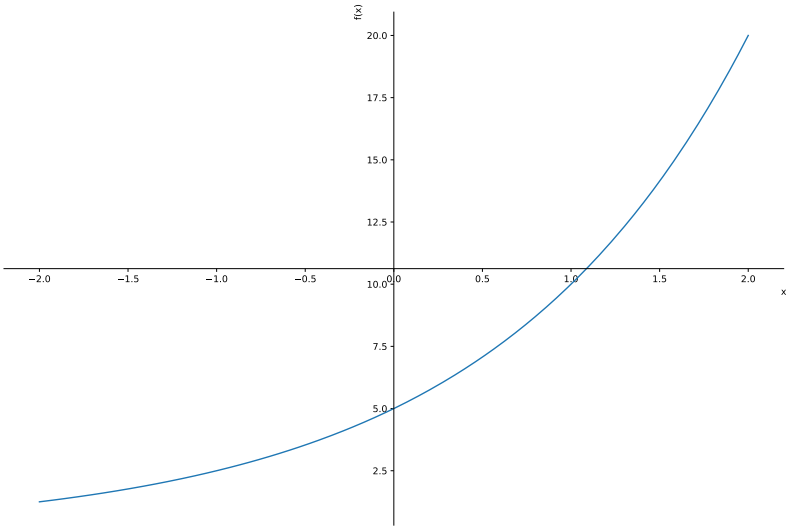


Figura 8.2: Exponencial crescente com b=5, a=2 via sympy

Se $a < 1$ a exponencial será dita descrescente (8.3 e 8.4)

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 0.5
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

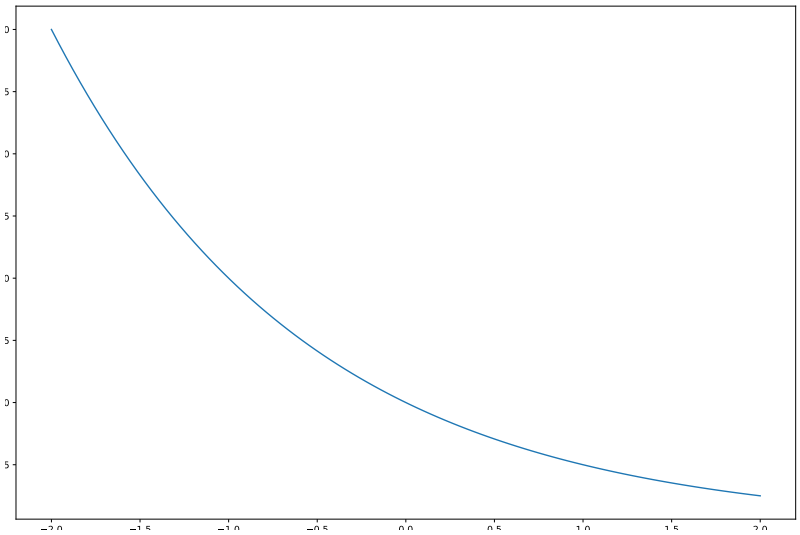


Figura 8.3: Exponencial decrescente com b=5, a=2 e k=-1

```
import sympy as sp
x = sp.symbols("x")
b = 5
a = 0.5
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

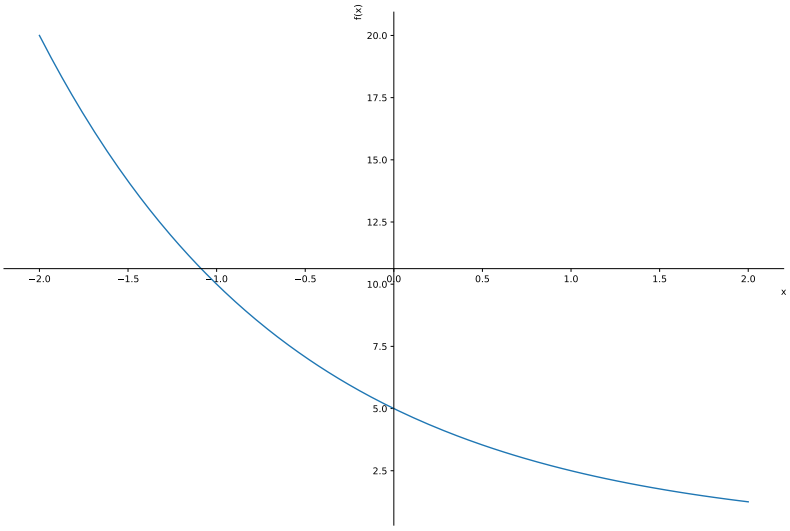


Figura 8.4: Exponencial crescente com b=5, a=2 via sympy

Caso $a = 1$ a função se transforma (degenera) na função constante $y = f(x) = b$ (vide 8.5 e 8.6).

```
import numpy as np
from matplotlib import pyplot as plt
b = 5
a = 1
x = np.linspace(-2,2,100)
y = b*(a**x)
plt.plot(x,y);
plt.show();
```

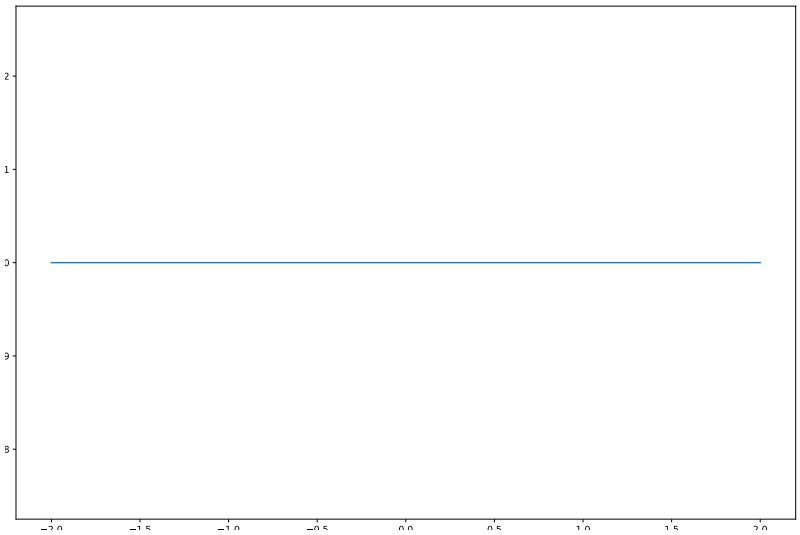


Figura 8.5: Exponencial 'degenerada' na função constante com b=5, a=1

```
import numpy as np
x = sp.symbols("x")
b = 5
a = 1
y = b*(a**x)
sp.plot(y, (x,-2, 2));
```

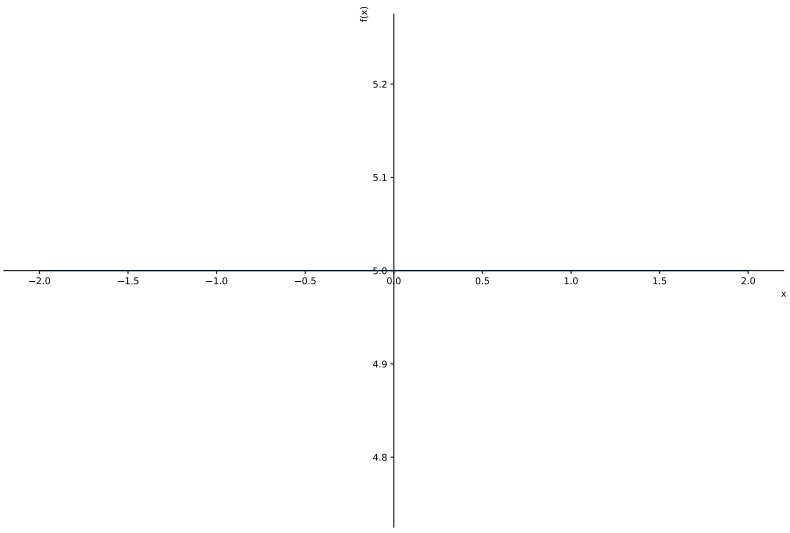


Figura 8.6: Exponencial 'degenerada' na função constante com b=5, a=1, via sympy

Um caso especial e muito importante ocorre quando $a = e$ onde e é o número de Euler, uma constante irracional cujo valor aproximado é 2,71828...

8.2 A Função Logaritmo

A função logaritmo é a inversa da função exponencial. Sendo assim se $y = a^x$ dizemos que $log_a(y) = x$. Caso $a = 10$ temos os logaritmos decimais, geralmente representados por $log(y) = x$ (e consequentemente $10^x = y$). Caso $a = e$ temos os logaritmos naturais ou neperianos, representados por $ln(y) = x$ (o que implica neste caso que $e^x = y$).

8.3 Propriedades dos Logaritmos

As principais propriedades algébricas dos logaritmos (independentes de sua base de cálculo) são:

1. $ln(A.B) = ln(A) + ln(B)$
2. $ln(\frac{A}{B}) = ln(A) - ln(B)$
3. $ln(A^k) = k.ln(A)$
4. $ln_B(A) = \frac{ln(A)}{ln(B)}$

A forma gráfica da função logarítmo pode ser vista a seguir para a base 2 (8.7) e para a base $\frac{1}{2}$ (8.8). Atente para o fato de que no Python, tanto na `numpy` quanto na `sympy` a função `log` diz respeito ao logaritmo natural.

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0.1,10,1000)
y = np.log(x)/np.log(2)
plt.plot(x,y);
plt.show();
```

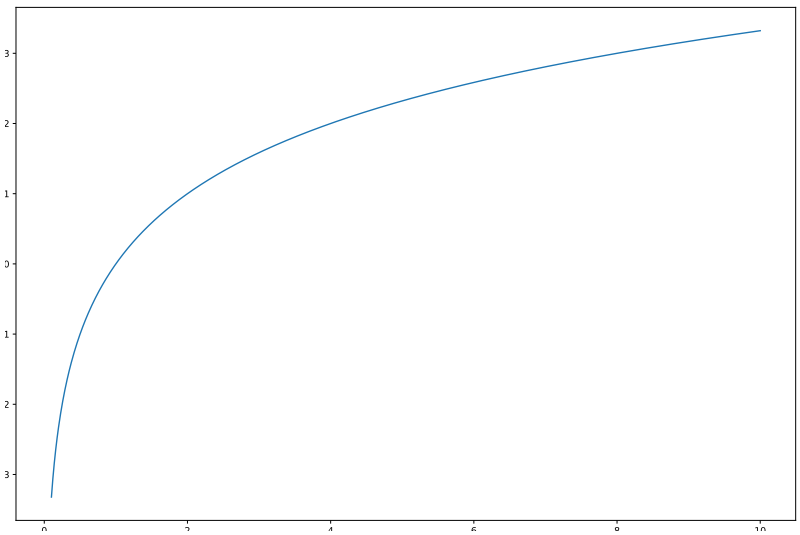


Figura 8.7: Função logaritmo na base 2

```
import numpy as np
from matplotlib import pyplot as plt
x = np.linspace(0.1,10,1000)
y = np.log(x)/np.log(0.5)
plt.plot(x,y);
plt.show();
```

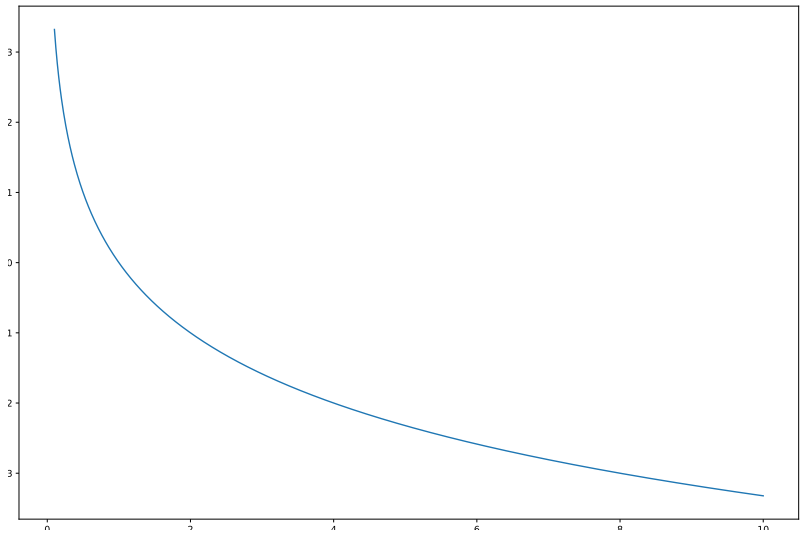


Figura 8.8: Função logaritmo na base 0.5

8.4 Aplicação em Geografia: Modelo de Crescimento Populacional

1. Suponha uma região que tenha população inicial $P_0 = 1000$, a qual irá crescer a uma taxa de 0.02 ao ano. Qual a expressão que irá determinar a quantidade aproximada de pessoas que estarão habitando esta região t anos após o instante inicial t_0?

Solução: $P(t) = P_0(1 + 0.02)^t$

1. Uma cidade tem hoje 20.000 habitantes e esse número cresce a uma taxa de 0.03 ao ano. Com base nestas informações pede-se:
   (a) Qual o número de habitantes daqui a 10 anos?
   (b) Se daqui a dez anos no número de habitantes for igual a 30.000, tendo partido de um total de 20.000 habitantes, qual seria a taxa de crescimento anual?

Solução:

A quantidade de habitantes em 10 anos é calculada a seguir (via `numpy`)

```
import numpy as np
p0 = 20000
r = 0.03
p = lambda t:p0*(1+r)**t
p(10)
```

```
26878.327586882446
```

A taxa de crescimento anual para uma população em 10 anos de 30.000 pessoas, começando o crescimento em 20.000 pessoas é calculada a seguir (via `sympy`). Observe que desta vez foi utilizado o solver numérico da `sympy` através da função `nsolve`. Este solver requer um ponto de partida para encontrar a solução, o qual escolhemos como zero.

```
import sympy as sp
r = sp.symbols("r")
p = 20000*(1+r)**10
sp.nsolve(p-30000,r,0)
```

```
0.0413797439924106
```

8.5 Aplicação em Finanças: Juros Compostos

A mesma lógica de crescimento populacional pode ser utilizada para calcular o valor futuro de um investimento submetido a uma taxa de juros composta. Com base nesta informação pede-se resolver o próximo exercício:

1. Um automóvel novo vale hoje R$20.000 e sofre desvalorização de 0.15 ao ano. Daqui a quanto tempo seu valor se reduzirá a metade do valor original?

```
import sympy as sp
t = sp.symbols("t")
p = 20000*(1-0.15)**t
sp.nsolve(p-10000,t,0)
```

```
4.26502428179873
```

8.6 Exercícios

A seção a seguir inclui exercícios de Matemática Financeira, dado o amplo uso das funções exponencial e logarítmo na resolução de problemas em finanças.

1. Sabendo que a relação entre o valor presente (VP) e o valor futuro (VF) de uma quantia, quando a taxa de juros (i) possui capitalização contínua, é dado por VF = VP.ei.t , onde t é o tempo medido em anos, determine a taxa de juros (i), sabendo que VF=200, VP=100 e t=2 anos.
2. O valor presente de um investimento é $100 e sabe-se que em 3 anos seu valor será de $200. Calcule qual é a taxa de juros contínua deste investimento.

3. Qual a taxa de juros com capitalização contínua que faz um valor de R$ 100.000 tornar-se R$90.000 em 2 anos?

4. Suponha que uma companhia emita títulos custando R$ 200,00 e pagando juros compostos mensalmente. Os juros são acumulados até que o título atinja a maturidade. Suponha que, após 5 anos, o título tenha um valor de R$ 500,00. Qual é a taxa anual de juros?

5. Suponha que um casal fez um investimento de R$ 4.000 a cada ano durante quatro anos numa aplicação que rende 8

6. Encontre o valor presente de R$ 10.000 a serem pagos ao final de 5 anos se o dinheiro for investido a juros de 12

7. Uma carteira de ações se valorizou de R$ 100.000 para R$117.000 em 2 anos. Que taxa de juros, composta continuamente, foi obtida com esse investimento?

8. Qual o valor futuro de um investimento de R$ 1.000,00 feito hoje, daqui a 2,5 anos, supondo que a taxa de juros seja de 7,5

9. Crie uma função Python que recebe como parâmetros: a) VP ("valor presente"), b) i ("taxa de juros") e c) n ('número de períodos do investimento') e retorna o valor futuro de um investimento após n períodos sob juros compostos.

10. Suponha duas quantias de 1000 e 2000 reais depositadas em contas bancárias distintas, as quais pagam juros simples iguais a : 10

11. Supondo que a população de uma cidade A seja da ordem de 100.000 habitantes com uma taxa anual de crescimento de 2.8

Capítulo 9

Limites

9.1 Introdução

Vamos agora aprofundar nosso estudo de funções. Como pode ser visto nos últimos exercícios do capítulo anterior, um tema recorrente é a análise da variação dos valores de uma função.

Para nos auxiliar neste processo vamos utilizar o próprio conceito de funções do Python. Uma função Python pode ser utilizada para obter o valor de uma expressão através do valor de uma variável. Sendo assim vamos criar uma função que receba um valor e execute o cálculo desejado para nós.

Podemos obter tal resultado primeiro precisamos avisar o Python do nome que terá nossa função, por exemplo f. Para apenas definir um nome como uma função basta fazer:

```
def f():
  return()
```

No caso acima informamos o Python que `f` é uma função (por meio de `def`), que esta função no momento não requer nenhum argumento (através dos parênteses) e que a mesma não executa nenhum cálculo, retornando como resultado um valor nulo (vazio) (através de `return()`). Vamos melhorar esta definição.

```
def f(x):
  return()
```

Agora `f` é uma função que irá receber um argumento `x`, porém ainda não executa nenhum cálculo com ele. Para torna-la operacional então fazemos:

```
def f(x):
  y = x**2 + 3*x + 40;
  return(y)
```

Agora temos uma função Python completa. Ela se chama `f`, precisa de um argumento, denominado `x`, executa a operação $x^2+3*x+40$ com este argumento, atribui o resultado desta operação à uma variável `y` e retorna o valor desta variável como resultado.

Para calcular o valor desta função em $x = 6$ podemos agora simplesmente fazer:

```
f(6)
```

```
94
```

9.2 Cálculo de valores repetidos de uma função.

No caso de uma planilha, para calcular uma série de valores de uma função podemos colocar os valores de interesse da variável x em uma coluna e simplesmente copiar a fórmula da função na coluna ao lado.

Se estamos trabalhando no Python, lançamos mão da possibilidade de atribuir uma sequência de valores para uma variável x (a qual passa a ser chamada de `vetor`) e utilizamos este vetor como argumento da função desejada. Sendo assim, se o vetor de entrada possuir seis valores (-2, -1, 0, 2, 4 e 5 por exemplo), a saida será também um vetor de seis elementos, cada um deles o resultado da função para o valor específico de entrada. Um equívoco comum entre usuários iniciantes de Python é supor que uma `lista` seja equivalente a um `vetor`. Listas não permitem de forma automática a execução de operações elemento a elemento. Para termos tal funcionalidade precisamos lançar mão dos arrays `numpy`. Veja a seguir:

```
import numpy as np
x = np.array([-2, -1, 0, 2, 4, 5])
```

Aqui fizemos x receber a sequência de números (-2, -1, 0, 2, 4 e 5) na forma de um `vetor`. Na verdade convertemos a lista `[-2, -1, 0, 2, 4, 5]` em um array `numpy` com os mesmos elementos. Agora vamos utilizar este vetor `x` como argumento de entrada de `result` e observar o resultado:

```
f(x)
```

```
array([38, 38, 40, 50, 68, 80])
```

Podemos dizer que a uma planilha Excel é mais rápida de utilizar. Porém quando os cálculos aumentam em complexidade o Python torna-se bem mais prático ou até mesmo essencial para a tarefa que queremos executar.

9.3 Aproximação Numérica via Python

Vamos agora trabalhar com problemas de matemática superior. No caso desta seção iremos mostrar como calcular limites de funções com o Python, seja de forma puramente numérica (com a `numpy`), seja de forma algébrica (com a `sympy`). Uma pressuposição deste texto é que o leitor é um usuário de planilha que deseja experimentar o uso do Python para resolver problemas de cálculo.

Entende-se por limite, o valor para o qual tende a função $f(x)$ a medida que a variável independente x aproxima-se de um determinado valor. Na maioria das vezes para determinar o limite de $f(x)$ quando x tende a um valor específico a, basta calcular o valor de $f(a)$.

Suponha que se queira calcular $\lim_{x\to 3} x^2 + 2\,x + 4$. Neste caso o resultado seria imediato, pois $f(3) = 3^2 + 2\,3 + 4 = 19$. Nem sempre o cálculo do limite é direto. As vezes a simples substituição da variável x pelo valor para o qual ela tende produz: a) uma operação proibida (uma divisão por zero), b) uma operação cujo resultado esta fora da capacidade computacional da máquina (uma multiplicação por números muito grandes) ou c) porque o software não está preparado para lidar com os números necessários (uma raiz quadrada de números negativos). Nestas situações é necessário realizar o cálculo numérico do mesmo ou eliminar a situação de erro através de manipulação algébrica.

Por exemplo, calcular $\lim_{x\to 0} \frac{1}{x}$. Para determinar este limite devemos calcular o valor de $\frac{1}{x}$ para valores de x cada vez mais próximos de 0 sem no entanto atingir o 0. Este é o chamado cálculo por aproximação. Observe a tabela abaixo:

x	1/x
1	1
0,1	10
0,01	100
0,001	1000

Pela tabela acima observa-se que a medida que x aproxima-se de 0 o valor de $\frac{1}{x}$ aumenta cada vez mais. Sendo assim podemos extrapolar esta relação e dizer que se fosse possível a x chegar a 0 (o menor valor), o valor de $\frac{1}{x}$ atingiria o maior número (uma impossibilidade numérica). Este conceito é chamado de *infinito* e representado pelo símbolo ∞. Sendo assim, do ponto de vista formal dizemos que $\lim_{x\to 0} \frac{1}{x} = \infty$

9.4 Exercícios

9.4.1 Cálculo de Limites

1. Funções polinomiais, com raiz quadrada e envolvendo indeterminações do tipo $+\infty - \infty$

(a) $\lim_{x\to-\infty} -x^3+3x+10$

(b) $\lim_{x\to\infty} \sqrt[3]{x+a}-\sqrt[3]{x}$

(c) $\lim_{n\to x_0} \frac{\sqrt[n]{x}-\sqrt[n]{x_0}}{x-x_0}$

(d) $\lim_{n\to\infty} \left|\sqrt{x+1}-\sqrt{x+3}\right|$

2. Funções racionais e envolvendo indeterminações do tipo $\frac{0}{0}$ ou $\frac{\infty}{\infty}$

(a) $\lim_{x\to\infty} \frac{1000x}{x^2-1}$

(b) $\lim_{x\to\infty} \frac{(2x+2)^3(3x-4)^2}{x^5-1}$

(c) $\lim_{x\to\infty} \frac{(2x^2+3x+4)}{\sqrt{x^4-1}}$

(d) $\lim_{x\to2} \frac{x^2-6x+8}{x^2-5x+6}$

(e) $\lim_{x\to\infty} \frac{3x-1}{x+4}$

(f) $\lim_{x\to\infty} \frac{3x^2-3}{x^2-4}$

(g) $\lim_{x\to\infty} \frac{x^2-6}{x^-4}$

(h) $\lim_{x\to\infty} \frac{6x^2-3}{x^2-4}$

(i) $\lim_{x\to\infty} \frac{4(x-1)(x-2)}{(x-3)(x-4)}$

(j) $\lim_{x\to\infty} \frac{2x^2+3x+4}{\sqrt{x^4-1}}$

(k) $\lim_{x\to1} \frac{x^3+5x^2-7x+9}{x^2+1}$

(l) $\lim_{x\to2} \frac{x^2+5x-14}{x^2-2x}$

(m) $\lim_{x\to\infty} \frac{x^2-x-1}{x^3+x}$

(n) $\lim_{x\to\infty} \frac{x^4-3x^2+7x-1}{2x^4+3x+9}$

(o) $\lim_{x\to2} \frac{x^3-2x^2-x+15}{x^2+4}$

(p) $\lim_{x\to2} \frac{x^2+5x-14}{x^2-2x}$

(q) $\lim_{x\to-\infty} \frac{x^3-3x^2+4x-2}{3x^3+3x+9}$

(r) $\lim_{x\to\infty} \frac{x^2-2x+1}{x^3-3x}$

(s) $\lim_{x\to2} \frac{-x^4+2x^2+8}{x-2}$

(t) $\lim_{x\to1} \frac{x^2-3x+2}{x^2+1}$

(u) $\lim_{x\to1} \frac{(2x+5)\ln(x)}{(3x-1)}$

(v) $\lim_{x\to 2} \frac{x^2+5x-14}{x^2-2x}$

(w) $\lim_{x\to\infty} \frac{x^2-x-1}{x^3+x}$

(x) $\lim_{x\to\infty} \frac{-x^4-3x^2+7x-1}{2x^4+3x+9}$

3. Funções exponenciais e envolvendo indeterminações do tipo 1^∞

(a) $\lim_{x\to 0} (1+ax)^{\frac{1}{x}}$

(b) $\lim_{x\to\infty} (1+\frac{0,4}{x})^x$

(c) $\lim_{x\to 0} (1+0,4\,x)^{\frac{1}{x}}$

(d) $\lim_{x\to\infty} e^{-x^2}$

(e) $\lim_{x\to-\infty} e^{x^2}$

(f) $\lim_{x\to\infty} \frac{e^x}{e^{2x-1}}$

(g) $\lim_{x\to\infty} \frac{e^{3x}}{e^{x-1}}$

(h) $\lim_{x\to\infty} \frac{e^x}{e^{3x-1}}$

(i) $\lim_{x\to\infty} \left(\frac{x}{x+1}\right)^x$

(j) $\lim_{x\to\infty} \left(\frac{x-1}{x+3}\right)^{x+2}$

4. Funções trigonométricas:

(a) $\lim_{x\to 0} \frac{\sin(3\,x)}{x}$

(b) $\lim_{x\to 0} 1+\sin(x)\frac{1}{x}$

(c) $\lim_{x\to a} \frac{\cos(x)-\cos(a)}{x-a}$

(d) $\lim_{x\to 0} \frac{\sin 3x}{x}$

(e) $\lim_{n\to 0} \frac{\cos(mx)-\cos(nx)}{x^2}$

(f) $\lim_{x\to 0} \frac{1-\cos(x)}{x^2}$

9.4.2 Esboço de Gráficos

Para as funções a seguir determine: a) domínio e imagem, b) raízes e pontos de cruzamento com o eixo dos y, c) limites à esquerda e à direita dos pontos de descontinuidade, d) limites para $x \to \pm\infty$, e) assíntotas verticais e horizontais, f) intervalos de crescimento e decrescimento, g) pontos de mínimo e máximo local e h) esboço do gráfico

1. Funções de Primeiro Grau

(a) $f(x) = -2x + 1$

(b) $g(x) = 3x - 5$

(c) $f(x) = -2x + 1$

(d) $g(t) = 3t - 2$

2. Funções Polinomiais

(a) $j(x) = x^2 - 2x + 1$

(b) $g(x) = (x - 4)^2 + 1$

(c) $m(t) = (t - 1)^3$

(d) $n(x) = (x + 1)^4 - 1$

(e) $h(t) = t^2 + 2$

(f) $h(t) = (t - 2)^2$

(g) $j(t) = (t + 1)^2 - 2$

3. Funções envolvendo raízes

(a) $f(t) = \sqrt{-4t + 17}$

(b) $n(x) = \sqrt{x^2 - 7x + 6}$

(c) $f(x) = \sqrt{x^2 - 5x + 4}$

(d) $l(x) = \sqrt{x}$

(e) $p(x) = \sqrt{x + 1}$

4. Funções Racionais

(a) $h(x) = \frac{x-2}{2x-6}$

(b) $j(t) = \frac{2t}{t^2+16}$

(c) $f(x) = \frac{1}{x}$

(d) $g(x) = \frac{2x-3}{x-1}$

(e) $h(x) = \frac{3x^2-7x+9}{4x-16}$

(f) $i(x) = \frac{4x^7-3x+15}{2x^7-6x}$

(g) $j(x) = \frac{-2}{x+3}$

(h) $l(x) = \frac{3}{x-1} + 1$

(i) $m(x) = \frac{x^2+x}{x}$

(j) $f(x) = \frac{x^3-x^2-x+2}{x^3-4x^2}$

(k) $f(x) = \frac{4x^3+x^2-x-5}{x^3-8x^2}$

(l) $f(x) = \frac{x^2+2x-3}{x^2+3x}$

(m) $f(x) = \frac{x^2-3x+2}{100-x^2}$

(n) Dada a função: $g(x) = \frac{x-6}{3-x}$. Para esta função determine também: a área do triângulo formado pela reta tangente ao gráfico em $g(0)$ e as assíntotas da função $g(x)$.

5. Funções Racionais e com Raiz Quadrada

(a) $p(x) = \sqrt{\frac{x^2-8x+15}{-2x+8}}$

(b) $q(x) = \sqrt{\frac{x^3-4x}{(x+2)(-x+1)}}$

(c) $r(x) = \sqrt[3]{\frac{x^2-4x+3}{3x(2x-10)}}$

(d) $v(x) = \sqrt{\frac{(-x+2)(x^2-9)}{(4x-8)(-5x+15)}}$

(e) $w(x) = \sqrt[4]{\frac{x^{14}}{x^2+16}}$

(f) $y(x) = \sqrt[5]{\frac{x^4}{x^3+8}}$

(g) $f(x) = \frac{\sqrt{x^2+2x-6}}{1-x}$

6. Funções Logaritmicas e Exponenciais

(a) $m(x) = \log(-4x - 12)$

(b) $f(k) = \ln(k^4 + 5k^2 + 6)$

(c) $r(x) = 2^x$

(d) $s(x) = \log_2(x)$

(e) $h(x) = \log(\frac{-x+4}{x+2})$.

(f) $y = 2e^{x^2-4x}$

7. Funções Definidas por Segmento

(a) $f(x) = \begin{cases} 2x - 5, \text{ se } x > 2 \\ x - 3, \text{ se } x \leq 2 \end{cases}$ em $p = 2$

(b) $g(x) = \begin{cases} 1/x, \text{ se } x > 0 \\ x^2, \text{ se } x < 0 \\ 1, \text{se } x = 0 \end{cases}$ em $p = 0$

(c) $h(x) = \begin{cases} x^2 + 5x - 1, \text{ se } x \neq 1 \\ 4x + 1, \text{ se } x = 1 \end{cases}$ em $p = 1$

(d) $f(x) = \begin{cases} 4x - 1, \text{ se } x > 2 \\ 7, \text{ se } x \leq 2 \end{cases}$ em $p = 2$

(e) $g(x) = \begin{cases} 4x, \text{ se } x < 0 \\ x^3, \text{ se } x > 0 \text{ em } p = 0 \\ 0, \text{se } x = 0 \end{cases}$

(f) $h(x) = \begin{cases} x^2 + 2x + 1, \text{ se } x \neq 1 \\ 2(x+1), \text{ se } x = 1 \end{cases} \text{ em } p = 1$

(g) $f(x) = \begin{cases} 3x - 1, \text{ se } x > 1 \\ x + 1, \text{ se } x \leq 1 \end{cases} \text{ em } p = 2$

(h) $f(x) = \begin{cases} 1/x, \text{ se } x > 0 \\ -1/x, \text{ se } x < 0 \text{ em } p = 0 \\ 0, \text{se } x = 0 \end{cases}$

9.4.3 Aplicações

1. Seja $g(x) = x^3 - x^2 - x + 4$ contínua e derivável. Julgue como verdadeiro (V) ou falso (F) as afirmações a seguir:
   (a) Dom = IR
   (b) Im = IR – {4}
   (c) A função cresce no intervalo]- ∞;-1/3] ∪ [1;+∞[.
   (d) A função tem concavidade para cima no intervalo [-1/3;+∞[
   (e) O ponto x = 1 é ponto de mínimo da função.
   (f) O ponto x = -1/3 é ponto de inflexão da função.
2. Decida se as afirmações que seguem são verdadeiras (V) ou falsas (F), justificando a sua resposta.
   (a) A função $p(x) = \frac{x^3-1}{1-x^2}$ possui como domínio qualquer $x \neq 1$ e a imagem da função $g(x) = \sqrt{x^2 - 5x + 6}$ é qualquer número real.
   (b) O ponto de equilíbrio de um produto cuja função custo total é $C(q) = 20q + 50$ e cuja receita total é $R(q) = 30q$ ocorre para 6 unidades vendidas.
   (c) Se a demanda por um produto é linear e sabemos que vendemos 100 unidades a R$ 2 e vendemos 50 unidades a R$ 3, então a função demanda é $p(q) = -\frac{q}{50} + 4$.
   (d) A inversa da função $f(x) = x^2 + 4$ tem como imagem o conjunto dos reais.
   (e) Uma função admite assíntota horizontal $y = a$ quando $\lim_{x\to\infty} f(x) = a$.

(f) A função $p(x) = \frac{x^3-1}{x^2-9}$ possui como domínio qualquer x diferente de 3 ou -3 e a imagem da função $g(x) = \sqrt{-x^2+5x-6}$ é qualquer número real não-negativo.

(g) O ponto de equilíbrio de um produto cuja função custo total é $C(q) = 20q + 100$ e cuja receita total é $R(q) = 25q$ ocorre para 20 unidades vendidas.

(h) Se a demanda por um produto é linear e sabemos que vendemos 100 unidades a R$2 e vendemos 50 unidades a R$3, então a função demanda é $p(q) = \frac{-1}{100}q + 4$.

(i) A inversa da função $f(x) = x^2 + 1$ tem como imagem apenas os números reais não-negativos.

(j) Uma função admite assíntota vertical $y = a$ quando $\lim_{x\to a} f(x) = \pm\infty$.

3. Em 2017 o Facebook deverá atingir a marca de 2 bilhões de usuários ativos, sendo que somente metade da população da Terra, ou seja, 3,5 bilhões de pessoas possuem acesso à Internet. A curva ascendente do número de usuários pode ser representada pela função logística $U(t) = \frac{L}{1+e^{-k(t-t_0)}}$. Onde t é o tempo em anos, $U\,(t)$ é número de bilhões usuários ativos. Para o caso do Facebook L é igual a 3,5 bilhões de usuários, k é igual 0,3 e t_0 é igual a 12 anos. Sendo assim a função logística para o caso do Facebook é: $U(t) = \frac{3,5}{1+e^{-0,3(t-12)}}$. Dessa forma responda:

   (a) Qual o domínio e os interceptos da função $U(t)$, considerando o caso do Facebook?

   (b) Faça o estudo do crescimento da função $U(t)$, e responda se há pontos de máximo e/ou mínimo.

   (c) Faça o estudo da concavidade da função $U(t)$, e responda se há pontos de inflexão. O que o(s) ponto(s) de inflexão encontrados significam para a previsão de infraestrutura do Facebook?

   (d) Faça análise das assíntotas da função $U(t)$. O que o $\lim_{t\to+\infty} U(t)$ significa para o Facebook?

   (e) Faça o esboço do gráfico da função $U(t)$ para o caso do Facebook e discuta que modificações na estratégia do Facebook deveriam ser adotadas para manter o crescimento de sua receita.

Capítulo 10

Derivadas

10.1 Introdução

Entende-se por derivada a taxa de variação de uma função $y = f(x)$ quando o valor da variável independente x se altera de um valor muito pequeno (isto é tende para zero). Simbolicamente dizemos que :

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Dado que a alteração na função ocorreu em um ponto posterior $x + h$ em relação ao ponto que estamos calculando a derivada, dizemos que a mesma é calculada *à frente* ou *forward* em inglês. Caso a alteração ocorra em um ponto anterior ao ponto de cálculo da derivada, dizemos que a mesma é calculada *para trás* ou *backward* em inglês. Neste caso o procedimento de cálculo será :

$$f'(x) = \lim_{h \to 0} \frac{f(x) - f(x-h)}{h}$$

No entanto existe uma outra forma de cálculo da derivada, a qual produz resultados até mais precisos, pois combina alterações para a frente e para trás. Neste caso dizemos que a derivada é *central* e calculamos a mesma de acordo com :

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}$$

10.2 Cálculo Numérico de Derivadas

Nesta seção serão apresentados exemplos que demonstram a grande concordância entre a aproximação numérica da derivada e o seu valor obtido algebricamente.

Sempre que possível será utilizada a aproximação pela derivada central.

1. Calcule a primeira e a segunda derivadas numérica de $x^3 - 20x^2 + 11x + 30$, apresentando o gráfico da função e suas duas derivadas no intervalo $x = [-1,5; 4,0]$.

O cálculo da função e suas duas derivadas bem como o gráfico associado são apresentados na 10.1.

```
import numpy as np
import pandas as pd
import matplotlib as mpl; mpl.use("TkAgg")
from matplotlib import pyplot as plt
x = np.linspace(-1.5, 4.0, 100);
f = lambda x0:x0**3-20*x0**2+11*x0+30;
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h);
dados = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x)});
g = dados.plot(x='x', y=['y','dy']);
plt.show();
```

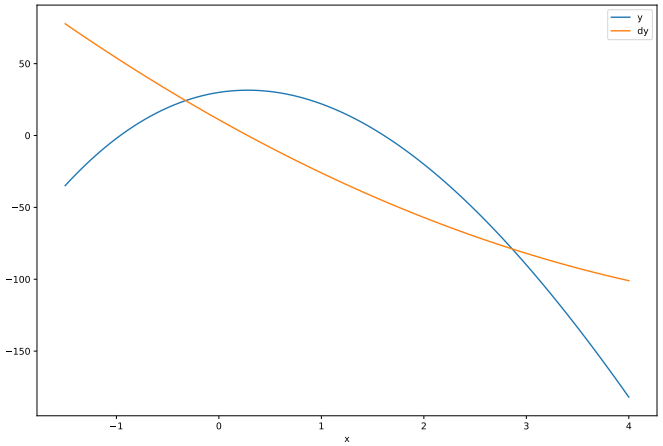


Figura 10.1: Cálculo e Gráfico da função e derivada de $f(x) = x^3 - 20x^2 + 11x + 30$

2. Calcule a derivada de $f(t) = \sin(\theta)$ por aproximação numérica (central) e compare o resultado com o valor analítico da derivada $f'(t) = \cos\theta$, apresentando ambos em um gráfico.

Observe que a aproximação é boa o suficiente para os gráficos das derivadas numérica e algébrica coincidirem.

```
import numpy as np
import matplotlib as mpl; mpl.use("TkAgg")
from matplotlib import pyplot as plt
x = np.linspace(-1.5, 4.0, 100);
f = lambda x0:np.sin(x0);
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h);

import pandas as pd
dados = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x),
                      'dy_exata':np.cos(x)});
g = dados.plot(x='x', y = ['dy', 'dy_exata']);
plt.show();
```

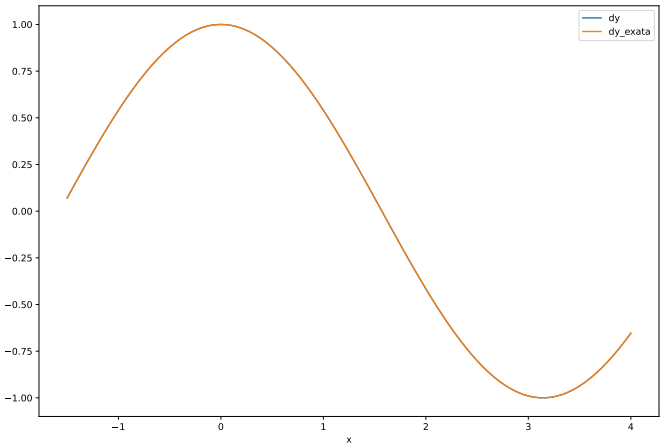


Figura 10.2: Gráfico da derivada de $f(t) = \sin(\theta)$ por aproximação numérica e função exata

10.3 Método de Newton-Raphson

3. Calcule uma das raízes de $f(x) = x^2 + 3\,x + 1$ através do método de Newton-Raphson. Utilize a derivada central para o cálculo de $f'(x)$.

O método de Newton-Raphson é um método iterativo, no qual a aproximação da derivada é calculada de acordo com o valor da aproximação atual da raiz x_i, do valor da função em x_i, $f(x_i)$ e da derivada da função, também em x_i, $f'(x_i)$, de acordo com a expressão: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$. (vide 10.1)

```
import numpy as np
f = lambda x0: x0**2+3*x0+1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)

x = [10]
for i in range(10):
  x.append(x[-1] - f(x[-1])/df(x[-1]))

import pandas as pd
dados = pd.DataFrame({'x':x, 'f':map(f,x), \
                      'df':map(df,x)})
```

Tabela 10.1: Tabela de Cálculo da Raiz de $f(x) = x^2 + 3x + 1$ através do método de Newton-Raphson

x	f	df
10.0000000	131.0000000	23.000000
4.3043478	32.4404537	11.608696
1.5098518	7.8092080	6.019704
0.2125773	1.6829211	3.425155
-0.2787643	0.2414166	2.442471
-0.3776054	0.0097696	2.244789
-0.3819575	0.0000189	2.236085
-0.3819660	0.0000000	2.236068
-0.3819660	0.0000000	2.236068
-0.3819660	0.0000000	2.236068
-0.3819660	0.0000000	2.236068

10.4 Gráficos da derivada

4. Sabendo que

$$x^3 + y^3 = 2xy \tag{10.1}$$

trace o gráfico de $\frac{dy}{dx}$.

Primeiro, podemos traçar o gráfico da função implícita $y = f(x)$ representada pela equação acima através da `sympy` e da função `plot_implicit`.

```
import sympy as sp
from sympy.plotting import plot_implicit
x, y = sp.symbols("x y")
f = x**3 + y**3 -2*x*y
g = plot_implicit(f, x_var=(x,-1.2,1.2), \
                     y_var=(y,-1.2,1.2));
```

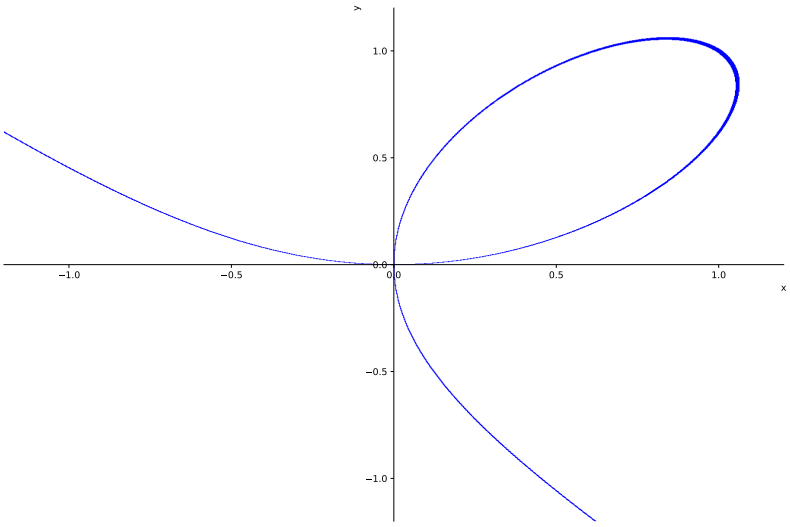


Figura 10.3: Gráfico da Função Implícita $x^3 + y^3 = 2xy$

O desenho do gráfico em 10.3 é obtido através de matemática intervalar. Ao invés de calcular um único par (x, y) e desenhar um ponto nas suas coordenadas, a função `plot_implicit` calcula pequenos retângulos e "pinta" o interior dos mesmos. O que pode-se fazer é calcular o ponto médio de cada sub-intervalo (tanto no eixo dos x quanto no eixo dos y) e considerar estes valores como os "verdadeiros" pares (x, y). Para obter os intervalos a partir dos valores armazenados na variável `g` fazemos:

```
intervs, _ = g[0].get_points()
intervs = np.array(intervs, dtype='object')
```

Convertemos a variável `intervs` para um array `numpy` o que torna sua manipulação mais fácil (acima). Em seguida vamos extrair de cada intervalo seu valor médio (através do método `.mid`) e salvamos o resultado em um data frame pandas.

```
import numpy as np
dados = [(x0.mid, y0.mid) for x0, y0 in intervs]
dados = pd.DataFrame(data = dados, columns=['x','y'])
dados.head()
```

```
          x         y
0 -1.177734  0.598828
1 -1.175390  0.596484
2 -1.175390  0.598828
3 -1.173046  0.596484
```

```
4 -1.173046   0.598828
```

Todo este procedimento pode ser compactado em uma única função Python, a qual recebe uma função implícita e os intervalos de cálculo para x e y e devolve o data frame correspondente.

```
import sympy as sp
from sympy.plotting import plot_implicit
x, y = sp.symbols("x y")
f = x**3 + y**3 -2*x*y
interv_x = (x,-1.2,1.2)
interv_y = (y,-1.2,1.2)

import pandas as pd
def gera_xy(Eq0, intervX, intervY):
  g = sp.plot_implicit(Eq0, x_var = intervX, \
                       y_var = intervY, \
                       show = False);
  intervs, _ = g[0].get_points()
  intervs = np.array(intervs, dtype='object')
  ptos = [(x0.mid, y0.mid) for x0, y0 in intervs]
  dados = pd.DataFrame(data = ptos, \
                       columns = ['x','y'])
  return(dados)

dados = gera_xy(f, interv_x, interv_y); dados.head()
```

```
          x         y
0 -1.177734  0.598826
1 -1.175391  0.596482
2 -1.175391  0.598826
3 -1.173047  0.596482
4 -1.173047  0.598826
```

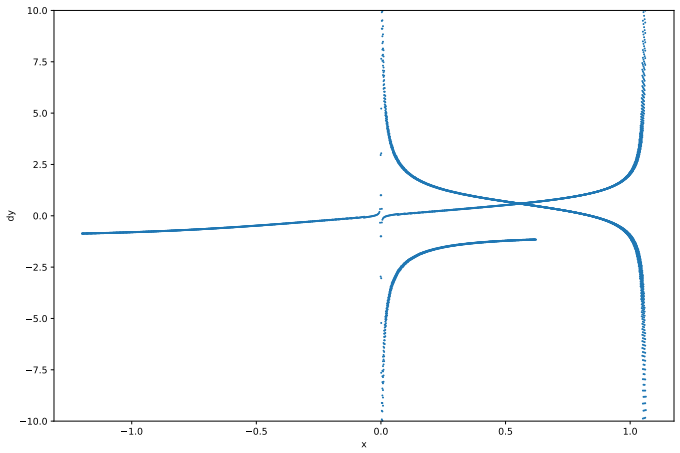
A derivada da expressão em 10.1 pode ser obtida através da função `idiff` da `sympy`.

```
import sympy as sp
df = sp.idiff(f, y, x, 1); df
```

```
(3*x**2 - 2*y)/(2*x - 3*y**2)
```

Com esta expressão e os valores disponíveis no data frame pandas, podemos calcular os valores da derivada em cada par (x, y) e em seguida plotar o seu gráfico. Por questões de desempenho, vamos criar uma função numérica com a mesma expressão de `dEq` para realizar as substituições de valores

```
df = lambda x0,y0:(3*x0**2 - 2*y0)/(2*x0 - 3*y0**2)
x, y = dados['x'].values, dados['y'].values
yL = [df(x0,y0) for x0, y0 in zip(x, y)]
dados['dy'] = np.array(yL, dtype='float')
g = dados.plot(x='x', y='dy', kind='scatter', s=1);
plt.ylim(-10,10);
plt.show();
```



1. Mostre que a inclinação da equação logística: $y = \frac{1}{1+e^{-ax}}$ no seu ponto intermediário é igual a $a/4$.

Vamos resolver este problema de forma numérica. Na listagem a seguir são apresentadas funções Python para cálculo dos valores da função logística e de sua derivada em um ponto x qualquer e para valores a escolher do parâmetro a. Como pode ser visto, sempre que a derivada é calculada para $x = 0$, seu valor coincide dentro de uma razoável precisão com o valor $a/4$, para a variando no intervalo $[-2, 2]$.

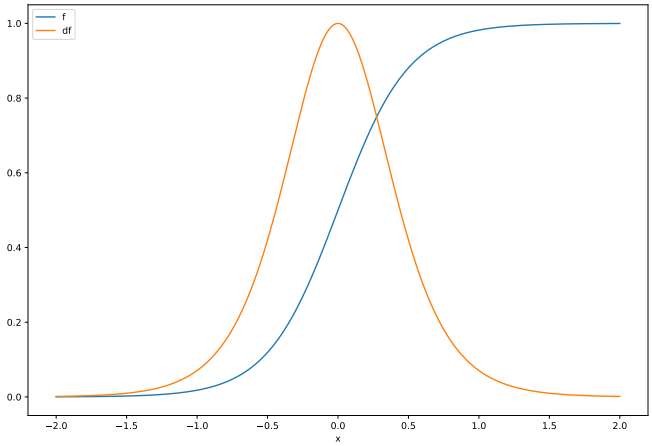
```
import numpy as np
a = np.linspace(-2,2,100);
f = lambda x0,a: 1/(1+np.exp(-a*x0));
df = lambda x0,a,h=0.0001: (f(x0+h,a)-f(x0-h,a))/(2*h)

import pandas as pd
dados = pd.DataFrame({'df(0,a)':df(0,a), 'a/4':a/4})
dados.head()
```

```
    df(0,a)       a/4
```

```
0 -0.500000 -0.500000
1 -0.489899 -0.489899
2 -0.479798 -0.479798
3 -0.469697 -0.469697
4 -0.459596 -0.459596
```

O gráfico da função logística e de sua derivada, para $a = 4$ são apresentados a seguir:

```
import numpy as np
x = np.linspace(-2,2,200)
f = lambda x0: 1/(1+np.exp(-4*x0))
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
dados = pd.DataFrame({'x':x, 'f':f(x), 'df':df(x)})
g = dados.plot(x='x', y=['f', 'df']);
plt.show();
```



10.5 Derivadas Parciais

Se a função tiver mais de uma variável independente, por exemplo $z = f(x, y)$ teremos taxas de variação independentes, uma para cada variável. Simbolicamente dizemos $z_x = \frac{\delta z}{\delta x}$ e $z_y = \frac{\delta z}{\delta y}$. O cálculo é muito parecido com o de uma derivada em uma única variável. Quando estivermos derivando em relação à x por exemplo, todas as outras variáveis independentes y por exemplo serão consideradas constantes.

Podemos ter derivadas segundas na mesma variàvel z_{xx} ou z_{yy} por exemplo ou em variáveis alternadas z_{xy}. Um resultado interessante é que, para funções

contínuas, $z_{xy} = z_{yx}$, ou seja, a ordem da derivação não influência o resultado final do cálculo da derivada.

10.6 Exercícios

10.6.1 Cálculo de Derivadas

Para os exercícios a seguir, calcule as derivadas das funções a seguir através a) da definição de derivada (utilizando a fórmula $f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x)-f(x)}{\Delta x}$, b) do método algébrico (utilizando as fórmulas básicas $f(x) = ax^n \rightarrow f'(x) = anx^{n-1}$ e $f(x) = be^{ax} \rightarrow f'(x) = abx^{ax}$) e c) da função `limit` da `sympy`.

1. $3x$
2. x^x
3. x^4
4. $\frac{1}{x}$
5. $\sqrt{x}$
6. $\ln(3x)$
7. $e^{\sin(x)}$
8. $\tan(x)$
9. $\frac{1}{\sin(x)}$
10. $\sqrt{x}$
11. e^{ax}
12. $e^x.x^2$
13. $e^x.ln(x)$
14. $e^{3x}.ln(x)$
15. $e^x.ln(2x)$
16. $\ln(1/x)$
17. $\ln(3/x^2)$
18. $\ln(\frac{1}{x^3})$
19. $\ln(\frac{4}{x^2})$
20. $\frac{\sqrt{x}}{ln(x)}$
21. $\frac{\sqrt{3x}}{ln(x)}$
22. $\frac{\sqrt{x}}{ln(2x)}$

23. $\frac{1}{(x^2+2)^3}$
24. $\frac{e^x}{x+1}$
25. $\frac{e^{2x}}{x+3}$
26. $\frac{3}{(x^2-1)^8}$
27. $(3x^2+x-2)^2$
28. $\frac{1}{x^{1/7}}$
29. $x^4+\frac{x^3}{3}+\frac{x^2}{4}+2x-1$
30. $\sqrt{x}+\sqrt[3]{x^2}$
31. $(t^2+5t-14)^4$
32. $(t^4+2t)^2.(1-t^2)$
33. $ln(\frac{x}{3+5x})+e^{(2x+1)^2}$
34. $\ln(2x^2+x).(4x+1)^2$
35. $\tan(t+4)$
36. $\frac{(x^2-x)^3}{(2x-1)^4}$
37. $\sin(\frac{u^2-1}{u+1})$
38. $-x^4+\frac{x^3}{3}-\frac{x^2}{4}+1$
39. $\sqrt{4x+1}-\sqrt[5]{x^3}$
40. $(t^4+5t+1)^3$
41. $(t^3+2t)(2+3t^2)$
42. $\frac{sin(x+1)}{\ln(2x-1)}$
43. $\ln(2x^2-3x)-e^{(4x-3)^2}$
44. $\ln(\frac{x^2-x}{2x+5})$
45. $\frac{(x^2-x)^7}{(2x-1)^4}$

10.6.2 Valor Aproximdado de Funções e Derivadas

1. Calcular a derivada da função $g(x)=(\frac{1}{x}+3).e^{2x}$ e determine o valor da derivada em t = 1,7.
2. Calcule o valor aproximado de $\frac{d\left[e^{2x}\right]}{dx}$ com h=0,1, no ponto x=2 ou o valor exato por derivação algébrica

10.6.3 Cálculo de Raízes de Equações - Método de Newton-Raphson

1. Partindo de $x = 0,1$, calcule duas outras aproximações para uma das raízes da função $f(x) = e^{-2x}\cos(2\pi x)$ através do método de Newton-Raphson.
2. Partindo de $x = 0$, use o método de Newton-Raphson para calcular uma aproximação para uma das raízes de $(x) = x^3 - 2x^2 + x - 1$ com pelo menos quatro dígitos de precisão.
3. Encontre a solução da equação $x^3 + 2x^2 + x + 1 = 0$ utilizando o método de Newton-Raphson. A solução encontrada deverá ter $\Delta x\% < 0,1\%$.
4. Resolver numericamente os problemas abaixo através do método de Newton-Raphson. As soluções devem ser obtidas com um erro percentual ϵ menor que $0,01\%$.
   (a) Calcular o valor de $\sqrt{5}$
   (b) Calcular as raízes de $x^2 + 4\,x + 3 + \sin(x) - x\,\cos(x)$

10.6.4 Aplicações em Geometria

1. Encontre o coeficiente angular da reta tangente à curva $y = x^3$ no ponto $(4, 64)$ e escreva a equação desta reta.
2. Em que ponto da curva $y^2 = 2x^3$ a tangente é perpendicular à reta $4x - 3y + 2 = 0$?
3. Calcule a distância entre os pontos para os quais as curvas $y = 4x^2 + 2x - 8$ e $y = x^3 - x + 10$ são paralelas entre si.
4. Calcule a área do triângulo formado pela reta tangente à circunferência $x^2 + y^{=}1$ no ponto $x = 0,5$ e os eixos coordenados.
5. Ar esta sendo bombeado continuamente para dentro de um balão esférico a uma taxa de $5cm^3/min$. Determine a taxa com a qual cresce o raio do balão quando seu diâmetro é de 20 cm.

10.6.5 Derivação Implícita

1. Suponha que você tenha a seguinte equação: $x^2 + y^2 = 1$. Sem isolar a variável y qual a expressão de $\frac{dy}{dx}$?
2. Calcule a derivada de y em relação à x na expressão $y^3 = x^2 - 1$ de forma implícita.
3. Calcular $\frac{dy}{dx}$ sabendo que $x^3 + y^3 = a^3$
4. Uma partícula se move sobre uma curva implicitamente definida por: $xy^4 - yx^4 = x - y^2$. Quando a partícula passa pelo ponto $(1, 1)$, sua

coordenada x está mudando 1/4 de unidade por segundo. A que taxa a coordenada y está mudando nesse ponto?

5. Calcule usando derivadas implícitas a derivada $\frac{dy}{dx}$ de $3y^2 + 2x^2 = R^2$ onde R é uma constante

10.6.6 Curvas de Nível e Derivadas Parciais

1. Desenhe as curvas de nível com z=0, 1 e 2 para as funções :
   (a) $z = f(x, y) = 2x - y$
   (b) $z = f(x, y) = -x^2 + 2y$
2. Calcule as expressões $\frac{\sigma x^3 y^2}{\sigma x}$ e $\frac{\sigma x^3 y^2}{\sigma y}$
3. Determine quais funções $f(x, y)$ a seguir satisfazem a equação $f_{xx} + f_{yy} = 0$
   (a) $f(x, y) = x^2 - y^2$
   (b) $f(x, y) = xy$
   (c) $f(x, y) = x.e^y - y.e^x$
   (d) $f(x, y) = ((x - 1)^2 + (y + 3)^2)^{(-\frac{1}{2})}$

10.6.7 Aplicações em Economia

1. A utilidade de um consumidor para adquirir x unidades de um produto e y unidades de um segundo produto é dada pela função de utilidade $U(x, y) = 6x^2y^3$. Suponha que o consumidor adquire mensalmente 2 unidades do primeiro produto e 2 unidades do segundo produto. Calcule $U(2, 2)$ e trace a curva de nível para esse valor.
2. Considere a função custo total $C(q) = q^2 - 4q - 2$. Com base nesta informação pede-se:
   (a) Calcule o custo marginal usando a definição de derivada. Calcule $C_{mg}(10)$ e interprete seu significado.
   (b) Determine a equação da reta tangente ao gráfico de C(q) nos pontos a = 2 e b = 4.
   (c) Encontre os intervalos de crescimento e decrescimento do custo total.
   (d) Mostre que a função custo total possui concavidade para cima em todo o seu domínio.
3. Seja $L(q) = q^3 - q^2 - q + 1$ a função lucro total para q unidades de pipoca vendidas pelo ambulante em frente à escola às quartas-feiras, conforme levantamento de mercado efetuado no primeiro semestre de 2008. O pipoqueiro conta com o talento e conhecimento matemático dos alunos do curso de Administração desta escola para conhecer:

(a) Para quais quantidades o lucro total é zero?

(b) Determine a função lucro marginal do pipoqueiro. Calcule e interprete $L_{mg}(5)$.

(c) Em qual(is) intervalo(s) a função lucro total é crescente? Qual quantidade minimiza o lucro do pipoqueiro?

(d) Em qual(is) intervalo(s) a função lucro total tem concavidade para cima?

(e) Utilizando um limite apropriado, mostre o que acontece com o lucro total quando o número de pipocas vendidas cresce infinitamente.

(f) Baseando-se nas descobertas dos itens anteriores, esboce o gráfico do lucro total em função da quantidade de pipocas vendidas.

4. Considere um país que tenha gastos anuais com saúde (em bilhões de dólares) dados pela função $f(t) = 27e^{0,106t}$, onde $t = 0$ para o ano de 1960. Calcule:

   (a) Qual o montante de gastos em 1972?

   (b) Qual a taxa de crescimento dos gastos em 1976?

   (c) Quando o país atingiu o montante de gastos de 120 bilhões de dólares?

   (d) Quando a taxa de crescimento era de 20 bilhões de dólares por ano?

5. Em cada dia a produção de uma mina de carvão, após t horas de operação, é de aproximadamente $p(t) = 40t + t^2 - -1/15t^3$ toneladas, para $0 <= t <= 12$. Qual é a taxa de produção (em toneladas de carvão por hora) em t = 5 horas?

6. Qual o significado de a) custo marginal, b) receita marginal e c) lucro marginal?

7. Uma fábrica produz $P(x, y) = 60x^{0,75}y^{0,25}$ unidades de um determinado produto por semana, onde x é a quantidade de trabalho (em homens-hora) e y é a quantidade de máquinas utilizadas. Com base nisto pede-se: a) determinar a produção para 81 unidades de mão de obra e 16 máquinas, b) se cada unidade de trabalho for remunerada pela sua produtividade marginal, determine a expressão para o valor do trabalho (isto é para os salários) e c) determine o valor dos salários quando forem utilizadas 81 unidades de mão de obra e 16 máquinas.

8. O lucro diário (em dólares) de um varejista que vende duas marcas de produtos similares é dado por $P(x, y) = (x - 40)(55 - 4x + 5y) + (y - 45)(70 + 5x - 7y)$, onde xé o preço unitário do primeiro produto e yo preço unitário do segundo produto. No momento a primeira marca vende por \$0,70 a unidade e a segunda marca por \$0,73 a unidade. Com base nestes dados pede-se:

(a) Encontrar a funções de lucro marginal Px e Py.

(b) Determinar o valor de ambos os lucros marginais nos níveis atuais de preço

(c) Utilizar o conceito de derivadas parciais para estimar a mudança no lucro diário que irá resultar se o varejista decidir aumentar o preço do primeiro produto em \$0,01 e o preço do segundo produto em \$0,02 simultaneamente.

9. Suponha uma função de demanda (preço como função da quantidade) de um determinado produto seja $P(q) = \frac{45}{ln(q)}$, determine a função faturamento marginal e o faturamento marginal para $q = 20$. (Atenção: pede-se o faturamento MARGINAL e não apenas o faturamento simples).

10. Seja x a quantidade de carros fabricados por uma montadora e $f(x)$ o custo em dólares para fabrica-los Qual o significado de $f(1000) = 15.000.000$ e $f'(1000) = 10.000$. Qual o custo marginal quando a quantidade fabricada é de 1000 carros.

11. O custo de fabricação de x unidades de um determinado produto é dado pela expressão $C(x) = 0,2x^3 - 0,3x^2 + 200x + 200$. Sabendo que a receita é dada pela expressão $R(x) = -4x^2 + 500x$, obtenha:

    (a) Uma aproximação para o custo de fabricação da 21ª unidade, a partir do custo de fabricação de 20 unidades e do valor do custo marginal em x=20.

    (b) Uma aproximação para a receita proveniente da 15ª unidade, a partir da receita total de 14 unidades, mais o valor da receita marginal em x=14

    (c) O valor de x que fará o custo marginal ser igual à receita marginal.

    (d) Uma interpretação para o valor de x obtido no ítem anterior.

12. Uma estimativa mostra que a produção mensal de e-books da obra "Vidas Secas" de Graciliano Ramos é dada pela função: $P(x, y, z) = 900x + 360y + 2x^2y + 10\text{yz} + 12x\sqrt{z}$ (em unidades), em que x é o número de editores, y o número de trabalhadores especializados e z o número de trabalhadores não especializados. No momento, a mão de obra disponível é constituída por 2 editores, 18 trabalhadores especializados e 36 trabalhadores não especializados. A editora decidiu contratar mais um único funcionário que pode ser um editor, um trabalhador especializado ou um trabalhador não especializado. Utilize o conceito de derivadas parciais para encontrar a melhor solução para a editora visando aumentar a produção.

13. Estima-se que a produção semanal de certa fábrica é dada pela função $Q(x, y) = 1200x + 600y + 2x^2y - x^3 - 0,5y^2$ unidades, sendo x o número de operários especializados e y o número de operários não especializados.

No momento, a mão de obra disponível é constituída por 20 operários especializados e 60 operários não especializados.

(a) Use o conceito de derivadas parciais para estimar a variação na produção se 1 operário não especializado for contratado e o número de operários especializados permanecer constante.

(b) Calcule a diferença exata: $Q(20, 61) - Q(20, 60)$

(c) Qual a diferença percentual entre os valores obtidos em a) e b) (Obs: utilize como valor de referência o resultado exato, isto é, da letra b).

14. Uma editora dispõe de R$ 60 000,00 para investir na produção e propaganda do e-book de Graciliano Ramos. Estima-se que se forem gastos x milhares de reais na produção e y milhares de reais na propaganda, aproximadamente $f(x, y) = 20x^{1,5}y$ unidades de e-books serão vendidas. Use o Multiplicador de Lagrange para analisar quanto a editora deve investir na produção e na propaganda para que o número de e-books vendidos seja o maior possível.

15. Atualmente 1.800 pessoas por dia utilizam uma certa linha de trem para ir trabalhar pagando $4 por passagem. O número de pessoas q dispostas a tomar o trem quando o preço da passagem é p é $q = 600(5 - \sqrt{p})$. A companhia que opera o sistema de transporte deseja aumentar o seu faturamento. a) A demanda é elástica ou inelástica quando $p = 4$? b) O preço deve ser aumentado ou não? Por quê?

16. Suponha que a demanda de um certo metal seja q=100-2p, onde p é o preço por quilograma e q é a quantidade demandada (em milhares de toneladas). (Dica: lembre-se que elasticidade é a taxa de variação de uma função em relação a si própria, acrescida de um sinal – (negativo).) Responda:

(a) Qual quantidade é demandada a $30 o quilo?

(b) Qual a função E(p) (Elasticidade preço)

(c) Qual a elasticidade em p=30. Interprete o resultado.

(d) Qual a elasticidade em p=20. Interprete o resultado.

17. Considere dois bens X e Y com preços p e q, respectivamente. Sejam $f(p, q) = a/p^2q$ a demanda para o bem X e $g(p, q) = b/(pq)$ a demanda para o bem Y. Descubra as condições sobre a e b para que os bens X e Y sejam substitutos e também as condições para que sejam complementares

18. Considere as funções demanda de dois produtos W e Z, dadas pelas equações $q_W = \frac{7p_Z}{1+8p_W}$ e $q_Z = \frac{2\sqrt{p_W^5}}{7+p_Z^3}$, onde p_W e p_Z são os preços unitários de W e Z.

(a) Calcule as demandas marginais parciais $\frac{\partial q_W}{\partial p_W}$, $\frac{\partial q_W}{\partial p_Z}$, $\frac{\partial q_Z}{\partial p_W}$, $\frac{\partial q_Z}{\partial p_Z}$.

(b) Os produtos W e Z são substitutos ou complementares? Justifique.

(c) Se existe um terceiro produto U com demanda dada pela equação $q_U = 117 - 7p_U + 2p_W$ onde p_U é o preço unitário de U, qual das opções abaixo reduz mais a demanda de U:

i. Diminuir em uma unidade o preço de W

ii. Aumentar em uma unidade o preço de U.

19. Dadas as funções demanda de dois produtos, determine se eles são substitutos ou complementares

(a) $D_1 = 500 - 6p_1 + 5p_2$, $D_2 = 200 + 2p_1 - 5p_2$

(b) $D_1 = 1000 - 0,02p_1^2 - 0,05p_2^2$, $D_2 = 800 - 0,001p_1^2 - p_1p_2$

(c) $D_1 = 200p_1^{-1/2}p_2^{-1/2}$, $D_2 = 300p_1^{-1/2}p_2^{-3/2}$

(d) $D_1 = 2000 + \frac{100}{p_1+2} - 25p_2$, $D_2 = 1500 - \frac{p_2}{p_1+7}$

(e) $D_1 = \frac{7p_2}{1+p_1^2}$, $D_2 = \frac{p_1}{1+p_2^2}$

(f) $q_W = \frac{2p_Z}{3+5p_W}$ e $q_Z = \frac{\sqrt[3]{P_w}}{2+p_Z^2}$

20. Uma notícia é transmitida pelos meios de comunicação de massa para uma audiência potencial de 50.000 pessoas. Após t dias $f(t) = 50.000(1-e^{-0.3t})$ pessoas tomaram conhecimento da notícia.

(a) Quantas pessoas terão conhecimento da notícia após 10 dias?

(b) Com qual taxa a notícia está se espalhando inicialmente?

(c) Quando 22.500 pessoas terão conhecimento da notícia?

(d) Aproximadamente quando a notícia estará sendo difundida a uma taxa de 2.500 pessoas por dia?

(e) Com que taxa a notícia estará se espalhando, quando metade da audiência já tiver conhecimento a respeito?

21. Uma notícia é espalhada boca a boa para uma audiência potencial de 10.000 pessoas. Após r dias, $f(t) = \frac{10.000}{1+50e^{-0.4t}}$ pessoas tinham conhecimento da notícia. Com qual taxa a notícia estará se espalhando quando metade da audiência potencial já tenha ouvido a notícia?

22. Quando o júri considerou o prefeito de uma certa cidade culpado por receber comissões ilegais, os jornais, rádio e televisão começaram imediatamente a veicular a notícia. No transcorrer de uma hora, um quarto dos cidadãos já tinha conhecimento da decisão. Estime quando três quartos da cidade terão conhecimento da notícia, sabendo que $P'(t) = k.[1 - P(t)]$, onde $P(t)$ é a proporção de habitantes da cidade que sabem da notícia no instante t e que no instante inicial nenhuma pessoa tinha conhecimento da notícia. Dica: Este problema pode ser expresso da seguinte forma: $P(0) = 0$; $P(1) = 0,25$;

$P'(t) = k.[1--P(t)]$. Utilize a fórmula $P(t+h) = P(t)+P'(t).h$ e encontre t para $P(t) = 0,75$. Utilize $h = 0,1$.

23. A proporção de pessoas que sabem de uma notícia após sua divulgação é dada por $P(t) = 1 - e^{-kt}$. Em uma determinada cidade, três horas após a divulgação de uma notícia, 25% dos habitantes da mesma já sabiam do ocorrido. Com base nesta informação, calcule quando 80% dos habitantes terão tomado conhecimento da mesma.

24. Sabe-se que a velocidade de difusão de uma informação é proporcional ao percentual de pessoas em um grupo que ainda não tomaram conhecimento da mesma. Em um grupo, demorou 4 horas para que 30% das pessoas tomassem conhecimento de uma notícia. Com base nestes dados, e utilizando o modelo descrito acima, pede-se determinar através da resolução da equação diferencial correspondente (de forma algébrica), quando 95% das pessoas deste grupo terão tomado conhecimento da referida notícia.

10.6.8 Exercícios de Séries

1. Sendo $S = 1 + 1/2 + 1/3 + 1/4 + ... + 1/N$, crie uma função Python que recebe n e retorna S com n termos. Se $n = 100$ o resultado deve ser 5.1972785077386305.

2. Sendo $S = 1/1 + 2/3 + 3/5 + 4/7 + 5/9 + ... + n/m$, crie uma função Python que recebe n e m e retorna o valor de S. Se forem calculados os 100 primeiros termos desta série o valor de S deve ser 52.14465865683989.

3. Crie uma função Python que recebe um número n e em seguida retorna os n primeiros termos da série de Fibonacci. A série de Fibonacci começa com 0 e 1 e o próximo termos é sempre a soma dos dois termos anteriores.

4. Crie uma função Python que recebe um número n e em seguida imprime a razão entre dois termos da série de Fibonacci do tipo $\frac{n(i+1)}{n(i)}$.

5. Crie uma função Python que recebe um número n e retorna os n primeiros números primos

6. Crie uma função Python que recebe um número n e retorna os n primeiros números primos que pertencem à série de Fibonacci.

7. Crie uma função Python que recebe um número n e retorna o valor de e calculado através da série de potências associada. OBS: $e^x \approx x^0/0! + x^1/1! + x^2/2! + x^3/3! + x^4/4! + ... + x^n/n!$.

8. Crie uma função Python que recebe um número n e retorna o valor de sen(1) onde sen(1) é dado pela série de potências associada. OBS: $sen(x) \approx x - x^3/3! + x^5/5! - x^7/7! + ... + \frac{(-1)^n x^{2n+1}}{(2n+1)!}$

9. Crie uma função Python que recebe um número n e retorna o valor decos(1) onde cos(1) é dado pela série de potências associada. OBS:

$cos(x) \approx 1 - x^2/2! + x^4/4! - x^6/6! + ... + \frac{(-1)^n x^{2n}}{(2n)!}$

10. O python pode trabalhar com números imaginários. Um número imaginário j é definido como $j = \sqrt{-1}$. Para realizar operações que envolvam a criação de números complexos a partir da função raiz quadrada importe o pacote cmath. Em seguida crie um programa que irá calcular o valor de e^j através da série de potências associada com 30 termos. Compare o resultado obtido pela série de potências com os valores obtidos para $cos(1)$ e $sen(1)$. O que pode-se concluir por estes resultados?

11. Crie uma função Python que recebe um número n e retorna a soma dos n primeiros termos de uma progressão aritmética de razão x.

12. Crie uma função Python que recebe um número n e retorna a soma dos n primeiros termos de um progressão geométrica de razão x.

13. Crie uma função Python que recebe um número n e retorna a soma dos termos de uma progressão geométrica, a qual irá começar em 1 e terá como razão x onde $x < 1$.

14. Crie uma função Python que recebe um número n e retorna a seguinte soma: $S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}$.

10.6.9 Séries Infinitas

1. Apresente o limite da série $S = R + R^2 + R^3 + R^4 + R^5 + ...$ para $R = \frac{1}{2}$

2. Considere a série $S = LR + LR^2 + LR^3 + LR^4 + LR^5 + ...$ onde L é o lucro recebido por período de uma determinada ação, S o valor atual da série de recebimentos infinita e $R = \frac{1}{1+i}$, onde i é a taxa de desconto (ou de juros). Responda as questões a seguir:

   (a) Qual seria condição para que P fosse um preço justo para a ação (em termos de S)

   (b) Seja P o preço pago pela ação. Qual o efeito esperado em P de um aumento de i? Demonstre sua afirmação em termos matemáticos.

3. Suponha que você compre uma ação por P=R$100,00 e que esta ação irá dar um lucro de L=R$ 10,00 todo o ano indefinidamente. Nestas condições o valor presente dos primeiros R$10,00 será $\frac{10}{1+i}$, dos segundos R$10,00 será $\frac{10}{1+i}^2$ e assim por diante. Encontre, utilizando a soma dos termos de uma série geométrica a relação entre P, L e i, e calcule o valor de i para o caso acima.

4. Considere uma perpetuidade que promete pagar US$ 100 no início de cada mês. Suponha que a taxa de juros seja de 12% compostos mensalmente. Então o valor presente de US$100 em k meses é $100(1,012)^{-k}$. Encontre a soma da série infinita.

5. Calcule o total de novos gastos criados pelo corte de US$ 10 Bilhões na arrecadação do Imposto de Renda, quando a propensão marginal ao consumo é de 95%. Se a propensão marginal ao consumo é de 96%, em quanto aumentam os novos gastos?

6. Calcule o efeito de um corte de US$ 20 bilhões no total do Imposto de Renda cobrado, quando a propensão ao consumomarginal da população é de 98%? Qual é o multiplicador neste caso?

7. Suponha que o Banco Central Americano compre de particulares US$ 100 milhões de obrigações do Governo Federal. Com isto ele introduz US$ 100 milhões de dinheiro novo e dá início a uma reação em cadeia por causa do sistema de encaixe monetário. Quando os US$ 100 milhões são depositados em contas bancárias, os bancos mantem 15% em reservas e podem emprestar os restantes 85% criando mais dinheiro novo, 0,85 $\times$ 100 milhões e assim sucessivamente, emprestando 0,85.0,85*100 milhões. Este processo repete-se por uma quantidade infinita de vezes. Calcule o montante total de dinheiro novo que pode ser criado, em teoria, por este processo além dos US$ 100 milhões originais.

10.6.10 Séries de Potências

1. Calcule através da expansão em série de potências de f(x), com pelo menos 5 termos não nulos, cada uma das funções f(x) a seguir:
   (a) e^x
   (b) e^{2x}
   (c) e^{-3x}
   (d) e^{3x}
   (e) e^{-x}
   (f) e^{2x}
   (g) $5e^{2x}$
   (h) e^{-2x}
   (i) $e^{-x/2}$
   (j) $\sin(x)$
   (k) $\cos(x)$
   (l) $\cos(2x)$
   (m) $\cos(-x)$
   (n) $\cos(3x)$
   (o) $\cos(\frac{\pi}{2} - 3x)$

(p) $\cos(\pi - 5x)$

(q) $\sqrt{4x+1}$

(r) $\frac{1}{x+2}$

(s) xe^{3x}

(t) $\sqrt{1-x}$

(u) $e^{2x}\sin(3x)$

10.6.11 Avançados

Os exercícios a seguir são de nível avançado. Podem ser ignorados em um curso inicial de cálculo:

1. Demonstre a fórmula de Euler, $e^{ix} = \cos x + isenx$
2. Calcule i^i
3. Sabe-se que $e^{i\pi} = -1$ e que $e^{i3\pi} = -1$. Isto não poderia levar-nos a concluir que $1 = 3$? Onde está o erro desta afirmação?
4. Qual o $\ln(-1)$?
5. Calcule o $\cos(ix)$. (Dica: expansão em série de potências).

Capítulo 11

Otimização

11.1 Otimização Numérica

Nos problemas de otimização buscamos encontrar o valor de uma ou mais variáveis independentes (x) as quais levem uma variável dependente y a atingir o maior ou o menor valor possível. Para tanto lançamos mão das relações entre as derivadas e a função original para determinar tal ponto com a máxima precisão possível.

As relações acima mencionadas são o fato de em seu ponto de máximo x_{max}, uma função $f(x_{max})$ terá primeira derivada nula (isto é igual a 0) $f'(x_{max}) = 0$ e segunda derivada negativa $f''(x_{max}) < 0$. Em um ponto de mínimo uma função terá também $f'(x_{min}) = 0$, porém $f''(x_{min}) > 0$. Se a função neste ponto x_{max} tiver tanto a primeira quanto a segunda derivadas nulas, dizemos que a mesma tem em x_{max} um ponto de inflexão. Com estes conhecimentos podemos partir para uma ampla gama de exercícios, como poderá ser visto nas seções a seguir.

1. Calcule o lucro máximo que poderá ser obtido em uma semana, por um fabricante o qual estima que o lucro ao produzir x unidades de uma comodity seja dado por $L(x) = -x^2 + 40x$ dólares/semana. Utilize a expressão da derivada central para a aproximação numérica da derivada. Lembre-se que no ponto de máximo da função sua derivada será igual a zero.

Como queremos calcular a raiz de $L'(x)$ através do método de Newton-Raphson, precisaremos dos valores da segunda derivada $L''(x)$.

Utilizaremos as seguintes aproximações numéricas para os valores de $f(x)$, $f'(x)$, $f''(x)$ 1. Para o cálculo da aproximação numérica seguinte da raiz : $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ 2. Para o cálculo numérico da derivada : $f'(x_i) = \frac{f(x_i+h)-f(x_i-h)}{2\,h}$ 3. Para o cálculo numérico da 2^a derivada : $f''(x_i) = \frac{f(x_i+h)+f(x_i-h)-2\,f(x_i)}{h^2}$

A sequência de cálculo pode ser vista na tabela 11.1. Como pode ser visto o resultado é $x = 20$ e $L(x) = 400$.

```
import numpy as np
import pandas as pd
f = lambda x: -x**2+40*x
df = lambda x,h=0.001: (f(x+h)-f(x-h))/(2*h)
df2 = lambda x,h=0.001: (df(x+h,h)-df(x-h,h))/(2*h)

x = [1]
for i in range(4):
  x.append(x[-1]-df(x[-1])/df2(x[-1]))

dados = pd.DataFrame({'x':x, 'f':map(f,x), \
                      'df':map(df,x), 'df2':map(df2,x)})
```

Tabela 11.1: Cálculo do ponto em que $L'(x) = 0$

x	f	df	df2
1	39	38	-2
20	400	0	-2
20	400	0	-2
20	400	0	-2
20	400	0	-2

Existem situações em que a função tem mais de um ponto de máximo e mínimo no seu domínio.

2. Calcule os pontos críticos da função $f(x) = -x^3 + 3.x^2 + 9.x - 1$.

Vamos criar uma função que executa o método de Newton-Raphson em busca da raiz da derivada de $f(x)$. Por default, a função irá executar a iteração 100 vezes com $h = 0.0001$ e irá retorna o valor de x que torna a derivada de $f(x)$ nula e os valores de $f(x)$, $f'(x)$ e $f''(x)$ neste ponto. Observe a seguir:

```
import numpy as np
f = lambda x0: -(x0**3) + 3 * (x0**2) + 9*x0 -1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
df2 = lambda x0,h=0.0001: (df(x0+h)-df(x0-h))/(2*h)
def newton(x0,n=100,h=0.0001):
  for i in range(n):
    x0 = x0 - df(x0)/df2(x0)
  return(x0, f(x0), df(x0), df2(x0))

np.round(newton(-20),5)
```

```
array([-1., -6.,  0., 12.])
```

O programa antes de ser executado teve como ponto inicial $x = -20$. Após a execução do mesmo obtivemos como resultados $x = -1$, $f(x) = -6$, $f'(x) = 0$ e $f''(x) = 12 > 0$. Logo para $x = -1$ temos $f(-1) = -6$, $f'(-1) = 0$, $f''(-1) = 12 > 0$. Dado que a primeira derivada é nula e a segunda positiva em $x = -1$ podemos dizer que $f(-1)$ será um mínimo local de $f(x)$. Trocando o valor inicial para $x = 20$ e reiniciando o processo iterativo chegaremos ao resultado mostrado a seguir. Podemos dizer então que $f(3) = 26$ é um máximo local de $f(x)$, pois $f'(3) = 0$ e $f''(3) = -12$.

```
import numpy as np
f = lambda x0: -(x0**3) + 3 * (x0**2) + 9*x0 -1
df = lambda x0,h=0.0001: (f(x0+h)-f(x0-h))/(2*h)
df2 = lambda x0,h=0.0001: (df(x0+h)-df(x0-h))/(2*h)
def newton(x0,n=100,h=0.0001):
  for i in range(n):
    x0 = x0 - df(x0)/df2(x0)
  return(x0, f(x0), df(x0), df2(x0))

np.round(newton(20),5)
```

```
array([  3.,  26.,   0., -12.])
```

11.2 Otimização Algébrica

A otimização pode ser calculada também de forma algébrica através da `sympy`.

3. Por exemplo, determine o mínimo de $f(x, y) = x + y$, quando $x.y = 100$ e $x > 0$ e $y > 0$.

Sem especificar as restrições de x e y maiores que 0 obtemos o resultado abaixo. Escolhemos a solução apenas com valores reais positivos para x e y.

```
import sympy as sp
x = sp.symbols("x", real=True)
y = 100/x
f = x+y
sp.solve(f.diff(x),x)
```

```
[-10, 10]
```

4. Suponha que 20.000 pessoas irão a um jogo de futebol se o preço do ingresso for de \$5,00 e que cada aumento de \$1.00 provocará uma diminuição na quantidade de torcedores de 500 pessoas. Quanto o preço deverá variar para maximizar a receita?

Para resolver, devemos primeiro encontrar a função demanda, a qual irá relacionar preço com quantidade. Podemos fazer isso supondo que p e x estejam em uma linha, a qual deverá passar pelos pontos $(p1 = 5; x1 = 20.000)$ e $(p2 = 6; x2 = 19.500)$. Isto irá formar a nossa restrição. Fazendo $R = p.x$ temos a função objetivo. Passando estes dados para o formato utilizado no exemplo anterior resolvemos facilmente o problema.

```
import sympy as sp
p = sp.symbols("p", real=True)
x = (19500 - 20000)/(6 - 5)*(p-5) + 20000
R = p*x
sp.solve(R.diff(p),p)
```

```
[22.5000000000000]
```

5. A equação de demanda de um monopolista é de $p = 1050 - 0.03x$ e a função de custo é $C(x) = 150x + 750.000$, onde x é o número de unidades produzidas. Com base nestes dados pede-se:

   (a) O valor de x que maximiza o lucro e o preço correspondente

   (b) Supondo que o governo reduza os impostos em \$1.000.000, de modo a aumentar a atividade econômica (provocando um aumento no lucro líquido do monopolista), qual é o resultado em termos de x e de p.

```
import sympy as sp
x, T = sp.symbols("x T")

p = 1050 - 0.03 * x
c = 150 * x + 750000
L = p * x - c + T

sp.solve(L.diff(x), x)
```

```
[15000.0000000000]
```

Os impostos são calculados através da variável T. Observe que o valor de x que maximiza o lucro do monopolista L é independente de T e igual a $x = 15000$. Sendo assim, uma redução dos impostos que afete apenas o lucro do monopolista, sem afetar diretamente a quantidade produzida pelo menos não terá efeito na atividade econômica. Mais um exemplo: O raio de uma esfera esta aumentando. Para qual valor de r, a taxa de aumento do volume dV/dt é igual a $64.\pi$ vezes a taxa de aumento de r?

Antes de iniciar a resolução do problema em planilha precisamos lembrar que $V = \frac{4}{3}\pi\, r^3$. Nosso problema agora é calcular o r para o qual $V' = 64\,\pi\, r'$. Porém $\frac{dV}{dt} = \frac{dV}{dr}\,\frac{dr}{dt} = \frac{dV}{dr}\, r'$. Como queremos $V' = 64\,\pi\, r'$, desejamos na verdade

calcular $\frac{dV}{dr}\frac{dr}{dt} = 64\,\pi\,\frac{dr}{dt}$. Simplificando o $\frac{dr}{dt}$ de ambos os lados da equação chegamos a conclusão que nosso problema final é calcular $\frac{dV}{dr} = 64\,\pi$. De posse destas relações montamos a sequência de comandos Python mostrada a seguir, ficando a resposta então $r = 4$ (quatro).

```
import sympy as sp
V, r = sp.symbols("V r")
V = 4/3*sp.pi*r**3
sp.solve(V.diff(r)-64*sp.pi, r)
```

```
[-4.00000000000000, 4.00000000000000]
```

6. Trace o gráfico de : $y = e^{-x^2}$ e de sua derivada no intervalo $x = [-3; 3]$ e encontre o ponto de máximo da derivada.

Para encontrar o máximo de $y'(x)$ devemos calcular o ponto x no qual $y''(x) = 0$. Porém para calcular uma das raízes de $y''(x)$ por um método de aproximação numérica tipo Newton-Raphson, devemos calcular $y'''(x)$ pois o método requer iterações do tipo :

$$x_{i+1} = x_i - \frac{f''(x_i)}{f'''(x_i)} \tag{11.1}$$

Na 11.1 é mostrado o gráfico de $y(x)$ e de $y'(x)$ e na 11.2 é feito o cálculo do máximo de $f'(x)$ (o que implica no cálculo da raiz de $f''(x)$, o que por sua vez requer o cálculo de $f'''(x)$).

```
import numpy as np
import matplotlib as mpl; mpl.use("TkAgg");
from matplotlib import pyplot as plt

x = np.linspace(-3,3,100);
f = lambda x0:np.exp(-x0**2);
df = lambda x0,h=0.0001 : (f(x0+h)-f(x0-h))/(2*h);

import pandas as pd
dados = pd.DataFrame({'x':x, 'y':f(x), 'dy':df(x)})
g = dados.plot(x='x', y=['y', 'dy'])
plt.show();
```

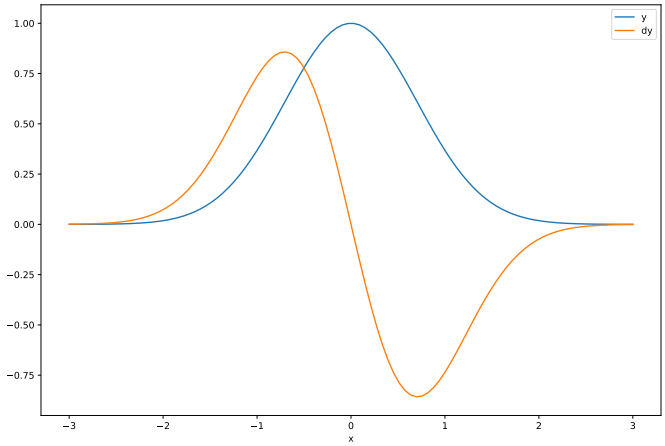


Figura 11.1: Gráfico de e^{-x^2} e de sua derivada

```
df2 = lambda x0, h=0.0001 : (df(x0+h)-df(x0-h))/(2*h)
df3 = lambda x0, h=0.0001 : (df2(x0+h)-df2(x0-h))/(2*h)

def newton(x=1, n=100):
  res = [x]
  res.append(res[-1] - df2(res[-1])/df3(res[-1]))
  return(res)

x = newton()

import pandas as pd
dados = pd.DataFrame({'x':x,
                      'y':map(f,x),
                      'dy':map(df,x),
                      'dy2':map(df2,x),
                      'dy3':map(df3,x)})
```

Tabela 11.2: Tabela para Cálculo do Ponto Máximo de dy

x	y	dy	dy2	dy3
1.0000000	0.3678794	-0.7357589	0.7357589	1.471510
0.4999975	0.7788027	-0.7787988	-0.7788104	3.893955

11.3 Otimização em Várias Variáveis

Assim como existe a teoria da otimização de funções de uma variável a partir do cálculo de derivadas, as funções de duas ou mais variáveis também podem ter seus pontos extremos calculados por técnicas similares.

No caso de funções de uma variável temos pontos de *máximo*, *mínimo* e *de inflexão*. No caso de funções de duas variáveis existem os pontos de *máximo*, *mínimo* e o chamado *ponto de sela*, no qual a função tem um máximo quando observada a partir de um dos eixos coordenados e um mínimo ao ser observada a partir do outro.

A forma algébrica de determinação de tais pontos é apresentada a seguir.

11.4 Pontos de máximo

Neste caso temos um ponto (x_0, y_0) no domínio de uma função $z = f(x, y)$ o qual atende às seguintes condições:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

$$z_{xx}(x_0, y_0) < 0$$

e

$$z_{yy}(x_0, y_0) < 0$$

E o determinante

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \end{vmatrix} > 0$$

11.5 Pontos de mínimo

Neste caso temos um ponto (x_0, y_0) no domínio de uma função $z = f(x, y)$ o qual atende às seguintes condições:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

$$z_{xx}(x_0, y_0) > 0$$

e

$$z_{yy}(x_0, y_0) > 0$$

E o determinante

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \\ \end{vmatrix} > 0$$

11.6 Pontos de sela

Neste caso temos um ponto (x_0, y_0) no domínio de uma função $z = f(x, y)$ o qual atende às seguintes condições:

$$z_x(x_0, y_0) = z_y(x_0, y_0) = 0$$

E o determinante

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \\ \end{vmatrix} < 0$$

11.7 Possíveis problemas

É muito raro, mas caso ocorra a situação em que o determinante:

$$D = \begin{vmatrix} z_{xx}(x_0, y_0) & z_{xy}(x_0, y_0) \\ z_{yx}(x_0, y_0) & z_{yy}(x_0, y_0) \\ \end{vmatrix} = 0$$

dizemos que o teste da segunda derivada falhou. Não serão estudados neste texto tais casos.

11.8 Exemplos

7. Analisar os pontos extremos da função: $f(x, y) = \frac{1}{x^2+y^2+3x-2y+1}$

Vamos realizar este exemplo a partir da `sympy`. Primeiro calculamos suas primeiras derivadas em relação a x e a y. Para z_x temos:

```
import sympy as sp
x, y = sp.symbols("x y")
z = 1/(x**2 + y**2 + 3*x -2*y +1)
sp.diff(z,x)
```

```
(-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
sp.diff(z,y)
```

```
(2 - 2*y)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

Calculamos agora (x_0, y_0) tal que $z_x(x_0, y_0) = 0$ e $z_y(x_0, y_0) = 0$.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
sp.solve([f.diff(x), f.diff(y)],[x,y])
```

```
{x: -3/2, y: 1}
```

Por garantia, calculamos os valores de x que tornam o denominador de z_x e z_y nulo para $y = 1$. Neste caso fazemos:

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1)
print("f = ", f);
```

```
f =  1/(x**2 + 3*x + y**2 - 2*y + 1)
```

```
dfx = f.diff(x);
print("dfx = ", dfx);
```

```
dfx =  (-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
dfx_y1 = dfx.subs(y,1)
print("dfx_y1 = ", dfx_y1);
```

```
dfx_y1 =  (-2*x - 3)/(x**2 + 3*x)**2
```

```
den_dfx_y1 = sp.fraction(dfx_y1)[1];
print("den_dfx_y1 = ", den_dfx_y1);
```

```
den_dfx_y1 =  (x**2 + 3*x)**2
```

```
sp.solve(den_dfx_y1,x)
```

```
[-3, 0]
```

Logo o ponto candidato é

$$(x, y) = (-\frac{3}{2}, 1)$$

Vamos agora calcular as derivadas segundas no ponto $(-3/2, 1)$. Começamos com z_{xx}. Neste caso:

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(x,2).subs(x,-3/2).subs(y,1)
```

```
-0.395061728395062
```

Calculamos agora, a segunda derivada em relação a y.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(y,2).subs(x,-3/2).subs(y,1)
```

```
-0.395061728395062
```

Em seguida obtemos o valor da derivada cruzada,

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f.diff(x).diff(y).subs(x,-3/2).subs(y,1)
```

```
0
```

Como último passo calculamos o determinante, o qual é maior que 0.

```
import sympy as sp
x, y = sp.symbols('x y');
f = 1/(x**2 + y**2 + 3*x -2*y + 1);
f_xx = f.diff(x,2)
f_yy = f.diff(y,2)
f_xy = f.diff(x).diff(y)
A = sp.Matrix([[f_xx, f_xy],[f_xy, f_yy]])
detA = sp.det(A)
detA.subs(x,-3/2).subs(y,1)
```

```
0.156073769242493
```

Este conjunto de resultados: $z_x(-\frac{3}{2},1) = 0$, $z_y(-\frac{3}{2},1) = 0$, $z_{xx}(-\frac{3}{2},1) = -\frac{31}{81}$, $z_{yy}(-\frac{3}{2},1) = -\frac{31}{81}$ e $D > 0$ nos permitem afirmar que o ponto $z(-\frac{3}{2},1)$ é um máximo.

8. Analisar os pontos extremos da função:

$$f(x,y) = xye^{\frac{-9y^2-16x^2}{288}}$$

A resolução em sympy pode ser vista a seguir. Primeiro calculamos os pontos candidatos, isto é, os pares de valores (x, y) para os quais $f_x = 0$ e $f_y = 0$

```
import sympy as sp
x, y = sp.symbols("x y")
f = x*y*sp.exp((-9*y**2-16*x**2)/288)
f_x = sp.diff(z,x);
print("f_x = ", f_x);
```

```
f_x =  (-2*x - 3)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
f_y = sp.diff(z,y);
print("f_y = ", f_y);
```

```
f_y =  (2 - 2*y)/(x**2 + 3*x + y**2 - 2*y + 1)**2
```

```
sols = sp.solve([f_x, f_y],[x,y]);
print("Ptos candidatos = ", sols)
```

```
Ptos candidatos =  {x: -3/2, y: 1}
```

Calculamos o valor do determinante de forma algébrica :

```
import sympy as sp
x, y = sp.symbols("x y")
f = x*y*sp.exp((-9*y**2-16*x**2)/288)
f_x = f.diff(x); f_y = f.diff(y);
f_xx = f.diff(x,2); f_yy = f.diff(y,2)
f_xy = f.diff(x).diff(y)
A = sp.Matrix([[f_xx, f_xy],[f_xy, f_yy]])
detA = sp.det(A); detA.simplify()
```

```
(-16*x**4*y**2 - 256*x**4 - 9*x**2*y**4 + 720*x**2*y**2 + 4608*x**2 - 81*y**
```

E os pontos de interesse (ou pontos candidatos):

```
sols = sp.solve([f_x, f_y],[x,y])
sols
```

```
[(-3, -4), (-3, 4), (0, 0), (3, -4), (3, 4)]
```

Em seguida criamos uma função para calcular o valor do determinante e das derivadas segundas em um ponto candidato e utilizamos a mesma para avaliar cada um deles.

```
def avalia(pto):
  x0 = pto[0]; y0 = pto[1]
  detAx0y0 = detA.subs(x,x0).subs(y,y0).evalf()
  fxx = f.diff(x,2).subs(x,x0).subs(y,y0).evalf()
  if detAx0y0<0:
    result = "sela"
  else:
    if fxx <0:
      result = "máximo"
    else:
      result = "mínimo"
  detAx0y0 = np.round(float(detAx0y0),5)
  fxx = np.round(float(fxx),5)
  return(pto, detAx0y0, fxx, result)
```

```
for sol in sols:
  print(avalia(sol))
```

```
((-3, -4), 0.54134, -0.98101, 'máximo')
((-3, 4), 0.54134, 0.98101, 'mínimo')
((0, 0), -1.0, 0.0, 'sela')
((3, -4), 0.54134, 0.98101, 'mínimo')
((3, 4), 0.54134, -0.98101, 'máximo')
```

9. Avaliar os pontos extremos da função:

$$f(x,y) = x.ln\left(\frac{y^2}{x}\right) - x.y^2 + 3.x$$

Neste exercício a tentativa de resolução do sistema de equações, diretamente pela `sympy` falha.

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
sp.solve([f.diff(x), f.diff(y)],[x,y])
```

```
[(x, -1), (x, 1)]
```

Se tentarmos resolver as equações de forma separada, conseguimos os pares de valores. Primeiro $f_y = 0$ para y. Obtemos como era de se esperar $y = \pm 1$

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
sp.solve(f.diff(y),y)
```

```
[-1, 1]
```

Substituindo y por -1 ou por $+1$ em $f_x = 0$ e resolvendo para x obtemos $x = e$:

```
sp.solve(f.diff(x).subs(y,-1),x)
```

```
[E]
```

Podemos então calcular as derivadas segundas de f e analisar os pontos $[e, -1]$ e $[e, +1]$. Abaixo calculamos a expressão $f_{xx}.f_{yy} - f_{xy}^2$

```
import sympy as sp
x, y = sp.symbols("x y", real = True)
f = x * sp.log((y**2)/x) - x*(y**2) + 3*x
fxx = f.diff(x,2)
fyy = f.diff(y,2)
fxy = f.diff(x).diff(y)
detA = fxx * fyy - fxy**2
detA.simplify()
```

```
-4*y**2 + 10 - 2/y**2
```

Avaliando as duas expressões em $(e, -1)$ vemos que este é um ponto de máximo.

```
detA.subs(x,sp.E).subs(y,-1), fxx.subs(x,sp.E).subs(y,-1).evalf()
```

```
(4, -0.367879441171442)
```

Da mesma forma, $(e, +1)$ também é um ponto de máximo.

```
detA.subs(x,sp.E).subs(y,1), fxx.subs(x,sp.E).subs(y,1).evalf()
```

```
(4, -0.367879441171442)
```

Isto só é possível se existir uma descontinuidade no intervalo que vai de $y = -1$ até $y = 1$. O resultado neste caso confere, pois em $y = 0$ a expressão $\ln(\frac{y^2}{x})$ tem um limite tendendo a $-\infty$.

```
import sympy as sp
x, y = sp.symbols("x y")
f = sp.log(y**2/x)
sp.limit(f,y,0)
```

```
-oo
```

10. Avaliar os pontos extremos da função:

$$\frac{x}{y^2 + x^2 + 4}$$

Para este problema vamos criar uma função Python que recebe uma expressão, calcula cada ponto crítico e determina se o mesmo é um ponto de sela, de mínimo ou de máximo.

```
import numpy as np
import sympy as sp
x, y = sp.symbols("x y")
z = x/(y**2 + x**2 + 4)

def avalia(f):
  fx = f.diff(x)
  fy = f.diff(y)
  ptos = sp.solve([fx, fy],[x,y])
  fxx= f.diff(x,2)
  fyy = f.diff(y,2)
  fxy = f.diff(x).diff(y)
  ptos = sp.solve([fx,fy],[x,y])
  results = []
  for pto in ptos:
    x0 = pto[0]
    y0 = pto[1]
    fxx0 = fxx.subs(x,x0).subs(y,y0).evalf()
    fyy0 = fyy.subs(x,x0).subs(y,y0).evalf()
    fxy0 = fxy.subs(x,x0).subs(y,y0).evalf()
    detA =  fxx0 * fyy0 - fxy0**2
    if detA < 0:
      result = "sela"
    else:
      if fxx0 < 0:
        result = "máximo"
      else:
        result = "mínimo"
    results.append([pto, detA.round(4), \
                    fxx0.round(4), result])
  return(results)

for solucao in avalia(z):
  print(solucao)
```

```
[(-2, 0), 0.0039, 0.0625, 'mínimo']
[(2, 0), 0.0039, -0.0625, 'máximo']
```

11.9 Exercícios de Máximos e Mínimos Univaridos

1. Dada a função $f\left(x\right) = 3x3 + x2 - 4x - 1$, utilize a fórmula: $f'\left(x\right) = \left[f\left(x+h\right) - f\left(x\right)\right]/h$, com $h = 0,1$ e encontre o valor de x que torna a derivada de $f\left(x\right)$ igual a 0.
2. Calcule os pontos de máximo e mínimo (se houver) das funções f(x) =
   (a) $x.\ln x$
   (b) $x - 1.\ln x$
   (c) $2x.\ln x$
3. Encontre os zeros, máximos, mínimos e plote o gráfico da função: $f(x) = \frac{1}{(x-0,1)^2+0,01} + \frac{1}{(x-1,2)^2+0,04} - 10$ no intervalo [-2;2]
4. Encontre o valor de x que torna $f(x) = -x^2 + 3x - 4$ um máximo, pelo método que achar mais conveniente
5. Calcule o máximo local para a função: $f(x) = x^3 - 5x^2 + 3x - 4$
6. Determine os coeficientes p e q na expressão $y = x^2 + px + q$ de forma que ele tenha um mínimo em $x = 1$ e $y = 3$
7. Dada a função $f\left(x\right) = 2x3 + x2 - 5x - 1$, utilize a fórmula $f'(x) = \frac{f(x+h)-f(x)}{h}$, com $h = 0,1$ e encontre o ponto de máximo da função $f(x)$.

11.10 Exercícios de Aplicação em Geometria

1. Uma pessoa encontra-se na borda de um lago circular de raio R. Ela pode nadar através do lago com velocidade V1 ou caminhar na sua borda com velocidade V2. Pede-se determinar a trajetória que irá levar a pessoa até a borda oposta do lago no menor tempo possível.
2. Qual o maior jardim que pode ser construído ao lado de uma casa com 40m de cerca?
3. Uma caixa de base quadrada, sem tampa, deve ter 12m2 de superfície total. Determine o comprimento da aresta da base e a altura da caixa de volume máximo.
4. Encontre a maior área que pode ser utilizada por uma pista de atletismo de 500m de perímetro, composta de dois semicírculos e duas pistas retas e paralelas
5. Um engenheiro está projetando um estádio esportivo para uma cidade e o campo retangular deve estar inscrito dentro de uma arquibancada representada em seu projeto pela seguinte elipse: $\frac{1}{16}x^2 + \frac{1}{9}y^2 = 1$ Quais as dimensões para que esse campo tenha a maior área possível? Prove

6. Um criador de gado está em um ponto A situado a 200 metros da margem de um rio. Ele deseja levar seu rebanho para beber água no rio e de lá quer seguir para um ponto de pastagem B situado a 400 metros da margem do rio. Suponha que a margem do rio seja em linha reta. A distância horizontal que separa o ponto onde o criador se encontra agora e o ponto de pastagem é de 1.000 metros. Pede-se calcular em qual ponto da margem o criador deverá levar o gado para beber água de modo que a distância total percorrida seja mínima.

7. Um arame de 10 cm de comprimento deve ser cortado em dois pedaços, um dos quais será torcido de modo a formar um quadrado e, o outro, a formar uma circunferência. De que modo deverá ser cortado para que a soma das áreas das regiões limitadas pelas figuras obtidas seja mínima?

8. 320 dólares estão disponíveis para serem gastos na construção da cerca em um jardim. A cerca no lado do jardim que faz frente com a rua custa \$6 por metro e a cerca nas outras três partes custa \$2 por metro. Calcule as dimensões do jardim que maximiza a área.

11.11 Exercícios de Aplicação em Economia

1. A função faturamento de uma firma que produz um único produto é: $R(x) = 200–1600/(x+8)–x$. Encontre o valor de x que resulta no faturamento máximo.

2. Uma firma que produz um único produto estima que sua função de custo diário (em unidades apropriadas) é $C(x) = x3–6x2+13x+15$, e sua função faturamento é $R(x) = 28x$. Encontre o valor de x que maximiza o lucro diário.

3. Considere a função lucro (L) como $L = R–C$ onde R é a receita e C o custo. Expresse a condição de lucro ótimo em termos do faturamento marginal e da receita marginal.

4. Após uma campanha publicitária, as vendas de um produto frequentemente aumentam e, após algum tempo, diminuem. Suponha que transcorridos t dias do fim da campanha, as vendas diárias sejam f(t) = -3tˆ2+9t+100. Qual é a taxa de crescimento das vendas no quarto dia? Após quantos dias as vendas atingiriam um máximo (ou mínimo)? Qual o valor das vendas neste momento? Este é um ponto de máximo ou mínimo local?

5. A equação de demanda de um certo bem é $p = (1/12)x^2–10x + 300$, para 0<=x<=60. Encontre o valor de x que corresponde ao preço que maximiza o faturamento.

6. Suponha que a equação de demanda de um monopolista é dada por $p = 150 - 0,02.x$, onde p é o preço, x é a quantidade vendida. Supondo que a função custo seja dada por $C(x) = 10x + 300$, encontre o valor de x

que maximiza o lucro. Prove que é um máximo calculando o valor da 2a derivada neste ponto.

7. Uma pesquisa de mercado indicou que a relação entre o preço do açucar (em dólares por quilograma) e a quantidade demandada (em milhares de toneladas) em um determinado país estão relacionados pela função: $p(x) = -0,25*x + 50$. Sabendo que o custo de produção $C(x)$ é dado pela expressão $C(x) = 2,25x^2 + 3x + 70$ pede-se: a) Montar em planilha a expressão do lucro esperado em função do preço praticado, isto é $L(p)$ e do lucro esperado em função da quantidade demandada, isto é $L(x)$ e b) Determinar o preço p e a quantidade x que tornam a produção de açucar o mais lucrativa possível

8. Alguns anos atrás foi estimado que a demanda para aço satisfazia aproximadamente à equação $p = 256 - 50x$ e o custo total para se produzir x unidades de aço era $C(x) = 182 + 56x$. (A quantidade x era medida em milhões de toneladas, e o preço e custo total eram medidos em milhões de dólares. Determine o nível de produção e o correspondente preço que maximiza o lucro

9. A equação de demanda mensal de uma companhia elétrica é estimada em: $p = 60 - (10 - 5).x$ em que p é medido em dólares e x é medido em milhares de quilowatts-hora. Os custos fixos da companhia são de \$7.000.000 por mês e os custos variáveis são de \$30 por 1.000 quilowatts-hora de eletricidade gerada, de modo que a função custo é: $C(x) = 7.106 + 30x$. Encontre o valor de x e o preço por 1.000 quilowatts-hora que maximiza o lucro da companhia.

10. Suponha que a função faturamento total de uma empresa é $R(x) = 300ln(x + 1)$, de forma que a venda de x unidades de um produto produz um retorno de $R(x)$ dólares. Suponha também que o custo total para se produzir x unidades é de $C(x)$ dólares com $C(x) = 2x$. Encontre o valor de x para o qual a função lucro $R(x) - C(x)$ será maximizada. Mostre que a função lucro tem um máximo relativo e não um mínimo relativo para um tal valor de x

11. Quando o preço médio de um ingresso para uma apresentação de ópera é de \$50 a audiência é de 4.000 pessoas. Quando o preço foi elevado para \$52, a audiência declinou para 3.800 pessoas. Considere a curva de demanda linear e com estes dados calcule o preço do ingresso que irá maximizar o faturamento do teatro.

12. O movimento médio de uma estrada é de 36.000 carros por dia quando o pedágio é de \$1 por carro. Uma pesquisa de opinião concluiu que o aumento do pedágio implicaria em 300 carros a menos para cada centavo de aumento. Qual o valor do pedágio para maximizar o faturamento?

13. Suponha que, em determinada rota, uma companhia aérea regional transporta 8000 passageiros por mês, cada um deles pagando \$50. A companhia

deseja aumentar a tarifa. Entretanto o departamento de pesquisa de mercado estima que para cada $1 de aumento, a companhia perderá 100 passageiros. Determine o preço que maximiza o faturamento.

14. Uma agência de viagens oferece um cruzeiro por várias ilhas do Caribe por um período de 3 dias e 2 noites. Para um grupo de 12 pessoas, o custo por pessoa é de $800. Para cada pessoa adicional, o custo é reduzido em $20 por pessoa. O número máximo de pessoas que pode ser considerado para o cruzeiro é de 23. Qual o tamanho do grupo que proporciona o maior faturamento para a agência de viagens?

15. Uma loja e móveis espera vender 640 sofás em intervalos regulares no próximo ano. O gerente planeja pedir estes sofás a um fabricante fazendo diversos pedidos de mesmo tamanho e igualmente espaçados durante o ano. O custo de cada entrega é de $160 e o custo de armazenagem, baseado no número médio de sofás em estoque é de $32 por ano para cada sofá. Determine a quantidade de sofás em cada pedido que minimizará o custo total de estocagem.

16. Uma loja e móveis espera vender 740 sofás em intervalos regulares no próximo ano. O gerente planeja pedir estes sofás a um fabricante fazendo diversos pedidos de mesmo tamanho e igualmente espaçados durante o ano. O custo de cada entrega é de $140 e o custo de armazenagem, baseado no número médio de sofás em estoque é de $42 por ano para cada sofá. Determine a quantidade de sofás em cada pedido que minimizará o custo total de estocagem.

11.12 Exercícios de Máximos e Mínimos Multivariados

1. Classifique os pontos críticos das seguintes funções:
   (a) $x^2 + 3xy + 4y^2 - 6x + 2y$
   (b) $x^2 + y^3 + xy - 3x - 4y + 5$
   (c) $x^3 + 2xy + y^2 - 5x$
   (d) $-x^2 + y^2 + 2xy + 4x - 2y$
   (e) $x^3 - 3x^2y + 27y$
   (f) $x^2 - 4xy + 4y^2 - x + 3y + 1$
   (g) $x^5 + y^5 - 5x - 5y$

2. Uma instituição financeira pretende abrir uma agência bancária perto de centros comerciais. Para isso a área de planejamento da instituição financeira verificou que existem três grandes centros comerciais nos seguintes pontos $A(1, 5)$, $B(0, 0)$ e $C(8, 0)$, sendo que as unidades estão em

quilômetros. Em que ponto $P(x, y)$ deve ser instalada a agência para que a soma das distâncias do ponto P aos pontos A, B e C seja a menor possível?

3. Um monopolista comercializa o seu produto em dois países e pode cobrar preços diferentes em cada país. Seja x o número de unidades a serem vendidas no primeiro, e y o número de unidades a serem vendidas no segundo país. Como consequência das leis de demanda, o monopolista precisa fixar o preço em 97-(x/10) dólares no primeiro país, e em 83-(y/20) dólares no segundo, para conseguir vender todas as unidades. O custo na produção dessas unidades é de 20.000 + 3(x+y). Encontre os valores de x e y que maximizam o lucro.

4. Seja $P(x, y) = 10 - 2x^2 + xy - y^2 + 5y$ uma função de produção, onde x e y são quantidades de dois insumos utilizados na fabricação da quantidade $P(x, y)$ de um produto. O preço unitário de cada insumo é $R\$3,00$ e o produto acabado é vendido por $R\$6,00$ à unidade. Calcule o lucro máximo, as quantidades x e y para as quais o mesmo acontece e comprove que o ponto encontrado é um ponto de máximo pelo teste da derivada segunda.

5. Certa empresa que produz sacola para carregar quaisquer artigos esportivos cobra preços diferentes para seus clientes varejistas e atacadistas. A função demanda para o mercado varejista é $p = 500–x$, em que p representa o preço de cada sacola e x a quantidade vendida. A função demanda para o mercado atacadista é $q = 350–1,5y$, em que q representa o preço unitário e y a quantidade vendida pela empresa. A função custo total é dada por: $C(x, y) = 50000 + 20(x + y)$. Todos os dados estão em reais. Que preços p e q a empresa deve estabelecer para maximizar seu lucro na comercialização das sacolas?

6. A loja O ESPORTISTA oferece todo tipo de artigos esportivos. Uma pesquisa feita pelo departamento de Marketing mostrou que, em geral, a demanda de cada produto não depende apenas do seu próprio preço, mas do preço do concorrente. Assim, se a raquete Nadal for vendida por x reais e raquete Federer por y reais, a demanda da raquete Nadal será 300 - 20x + 30y raquetes por ano e a demanda da raquete Federer será 200 + 40x – 10y raquetes por ano.

   (a) Expresse a receita total anual da loja R(x, y) em termos de x e y.

   (b) A loja decide aumentar rapidamente as vendas. Use o conceito de derivadas parciais para decidir se a loja deve aumentar o preço da raquete Nadal ou o preço da raquete Federer. Considere que atualmente a raquete Nadal custa R$ 100,00 e a raquete Federer, R$ 200,00.

7. Uma fábrica decide produzir quadros em duas versões: quadro negro e quadro branco. O custo unitário para produzir quadros negros e R$ 40,00 e o custo unitário para produzir quadros brancos, R$ 60,00. O Departamento de Marketing da fábrica estima que se quadros negros forem vendidos a

x reais e quadros brancos a y reais, cada um, então serão vendidos 500(y - x) exemplares de quadros negros e 45.000 + 500(x - 2y) exemplares de quadro branco. Determine o preço de venda que a fábrica deve estabelecer para cada versão de modo a maximizar seu lucro. Justifique.

8. Uma companhia produz e vende dois produtos, denominados por I e II, os quais são vendidos por $10 e $9 por unidade. O custo para produzir x unidades do produto I e y unidades do produto II é: $C(x, y) = 400 + 2x + 3y + 0,01(3x^2 + xy + 3y^2)$. Encontre os valores de x e y que maximizam o lucro da companhia. Comprove pelo teste da derivada segunda, que o ponto encontrado é um máximo.

11.13 Exercícios de Mínimos Quadrados

1. Dada a tabela de pontos $\begin{bmatrix} x & y \\ 1 & 1 \\ 2 & 6 \\ 3 & 9 \end{bmatrix}$, encontre os parâmetros b0 e b1 da reta $y = b0 + b1.x$, a qual passa o mais próximo possível dos três pontos, pelo método que achar mais conveniente.

2. Um laboratório produz um tipo de soro em quantidade x. A tabela 11.13 mostra as quantidades (em litros) demandadas em função do preço de cada litro. Com base nos dados apresentados, calcule a expressão de regressão linear da demanda e a demanda (em litros) que pode ser esperada para um preço de R$ 15,00 por litro.

$$\begin{bmatrix} P\left(\frac{R\$}{Litro}\right) & D\,(Litros) \\ 2,00 & 5,00 \\ 6,00 & 4,58 \\ 11,00 & 4,00 \\ 18,00 & 3,00 \\ 23,00 & 2,00 \end{bmatrix}$$

3. Uma empresa pretende lançar um produto farmacêutico em dois países, x e y. A fábrica terá um custo fixo anual de US$ 500.000 por ano e um custo de produção unitário de US$ 2,0 por unidade de produção. Podem ser vendidas quantidades fracionárias do produto. Pesquisas de mercado para avaliar a sensibilidade da demanda ao preço unitário de venda do produto nos países x e y apresentaram os seguintes resultados (todos os preços e custos estão em dólares americanos). Com base nos dados apresentados a seguir pede-se:

   (a) Estimar a função demanda para o produto na região x, supondo uma relação linear entre o preço e a quantidade

(b) Estimar a função demanda para o produto na região y, supondo uma relação linear entre o preço e a quantidade

(c) Determinar o preço a ser praticado na região x e na região y de modo a maximizar o lucro total da empresa

(d) Calcular o valor do lucro máximo da empresa

$$\begin{bmatrix} Px & Qx \\ 10 & 34 \\ 15 & 32 \\ 20 & 24 \end{bmatrix}$$

$$\begin{bmatrix} Py & Qy \\ 12 & 31 \\ 18 & 30 \\ 24 & 25 \end{bmatrix}$$

4. Suponha que uma fábrica de automóvel, para analisar o consumo de combustível de um modelo específico, efetuou 7 viagens, tendo-se registrado a distância percorrida (km) e o consumo (l), obtendo-se, então, os 7 pares de valores seguintes disponíveis na tabela 11.3. Com base nestes dados responda:

   (a) Escreva a equação da reta de regressão estimada que relaciona a distância em relação ao consumo (aproxime sua resposta para duas casas decimais).

   (b) Com 16 litros de combustível, qual das duas distâncias lhe parece mais provável de ser percorrida: 190 km ou 205 km? Explique.

   (c) Supondo que o preço do litro de gasolina seja R$ 2,52, qual o valor gasto (estimado) em um trajeto de 820 km?

Tabela 11.3: Pares Distância x Consumo

y_distancia	x_consumo
20	2
40	3
80	5
120	9
160	12
200	14
250	18

5. A partir da tabela 11.4 , relacione a variável Número de Automóveis a partir de uma expressão com a variável Tempo, do tipo Número = b0 + b1.Tempo utilizando a fórmula matricial $\vec{b} = (X^t.X)-1.(X^t.Y)$

Tabela 11.4: Tabela de Vendas Anuais de Carros

Ano	Carros
1980	104.6
1985	114.7
1989	122.8
1990	123.3
1991	123.3
1992	120.3
1993	121.1
1994	122.0

6. A partir da tabela 11.5, relacione a variável Vendas a partir de uma expressão com a variável Propaganda, do tipo $V = \beta_1.P + \beta_0$, utilizando a fórmula matricial $\vec{\beta} = (X^T.X)^{-1}.(X^T.Y)$

Tabela 11.5: Vendas e Propaganda

P	V
10	155.35
11	161.55
12	169.67
13	185.77
14	174.05
15	186.03
16	183.03
17	193.57
18	185.97
19	197.98
20	203.31

7. Ache a função de regressão exponencial $y = ae^{bx}$ que minimiza o erro quadrado para os pontos da tabela 11.13:

$$\begin{bmatrix} x & y \\ 0 & 3,10 \\ 1 & 5,50 \\ 2 & 17,60 \\ 3 & 57,30 \\ 4 & 199,70 \end{bmatrix}$$

8. Encontre a e b na regressão exponencial $y = b.x^a$ na tabela abaixo

Renda	Pop com Renda maior que
50	25168
60	18224
70	13533
80	9950
90	7642
100	6129

Tabela 11.6: Distribuição de Pareto

P/L	Risco	Retorno
7,4	1,0	7,6
11,1	1,3	13,0
8,7	1,1	8,9
11,2	1,2	10,9
11,6	1,7	12,1
12,2	1,3	12,8
12,5	1,2	11,3
12,5	1,3	14,1
13,0	1,6	14,8
13,4	1,4	16,7

Tabela 11.7: Retorno x PL e Risco

$$\begin{bmatrix} x & y \\ 0,1 & 2,4 \\ 1,0 & 3,6 \\ 2,0 & 5,0 \\ 3,0 & 7,2 \\ 4,0 & 12,1 \\ 5,0 & 17,5 \end{bmatrix}$$

9. Estudando a distribuição de riqueza em uma população, os economistas assumem que a mesma é distribuída segundo a curva de Pareto $y = b/x^a$, onde a e b são parâmetros a determinar. A tabela 11.6 contém os dados do Censo Americano para 2001. Nesta tabela, x representa a renda anual em milhares de dólares e y representa o número de homens (em milhares) com renda anual maior ou igual a x. Ache a curva de Pareto $y = b/x^a$, que melhor se encaixa nos dados. Calcule a regressão através do método que achar mais conveniente e detalhe todos os passos da sua solução.

10. Tomando por base os valores da tabela 11.7, encontre a melhor estimativa para o Retorno quando o P/L igual a 9,5 e o risco igual a 11,0. Utilize como expressão de estimação $Retorno = b_0 + b_1.(P/L) + b_2.(Risco)$.

11. A partir da tabela 11.8, relacione a variável Preço (y) à variável Área do

Área	Preço
104	69
114	73
124	76
145	78
166	86
187	97
197	103
197	108
218	129
218	137
228	160

Tabela 11.8: Preço x Área de Imóveis

Imóvel (x), de acordo com a expressão de regressão $y = b_0 + b_1.x + b_2.x_2 + b_3.x_3$, utilizando a fórmula matricial $\vec{b} = (X^t.X)^{-1}.(X^t.Y)$

12. A partir da tabela 11.9, monte uma expressão de classificação das empresas listadas, utilizando o Solver na obtenção da expressão e utilize tal critério para classificar as empresas da própria tabela:

13. A partir da tabela 11.10 a seguir, monte uma expressão de classificação para determinar se uma pessoa será ou não um possível comprador de cortadores de grama. Utilize o Solver na obtenção da expressão e utilize tal critério para classificar as pessoas da própria tabela como Proprietários ou não. Obs : Propr.=1 Pessoa NÃO tem um cortador de grama, Propr=2 Pessoa TEM um cortador de grama.

11.14 Exercícios de Otimização Condicionada

1. Ache o máximo de:
   (a) $f = x^2 + y^2$ se $x + y = 6$
   (b) $f = 49 - x^2 - y^2$ se $x + 3y = 10$
   (c) $f = 3x + 4y$ se $x^2 + y^2 = 1$
   (d) $f = \sqrt{(x-1)^2 + (y-2)^2}$ se $x^2 + y^2 = 45$
   (e) $f = \sqrt{(x-14)^2 + (y-1)^2}$ se $y = x^2 + 2x - 3$

2. A administração de uma loja quer construir um cercado retangular com 600m2 no estacionamento da loja para exibir um equipamento. Em três dos lados serão construídas cercas de madeira, a um custo de $14 por metro de comprimento. O quarto lado será construído utilizando blocos de cimento

Empresa	Grupo	Indice 1	Indice 2
1	Bom	8,1	0,6
2	Bom	6,6	1,0
3	Bom	5,8	0,7
4	Bom	12,3	0,8
5	Bom	4,5	0,7
6	Bom	9,1	0,7
7	Bom	1,1	0,6
8	Bom	8,9	0,8
9	Bom	0,7	0,6
10	Bom	9,8	0,7
11	Ruim	7,3	0,6
12	Ruim	14,0	0,5
13	Ruim	9,6	0,7
14	Ruim	12,4	0,4
15	Ruim	18,4	0,5
16	Ruim	8,0	0,5
17	Ruim	12,6	0,3
18	Ruim	9,8	0,7
19	Ruim	8,3	0,5
20	Ruim	20,6	0,8

Tabela 11.9: Empresas Boas e Ruins x 2 Índices

Observation	Renda	Tam.Res.	Propr.
1	60,0	18,4	1
2	85,5	16,8	1
3	64,8	21,6	1
4	61,5	20,8	1
5	87,0	23,6	1
6	110,1	19,2	1
7	108,0	17,6	1
8	82,8	22,4	1
9	69,0	20,0	1
10	93,0	20,8	1
11	51,0	22,0	1
12	81,0	20,0	1
13	75,0	19,6	2
14	52,8	20,8	2
15	64,8	17,2	2
16	43,2	20,4	2
17	84,0	17,6	2
18	49,2	17,6	2
19	59,4	16,0	2
20	66,0	18,4	2
21	47,4	16,4	2
22	33,0	18,8	2
23	51,0	14,0	2
24	63,0	14,8	2

Tabela 11.10: Propriedade x Renda e Tamanho de Imóvel

a um custo de $28 por metro de comprimento. Encontre as dimensões do cercado que minimizará o custo total dos materiais de construção.

3. A equação de demanda de um monopolista é p = 200 -3x e a função custo é $C(x) = 75 + 80x – x2$ para $0 \leq x \leq 40$. Determine:
   (a) O Valor de x e o preço correspondente que maximiza o lucro
   (b) Suponha que o governo cobre um imposto do monopolista de $4 por unidade produzida. Determine o novo preço que maximiza o lucro
   (c) Suponha que o governo cobre um imposto de T dólares por unidade produzida, de forma que a nova função de custo seja $sejaC(x) = 75 + (80 + T)\, x – x2 para 0 \leq x \leq 40$. Determine o novo valor de x que maximiza o lucro do monopolista como uma função de T
   (d) Admitindo que o monopolista diminua sua produção para o nível do item c) expresse a arrecadação do governo gerada pela cobrança do imposto como uma função de T.
   (e) Determine o valor de T que irá maximizar a arrecadação recebida pelo governo
4. Desenvolva um modelo em planilha eletrônica para calcular as coordenadas dos vértices do maior retângulo que possa ser inscrito no círculo $x^2 + y^2 = 4$. Utilize o Solver para determinar as coordenadas dos vértices do mesmo
5. Usando o Método dos multiplicadores de Lagrange, encontre os três números positivos cuja soma seja 15 e cujo produto seja o maior possível.
6. Um clube poliesportivo pretende adquirir na loja O ESPORTISTA, calções e meias para a prática de futebol dos seus associados. Cada par de meias custa R$ 20,00 e cada calção, R$ 10,00. A função utilidade do clube para adquirir x meias e y calções, pode ser expressa por: $U(x, y) = 10x^{0,5}y^{0,5}$.
   (a) Use o multiplicador de Lagrange para determinar a combinação ótima de consumo do clube poliesportivo. Justifique a resposta.
   (b) Calcule o valor de λ correspondente aos valores de x e y do item A.
7. A função de produção de uma empresa (isto é o número de unidades produzidas) é dada por: $f(x, y) = 64x^{\frac{3}{4}}y^{\frac{1}{4}}$, onde x e y são o número de unidades de trabalho e capital utilizadas. Suponha que o trabalho custe $96 por unidade, o capital $162 por unidade e a empresa deseja produzir um total de 3.456 unidades. Com base nestas informações pede-se calcular o número de unidades de trabalho e capital que irá minimizar o custo total de produção.
8. A utilidade de um consumidor para adquirir x unidades de um produto e y unidades de um segundo produto é dada pela função de utilidade $U(x, y) = x^2y$. Suponha que o consumidor adquire mensalmente 3 unidades do primeiro produto e 3 unidades do segundo produto.

(a) Calcule U(3,3) e trace a curva de nível para esse valor.

(b) Suponha que em certo mês, o consumidor decide gastar R$ 120,00 para adquirir e estocar certa quantidade dos dois produtos. Qual a quantidade de cada um que o consumidor deverá comprar para maximizar a sua utilidade? O preço por unidade do primeiro produto é R$ 4,00 e do segundo produto, R$ 5,00?

9. A quantidade de espaço requerida por uma empresa é $f(x, y) = \sqrt{6x^2 + y^2}$, em que x e y são respectivamente o número de unidades de mão de obra e de capital utilizadas. Suponha que a mão de obra custa $480 por unidade e capital custa $40 por unidade e que a empresa tem $5.000 para gastar. Determine a quantidade de mão-de-obra e capital que minimizam o espaço total utilizado.

10. Suponha que uma firma produza dois produtos, A e B que utilizam a mesma matéria-prima e quantidades determinadas de mão de obra, de modo que a quantidade x do produto A e a quantidade y do produto B deverão satisfazer $9x^2 + 4y^2 = 18.000$. Se o lucro de cada unidade de A é de $3 e o de cada unidade de B é de $4, determine as quantidades que devem ser produzidas para atender a restrição de produção e maximizar o lucro da empresa.

11. Um funcionário da loja decidiu aproveitar suas férias, em julho, e frequentar um bar e uma discoteca. Ele paga R$ 10,00 para entrar no bar e R$60,00, na discoteca. O seu salário mensal é R$ 2 400,00. Ele decidiu gastar 10% do seu salário para pagar as entradas. Considere que a função utilidade seja dada por $U(x, y) = xy$ em que x é o número de vezes que entra no bar e y o número de vezes que entra na discoteca.

    (a) Em quantas vezes deve ir ao bar e em quantas à discoteca para maximizar a função utilidade (satisfação)?

    (b) Suponha agora que o funcionário decida sair todas as noites durante os 31 dias do mês de julho. Sem limitar a quantia que vai gastar. Ele vai ao bar, à discoteca ou ao bar e à discoteca. Lógico, considere que ele não vai entrar e sair do bar ou da discoteca na mesma noite.

    (c) Quantas vezes ele deve ir ao bar ou à discoteca de forma a maximizar a sua função utilidade $U(x, y) = xy$?

12. Suponha que cada unidade de capital (x) custe $150 e cada unidade de mão de obra (y) custe $250. Sabendo que o total produzido é dado pela função $P(x, y) = x^{1/4}.y^{3/4}$ e que você possui um orçamento de $50.000, quais são as quantidades x e y que maximizam a sua produção?

13. Um fabricante produz dois tipos de máquinas: x e y. Suponha que a função lucro seja dada por $L(x, y) = x^2 + 3xy - 6y$. Quantas unidades x e y devem ser produzidas se o fabricante tem recursos suficientes para fabricar no máximo um total de 42 máquinas?

14. Use multiplicadores de Lagrange para encontrar expressões para x e y que maximizem a produção dada pela função de Cobb-Douglas $P(x, y) = K.x^{\alpha}.y^{\beta}$ onde K, α e β são constantes positivas. A função $P(x, y)$ está sujeita à restrição de custo $x.p_x + y.p_y = D$ onde p_x, p_y e D são constantes.

15. Temos dois pontos em um terreno, ponto A e ponto B. No ponto A está o gado. No ponto B está o estábulo. Temos um rio passando perto dos pontos A e B. Você pode pensar neste rio como uma $y = f(x)$ Com base nestes dados, os pecuaristas pediram a um matemático que desenvolvesse um método para determinar qual o ponto do rio levar o gado para beber água e depois leva-lo ao estábulo, de forma que a distância que o gado percorrerá seja mínima.

11.15 Exercícios de Otimização Linear Lagrange, Grafico, Simplex, Solver

1. Uma fábrica produz dois tipos de caixas, utilizando os insumos *trabalho* e *metal*. Para produzir uma caixa do tipo 1 são necessários 10 homens-hora e para uma caixa do tipo 2 são necessários 2 homens-hora. Cada caixa do tipo 1 requer 10 folhas de metal, e cada caixa do tipo 2 requer 5 folhas de metal. Em um dado período de tempo, a fábrica dispõe de 200 homens-hora e 260 folhas de metal. Se cada caixa do tipo 1 é vendida por R$ 200,00 e cada caixa do tipo 2 é vendida por R$ 90,00, qual a produção que maximiza a receita de vendas neste período?

2. Uma certa empresa fabrica dois produtos P1 E P2. O lucro unitário do produto P1 é 1000 reais e o lucro unitário de P2 é de 1800 reais. A empresa precisa de 20 horas para fabricar uma unidade de P1 e 30 Horas para fabricar uma unidade de P2. O tempo anual de produção disponível é de 1200 horas. A demanda esperada para cada produto é de 40 unidades anuais de P1 e 30 Unidade anuais de P2.

   (a) Qual será o lucro se a empresa decidir fabricar 10 unidades do produto 1 e 20 unidades do produto 2.

   (b) É viável fabricar tais quantidades?

   (c) Quanto a empresa deverá fabricar do produto P1 e do produto P2 para maximizar seu lucro?

3. O gerente de marketing do refrigerante Montanha Gelada precisa decidir quantos anúncios na TV e em revistas devem ser exibidos durante o próximo trimestre. Cada spot de TV custa R$ 5 mil e deve aumentar as vendas em 300 mil latas. Cada anúncio de revista custa R$ 2 mil e deve aumentar as vendas em 500 mil latas. Um total de R$ 100 mil pode ser gasto em anúncios de TV e revistas; no entanto, a Montanha Gelada não quer gastar mais de R$ 70 mil em comerciais de TV e não mais de R$ 50 mil em

anúncios em revistas. Além disso, a Montanha Gelada ganha um lucro de R$ 0,05 em cada lata que é vendida. Pede-se com base nestas informações:

(a) Formular um modelo algébrico para esse problema.

(b) Esboçar a região de solução para este modelo.

(c) Encontre a solução ideal (quantidades de anúncios em TV e Revista) para este problema através do método gráfico, além do lucro máximo correspondente.

4. Uma empresa fabrica 2 tipos de banheira, AquaSpa e HydroLux. Cada banheira AquaSpa precisa de 2 bombas, 12h de trabalho e 16m de canos, proporcionando um lucro unitário de $400 reais. Cada banheira HydroLux precisa de 1 bomba, 8h de trabalho e 12m de cano, proporcionando $300 reais de lucro unitário. Existem 200 bombas, 1566h e 2880m de cano em estoque. Pede-se de modo a maximizar o lucro total:

   (a) Expressar o problema na forma algébrica

   (b) Esboçar o gráfico que representaria a solução do problema

5. Uma empresa de móveis pode produzir dois tipos de mesa: a mesa tipo A, que consome 1 litro de tinta, 9 horas de trabalho e 12 metros de madeira, ou mesa tipo B que consome 1 litro de tinta, 6 horas de trabalho e 16 metros de madeira. O Lucro das mesas é 350 reais para o tipo A e 300 para B. A empresa possui disponível 200 litros de tinta, 1566 horas de trabalho e 2880 metros de madeira. Com base nestes dados pergunta-se:

   (a) Qual deve ser a produção visando o maior lucro?

   (b) Qual dos produtos em estoque pode ser reduzido sem afetar o lucro máximo?

6. Uma rede de televisão local tem o seguinte problema: foi descoberto que um programa A com 20 minutos de música e 1 minuto de propaganda atrai 30 mil telespectadores, enquanto um programa B, com 10 minutos de música e um minuto de propaganda chama atenção de 10 mil telespectadores. No decorrer de uma semana, o patrocinador insiste no uso de no mínimo 5 minutos para sua propaganda e que não há verba para mais de 80 minutos de música. Quantas vezes por semana cada programa de deve ser levado ao ar para se atingir o número máximo de telespectadores.

7. Para uma boa alimentação, o corpo necessita de vitaminas e proteínas. A necessidade mínima de vitaminas é de 32 unidades por dia e a de proteínas é de 36 unidade por dia. Uma pessoa tem disponível carne e ovos para se alimentar. Cada unidade de carne contém 4 unidades de vitaminas e 6 unidades de proteínas. Cada unidade de ovo contém 3 unidades de vitaminas e 6 unidades de proteínas. Qual a quantidade diária de carne e ovos que dever ser consumida para suprir as necessidades de vitaminas e proteínas com o menor custo possível. Se cada unidade de carne custa 3

reais e cada unidade de ovo custa 2,5 reais. Resolva o problema através do método gráfico.

8. Uma organização humanitária se propõe a realizar projetos de melhoria da produção agrícola em dois países subdesenvolvidos. Estes projetos requerem o envio de tratores, especialistas e dinheiro. A organização conta com 20 tratores, 40 especialistas e 270 milhões de dólares. As necessidades por projetos e país são as seguintes: a) País A: 2 tratores, 2 especialistas e 30 milhões de dólares por projeto (ou seja, cada projeto necessita 2 tratores, 2 especialistas e 30 milhões de dólares), b) País B: 1 trator, 3 especialistas e 10 milhões de dólares por projeto.

   (a) Qual é o número total máximo de projetos que essa organização pode realizar?

   (b) Sabe-se que, para cada projeto no país A são beneficiadas 6 000 pessoas, enquanto que os projetos realizados no país B beneficiam 2 500 pessoas cada. Qual é o maior número de pessoas que podem ser beneficiadas?

9. A empresa PC Tech monta e depois testa dois modelos de computadores, Basic e XP. Para o próximo mês, a empresa quer decidir quantos modelos de cada um montar e depois testar. Nenhum computador está no inventário em relação ao mês anterior e, como esses modelos serão alterados após este mês, a empresa não deseja manter nenhum inventário após este mês. Ele acredita que o máximo que pode vender este mês são 600 Básico e 1200 XPs. Cada Basic é vendido por U$ 300 e cada XP é vendido por U$ 450. O custo das peças de um Basic é de U$ 150; para um XP, custa U$225. É necessária mão de obra para montagem e teste. Existem no máximo 10.000 horas de montagem e 3000 horas de teste disponíveis. Cada hora de trabalho para montagem custa U$ 11 e cada hora de trabalho para teste custa U$ 15. Cada Básico requer cinco horas para montagem e uma hora para teste, e cada XP requer seis horas para montagem e duas horas para teste. A PC Tech quer saber quantos modelos cada um deve produzir (montar e testar) para maximizar seu lucro líquido, mas não pode usar mais horas de trabalho do que o disponível e não quer produzir mais do que pode vender. Pede-se: a) formular um modelo algébrico para esse problema, b) esboçar a região de solução para este modelo, c) encontrar a solução ideal para este problema usando o método gráfico (análise dos pontos de fronteira).

10. Uma empresa de cosméticos produz dois tipos de perfume, Sense e Sensibility. O lucro líquido obtido pela venda de cada um dos perfumes é de R$ 12,00 e R$ 60,00 respectivamente. O processo de fabricação envolve as etapas de mistura, controle de qualidade e embalagem. O processo de mistura requer 15 minutos para cada unidade do Sense e 30 minutos para cada unidade do Sensibility. O processo de controle de qualidade requer 6 minutos para cada unidade do Sense e 45 minutos para cada

Recurso:	Mold1	Mold2	Mold3	Mold4
Trabalho (horas)	2	1	3	2
Metal (kg)	3	2	1	2
Vidro (kg)	6	2	1	2
Preço Unit.	28,50	12,50	29,25	21,50

Tabela 11.11: Dados da Empresa Monet Frames

unidade do Sensibility. Já o processo de embalagem necessita de 6 e 24 minutos respectivamente, para cada unidade do Sense e do Sensibility produzida. O tempo disponível por semana para os processos de mistura, controle de qualidade e embalagem é de 36, 22 e 15 horas. A empresa deseja determinar o quantas unidades deverá produzir de cada um dos perfumes, para maximizar o seu Lucro, respeitando às limitações de tempo nas etapas do processo produtivo.

11.16 Exercícios de Modelagem e Aplicações Gerenciais

1. A companhia Monet produz quatro tipos diferentes de molduras para quadros. Cada moldura requer uma determinada quantidade de trabalho, metal e vidro, é vendida por um preço específico e possui um limite de demanda distinto dos demais, conforme a tabela 11.11. A Monet pode obter até 4.000 horas de trabalho por R$ 8,00 por hora, 6.000 kg de metal por $0,50 por kg e 10.000 kg de vidro por R$ 0,75 por kg. Determine o quanto fabricar durante a próxima semana, de modo a maximizar o lucro, sem deixar molduras sobrando na empresa.

2. Uma refinaria produz três tipos de gasolina: verde, azul e comum. Cada tipo requer gasolina pura, octana e aditivo que são disponíveis nas quantidades de 9.600.000, 4.800.000 e 2.200.000 litros por semana, respectivamente. As especificações de cada tipo são apresentadas abaixo. Como regra de produção, baseada em demanda de mercado, o planejamento da refinaria estipulou que a quantidade de gasolina comum deve ser no mínimo igual a 16 vezes a quantidade de gasolina verde e que a quantidade de gasolina azul seja no máximo igual a 600.000 litros por semana. A empresa sabe que cada litro de gasolina verde, azul e comum dá uma margem de contribuição para o lucro de $0,30, $0,25 e $0,20 respectivamente, e seu objetivo é determinar o programa de produção que maximiza a margem total de contribuição para o lucro.

   (a) um litro de gasolina verde requer 0,22 litro de gasolina pura, 0,50 litro de octana e 0,28 litro de aditivo;

   (b) um litro de gasolina azul requer 0,52 litro de gasolina pura, 0,34 litro

Programa:	Friends	MNF	Malcolm	Sports	TRL	Lifetime	CNN	JAG	Min.
H:18-35	6,0	6,0	5,0	0,5	0,7	0,1	0,1	1,0	60,0
H:36-55	3,0	5,0	2,0	0,5	0,2	0,1	0,2	2,0	60,0
H:$\geq$ 55	1,0	3,0	0,0	0,3	0,0	0,0	0,3	4,0	28,0
F:18-35	9,0	1,0	4,0	0,1	0,9	0,6	0,1	1,0	60,0
F:36-55	4,0	1,0	2,0	0,1	0,1	1,3	0,2	3,0	60,0
F:$\geq$ 55	2,0	1,0	0,0	0,0	0,0	0,4	0,3	4,0	28,0
Custo Un.:	$160	$100	$80	$9	$13	$15	$8	$85	

Tabela 11.12: Dados da Empresa General Flakes

Nutrientes	Ração	Milho	Alfafa	Mínimo necessário
Carboidrato	90	20	40	190
Proteína	30	80	60	210
Vitamina	10	20	60	160
Custo	84	72	60	

Tabela 11.13: Dados de Misturas para Rações

de octana e 0,14 litro de aditivo;

(c) um litro de gasolina comum requer 0,74 litro de gasolina pura, 0,20 litro de octana e 0,06 litro de aditivo.

3. A empresa General Flakes anuncia seu produto em uma série de comerciais de televisão, o qual varia de custo de acordo com o programa. Os telespectadores foram segmentados em seis categorias. O custo (em $1.000s por anúncio) e a quantidade de telespectadores em cada segmento (em 1Ms de pessoas) que um anúncio em um determinado programa proporciona são dados na tabela 11.12 onde `H=Homem` e `M=Mulher`. A empresa deseja saber quantos anúncios ela deverá colocar em cada programa para maximizar a exposição total, obtendo a exposição mínima desejada em cada segmento de telespectadores, com um custo máximo de $2,1 milhões.

4. Um fazendeiro está criando porcos comercialmente. Ele deseja determinar as quantidades de ração, milho e alfafa necessárias para alimentar os animais. Os porcos comerão quaisquer misturas destes três alimentos, assim o objetivo é determinar qual mistura conterá a quantidade mínima de alguns nutrientes pelo menor custo. A quantidade de cada nutriente contida em um quilo de cada um dos três alimentos, as quantidades mínimas necessária diariamente e os preços estão sumarizados na tabela 11.13. (Dica: Suponha que o fazendeiro deseja obter 1.000 kg de ração no total e obtenha os percentuais a partir daí).

5. Uma empresa química fabrica um composto que é muito utilizado em laboratórios de química de faculdades e universidades. Este composto deve conter pelo menos 20% de ácido sulfúrico, pelo menos 30% de óxido

Composto	Ácido Sulfúrico	Óxido de Ferro	Potássio
1	20%	60%	20%
2	40%	30%	30%
3	10%	40%	50%

Tabela 11.14: Dados de Mistura de Produtos Químicos

Nutriente	ração 1	ração 2	ração 3	ração 4
Milho	30%	5%	20%	10%
Grãos	10%	30%	15%	10%
Minerais	20%	20%	20%	30%
Custo/quilo	$0,25	$0,30	$0,32	$0,15

Tabela 11.15: Mistura de Matéria Prima para Rações

de ferro e pelo menos 30% de potássio, sendo que o potássio não deve ultrapassar a 45%. O departamento de marketing estimou que necessitará de 600Kg deste composto para suprir a demanda do próximo período. A empresa pode comprar três outros compostos básicos para misturar e fazer o seu produto. A composição destes compostos é dada na tabela 11.14. Os compostos 1,2 e 3 custam respectivamente $5,00, $5,25 e 5,50 por quilo. Pede-se :

(a) Modelar o problema em planilha

(b) Determinar a melhor mistura que atenda os requisitos para o produto final e que tenha o menor custo possível.

(c) Apresentar o custo mínimo obtido na resposta abaixo.

6. Uma empresa inovou na venda de rações para animais, preparando seus produtos conforme as especificações do cliente, utilizando entre outros componentes o milho, grãos e minerais. Esta "personalização"tem um grande valor agregado, pois a ração apropriada para um determinado animal, muda regularmente dependendo das condições climáticas, condições do pasto, etc. A empresa estoca quatro tipos de ração que ela pode misturar para atingir as especificações dos clientes. A tabela 11.15 sumariza a composição dos quatro tipos e seu custo. A empresa recebeu um pedido de 8.000 quilos de ração de um fazendeiro local. O fazendeiro quer que a ração contenha pelo menos 20% de milho, 15% de grãos e 15% de minerais. Qual mistura deverá ser feita para minimizar o custo?

7. A Sucofresco é uma empresa que cultiva laranjas e prepara suco para comercialização. Suas plantações estão em três cidades diferentes, tendo 275.000 alqueires em Laranjal, 400.000 alqueires em Rio Claro e 300.000 alqueires em Santa Clara. A Sucofresco tem fábricas para o processamento da laranja em três localidades, sendo que a de São Carlos tem capacidade para processar o equivalente a 200.000 alqueires, a de Jandira 600.000 e a

Pomar	São Carlos	Jandira	Santa Rita
Laranjal	21	50	40
Rio Claro	35	30	22
Santa Clara	55	20	25

Tabela 11.16: Distâncias entre Fazendas e Usinas

de Santa Rita 225.000. Ela contrata uma empresa local para transportar a laranja do pomar às fábricas, que cobra uma taxa fixa por quilômetro transportado do hectare equivalente de laranja. A tabela 11.16 sumariza as distâncias dos pomares às fábricas. A Sucofresco quer determinar quantos alqueires enviará de cada pomar para cada fábrica a fim de minimizar a o custo do transporte. Para tal pede-se, montar o problema em forma gráfica, desenhar a planilha de resolução, apresentar os parâmetros do solver e a resolução numérica para o mesmo.

Capítulo 12

Integrais

12.1 Integração Algébrica

A integração algébrica pode ser realizada no Python através do pacote `sympy`. Abaixo é calculada a integral da função $sin(x)$ no intervalo $[0, \pi]$

```
import sympy as sp
x = sp.symbols("x")
sp.integrate(sp.sin(x),[x,0,sp.pi])
```

```
2
```

12.2 Regra dos Trapézios

A seguir calculamos o mesmo valor através da regra dos trapézios dividindo o intervalo de integração em $n = 2000$ subintervalos.

```
import numpy as np
def fL(x):
  return(np.sin(x))

n = 2000

x0 = 0
x1 = np.pi
dx = (x1 - x0)/n
x = np.linspace(x0,x1,n)

integral = 0
for xi in x:
```

```
  integral = integral + (fL(xi+dx) + fL(xi))/2*dx

integral
```

```
1.998998355477553
```

A título de comparação realizamos o mesmo cálculo para $\int(\frac{1}{x}dx$ no intervalo $[1,2]$ cujo resultado exato (obtido por integração algébrica) sabemos ser $ln(2)$.

```
import sympy as sp
x = sp.symbols("x")
sp.integrate(1/x,[x,1,2]), \
   sp.integrate(1/x,[x,1,2]).evalf()
```

```
(log(2), 0.693147180559945)
```

```
import numpy as np
def fL(x):
  return(1/x)

n = 20000

x0 = 1
x1 = 2
dx = (x1 - x0)/n
x = np.linspace(x0,x1,n)

integral = 0
for xi in x:
  integral = integral + (fL(xi+dx) + fL(xi))/2*dx

integral
```

```
0.6931375236696611
```

12.3 Quadratura Gaussiana de 2 Pontos

Outra técnica de integração numérica, em especial para funções contínuas no intervalo $[-1,+1]$ de resultados excelentes e extramente rápida de ser calculada é a quadratura gaussiana. Abaixo são calculados os parâmetros da mesma para uma interpolação de 2 pontos.

```
import sympy as sp
x = sp.symbols("x")
I1 = sp.integrate(1,[x,-1,1]); I1
```

```
2
```

```
I2 = sp.integrate(x,[x,-1,1]); I2
```

```
0
```

```
I3 = sp.integrate(x**2,[x,-1,1]); I3
```

```
2/3
```

```
I4 = sp.integrate(x**3,[x,-1,1]); I4
```

```
0
```

```
a, x0, b, x1 = sp.symbols("a x0 b x1")
Eq0 = a*1 + b*1 - I1; Eq0
```

```
a + b - 2
```

```
Eq1 = a*x0 + b*x1 - I2; Eq1
```

```
a*x0 + b*x1
```

```
Eq2 = a*x0**2 + b*x1**2 -I3; Eq2
```

```
a*x0**2 + b*x1**2 - 2/3
```

```
Eq3 = a*x0**3 + b*x1**3 - I4; Eq3
```

```
a*x0**3 + b*x1**3
```

```
sols = sp.solve([Eq0, Eq1, Eq2, Eq3],[a,x0,b,x1]);
for sol in sols:
  print(sol)
```

```
(1, -sqrt(3)/3, 1, sqrt(3)/3)
(1, sqrt(3)/3, 1, -sqrt(3)/3)
```

Solução exata para $\int_0^{\pi} sin(x)dx = 2$

```
import numpy as np
sp.integrate(sp.sin(x),[x,0,sp.pi])
```

```
2
```

Solução por quadratura gaussiana de 2 pontos

```
x, z = sp.symbols("x z")
A = sp.Matrix([[x,z,1],[0,-1,1],[sp.pi,1,1]]); A
```

```
Matrix([
[ x,  z, 1],
[ 0, -1, 1],
[pi,  1, 1]])
```

```
g = sp.solve(sp.det(A),x)[0]; g
```

```
pi*(z + 1)/2
```

```
g.diff(z) #dx/dz
```

```
pi/2
```

```
c1 = sols[0][0]; print(c1)
```

```
1
```

```
p1 = sols[0][1]; print(p1)
```

```
-sqrt(3)/3
```

```
c2 = sols[0][0]; print(c2)
```

```
1
```

```
p2 = sols[0][3]; print(p2)
```

```
sqrt(3)/3
```

```
aprox = c1*sp.sin(g.subs(z,p1))*g.diff(z)
aprox = aprox + c2*sp.sin(g.subs(z,p2))*g.diff(z)
aprox.evalf()
```

```
1.93581957465114
```

12.4 Quadratura Gaussiana de Três Pontos

```
import sympy as sp
x = sp.symbols("x")
I0 = sp.integrate(1,[x,-1,1]); print(I0)
```

```
2
```

```
I1 = sp.integrate(x,[x,-1,1]); print(I1)
```

```
0
```

```
I2 = sp.integrate(x**2,[x,-1,1]); print(I2)
```

```
2/3
```

```
I3 = sp.integrate(x**3,[x,-1,1]); print(I3)
```

```
0
```

```
I4 = sp.integrate(x**4,[x,-1,1]); print(I4)
```

```
2/5
```

```
I5 = sp.integrate(x**5,[x,-1,1]); print(I5)
```

```
0
```

```
a, x0, b, x1, c, x2 = sp.symbols("a x0 b x1 c x2")
Eq0 = a + b + c - I0; print(Eq0)
```

```
a + b + c - 2
```

```
Eq1 = a*x0 + b*x1 + c*x2 - I1; print(Eq1)
```

```
a*x0 + b*x1 + c*x2
```

```
Eq2 = a*x0**2 + b*x1**2 + c*x2**2 - I2; print(Eq2)
```

```
a*x0**2 + b*x1**2 + c*x2**2 - 2/3
```

```
Eq3 = a*x0**3 + b*x1**3 + c*x2**3 - I3; print(Eq3)
```

```
a*x0**3 + b*x1**3 + c*x2**3
```

```
Eq4 = a*x0**4 + b*x1**4 + c*x2**4 - I4; print(Eq4)
```

```
a*x0**4 + b*x1**4 + c*x2**4 - 2/5
```

```
Eq5 = a*x0**5 + b*x1**5 + c*x2**5 - I5; print(Eq5)
```

```
a*x0**5 + b*x1**5 + c*x2**5
```

```
sols = sp.solve([Eq0, Eq1, Eq2, Eq3, Eq4, Eq5], \
                [a,x0,b,x1,c,x2])
for sol in sols:
  print(sol)
```

```
(5/9, -sqrt(15)/5, 5/9, sqrt(15)/5, 8/9, 0)
(5/9, -sqrt(15)/5, 8/9, 0, 5/9, sqrt(15)/5)
(5/9, sqrt(15)/5, 5/9, -sqrt(15)/5, 8/9, 0)
(5/9, sqrt(15)/5, 8/9, 0, 5/9, -sqrt(15)/5)
(8/9, 0, 5/9, -sqrt(15)/5, 5/9, sqrt(15)/5)
(8/9, 0, 5/9, sqrt(15)/5, 5/9, -sqrt(15)/5)
```

```
from IPython import display
from IPython.display import Math, HTML
Math(sp.latex(sols))
```

```
<IPython.core.display.Math object>
```

```
c1 = sols[0][0]; print(c1)
```

```
5/9
```

```
p1 = sols[0][1]; print(p1)
```

```
-sqrt(15)/5
```

```
c2 = sols[0][2]; print(c2)
```

```
5/9
```

```
p2 = sols[0][3]; print(p2)
```

```
sqrt(15)/5
```

```
c3 = sols[0][4]; print(c3)
```

```
8/9
```

```
p3 = sols[0][5]; print(p3)
```

```
0
```

Exata

```
import numpy as np
sp.integrate(sp.sin(x),[x,0,np.pi])
```

```
2.00000000000000
```

Por quadratura

```
x, z = sp.symbols("x z")
A = sp.Matrix([[x,z,1],[0,-1,1],[sp.pi,1,1]]); A
```

```
Matrix([
[ x,  z, 1],
[ 0, -1, 1],
[pi,  1, 1]])
```

```
g = sp.solve(sp.det(A),x)[0]; g
```

```
pi*(z + 1)/2
```

```
g.diff(z) #dx/dz
```

```
pi/2
```

```
((c1*sp.sin(g).subs(z,p1) + \
  c2*sp.sin(g).subs(z,p2) + \
  c3*sp.sin(g).subs(z,p3))*g.diff(z)).evalf()
```

```
2.00138891360774
```

12.5 Exercícios

12.5.1 Integrais Indefinidas

1. $\int 9x^8\,dx$
2. $\int 3x\,dx$
3. $\int e^{-3x}\,dx$
4. $\int 3dx$
5. $\int -4x\,dx$
6. $\int \frac{dx}{x^2}$
7. $\int x^2\cos(x)\,dx$
8. $\int xe^{-x^2}\,dx$
9. $\int \frac{dx}{(x-1)(x+2)(x-3)}$
10. $\int e^x\sin(x)\,dx$
11. $\int \frac{x^3-5x^2+3x+\ln(x)}{x}\,dx$
12. $\int \frac{\ln(4x)}{\sqrt{x}}\,dx$
13. $\int \frac{x}{\sqrt{x^2+1}+x}\,dx$
14. $\int_0^1 t.e^{-t}\,dt$
15. $\int \frac{dx}{x^2+7}$
16. $\int xsin(x)\cos(x)\,dx$
17. $\int \ln(x+a)\,dx$
18. $\int \frac{\ln(4x)}{\sqrt{x}}\,dx$
19. $\int \frac{x^3-5x^2+3x+\ln(x)}{x}\,dx$
20. $\int \frac{dx}{x^2\sqrt{4-x^2}}$
21. $\int x\arcsin(x)\,dx$

12.5.2 Integrais Definidas

1. Calcule o valor da integral de f(x)=x^3 para x variando de 2 até 5.
2. Calcule o valor da integral de f(x)=e^{-3x} para x variando de 0 até 3.
3. Existe uma reta que passa pela origem e que divide a região limitada pela parábola $y = x - x^2$ e o eixo x em duas regiões de áreas iguais. Qual é a inclinação dessa reta?

4. Prove por meio de integração que a área de um círculo é pi.R^2
5. Prove por meio de integração que o volume de uma esfera é $\frac{4\pi R^3}{3}$
6. Prove através de integração que o volume de um cone de altura h e raio da base R é dado por $V = \frac{\pi R^2 h}{3}$
7. Calcular o volume que se encontra entre os sólidos formados pela rotação das curvas $y = x^2$ e $y = \sqrt{x}$
8. Calcular o volume formado pela rotação da curva $y = \sin^2(x)$ de $x = 0$ até $x = \pi$ ao redor do eixo dos x.
9. Encontre a área da região entre as curvas $y = x^2$ e $y = \sqrt{x}$
10. Determine a área da região limitada pela curva $y = \sqrt{x}$ e pelas retas $x = 4$ e $y = \frac{-x}{4}$.

12.5.3 Integração Numérica

1. Calcule $\int_1^3 \frac{1}{t}\,dt$ pelo método algébrico e pelo método numérico (neste caso com h=0,1) e compare os dois resultados, dando a diferença percentual entre ambos. Dicas: $\frac{dln(t)}{dt} = \frac{1}{t}$ e Dif% entre A e B é igual a: (A-B)/B, onde B é o valor exato e A o valor aproximado.
2. Seja $g(x)$ a antiderivada de $f(x) = x + 5 - 2\sqrt{x^2+3}$. Tal que $g(0) = -4$ (aqui cabe uma dica: quando x é grande (positivo ou negativo), então $\sqrt{x^2+3} \cong |x|$). Encontre os valores de x para os pontos críticos de $g(x)$. Use o teste da primeira ou da segunda derivada para mostrar se os pontos encontrados são mínimos, máximos ou nenhum dos dois. (indique claramente que teste usou. Encontre os valores de x para os pontos de inflexão de $g(x)$. Indique a concavidade nos intervalos

12.5.4 Aplicações em Economia

1. Foi constatado em 1940 que a densidade populacional, em um raio de r kms do centro de Nova York, era de aproximadamente 120eˆ(-0,2r) mil pessoas por km quadrado. Estime a população que vivia em uma área situada entre 1km e 2km do centro de Nova York.
2. A taxa de consumo de água mundial t anos após 1960 foi de aproximadamente $860.e^{0,04t}$ km^3 por ano. Quanto de água foi utilizado entre 1960 e 1995?
3. Os Estados Unidos tem consumido minério de ferro a uma taxa de R'(t) milhões de toneladas por ano no instante t, em que t=0 corresponde a 1980. Sabendo que $R'(t) = 94e^{0,016t}$, calcule o valor total de gás consumido entre 1985 e 1995 através do Excel. Calcule este mesmo valor de forma algébrica e compare os dois resultados.

4. Desde 1987 a taxa de produção de gás natural nos Estados Unidos tem sido de aprox. R'(t) quatrilhões de unidades térmicas britânicas por ano no instante t, em que t=0, correspondendo a 1987, e R'(t) = 17,04$e^{0,016t}$. Com base nestes dados, calcule o valor consumido entre 1987 e 1992.

5. Suponha que a taxa do consumo de petróleo com a crise de 1974 seja dada pela formula $R_1(t) = 21,3\ e^{0,039(t-4)}$ bilhões de barris/ano para t >=4 onde t = 4 corresponde a 1974. Se a taxa de consumo sem a crise fosse de $R_0(t) = 17,2\ e^{0,074t}$ bilhões de barris/ano, para t>=0 onde t = 0 corresponde a 1970, calcule a quantidade total de petróleo que não foi consumida entre 1976 e 1980.

6. A taxa de variação do consumo de petróleo dos EUA nos anos 80 (em bilhões de barris por ano) pode ser representa pela função: $C(t) = 27,08e^{\frac{t}{25}}$. Onde t é o número de anos após 1° de Janeiro de 1980. Sendo assim determine.

   (a) O consumo total de petróleo nos EUA de 1° de Janeiro de 1980 a 1° de Janeiro de 1990.

   (b) Sabendo que a taxa de variação da produção de petróleo nos EUA na década de 80 obedeceu a seguinte função: $P(t) = \ln(10t) + 25$, e que os EUA importaram todo seu déficit de produção de petróleo, determine quantos barris de petróleo os EUA tiveram de importar na década de 80. DICA: Utilize Integrais definidas.

7. O custo marginal de uma companhia é 0,019x^2-2x+79 onde x é o número de unidades produzidas em um dia. A companhia tem custos fixos de $1100 por dia. Pede-se:

   (a) Determinar o custo para se produzir x unidades por dia

   (b) Suponha que o nível de produção atual é de x=35. Determine em quanto os custos aumentariam, se o nível de produção fosse elevado para x=47

8. Uma pequena loja de gravatas chega a conclusão de que quando o nível de vendas é de x gravatas por dia, seu lucro marginal é de MP(x) dólares por gravata, em que MP(x) = 1,30 + 0,06x – 0,0018x^2 Além disso a loja irá perder US$ 95 por dia se o nível de produção for de 0 gravatas. Encontre o lucro obtido com a operação da loja quando o nível de vendas é de x gravatas por dia

9. Um produtor de sabão estima que seu custo marginal com a produção de sabão em pó seja de 0,2x + 1 centena de dólares por tonelada produzida, quando o nível de produção é de x toneladas por dia. Os custos fixos são de US$200 por dia. Encontre o custo para se produzir 20 toneladas de sabão em pó por dia.

10. Suponha que o processo de perfuração de um poço de petróleo tenha um custo fixo de US$ 10.000 e um custo marginal de C'(x) = 1000 + 50x

dólares por metro, onde x é a profundidade em metros. Encontre uma expressão para C(x), o custo total para se perfurar x metros. (Observação C(0) = 10.000)

11. O lucro marginal de uma certa companhia e MP1(x) = -x^2 + 14x -24. A companhia espera que o nível de produção diária aumente de x=6 para x=8 unidades. A direção está considerando um plano que deveria ter como efeito uma mudança no lucro marginal para M2(x) = -x^2 + 12x -20. A companhia deve ou não adotar esse plano? Determine a área entre os gráficos das duas funções lucro marginal de x=6 a x=8. Interprete esta área no contexto da economia

12. O consumo mundial de cigarros (em trilhões de cigarros por ano) desde 1960 é aproximadamente dado pela função c(t) = 0,1t + 2,4 em que t=0 corresponde a 1960. Determine o número de cigarros vendidos de 1980 a 1998.

13. O corte de floresta é um dos maiores problemas enfrentados no sul do Saara, na África. Mesmo que a principal razão para o corte de florestas tenha sido o estabelecimento de fazendas, a crescente demanda de carvão vegetal vem se tornando um fator importante. A taxa de consumo de carvão vegetal (em milhões de metros cúbicos por ano) no Sudão t anos após 1980 é dada aproximadamente pela função c(t) = 76,2 $e^{0,03t}$. Considerando-se que a taxa de crescimento de novas árvores (em milhões de metros cúbicos por ano) no Sudão após t anos após 1980 é dada aproximadamente pela função g(t) = 50 – 6,03 $e^{0,09t}$, obtenha o valor de floresta destruída líquida, isto é levando-se em conta o reflorestamento e o desmatamento no país.

14. A área administrativa de uma prefeitura precisa fazer o cálculo do IPTU de um terreno de área R limitado pelos gráficos das funções $f(x) = x^2 - 2x - 1eg(x) = -e^x - 1$ e pelas retas verticais $x = -1ex = 1$.

    (a) Usando técnicas de integral de definida, calcule a área do terreno R (em milhares de m^2).

    (b) A fábrica que está instalada nesta área R vai produzir um bem especifico que possui a seguinte função custo $C(x) = 100x^2 - 100x + IPTU$, considerando que o IPTU da área custa R$ 100 por cada mil metros quadrados, calcule a produção x que minimiza $C(x)$ e o custo final para esta produção.

12.5.5 Aplicações em Estatística

1. Sabendo que a probabilidade de uma variável normalizada (z) estar situada entre a e b é dada pela expressão: $P(a < z < b) = \int_a^b \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}} dz$, e que z = (x-Média)/Desvio_Padrão, calcule a probabilidade de uma variável x de média 3,5 e desvio padrão 0,9 estar situada entre 4 e 5.

Capítulo 13

Equações Diferenciais Ordinárias

A resolução numérica de equações diferenciais fornece valores para a função solução, calculando a mesma ponto a ponto, a partir de um valor inicial qualquer. Sendo assim a estrutura básica do processo de resolução é : dada uma equação diferencial $y' = f(t, y)$ e um valor inicial $y(t_0)$ para a função de solução $y(t)$, pede-se determinar o valor de $y(t)$ quando $t = t_1$. Existem vários métodos numéricos para resolver tal problema. Se a função solução for uma função de uma única variável a equação diferencial chama-se ordinária, sendo abreviada por **EDO**. Iremos abordar alguns exemplos de métodos de resolução numérica de EDOs a seguir.

13.1 O Método de Euler

A ciência moderna passou a ter um efeito no dia-a-dia das pessoas quando o homem tornou-se capaz de prever o que aconteceria no mundo real a partir de modelos teóricos de alta precisão. Dado que no mundo, qualquer coisa que se queira medir sofre uma variação ao longo do tempo, o modelo preferido dos pesquisadores é aquele que prevê o que ocorrerá no futuro, a partir da determinação de como nossa variável de interesse altera o seu valor. Isto no entanto é muito mais simples do que parece de ser enunciado. O que vamos dizer é apenas que o valor futuro de algo que estamos medindo será igual ao seu valor atual mais uma variação. Por sua vez esta variação será dada pelo produto do ritmo ou da sua taxa de variação vezes o tempo que ela passou variando neste ritmo.

Vamos chamar o valor atual de nossa medida, por exemplo o número de pessoas em uma determinada região de $f(t)$. Este símbolo quer dizer apenas que no instante t o número de pessoas era $f(t)$. Para encontrar $f(t)$ a partir de t

geralmente devemos realizar algumas operações matemáticas. Por exemplo se dizemos que $f(t) = t^2 + 2$, isto indica que quando $t = 3$, $f(t)$ será igual a 11. Por outro lado se dizemos $f(t + h)$ isto quer dizer que teremos que calcular $(t + h)^2 + 2$.

Agora suponha que você não conhece a forma matemática de $f(t)$ e sim a forma de $f'(t)$ isto é de sua taxa de variação além do valor numérico de $f(t)$ em algum instante de tempo qualquer, por exemplo em $t = 0$ e você quer calcular o valor de $f(t)$ em um instante próximo, porém no futuro, digamos 0,1 unidades de tempo para frente. Sendo assim você faz $f(t + h)$ isto é $f(0 + 0, 1)$ aproximadamente igual ao seu valor atual, isto é $f(0)$ mais sua taxa de variação no momento atual $f'(t)$ vezes o tempo que você vai para o futuro h isto é 0,1 unidades de tempo. Em linguagem matemática escrevemos simplesmente :

$$f(t + h) \approx f(t) + f'(t).h$$
$$f(0 + 0, 1) = f(0, 1) \approx f(0) + f'(0).0, 1$$

E se quisermos calcular o valor da função em $t = 0, 2$ como proceder? Pense o seguinte, temos agora o valor de $f(0, 1)$ e uma forma matemática de calcular $f'(0, 1)$. Com isto reaplicamos o raciocínio descrito acima e fazemos :

$$f(t + h) \approx f(t) + f'(t).h$$

$$f(0, 1 + 0, 1) = f(0, 2) \approx f(0, 1) + f'(0, 1).0, 1$$

Este raciocínio pode ser aplicado quantas vezes quisermos (desde de que h seja pequeno, algo como 0,1 na maioria dos casos) e assim podemos calcular o valor de $f(t)$ em qualquer instante futuro. Por exemplo, se queremos calcular $f(t)$ em $t = 1$ continuamos o processo anterior:

$$f(0, 2 + 0, 1) = f(0, 3) \approx f(0, 2) + f'(0, 2).0, 1$$

$$f(0, 3 + 0, 1) = f(0, 4) \approx f(0, 3) + f'(0, 3).0, 1$$

$$f(0, 4 + 0, 1) = f(0, 5) \approx f(0, 4) + f'(0, 4).0, 1$$

$$f(0, 5 + 0, 1) = f(0, 6) \approx f(0, 5) + f'(0, 5).0, 1$$

$$f(0, 6 + 0, 1) = f(0, 7) \approx f(0, 6) + f'(0, 6).0, 1$$

$$f(0,7+0,1) = f(0,8) \approx f(0,7) + f'(0,7).0,1$$

$$f(0,8+0,1) = f(0,9) \approx f(0,8) + f'(0,8).0,1$$

e finalmente :

$$f(0,9+0,1) = f(1,0) \approx f(0,9) + f'(0,9).0,1$$

Perceba que o processo é muito simples, porém tedioso e repetitivo. É neste ponto que entra um software de implementação rápida de funções como o Python. Através dele podemos repetir estes cálculos com um mínimo de esforço e concentrar nosso tempo na modelagem do problema, isto é em determinar a partir do enunciado do mesmo, a forma da taxa de variação de $f(t)$, isto é $f'(t)$.

1. Crescimento Populacional. Suponha que em uma determinada região, a quantidade de pessoas $P(t)$ no instante $t = 0$ seja de 1.000 indivíduos, isto é $P(0) = 1000$ e sua taxa de variação seja de 0.03 da população atual por ano, ou seja $P'(t) = 3/100.P(t)$. Com base nestes dados estime a população no ano $t = 20$. Considere $h = 1/10$ como valor do passo de avanço.

A listagem Python com a resolução deste problema é apresentada a seguir, o data frame Pandas com os resultados pode ser visto na **??** e o gráfico resultante pode ser visto na 13.1. O resultado é de 1.820 pessoas.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

def euler(xi, yi, xf, h = 0.1, desenha=True):
  yL = lambda y0:0.03*y0
  x = [xi]; y = [yi]
  while x[-1] ≤ xf:
    x0 = x[-1]; x1 = x0 + h; x.append(x1)
    y0 = y[-1]; yL0 = yL(y0); y1 = y0 + yL0*h
    y.append(y1)
  dados = pd.DataFrame({'x':x, 'y':y})
  if desenha:
    g = dados.plot(x='x', y='y')
    plt.show();
  return(dados)

dados = euler(0,1000,20,0.1)
```

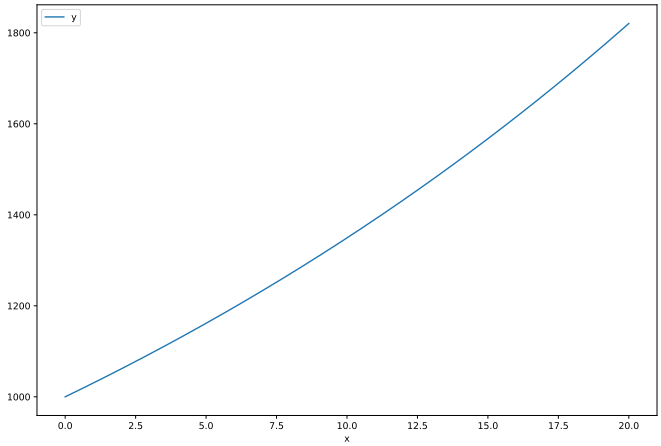


Figura 13.1: Gráfico do Problema de Crescimento Populacional

2. Crescimento Populacional com limite

O exemplo anterior tem uma deficiência pois a população pode crescer sem limite, o que não ocorre na prática. Na realidade, ao se aproximar de um valor ideal e em seguida ultrapassa-lo, a taxa de crescimento cai e em seguida passa a assumir valores negativos, provocando queda na população.

O primeiro a pensar sobre este tema foi Malthus ao perceber que a produção de alimentos cresce em um ritmo linear (na ausência de mudanças tecnológicas significativas), isto é proporcional a quantidade de área plantada e a população crescia em um ritmo exponencial isto é proporcional ao seu valor atual (na ausência de mecanismos eficazes de controle da natalidade).

Neste exemplo a taxa de variação da população ao longo do tempo é dada por $y' = a * (M - y).y$. Perceba que agora o gráfico da derivada é uma parábola. Sendo assim, supondo que em t=0, y seja pequeno, porém positivo, a derivada será também positiva e pequena, ou seja a função terá um crescimento lento. A medida que o tempo passa, e y cresce, a derivada y' atingirá um valor máximo (quando a função y terá sua inclinação máxima).

Em seguida a derivada y' será ainda positiva porém cada vez menor, ou seja y terá uma inclinação cada vez menor, até que para um determinado instante de tempo y' atingirá o zero e y será horizontal.

Sendo assim a função $y(t)$ que satisfaz a equação $y' = a * (M - y).y$ é uma função com crescimento exponencial para pequenos valores de y e crescimento limitado a partir de um determinado valor de y.

```
import numpy as np
```

```
import pandas as pd

M = 10000
a = 0.03
y = np.linspace(0,M,10000)
yL = a*(M-y)*y

from matplotlib import pyplot as plt
g = plt.plot(y,yL);
plt.show()
```

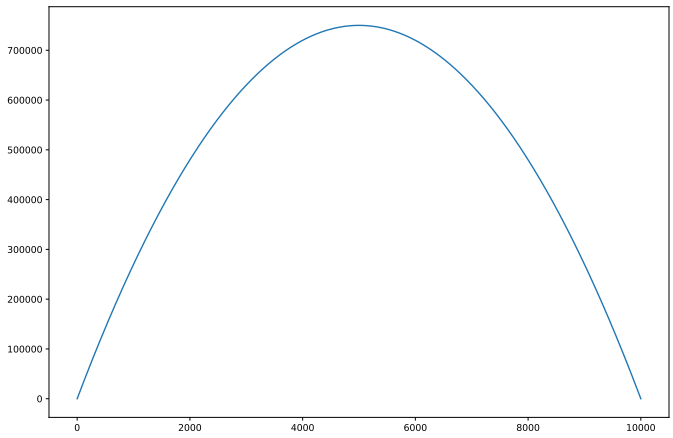


Figura 13.2: Grafico de y' vs. y na Função Logística

O desenvolvimento da solução algébrica é apresentado a seguir:

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

$$\frac{dy}{(M-y)y} = adt$$

$$\frac{1}{M(M-P)} + \frac{1}{MP}dp = adt$$

$$\frac{dp}{(M-P)} + \frac{dp}{P} = aMdt$$

$$-ln(M-y) + ln(y) = aMt + C$$

$$ln(\frac{y}{M-y}) = aMt + C$$

$$\frac{y}{M-y} = e^{aMt+C}$$

$$y = \frac{M}{1+Ce^{-aMt}}$$

Aplicando a condição inicial $y(0) = 100$ temos:

$$y_0 = \frac{M}{1+C}$$

$$C = \frac{M-y_0}{y_0}$$

o que nos leva a:

$$y = \frac{M}{1+\frac{M-y_0}{y_0}e^{-aMt}}$$

Na figura 13.3 e na **??** podem ser vistos os gráficos e a tabela de $y(t)$ obtidos através do método de Euler e da função de solução exata, para $M = 10.000$, $y(0) = 100$, $a = 5.10^{-6}$ e $t = [0, 200]$. Como pode ser visto, o método de Euler apresenta uma grande concordância com a solução exata, neste caso.

```
import numpy as np

def euler(xi, yi, xf, h, desenha=True):
  M = 10000; a = 5*10**(-6); y0 = 100.0; C = (M-y0)/y0
  yL = lambda x: a*(M-x)*x
  x = [xi]; y = [yi];
  while x[-1] <= xf:
    x.append(x[-1]+h)
    y.append(y[-1] + yL(y[-1])*h)
  dados = pd.DataFrame({'x':x, 'y':y})
  f = lambda x: M/(1+C*np.exp(-a*M*x))
  dados['yexata'] = list(map(f,dados['x'].values))
  if desenha:
    g = dados.plot(x='x', y=['y','yexata'])
    plt.show()
  return(dados)

dados = euler(0,100,200,0.1)
```

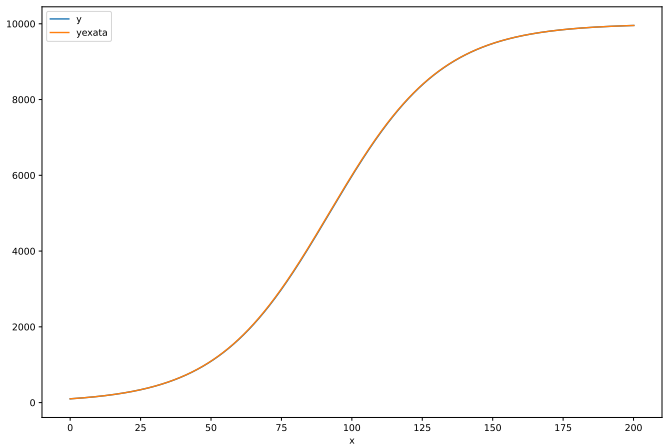


Figura 13.3: Gráfico com a Solução da Equação Logística pelo Método de Euler

Quais as limitações do método de Euler? A principal delas esta relacionada ao tamanho do passo h. Para que o método funcione corretamente o passo deve ser suficientemente pequeno. Porém isto em alguns casos pode levar a um aumento do erro por arredondamento, devido a grande quantidade de operações necessárias para a obtenção de um resultado. Por outro lado, tamanhos de passo muito grandes podem levar o método a instabilidade. Sendo assim, para a utilização do método de Euler é necessário um compromisso entre a precisão necessária e a estabilidade do processo de cálculo.

3. Apresente os gráficos de solução da equação logística para os valores h de passo iguais da 1, 35 e 50.

Com $h = 1$ temos uma grande concordância da solução numérica com a solução exata. Com $h = 35$ a solução começa a oscilar e se torna totalmente distinta do resultado correto. Esta oscilação ocorre devido à distância h que esta sendo utilizada em cada iteração. A partir de um certo valor tudo se passa como se de um instante para o outro a população tivesse ultrapassado o valor limite, portanto a derivada torna-se negativa, porém como o passo é muito longo a função torna-se muito pequena, necessitando de outro ajuste radical de valor. Este processo como pode ser visto, pode ou não ser corrigido ao longo do tempo. Se o passo ainda for menor que um valor limite, a oscilação é lentamente eliminada, porém se o passo for suficientemente grande, isto pode levar a solução à uma oscilação permanente.

```
from matplotlib import pyplot as plt
fig, ax = plt.subplots()
```

```
hs = [1, 35, 50]
for i, h in enumerate(hs):
  dados = euler(0,100,350,h,False)
  dados.rename({'y':'h'+str(h)}, axis=1, inplace=True)
  if i == 1:
    ax = plt.plot(dados['x'].values, \
                  dados['yexata'].values, \
                  label = 'exata');
  ax = plt.plot(dados['x'].values, \
                dados['h'+str(h)].values, \
                label = 'h'+str(h));

g = plt.legend();
plt.show()
```

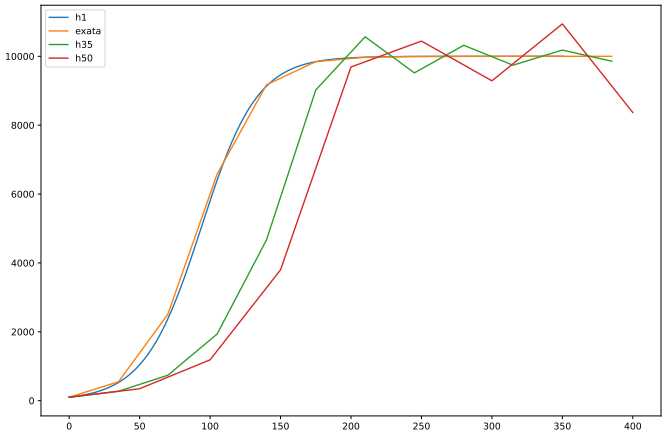


Figura 13.4: Soluções da Equação Logística para Diferentes Passos h

4. Oscilação de Um Sistema Massa Mola

Neste problema vamos analisar o caso de um sistema massa-mola, horizontal, com atrito dinâmico, regido pela equação diferencial :

$$m\frac{d^2x}{dt^2} = -kx - \mu\frac{dx}{dt} \tag{13.1}$$

$$x(0) = x_0 \tag{13.2}$$

$$x'(0) = v_0 \tag{13.3}$$

Onde m é a massa da mola, k é a constante elástica da mola, x_0 o deslocamento inicial do sistema, v_0 a velocidade inicial e t a variável independente. O objetivo deste problema é traçar o gráfico da coordenada x do centro de massa do objeto suspenso pela mola ao longo do tempo, considerando diferentes combinações de massa, constante elástica e deslocamento inicial.

Para resolver este problema numericamente no Python, transformamos a equação diferencial ordinária linear de segunda ordem acima em um sistema de equações diferenciais ordinárias lineares de primeira ordem fazendo :

$$v' = -\frac{k}{m}x - \frac{\mu}{m}v \tag{13.4}$$

$$x' = v \tag{13.5}$$

Em seguida resolvemos o sistema através do método de Euler, para o seguinte conjunto de parâmetros :

$$\begin{aligned} \mu &= 0,28 \\ k &= 0,1 \\ m &= 6kg \\ x(0) &= 5m \\ v(0) &= -10m/s \\ h &= 0,05 \end{aligned}$$

Lembrando que o método de Euler resolve uma equação diferencial calculando o valor da função ponto a ponto através da seguinte relação : $f(t+h) \approx f(t)+f'(t).h$. Em outras palavras, o método afirma que o valor de f(t) em um ponto t+h é obtido a partir do valor atual de f(t) e do valor atual de f'(t) calculando-se f(t+h) a partir relação acima.

O data frame com os resultados pode ser visto na **??** e o gráfico correspondente ao valor da posição no eixo horizontal ao longo do tempo pode ser visto na 13.5.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.28*v)/6
xL = lambda t,x,v : v

t, x, v = 0, 5, -10
tf, dt = 100, 0.1
sol = []
```

```
while t<tf:
  sol.append([t,x,v])
  t, x, v = t + dt, x + xL(t,x,v,)*dt, v + vL(t,x,v)*dt
sol = pd.DataFrame(data = sol, columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v'])
plt.show()
```

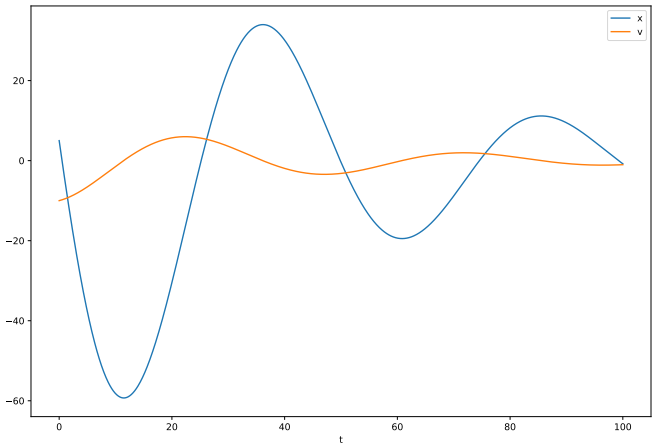


Figura 13.5: Resolução do Sistema de Oscilação Massa Mola

5. Limitações do método de Euler: amplitude crescente quando $\mu = 0$ (amortecimento igual a 0)

Fazendo $m = 6$ e $\mu = 0$ (o que equivale a tornar o amortecimento nulo) obtemos a solução abaixo. Como pode ser visto o gráfico resultante da posição ao longo do tempo tem amplitude crescente, o que significa que o sistema estaria ganhando energia, o que claramente não é o caso.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.0*v)/6
xL = lambda t,x,v : v

t, x, v = 0, 5, -10
tf, dt = 1000, 0.1
```

```
sol = []

while t<tf:
  sol.append([t,x,v])
  t, x, v = t + dt, x + xL(t,x,v,)*dt, v + vL(t,x,v)*dt
sol = pd.DataFrame(data = sol, columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v']);
```

```
/home/gustavo/miniconda3/envs/deeplearn/lib/python3.9/site-packages/pandas/p
  fig = self.plt.figure(figsize=self.figsize)
```

```
plt.show()
```

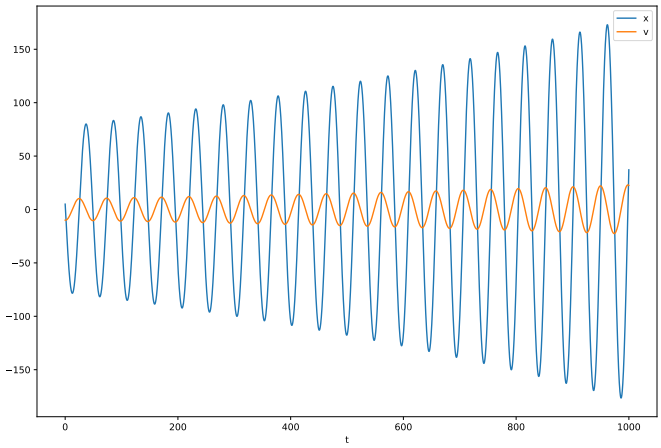


Figura 13.6: Amplitude de oscilação crescente quando $\mu = 0$

O que ocorre neste caso é que o método de Euler produz resultados a príncipio com um pequeno erro. No entanto a natureza recursiva do método (lembre-se que para calcular o valor em f(t=3) devemos calcular antes o valor em f(t=3-h) e assim sucessivamente) leva a um acúmulo dos erros. A primeira vista, a solução seria utilizar um valor bem menor para o passo h, pois isto iria diminuir o erro em cada cálculo. No entanto esta abordagem tem dois incovenientes : 1^o) ela aumenta a quantidade de operações necessárias, o que aumenta o tempo e a potência de processamento necessária e 2^o) ao aumentar a quantidade de operações necessárias, entra em cena o erro de arredondamento inerente às operações feitas em um computador. Esta limitação será abordada em maiores

detalhes nas próximas seções quando for apresentado o método mais utilizado na resolução numérica de equações diferenciais, chamado de Método de Runge-Kutta de 4^a ordem.

13.2 Método de Runge-Kutta de 4^a ordem

No exemplo 13.1 (Crescimento Populacional Logístico) foi mostrado que o método de Euler possui limitações relativas ao tamanho de passo. Outra limitação surgiu no exemplo 5, quando o amortecimento μ foi igualado a zero. Para minimizar tais problemas, existe outro método mais preciso, chamado de Método de Runge-Kutta de 4^a ordem. Este método na verdade pertence a uma família de métodos, todos denominados Métodos de Runge-Kutta em homenagem aos matemáticos alemães que desenvolveram tal conjunto de técnicas no início do século XX. O mais famoso (por ser o de maior aplicabilidade) é o método de 4^{a} ordem que descrevemos a seguir. Neste método o valor de $f(t+h)$ é calculado a partir do seguinte conjunto de operações :

$$\begin{aligned} y' &= f(t,y) \\ k_1 &= f(t,y) \\ k_2 &= f(x+\frac{h}{2}; y+\frac{h}{2}.k_1.h) \\ k_3 &= f(x+\frac{h}{2}.h; y+\frac{h}{2}.k_2.h) \\ k_4 &= f(x+h; y+k_3.h) \\ y(t+h) &= y(t)+\frac{(k_1+2k_2+2k_3+k_4)}{6} \end{aligned}$$

Vamos exemplificar este método calculando a solução no caso do crescimento populacional com limite, exemplo 13.1.

6. Crescimento Populacional com limite, resolução através do método de Runge-Kutta de 4^a ordem.

Neste caso estaremos resolvendo a equação diferencial :

$$y(t) = \frac{M}{1+ke^{-aMt}} \tag{13.6}$$

$$k = \frac{M-y_0}{y_0} \tag{13.7}$$

Para o seguinte conjunto de parâmetros e condições :

$$M = 200.000 \tag{13.8}$$
$$y_0 = 1.000 \tag{13.9}$$
$$a = 5.10^{-6} \tag{13.10}$$

A seguir na 13.7, para um passo de solução $\Delta t = 50$ são apresentados três curvas de solução: exata, Runge-Kutta e Euler. Observe que apesar de apresentar um erro razoável na parte intermediária, o método de Runge-Kutta ainda assim converge para o valor limite correto, enquanto o método de Euler diverge.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

a = 5e-06; M = 10000; p0 = 100
pL = lambda t,p : a*(M-p)*p
euler = lambda t,p,dt : [t+dt, p+pL(t,p)*dt]
exata = lambda t : M / (1 + (M-p0)/p0 * np.exp(-a*M*t))

def rk4(t,p,dt):
  k1 = pL(t,p)
  k2 = pL(t+dt/2, p+k1*dt/2)
  k3 = pL(t+dt/2, p+k2*dt/2)
  k4 = pL(t+dt, p+k3*dt)
  p = p + dt/6*(k1+2*k2+2*k3+k4)
  t = t+ dt
  return([t,p])

def edo(t, p, dt, tf, metodo):
  sol = [[t,p]]
  while t≤tf:
    t,p = metodo(t, p, dt)
    sol.append([t,p])
  sol = pd.DataFrame(data = sol, columns=['t','p'])
  return(sol)

ft = edo(t=0, p=100, dt=50, tf=1000, metodo=euler)
ft.rename(columns = {'p':'euler'},inplace=True)
ft_rk4 = edo(t=0, p=100, dt=50, tf=1000, metodo=rk4)
ft['rk4'] = ft_rk4['p']
ft['exata'] = list(map(exata,ft['t']))
graf = ft.plot(x='t',y=['euler','rk4','exata'])
plt.show()
```

X	V
$x' = f(t, x, v)$	$v' = g(t, x, v)$
$k_1 = f(t, x, v)$	$k_1 = g(t, x, v)$
$k_2 = f(t + \frac{h}{2}; x + k_1\frac{h}{2}; v + k_1\frac{h}{2})$	$k_2 = g(t + \frac{h}{2}; x + k_1\frac{h}{2}; v + k_1\frac{h}{2})$
$k_3 = f(t + \frac{h}{2}; x + k_2\frac{h}{2}; v + k_2\frac{h}{2})$	$k_3 = g(t + \frac{h}{2}; x + k_2\frac{h}{2}; v + k_2\frac{h}{2})$
$k_4 = f(t + h; x + k_3.h; v + k_3.h)$	$k_4 = g(t + h; x + k_3.h; v + k_3.h)$
$x(t + h) = x(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$	$v(t + h) = v(t) + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Tabela 13.1: Sequência de passos para aplicação do Método de Runge-Kutta de 4^a ordem ao exemplo 5

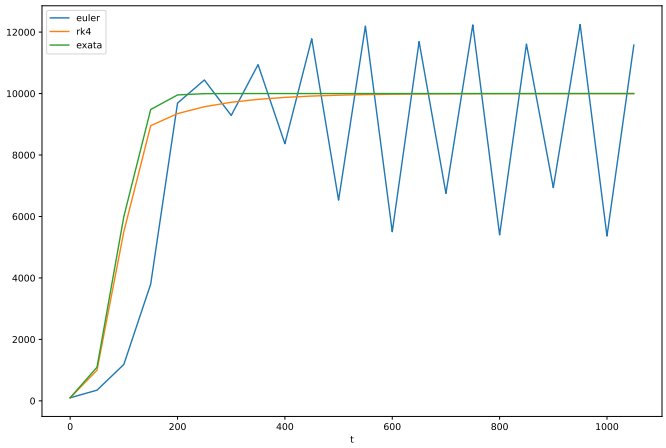


Figura 13.7: Solução da Equação Logística Exata e Numérica pelos Métodos de Euler e Runge-Kutta 4a Ordem

7. Sistema Massa-Mola: Solução quando $\mu = 0$ pelo Método de Runge-Kutta 4a Ordem

No exemplo 5, o sistema Massa-Mola foi analisado para o caso em que o amortecimento μ tornava-se zero. É de se esperar que a amplitude da oscilação se mantenha constante, pois dado que $\mu = 0$ o sistema não ganha nem perde energia. Vamos resolver este caso através do Método de Runge-Kutta de 4^a ordem e verificar se ele produz um resultado correto nesta situação.

A seguir é apresentada a sequência de cálculos para um RK4 em um sistema de duas equações de primeira ordem, tal como foi feito com o Método de Euler, quando aplicado ao sistema do exemplo 5.

A implementação é mostrada passo a passo a seguir, com $h = 0,05$, $u = 0$, $k = 0,1$, $m = 4kg$, $x(0) = 50m$ e $v(0) = 50m/s$

$$\begin{aligned}
x' &= f(t,x,v) = v \\
x'(0) &= v(0) = 50 \\
k1 &= v(0) = 50 \\
k2 &= v(0) + 0,5.h.k1 = 50 + 0,5.0,05.50 = 51,25 \\
k3 &= v(0) + 0,5.h.k2 = 50 + 0,5 * 0,05 * 51,25 = 51,281 \\
k4 &= v(0) + h.k3 = 50 + 0,05 * 51,28125 = 52,564 \\
x(0+0,05) &= x(0) + h\tfrac{(k1+2.k2+2.k3+k4)}{6} \\
&= 50 + 0,05\tfrac{(50+51,25+51,281+52,564)}{6} = 52,564 \\
x(0,05) &= 52,564
\end{aligned} \tag{13.11}$$

$$\begin{aligned}
v' &= g(t,x,v) = -\tfrac{K}{m}x - \tfrac{u}{m}.v \\
v' &= g(t,x,v) = -\tfrac{0,1}{4}x(t) - \tfrac{0}{4}v(t) = -0,025.x(t) \\
v' &= -0,025.x \\
x(0) &= 50 \\
v(0) &= 50 \\
v'(0) &= -0,025.x(0) = -0,025.50 = -1,25 \\
k1 &= g(t,x,v) = v'(0) = -1,25 \\
k2 &= g(t+0,5.h.k1; x+0,5.h.k1; v+0,5h.k1) = -0,025(x+0,5.h.k1) \\
k2 &= -0,025 * [50 + 0,5 * 0,05 * (-1,25)] = -1,249 \\
k3 &= g(t+0,5.h.k2; x+0,5.h.k2; v+0,5h.k2) = -0,025(x+0,5.h.k2) \\
k3 &= -0,025[50 + 0,5 * 0,05 * (-1,249)] = -1,249 \\
k4 &= g(t+h.k3; x+h.k3; v+h.k3) = -0,025(x+h.k3) \\
k4 &= -0,025 * [50 + 0,05 * (-1,249)] = -1,248 \\
v(0+0,05) &= v(0) + h\tfrac{(k1+2.k2+2.k3+k4)}{6} = 50 + 0,05\tfrac{(-1,25-1,249-1,249-1,248)}{6} = 49,938 \\
v(0,05) &= 49,938
\end{aligned} \tag{13.12}$$

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

vL = lambda t,x,v : -(0.1*x+0.0*v)/6
xL = lambda t,x,v : v

t, x, v = 0, 5, -10
tf, dt = 1000, 0.1
sol = []

while t<tf:
  sol.append([t,x,v])
  [k1, l1] = [fg(t,x,v) for fg in [xL,vL]]
```

```
  [k2, l2] = [fg(t+dt/2, \
               x+k1*dt/2, \
               v+l1*dt/2) for fg in [xL,vL]]
  [k3, l3] = [fg(t+dt/2, \
               x+k2*dt/2, \
               v+l2*dt/2) for fg in [xL,vL]]
  [k4, l4] = [fg(t+dt, \
               x+k3*dt, \
               v+l3*dt) for fg in [xL,vL]]
  t, x, v = t + dt, \
            x + (k1+2*k2+2*k3+k4)*dt/6, \
            v + (l1+2*l2+2*l3+l4)*dt/6
sol = pd.DataFrame(data = sol, \
                   columns = ['t','x','v'])

graf = sol.plot(x='t',y=['x','v'])
plt.show()
```

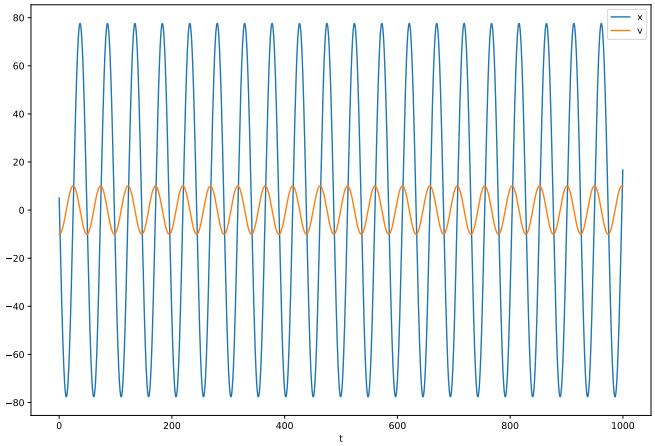


Figura 13.8: Sistema Massa Mola com $\mu = 0$, solução pelo Método de Runge-Kutta de 4a Ordem

8. Periodo de Oscilação de Um Sistema Massa-Mola

Nosso objetivo é determinar como o periodo de oscilação do sistema se altera a medida que a massa do objeto aumenta. Para tanto vamos determinar os instantes em que o sistema cruza dois picos quaisquer e calcular a diferença entre os picos consecutivos. Ao final definimos o periodo como a média desta sequência de valores.

Para determinar os instantes em que a função atingiu um ponto extremo (máximo ou mínimo) levamos em consideração que neste ponto a primeira derivada da função irá trocar de sinal (em termos algébricos dizemos que $x'(t+\Delta h) * x'(t) < 0$).

A resolução deste problema, para um sistema com os mesmos parâmetros do exemplo anterior e distintos valores de m pode ser vista a seguir:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

m = 6
vL = lambda t,x,v : -(0.1*x+0.28*v)/m
xL = lambda t,x,v : v

t, x, v, dt, tf = 0, 5, -10, 0.1, 100

def sedol(t,x,v,dt,tf):
  sol = [[t,x,v]]
  while t<tf:
    [k1, l1] = [fg(t,x,v) for fg in [xL,vL]]
    [k2, l2] = [fg(t+dt/2, \
                x+k1*dt/2, \
                v+l1*dt/2) for fg in [xL,vL]]
    [k3, l3] = [fg(t+dt/2, \
                x+k2*dt/2, \
                v+l2*dt/2) for fg in [xL,vL]]
    [k4, l4] = [fg(t+dt, \
                x+k3*dt, \
                v+l3*dt) for fg in [xL,vL]]
    t, x, v = t + dt, \
              x + (k1+2*k2+2*k3+k4)*dt/6, \
              v + (l1+2*l2+2*l3+l4)*dt/6
    sol.append([t,x,v])
  sol = pd.DataFrame(data = sol, \
                     columns = ['t','x','v'])
  return(sol)

periodos = []
for m in np.linspace(1,12,12):
  ft_runge = sedol(t, x, v, dt, tf)
  ft_runge = ft_runge[ft_runge['v'] * \
                      ft_runge['v'].shift(1) < 0]
  periodo = (ft_runge.t - \
             ft_runge.t.shift(1)).mean()
  periodos.append([m,periodo])
```

```
periodos = pd.DataFrame(data=periodos, \
                        columns=['massa','periodo'])
graf = periodos.plot(x='massa', \
                     y='periodo', \
                     kind='scatter')
plt.show()
```

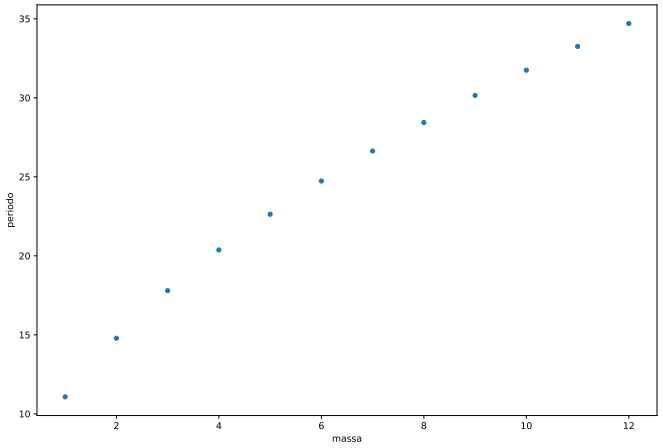


Figura 13.9: Periodo de Oscilação de Um Sistema Massa Mola

9. Sistema com dois corpos

Vamos agora aumentar a complexidade do problema. Teremos não mais um corpo oscilando ao redor do seu ponto de equilíbrio, mas sim dois corpos conectados por uma mola entre si, podendo andar livremente por um plano, com atrito. Nesta condição teremos o seguinte conjunto de equações definindo o problema :

$$m_1.x_1'' = (x_2 - x_1)K - x_1'.\mu_1$$

$$m_2.x_2'' = F - (x_2 - x_1)K - x_2'.\mu_2$$

Fazendo $v_1 = x_1'$ e $v_2 = x_2'$ teremos

$$v_1' = (x_2 - x_1)\frac{K}{m_1} - \frac{(v_1\mu_1)}{m_1}$$

$$v_2' = \frac{F}{m_2} - (x_2 - x_1)\frac{K}{m_2} - \frac{(v_2\mu_2)}{m_2}$$

Adotando: - $m_1 = 10kg$ - $m_2 = 5kg$ - $k = 0,1$ - $u_1 = u_2 = 0,15$ - $x_1(0) = x_2(0) = 0$ - $v_1(0) = 0$ - $v_2(0) = -3m/s$ - F um pulso de $1N$ de $t = 20$ segundos a $t = 50$ segundos,

teremos a evolução temporal para os respectivos centros de massa dos corpos x_1 e x_2, mostrada a seguir:

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

m1, m2, k, u1, u2 = 10, 5, 0.1, 0.15, 0.15
x1, x2, v1, v2 = 0, 0, 0, -3

x1L = lambda t, x1, v1, x2, v2, F : v1
v1L = lambda t, x1, v1, x2, v2, F : \
             (x2-x1)*k/m1 - v1*u1/m1
x2L = lambda t, x1, v1, x2, v2, F : v2
v2L = lambda t, x1, v1, x2, v2, F : \
             F/m2 -(x2-x1)*k/m2 -v2*u2/m2

def sedol(t, x1, v1, x2, v2, dt, tf):
  sol = [[t,x1,v1,x2,v2]]
  while t<tf:
    if (t >= 20) and (t <= 50):
      F = 1
    else:
      F = 0
    [a1, b1, c1, d1] = [fg(t,x1,v1,x2,v2,F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a2, b2, c2, d2] = [fg(t+dt/2, \
                           x1+a1*dt/2, \
                           v1+b1*dt/2, \
                           x2+c1*dt/2, \
                           v2+d1*dt/2, F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a3, b3, c3, d3] = [fg(t+dt/2, \
                           x1+a2*dt/2, \
                           v1+b2*dt/2, \
                           x2+c2*dt/2, \
                           v2+d2*dt/2, F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    [a4, b4, c4, d4] = [fg(t+dt, x1+a3*dt, \
                           v1+b3*dt, x2+c3*dt, \
                           v2+d3*dt, F) \
                  for fg in [x1L,v1L,x2L,v2L]]
    t, x1, v1, x2, v2 = t + dt, \
```

```
                            x1 + (a1+2*a2+2*a3+a4)*dt/6, \
                            v1 + (b1+2*b2+2*b3+b4)*dt/6, \
                            x2 + (c1+2*c2+2*c3+c4)*dt/6, \
                            v2 + (d1+2*d2+2*d3+d4)*dt/6
    sol.append([t,x1,v1,x2,v2])
  sol = pd.DataFrame(data = sol, \
                columns = ['t','x1','v1','x2','v2'])
  return(sol)

ft = sedol(0, x1, v1, x2, v2, 0.1, 100)
graf = ft.plot(x='t', y=['x1','x2'])
plt.show()
```

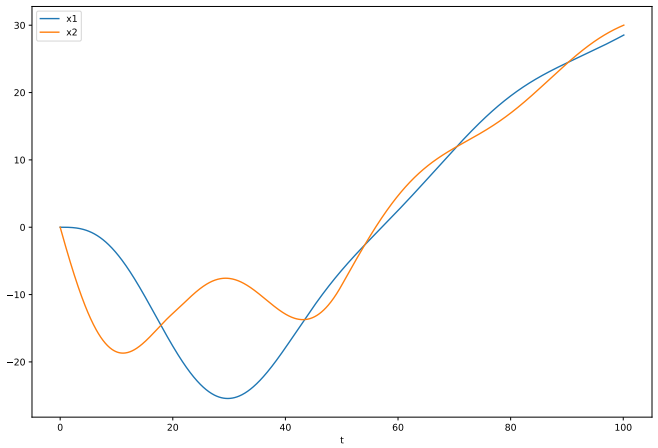


Figura 13.10: Sistema com Dois Corpos

10. Sistema com três corpos

Vamos agora resolver um sistema com três variáveis, através do método de Runge-Kutta de 4a ordem implementado com operações matriciais, para diminuir a complexidade das expressões. A base do raciocínio é que do ponto de vista matricial as três expressões podem ser combinadas em uma única operação matricial, $\vec{\mathbf{X}}' = \mathbf{A}.\vec{\mathbf{X}}$. O sistema é mostrado a seguir:

$$\begin{aligned} x_1' &= -0,1\,x_1 - 1\,x_2 - 0,4\,x_3 \\ x_2' &= 0,003\,x_1 - 0,4\,x_2 + 2\,x_3 \\ x_3' &= 0,2\,x_1 - 0,02\,x_2 - 1\,x_3 \\ x_1(0) &= 1; x_2(0) = 2; x_3(0) = 3 \end{aligned} \tag{13.13}$$

- Implementação da operação matricial $\vec{\mathbf{X}}' = \mathbf{A}.\vec{\mathbf{X}}$

O método de Runge-Kutta 4a ordem requer em seguida a implementação do seguinte conjunto de operações, para cada uma das variáveis :

$$\begin{aligned} k1 &= f\left[t; x_1; x_2; x_3\right] \\ k2 &= f\left[t + \tfrac{h}{2}; x_1 + k_1\,\tfrac{h}{2}; x_2 + l_1\,\tfrac{h}{2}; x_3 + m_1\,\tfrac{h}{2}\right] \\ k3 &= f\left[t + \tfrac{h}{2}; x_1 + k_2\,\tfrac{h}{2}; x_2 + l_2\,\tfrac{h}{2}; x_3 + m_2\,\tfrac{h}{2}\right] \\ k4 &= f\left[t; x_1 + k_3\,h; x_2 + l_3\,h; x_3 + m_3\,h\right] \\ x_1(t+h) &= x_1(t) + \tfrac{1}{6}\left(k_1 + 2\,k_2 + 2\,k_3 + k_4\right) \end{aligned} \tag{13.14}$$

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

h = np.array([0.1]).reshape(1,1)

C = np.array([[-0.100, -1.00,  -0.4],
              [ 0.003, -0.40,   2.0],
              [ 0.200, -0.02,  -1.0]])

t0 = np.zeros(1).reshape(1,1)
x0 = np.array([1,0,3]).reshape(3,1)
sol = np.concatenate((t0,x0),axis=0)

for i in range(300):
  k1 = C.dot(x0)
  k2 = C.dot(x0 + k1*h/2)
  k3 = C.dot(x0 + k2*h/2)
  k4 = C.dot(x0 + k3*h)

  t0 = t0 + h
  x0 = x0 + h/6*(k1+2*k2+2*k3+k4)
  sol0 = np.concatenate((t0,x0),axis=0)
  sol = np.concatenate((sol,sol0),axis=1)

sol = sol.T
df = pd.DataFrame(data = sol, \
                  columns = ['t','x1','x2','x3'])
```

Por último são mostrados os gráficos da evolução temporal das variáveis (13.11) e o diagrama de fases (variáveis dependentes entre si, na 13.12).

```
g = df.plot(x='t', y=['x1','x2','x3'])
plt.show()
```

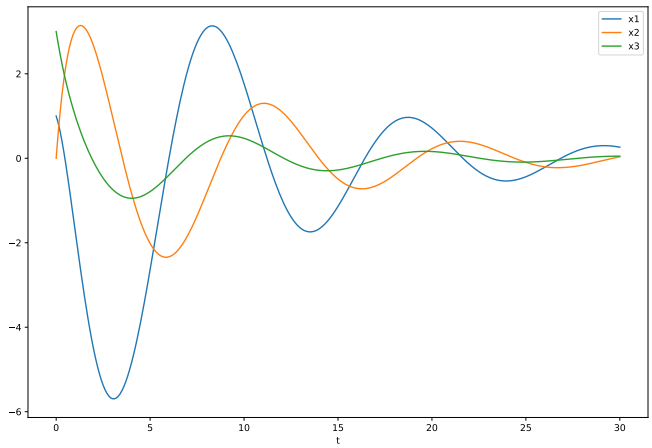


Figura 13.11: Evolução Temporal das Variáveis Sistema com Três Corpos

```
g = df.plot(x='x1', y=['x2','x3'])
plt.show()
```

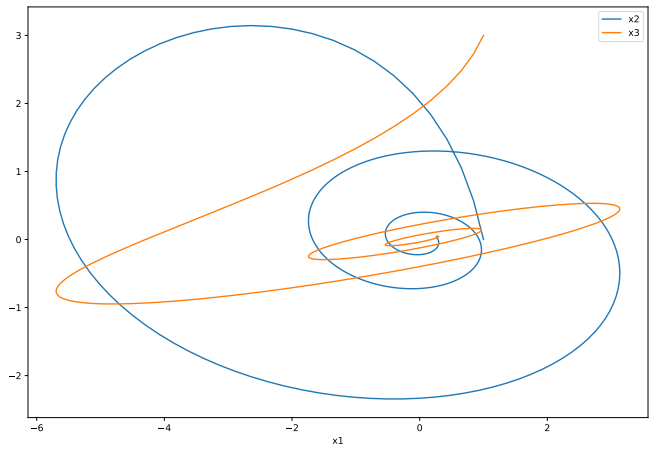


Figura 13.12: Diagrama de Fases, Variáveis x_2 e x_3 vs. x_1, Sistema com Três Corpos

11. Desenho da família de soluções de $y' = \sqrt{|y^2 - 1|}$

Neste caso desejamos desenhar diversos gráficos de $y(t)$, para diferentes condições iniciais $y_0 = y(0)$.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

def fL(t,y):
  result = np.sqrt(np.abs(y**2-1))
  return(result)

def runge4(t,y0,dt):
  k1 = fL(t,y0)
  k2 = fL(t+dt/2, y0 + k1*dt/2)
  k3 = fL(t+dt/2, y0 + k2*dt/2)
  k4 = fL(t+dt, y0 + k3*dt)
  y1 = y0 + dt/6*(k1+2*k2+2*k3+k4)
  return(y1)

ti = 0; tf = 3; n = 1000
ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]
y0s = np.linspace(-0.9,0,10)

sols = np.concatenate(([ti],y0s)).reshape(1,-1)

for t0 in ts:
  y1s = [ runge4(t0,y0,dt) for y0 in sols[-1,1:]]
  t1 = t0+dt
  sol1 = np.concatenate(([t1],y1s)).reshape(1,-1)
  sols = np.append(sols,sol1,axis=0)

nomes = list('t')
for i in range(len(y0s)):
  nome = 'y' + str(i)
  nomes.append(nome)

df = pd.DataFrame(data=sols, columns=nomes)
g = df.plot(x = 't', y = df.columns[1:])
plt.show()
```

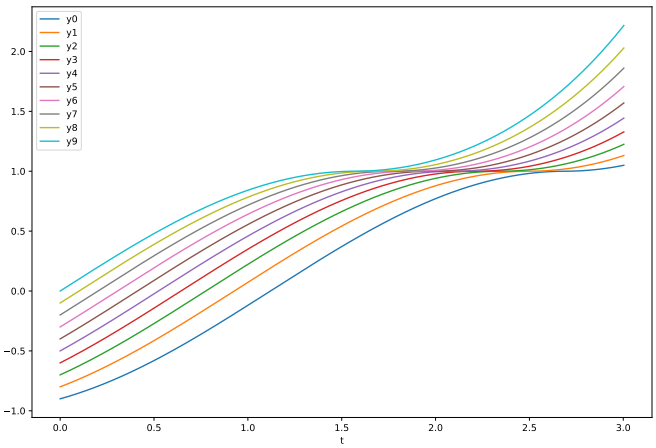


Figura 13.13: Família de Soluções da Equação de Bernoulli

13.3 Método das Diferenças Finitas

Este é um método que parte do seguinte raciocínio : a derivada tanto pode ser calculada pela definição, isto é $\frac{f(t+h)-f(t)}{h}$ quanto pela média da sua própria expressão, isto é $\frac{f'(t+h)+f'(t)}{2}$. Igualando os termos teremos :

$$\frac{f(t+h)-f(t)}{h} = \frac{f'(t+h)+f'(t)}{2} \tag{13.15}$$

Isolando $f(t+h)$ teremos :

$$f(t+h) = \frac{h}{2}\left[f'(t+h)+f'(t)\right]+f(t) \tag{13.16}$$

Como o valor de $f'(t+h)$ para ser calculado depende do valor de $f(t+h)$ temos um processo iterativo que converge linha a linha para o valor correto.

12. Compare a resolução pelo método de Euler, Runge-Kutta 4a Ordem e Diferenças Finitas, com $h = 0,5$ para a solução da equação que representa o balanço de massa em um reator, a saber :

$$V\frac{dc}{dt} = F - Q\,c - k\,V\,c^2 \tag{13.17}$$

onde $V = 12m^3$ é o volume do reator, c é a concentração a ser determinada ao longo do tempo, $F = 175g/min$ é a taxa de alimentação, $Q = 1m^3/min$

é a vazão e $k = 0,15m^3/g/min$ é a taxa de reação que "consome" o produto. Deseja-se resolver a equação até que a concentração c do produto químico no interior do reator fique estável, partindo de diferentes valores para a concentração inicial $c(0)$, a qual deve ir de 0 até 4. Analise também o caso (hipotético) de uma concentração inicial negativa comparando a solução obtida por cada um dos três métodos.

```
import numpy as np
import pandas as pd

def cL(c):
  V = 12; F = 175; Q = 1; k = 0.15
  resultado = (F - Q*c - k*V*(c**2))/V
  return(resultado)

def euler(c0,dt):
  c1 = c0 + cL(c0)*dt
  return(c1)

def runge4(c0,dt):
  k1 = cL(c0)
  k2 = cL(c0 + k1*dt/2)
  k3 = cL(c0 + k2*dt/2)
  k4 = cL(c0 + k3*dt)
  c1 = c0 + dt/6*(k1+2*k2+2*k3+k4)
  return(c1)

def diffin(c0,dt):
  # f(t+h) = h/2 * [ fL(t+h) + fL(t) ] + f(t)
  cL0 = cL(c0)
  c1 = c0 + cL0*dt
  i = 0; n_iter_max = 100;
  dif_perc = 1; dif_perc_max = 1e-6
  while ((i<n_iter_max) and (dif_perc > dif_perc_max)):
    cL1 = cL(c1)
    c1_new = dt/2 * ( cL1 + cL0 ) + c0
    dif_perc = abs(c1_new - c1)/c1_new
    c1 = c1_new
  return(c1)

ti = 0; tf = 2; n = 5;
ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]

c0 = 0
sols = np.array([ti, c0, c0, c0]).reshape(1,-1)
for t in ts:
```

```
  c0_euler = sols[-1,1]
  c1_euler = euler(c0_euler, dt)
  c0_runge = sols[-1,2]
  c1_runge = runge4(c0_runge, dt)
  c0_diffin = sols[-1,3]
  c1_diffin = diffin(c0_diffin, dt)
  sols0 = np.array([t+dt, \
                    c1_euler, \
                    c1_runge, \
                    c1_diffin]).reshape(1,-1)
  sols = np.concatenate((sols, sols0), axis=0)

df = pd.DataFrame(data=sols, \
                  columns=['t','euler',\
                           'runge','dif_fin'])
graf = df.plot(x='t', y=['euler','runge',\
                         'dif_fin'])
plt.show()
```

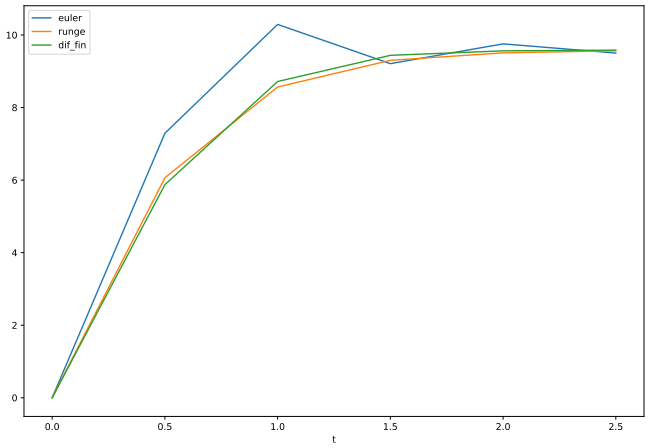


Figura 13.14: Resolução de uma EDO pelo Método das Diferenças Finitas

Além disso as figuras 13.15 e 13.16 mostram os gráficos da evolução temporal de $c(t)$ obtidos com os três métodos para os casos iniciais $c(0) = 0$ e $c(0) = -8$, lembrando sempre que esta última situação (onde a concentração inicial é negativa) é puramente teórica. Como pode-se ver, existe uma grande concordância entre o resultado prático obtido pela aplicação do método de Runge-Kutta de

4a ordem com precisão extendida e o método das diferenças finitas. Para este exemplo vamos criar uma função que irá gerar a família de soluções.

```
import numpy as np
import pandas as pd

def cL(c0):
  V = 12; F = 175; Q = 1; k = 0.15
  resultado = (F - Q*c0 - k*V*(c0**2))/V
  return(resultado)

def euler(c0, dt):
  c1 = c0 + cL(c0)*dt
  return(c1)

def runge(c0, dt):
  k1 = cL(c0)
  k2 = cL(c0 + k1*dt/2)
  k3 = cL(c0 + k2*dt/2)
  k4 = cL(c0 + k3*dt)
  c1 = c0 + dt/6*(k1+2*k2+2*k3+k4)
  return(c1)

def difin(c0,dt):
  dif_max = 1e-6
  cL0 = cL(c0)
  c1 = c0 + cL0 * dt

  for i in range(100):
    cL1 = cL(c1)
    c1_new = c0 + (cL1 + cL0)/2 * dt
    dif_per = abs((c1_new - c1)/c1_new)
    c1 = c1_new
    if dif_per < dif_max:
      break
  return(c1)

def familia(c0, ti, tf, n):
  ts = np.linspace(ti, tf, n); dt = ts[1] - ts[0]
  sols = np.array([ti, c0, c0, c0]).reshape(1,-1)
  for t in ts:
    t1 = sols[-1,0] + dt
    c0_euler = sols[-1,1]
    c1_euler = euler(c0_euler,dt)
    c0_runge = sols[-1,2]
    c1_runge = runge(c0_runge,dt)
```

```
    c0_difin = sols[-1,3]
    c1_difin = difin(c0_difin,dt)
    sol0 = np.array([t1, \
                     c1_euler, \
                     c1_runge, \
                     c1_difin]).reshape(1,-1)
    sols = np.concatenate((sols, sol0), axis=0)
  df = pd.DataFrame(data = sols, \
                    columns = ['t','euler',\
                               'runge','difer'])
  return(df)

c0 = 0
ti = 0; tf = 2; n = 5
df = familia(c0, ti, tf, n)
graf = df.plot(x = 't', \
               y = ['euler','runge','difer'])
plt.show()
```

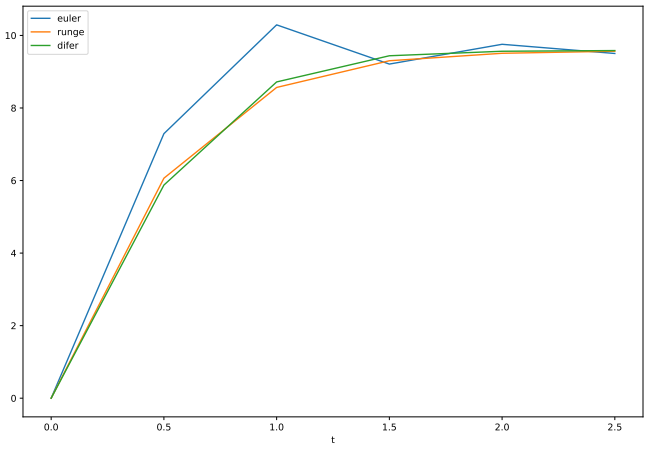


Figura 13.15: Evolução Temporal de $c(t)$ para $c(0) = 0$

```
c0 = -8
ti = 0; tf = 2; n = 5
df = familia(c0, ti, tf, n)
graf = df.plot(x = 't', \
               y = ['euler','runge','difer'])
plt.show()
```

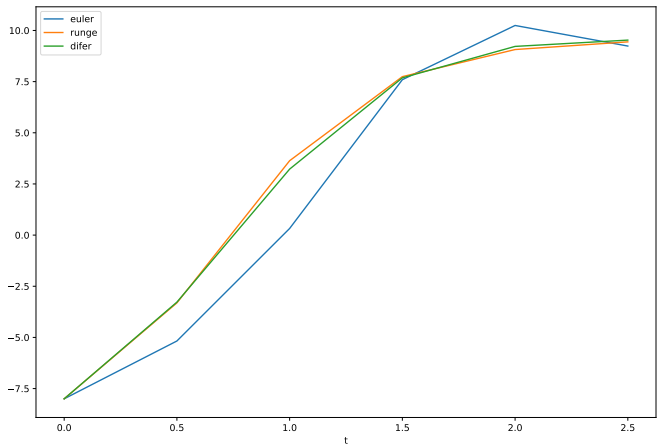


Figura 13.16: Evolução Temporal de $c(t)$ para $c(0) = 0$

13. Resolver a equação diferencial $mx'' + u.x' = F(t)$ para $m = 2$, $u = 0,2$ e passo de integração $h = 0,5$ no intervalo $t = [0; 50]$ pelo método das Diferenças Finitas. Considere que a força $F(t)$ atua de acordo com a seguinte regra de formação :

$$F(t) = \begin{cases} 0,5 & \text{se} \quad 0 \leq t < 4 \\ 1 - \frac{t}{8} & \text{se} \quad 4 \leq t < 8 \\ 0 & \text{se} \quad 8 \leq t \end{cases} \tag{13.18}$$

Solução Primeiro é feita uma transformação na equação de forma a torna-la um sistema de equações diferenciais lineares de 1^a ordem. Esta transformação é mostrada nas equações 13.19 e 13.20.

$$\begin{aligned} mx'' + u.x' &= F(x) \\ x''m &= F(t) - u.x' \\ x'' &= \frac{F(t) - u.x'}{m} \end{aligned} \tag{13.19}$$

$$\begin{cases} z' = \frac{F(t) - u.z}{m} \\ x' = z \end{cases} \tag{13.20}$$

Em seguida o sistema com duas equações é resolvido a seguir e o resultado pode ser visto na 13.17

```
import numpy as np
import pandas as pd
u = 0.2; m = 2
```

```
def F(t):
  if (0≤t) and (t<4):
    return(0.5)
  elif (4≤t) and (t<8):
    return(1-t/8)
  else:
    return(0)

def xL(x,z,t):
  return(z)

def zL(x,z,t):
  return((F(t)-u*z)/m)

def euler(x0,z0,t0,dt):
  x1 = x0 + xL(x0,z0,t0)*dt
  z1 = z0 + zL(x0,z0,t0)*dt
  return(x1,z1)

def runge(x0,z0,t0,dt):
  k1 = xL(x0,z0,t0)
  l1 = zL(x0,z0,t0)

  k2 = xL(x0+k1*dt/2, z0+l1*dt/2, t0+dt/2)
  l2 = zL(x0+k1*dt/2, z0+l1*dt/2, t0+dt/2)

  k3 = xL(x0+k2*dt/2, z0+l2*dt/2, t0+dt/2)
  l3 = zL(x0+k2*dt/2, z0+l2*dt/2, t0+dt/2)

  k4 = xL(x0+k3*dt, z0+l3*dt, t0+dt)
  l4 = zL(x0+k3*dt, z0+l3*dt, t0+dt)

  x1 = x0 + dt/6*(k1+2*k2+2*k3+k4)
  z1 = z0 + dt/6*(l1+2*l2+2*l3+l4)
  return(x1,z1)

def difin(x0,z0,t0,dt):
  xL0 = xL(x0,z0,t0)
  zL0 = zL(x0,z0,t0)

  x1 = x0 + xL0*dt
  z1 = z0 + zL0*dt
  t1 = t0 + dt

  for i in range(100):
```

```
    xL1 = xL(x1,z1,t1)
    zL1 = zL(x1,z1,t1)

    xL1 = (xL1 + xL0)/2
    zL1 = (zL1 + zL0)/2

    x1 = x0 + xL1*dt
    z1 = z0 + zL1*dt

  return(x1,z1)

def familia(ti, tf, n, x0, z0):
  ts = np.linspace(ti,tf,n)
  sol = np.array([0,x0,z0,x0,\
                  z0,x0,z0]).reshape(1,-1)

  for t0 in ts:
    t0 = sol[-1,0]
    t1 = t0 + dt

    x0 = sol[-1,1]; z0 = sol[-1,2]
    x1_euler, z1_euler = euler(x0,z0,t0,dt)

    x0 = sol[-1,3]; z0 = sol[-1,4]
    x1_runge, z1_runge = runge(x0,z0,t0,dt)

    x0 = sol[-1,5]; z0 = sol[-1,6]
    x1_difin, z1_difin = difin(x0,z0,t0,dt)

    sol0 = np.array([t1,\
                     x1_euler,\
                     z1_euler,\
                     x1_runge,\
                     z1_runge,\
                     x1_difin,\
                     z1_difin]).reshape(1,-1)
    sol = np.concatenate((sol,sol0),axis=0)

  df = pd.DataFrame(data = sol,
                    columns = ['t','x_euler',\
                    'z_euler','x_runge',\
                    'z_runge','x_difin','z_difin'])
  return(df)

dt = 0.5; ti = 0; tf = 50; x0 = 0; z0 = 0
n = int((tf - ti) / dt + 1)
```

```
df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

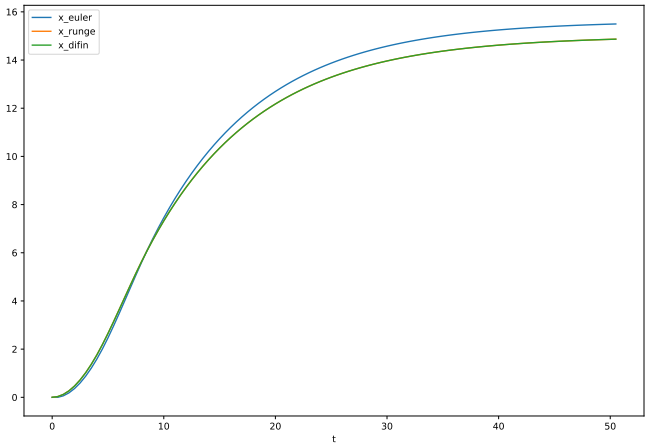


Figura 13.17: Resolução da Equação Diferencial com Forças Distintas por Intervalo de Tempo

14. Resolver a equação diferencial não linear do pêndulo $\theta''(t) = -g/l * sin(\theta)$ para $\theta(0) = 10^o$ e $\theta'(0) = 0$ pelos métodos de Euler, Runge-Kutta 4a ordem, XNumbers com Runge-Kutta 4a ordem e Diferenças Finitas. Comparar os resultados obtidos com a solução analítica da equação linearizada $\theta''(t) = -g/l*\theta$ a qual é válida para pequenos valores de θ. Considerar $g = 9,8m/s^2$ e $l = 2m$.

Solução Vamos reutilizar a maior parte do código anterior. Apenas precisamos alterar as definições de x e z. No gráfico abaixo é mostrada a resolução pelos três métodos. No caso do método de Euler, Como pode ser visto pelo gráfico, mesmo com um passo de apenas $0,005$ dois ciclos são suficientes para provocar o aparecimento de uma divergência. Os erros neste caso são provocados por um acúmulo de erros de arredondamento (devido ao montante de cálculos) e de truncamento (devido ao método em sí).

```
import numpy as np
import pandas as pd
g = 9.8; L = 2
```

```
#Theta será x
def xL(x,z,t):
  return(z)

#Theta' será z
def zL(x,z,t):
  return(-g/L*np.sin(x))

x0_graus = 10; x0 = x0_graus * np.pi / 180
z0_graus_s = 0;  z0 = z0_graus_s * np.pi / 180

dt = 0.2; ti = 0; tf = 5
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

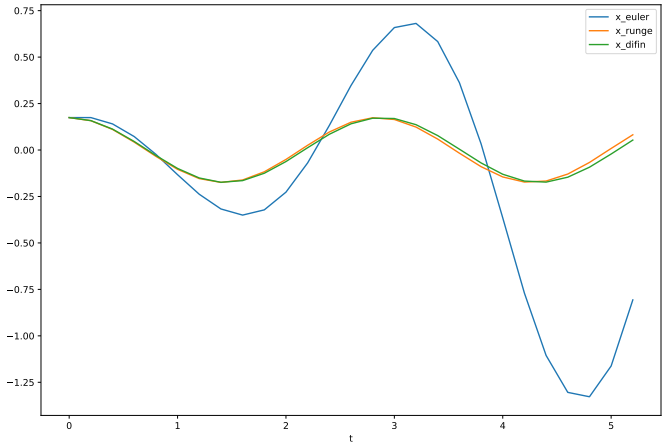


Figura 13.18: Solução da Equação Diferencial Não Linear do Pêndulo

15. Resolver o sistema de equações diferenciais não linear :

$$\begin{cases} x' = 1,2\,x - 0,6\,x\,y \\ y' = -0,8\,y + 0,3\,x\,y \\ x(0) = 2; y(0) = 1 \end{cases} \tag{13.21}$$

também conhecido por Modelo Predador-Presa, pois pode ser usado para

descrever o equilíbrio ecológico entre uma espécie de predador (Y) e uma população de presas (X). Faça além do gráfico da evolução temporal de X e de Y, o gráfico de fase X-Y. Resolva o sistema pelo método de Runge-Kutta de 4a ordem, via XNumbers.

Solução O modelo descreve uma população de presas X a qual tem uma taxa de crescimento associada (no caso 1,2) e uma taxa de mortalidade decorrente dos encontros predador-presa (no caso 0,6), descritos pelo fator $x.y$ na primeira equação. A população de predadores por sua vez, tende a diminuir com seu próprio aumento populacional (por causa da competição entre os próprios predadores, fator -0,8 na segunda equação) e a crescer devido aos encontros predador-presa, também descrito na segunda equação pelo fator $x.y$ e pelo fator 0,3.

A solução é apresentada nos gráficos a seguir. No primeiro gráfico é apresentada a evolução temporal apenas da solução obtida através do método de runge kutta. Perceba a nítida oscilação de populações a medida que as quantidades relativas de predadores e presa variam ao longo do tempo. Na figura seguinte é apresentado o diagrama de fase para a solução via runge kutta (linha contínua) e pelo método de Euler (linha tracejada). Observe que o acúmulo de erros provocado pelo método de Euler faz com que o gráfico não "feche'', gerando oscilações crescentes.

```
import numpy as np
import pandas as pd

#Theta será x
def xL(x,z,t):
  return(1.2*x-0.6*x*z)

#Theta' será z
def zL(x,z,t):
  return(-0.8*z+0.3*x*z)

x0=2; z0=1

dt = 0.02; ti = 0; tf = 10
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
graf = df.plot(x='t', y=['x_euler',\
                         'x_runge','x_difin'])
plt.show()
```

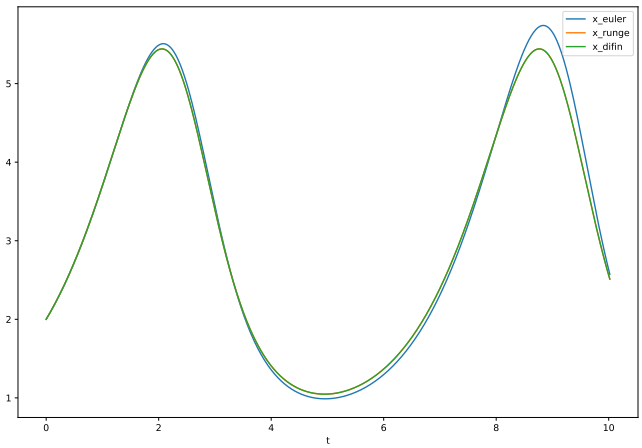


Figura 13.19: Solução do Modelo de Lotka-Voltera (Predador-Presa)

```
graf1 = df.plot(x='x_euler', y='z_euler')
graf2 = df.plot(x='x_runge', y='z_runge')
graf3 = df.plot(x='x_difin', y='z_difin')
plt.show()
```

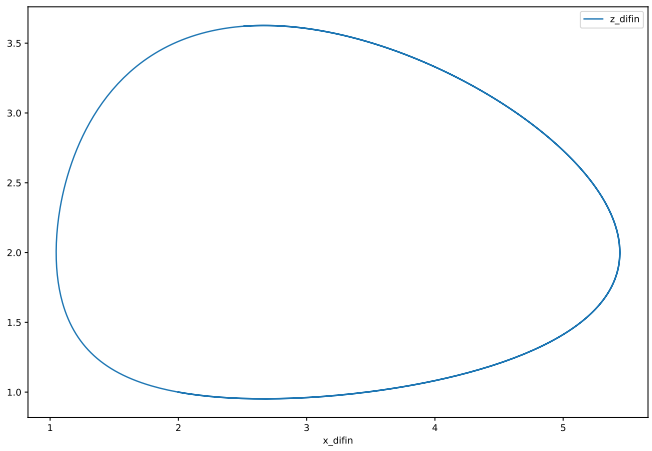


Figura 13.20: Gráfico de fase das variáveis dependentes, x e y, Modelo Predador-Presa, resoluções pelo Método de Euler, linha tracejada e por runge kutta 4a ordem, linha contínua

16. O sistema de equações apresentado em 13.22 é conhecido pelo nome de Sistema de Equações de Van der Walls e suas equações são classificados como rígidas. Resolva o mesmo para $t = [0; 100]$.

$$\begin{cases} y_1' = y_2 \\ y_2' = u\,(1 - y_1^2)\,y_2 - y_1 \\ y_1(0) = y_2(0) = 1 \end{cases} \tag{13.22}$$

Solução As características intrínsecas destas equações tornam elas de difícil solução numérica, requerendo um método especial chamado de *Piecewise-Integration*. Abaixo é apresentado o gráfico da evolução temporal da variável y_1, equações de Van de Walls. O leitor pode tentar resolver este sistema pelo método de Runge-Kutta 4a ordem e verificar que o mesmo falha totalmente. Isto ocorre porque sendo a inclinação da função quase vertical, isto provoca erro no sistema de aproximações da inclinação da função solução ao utilizarmos o método de runge kutta.

```
#A SER IMPLEMENTADO
```

13.4 Comparação entre os Métodos de Resolução

Nesta seção iremos comparar a eficiência dos métodos numéricos em planilha, aplicando os mesmos a resolução a longo prazo em um sistema cuja saída apresentará uma função oscilante. Esta combinação irá levar ao extremo a capacidade de resolução e a precisão dos métodos em planilha, pois além de buscarmos uma solução para um longo período de tempo, ela irá oscilar, o que fará com que a derivada assuma valores positivos e negativos.

17. Calcular a corrente $i(t)$ para um circuito RLC série, com tensão de entrada $E(t) = E_0 \sin \omega\, t$ para um período de tempo t de 0 a 100 segundos. O passo de integração será $h = 0,1$ segundos. Deseja-se comparar a solução exata com as soluções obtidas com os métodos de : Euler, Diferenças Finitas, Runge-Kutta 4a ordem e por um Sistema de EDOs de 1a ordem, ambos os últimos dois métodos pelas funções implementadas pelo pacote XNumbers.

Resolução Teórica

Aplicando a Lei de Kirchoff das malhas no circuito RLC série, teremos :

$$L\,\frac{di}{dt} + R\,i + \frac{q}{C} - E(t) = 0 \tag{13.23}$$

Isolando $E(t)$:

$$L\,\frac{di}{dt} + R\,i + \frac{q}{C} = E(t) \tag{13.24}$$

Trocando a notação de derivada :

$$L.i' + R.i + q/C = E(t) \tag{13.25}$$

Isolando $L\,i'$:

$$L.i' = -R.i - q/C + E(t) \tag{13.26}$$

Passando L para o outro lado dividindo :

$$i' = -\frac{R}{L}\,i - \frac{1}{C\,L}\,q + \frac{E(t)}{L} \tag{13.27}$$

Sistema de EDOs de 1^a ordem em i' e q' :

$$\begin{aligned} i' &= -\frac{R}{L}\, i - \frac{1}{C\,L}\, q + \frac{E(t)}{L} \\ q' &= i \end{aligned} \tag{13.28}$$

Supondo $E(t) = E_0 \sin \omega\, t$ e passando para a forma matricial teremos :

$$\begin{bmatrix} i' \\ q' \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{C\,L} \\ 1 & 0 \end{bmatrix} . \begin{bmatrix} i \\ q \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} E(t) \tag{13.29}$$

Resolvendo de maneira literal para a variável q, com $R = 0\,\Omega$, $L = 1\,H$, $C = 0,25\,F$, $E_0 = 1\,V$ e $\omega = \sqrt{3,5}$ teremos (a resolução não é apresentada neste texto):

$$q = -\frac{k\,\omega}{p} \sin p\,t + k \sin \omega\, t \tag{13.30}$$

onde

$$p = \frac{1}{\sqrt{L\,C}} \tag{13.31}$$

e

$$k = \frac{E_0}{L\,(p^2 - \omega^2)} \tag{13.32}$$

Código para Solução

A seguir é apresentado o código para resolução numérica e cálculo dos valores da solução exata. A maior parte do código (embutido na função *familia*) é reutilizada.

```
import numpy as np
import pandas as pd

R = 0; L=1; C=0.25; E0=1; w = np.sqrt(3.5)
p = 1/np.sqrt(L*C); k = E0/(L*(p**2 - w**2))

def x_exata(t):
  return(-k*w/p*np.sin(p*t)+k*np.sin(w*t))

#q será x; xL = q'
def xL(x,z,t):
  return(z)

#i será z, pois i = q'
def zL(x,z,t):
```

```
  E = E0 * np.sin(w*t)
  iLinha = -R/L*z-1/(C*L)*x+E/L
  return(iLinha)

x0=0; z0=0

dt = 0.02; ti = 0; tf = 400
n = int((tf - ti) / dt + 1)

df = familia(ti, tf, n, x0, z0)
df['x_exata'] = [x_exata(t) for t in df['t']]
```

Método de Euler

A figura a seguir mostra o gráfico comparando a solução exata com a solução obtida pelo método de Euler. Como pode ser visto, o acúmulo de erros faz com que a solução numérica divirja da solução exata já no primeiro ciclo de oscilação rápida, tornando o método pouco útil para a análise da solução após 5 segundos.

```
t_fim = 40
graf = df[df['t']<t_fim].plot(x='t', \
                              y=['x_euler',\
                                 'x_exata'])
plt.show()
```

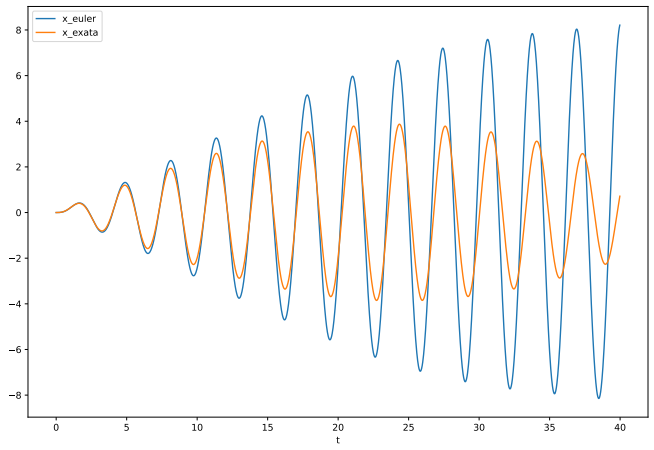


Figura 13.21: Solução do Circuito Oscilante pelo Método de Euler

Métodos de Runge-Kutta e Diferenças Finitas

Na 13.22 é apresentado o gráfico comparando a solução exata com a solução numérica via runge kutta 4a ordem e diferenças finitas. Como pode ser visto, ambos os métodos proporcionam uma grande estabilidade na solução. Deve-se considerar que o método de Runge Kutta atinge precisão similar ao das diferenças finitas com um número muito menor de operações.

```
graf = df[df['t']<100].plot(x='t', \
                            y=['x_runge',\
                               'x_difin',\
                               'x_exata'])
plt.show()
```

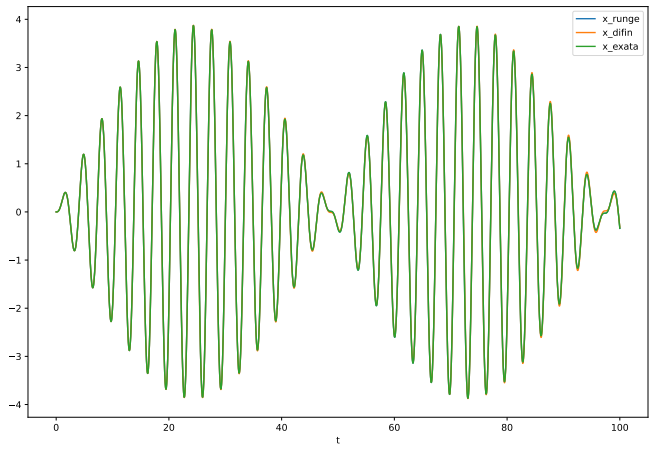


Figura 13.22: Solução do Circuito Oscilante pelos Métodos de Runge-Kutta 4a Ordem e das Diferenças Finitas

13.5 Exercícios

1. Seja o seguinte conjunto de equações: $Y'(t) = c.X(t).a.Y(t) - d.X(t)eX'(t) = -a.Y.X + b.Y$ com as seguintes condições iniciais X(0)=0, Y(0)=100.000. Sabendo $quec = 0,5\%; d = 1,0\%; a = 0,003\%; b = 0,5\%$ aça os gráficos de $X(t)eY(t)$ e o gráfico de Y com X no eixo horizontal. Dica: Este último gráfico deve gerar uma espiral no primeiro quadrante. Utilize $h = 1$ e t variando $de0a3000$.

2. A população de uma cidade t anos após 1990 satisfaz a equação diferencial y'= 0.02y. Qual é a constante de crescimento? Com que rapidez a população crescerá quando atingir 3 milhões de pessoas? Em qual nível populacional a população estará crescendo a uma taxa de 100.000 pessoas por ano?

3. Resolva numericamente (pelos métodos de Euler e Runge-Kutta de 4ª ordem) as seguintes equações diferenciais:

   (a)
   $$y^{+}2y = 1; y\,(0) = 1$$

   (b)
   $$y^{+}\frac{y}{1+t} = 20; y\,(0) = 10$$

4. Resolva pelo método algébrico de separação de variáveis, pelo método numérico de Euler e pelo método de Runge-Kutta 4ªordem, a equação diferencial ordinária $y\,(x) = 0,4y\,(x)$ com $y\,(0) = 10$, para o intervalo $x = [0; 10]$ com $Dt = 0,1$. Compare o erro percentual entre os métodos numéricos e a solução exata.

5. Suponha que o preço de uma commodity se modifique de forma que sua taxa de variação seja proporcional a escassez do produto dada por $\frac{dp}{dt} = k\,(D - S)$, onde D e S são as funções de demanda e oferta do produto dadas em função do preço por *D=7-p* e *S=p+1.* Se o preço é \$6 quando t=0 e \$4 quando t=4 encontre p(t). Mostre a medida que o tempo aumenta o preço se aproxima do valor para o qual a Demanda e a Oferta ficam iguais.

6. Resolver numericamente as equações diferenciais abaixo implementando os métodos de Euler, Diferenças Finitas e Runge-Kutta de 4a ordem , desenhando o gráfico da função solução. Utilize como passo de integração $h = 0,1$.

   (a) $y' = y\,x^2 - 1,1\,y$
   $y(0) = 1$

   (b) $y' = z$ e $z' = -4\,z - 3\,y$
   $y(0) = 3$ e $z(0) = 4$

   (c) $y' = z$ e $z' = -4\,z - 3\,y + 3\,exp(-2\,t)$
   $y(0) = 1$ e $z(0) = -1$

   (d) $y' = z$ e $z' = -6\,z - 9\,y + 0$
   $y(0) = -1$ e $z(0) = 1$

   (e) $y' = z$ e $z' = -5\,z - 6\,y + 3\,exp(-2\,t)$
   $y(0) = 1$ e $z(0) = -1$

7. Desenhar a família de soluções das equações diferenciais: $y' = y - x^2\,y^2$, solução exata: $y(x) = \frac{1}{x^2 - 2\,x + 2 + \left(\frac{1}{y_0} - 2\right)e^{-x}}$ desenhando a família de soluções tanto da solução exata quanto da numérica (Runge-Kutta 4a ordem). Comparar a diferença percentual entre o valor numérico aproximado obtido com o método de Runge-Kutta de 4a ordem e o valor exato.

Capítulo 14

Análise de Fourier

14.1 Séries de Fourier

A série de Fourier permite decompor um sinal no tempo, periódico, com período igual a T em uma soma ponderada de senos e cossenos. A série é representada pela relação :

$$f(t) = a_0 + \sum_{k=0}^{\infty} a_k \, cos\left(\frac{2\,\pi}{T}\,k\,t\right) + b_k \, sin\left(\frac{2\,\pi}{T}\,k\,t\right) \tag{14.1}$$

onde os coeficientes a_k e b_k são calculados pelas seguintes fórmulas :

$$a_0 = \int_{-\frac{T}{2}}^{+\frac{T}{2}} f\,(t)\,dt \tag{14.2}$$

$$a_k = \int_{-\frac{T}{2}}^{\frac{T}{2}} f\,(t)\; cos\left(\frac{2\,\pi}{T}\,k\,t\right) dt \tag{14.3}$$

$$b_k = \int_{-\frac{T}{2}}^{\frac{T}{2}} f\,(t)\; sin\left(\frac{2\,\pi}{T}\,k\,t\right) dt \tag{14.4}$$

Caso a função $f(t)$ seja dada como um conjunto de pontos (e não através de uma expressão algébrica) os coeficientes a_k e b_k poderão ser calculados através do método dos mínimos quadrados, uma vez que a expressão 14.1 é uma expressão linear para os coeficientes a_k e b_k. Para o cálculo através da linguagem Python, tanto pode-se utilizar a função `fft` da `numpy`, integração numérica ou regressão linear pela fórmula matricial $\beta = (X^t.X)^{-1}.(X^t.Y)$, como será mostrado nos exemplos a seguir.

1. Calcular os coeficientes a_k e b_k da série de Fourier que representa a função $f(t) = t$ para $t = [-1, 1]$.

Esta função representa um segmento de reta que se repete indefinidamente no tempo. Para calcular os coeficientes, determinamos primeiro que T é igual a $1 - (-1) = 2$ e em seguida fazemos :

$$a_0 = \int_{-1}^{+1} t dt = \left[\frac{t^2}{2}\right]_{-1}^{+1} = 0 \tag{14.5}$$

$$a_k = \int_{-1}^{+1} t\, cos\,(\pi\, k\, t)\, dt = \left[\frac{\pi\, k\, t\, sin\,(\pi\, k\, t) + cos\,(\pi\, k\, t)}{\pi^2\, k^2}\right]_{-1}^{+1} = 0 \tag{14.6}$$

$$b_k = \int_{-1}^{+1} t\, sin\,(\pi\, k\, t)\, dt = \left[\frac{sin\,(\pi\, k\, t) - \pi\, k\, t\, cos\,(\pi\, k\, t)}{\pi^2\, k^2}\right]_{-1}^{+1} = \frac{2\ (sin\,(\pi\, k) - \pi\, k\, cos\,(\pi\, k))}{\pi^2\, k^2} \tag{14.7}$$

O cálculo dos coeficientes e a reconstituição do sinal através da `numpy` é apresentado a seguir. Nosso objetivo é reconstituir um sinal $f(t) = t$, o qual possui um período fundamental $T = 2$, com até o $n$ésimo $n = 5$ par de coeficientes em um intervalo t_i, t_f igual neste caso a $-1, +1$. Serão utilizados $n_p = 100$ pontos por default para reconstituição do sinal neste intervalo.

Uma vez definido o objetivo de nossa função (reconstituir o sinal com precisão de n pares de coeficientes da série de Fourier), nossa primeira providência é calcular os coeficientes associados. Para isso vamos supor que exista uma função $gera_cns$ a qual recebe a função $f(t)$, o número de pares de coeficientes a calcular n e o periodo fundamental T devolvendo os n pares de coeficientes a_n, b_n desejados em um array `numpy`.

Com os coeficientes (cns), o periodo fundamental (T), o intervalo de reconstituição do sinal ($[t_i, t_f]$) e o número de pontos desejados np vamos supor que exista uma função que recebe estes valores como parâmetros e retorna um data frame pandas com três colunas t, $f(t)$ e $f_n(t)$, isto é, com o valor de t, da função original $f(t)$ e da função reconstruida $f_n(t)$.

Vamos trabalhar agora no desenvolvimento da função $gera_cns$. A função `gera_cns` recebe a função $f(t)$ a gerar os coeficientes, o número n de pares de coeficientes a serem gerados e o periodo fundamental T. Em seguida ela devolve um array `numpy` com os pares de coeficientes correspondentes. Para isso ela supõe a existência de uma função $c_n(i, T)$ a qual recebe a ordem i do par de coeficientes a ser gerado, o periodo fundamental T e devolve o par correspondente. Esta função será executada n vezes e será desenvolvida na próxima listagem.

Com foi dito no parágrafo anterior, a função `c_n` recebe a função $f(t)$, o índice i e o período fundamental T e devolve uma tupla de valores com o par de valores

a_n, b_n correspondente. O processo envolve integração numérica. Por default utilizamos um passo de integração de $dt = 1e-4$ e supomos que um coeficiente seja igual a 0 se ele for menor que $1e-12$

Vamos agora trabalhar na função `gera_fn` a qual irá receber a função $f(t)$, o array `numpy` de nome `cns` com os n pares de coeficientes da série de Fourier associada, o período fundamental T, o intervalo de integração t_i, t_f e o número de pontos n_p desejado na reconstrução do sinal e irá devolver um data frame com três colunas t, $f(t)$ e $f_n(t)$.

Esta função supõe a existência de outra função `f_n` a qual recebe a função $f(t)$, o array `numpy` com os pares de valores associados dos coeficientes *cns*, o período fundamental T e um valor de t e calcula o valor do sinal reconstruido neste ponto.

A seguir testamos esta sequência de funções para o sinal $f(t) = t$, com $n = 5$, $T = 2$, no intervalo $t_i = -1$ até $t_i = +1$ e o intervalo dividido em 100 pontos. (Vide 14.1)

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

f = lambda t:t
n = 5; T = 2; ti = -1; tf = 1; n_p = 100

def c_n(f, i, T, dt=1e-4, zero=1e-12):
  soma_a = 0; soma_b = 0
  ti = -T/2; tf = +T/2; t = ti
  while t ≤ tf:
    soma_a = soma_a + f(t)*np.cos(2*np.pi*i*t/T)*dt
    soma_b = soma_b + f(t)*np.sin(2*np.pi*i*t/T)*dt
    t = t + dt
  if abs(soma_a)<zero:
    soma_a = 0
  if abs(soma_b)<zero:
    soma_b = 0
  return(soma_a, soma_b)

def gera_cns(f, n, T):
  cns = []
  for i in range(n):
    par = c_n(f, i, T)
    cns.append(par)
  cns = np.array(cns)
  return(cns)

def f_n_t(cns, T, t):
```

```
  soma = 0
  for k, cn in enumerate(cns):
    soma = soma + cn[0]*np.cos(2*np.pi*k*t/T) + \
                  cn[1]*np.sin(2*np.pi*k*t/T)
  return(soma)

def gera_fn(cns, T, ti, tf, n_p):
  t = [ti]
  f_fourier = [f_n_t(cns, T, t[-1])]
  h = (tf-ti)/n_p
  while t[-1] <= tf:
    t.append(t[-1] + h)
    f_fourier.append(f_n_t(cns, T, t[-1]))
  dados = pd.DataFrame({'t':t, \
                        'f_fourier':f_fourier})
  return(dados)

def f_n(f, n, T, ti, tf, n_p):
  cns = gera_cns(f, n, T)
  dados = gera_fn(cns, T, ti, tf, n_p)
  return(dados)

dados = f_n(f, n, T, ti, tf, n_p)
dados['f_exata'] = list(map(f, dados['t'].values))
g = dados.plot(x='t', y=['f_exata', 'f_fourier'])
plt.show()
```

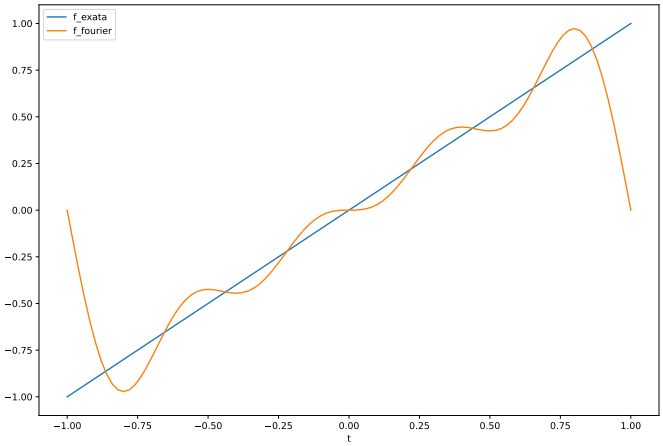


Figura 14.1: Reconstituição do sinal $f(t) = t$ com n=5, no intervalo $(-1, 1)$

Na figura seguinte (14.2) é apresentado o cálculo por meio do método dos mínimos quadrados. Primeiro são geradas as bases para a regressão (as colunas com o termo constante, igual a 1, e as 5 colunas para $cos(\frac{2\pi nt}{T})$ e $sin(\frac{2\pi nt}{T})$. Isto constitui a matrix $\mathbf{X}$. A matriz $\mathbf{Y}$ é gerada a partir da função $f(t)$ para os n_p pontos do intervalo entre $[-1, 1]$. Em seguida é aplicada a fórmula matricial $\beta = (X^tX)^{-1}(X^tY)$. A partir dos valores *beta* nesta fórmula calculamos os valores da função reconstruida facilmente através de $X.b$.

```
import numpy as np
import pandas as pd

ti = -1
tf = +1
n_p = 100
f = lambda t:t
T = 2

dados = pd.DataFrame({'t':np.linspace(ti,tf,n_p)})
dados['f_exata']=list(map(f,dados['t'].values))
dados['a0'] = 1

for k in range(1,6):
  dados['a'+str(k)] = \
       np.cos(2*np.pi*k*dados['t'].values/T)
  dados['b'+str(k)] = \
       np.sin(2*np.pi*k*dados['t'].values/T)
```

```
X = dados.loc[:,'a0':].values; X.shape
```

```
(100, 11)
```

```
Y = dados.loc[:,['f_exata']].values; Y.shape
```

```
(100, 1)
```

```
b = np.linalg.inv(X.T.dot(X)).dot(X.T.dot(Y));
dados['f_fourier'] = X.dot(b)
g = dados.plot(x='t',y=['f_exata', 'f_fourier'])
plt.show()
```

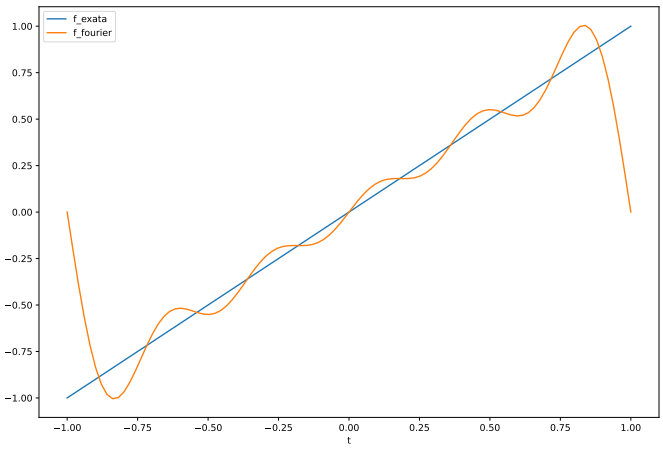


Figura 14.2: Cálculo dos Coeficientes da Série de Fourier pelo Método dos Mínimos Quadrados

14.2 Gráficos de Resposta em Frequência

Uma das técnicas mais utilizadas em Engenharia Elétrica é a análise da resposta em frequência. Matematicamente, trata-se do gráfico de uma função de variável complexa $f(z)$ onde o argumento da função assume valores puramente imaginários $j\omega$, gerando então a função $f(j\omega)$. Para efeito de apresentação e melhor entendimento, o gráfico é apresentado com a ordenada ω em escala logarítmica.

2. Dada a função $Z(\omega) = 10 + \frac{10^4 - j\,\frac{10^6}{\omega}}{10 + j\left(0{,}1\,\omega - \frac{10^5}{\omega}\right)}$ desenhar o gráfico da parte real, da parte imaginária e do módulo de $f(\omega)$

Solução O python suporta o uso de números complexos na sua forma nativa. A listagem com os cálculos e o gráfico correspondente (em escala linear) podem ser vistos nas figuras 14.3 e 14.4.

```
f = lambda w: 10 + \
              (1e4 - 1j*1e6/w)/(10 + 1j*(0.1*w-1e5/w))
dados = pd.DataFrame({'w':np.linspace(600,1600,1000)})
dados['f_'] = list(map(f,dados['w']))
dados['f_re'] = np.real(dados['f_'])
dados['f_im'] = np.imag(dados['f_'])
dados['f_abs'] = np.abs(dados['f_'])

g = dados.plot(x='w', y=['f_re', 'f_im'])
plt.show()
```

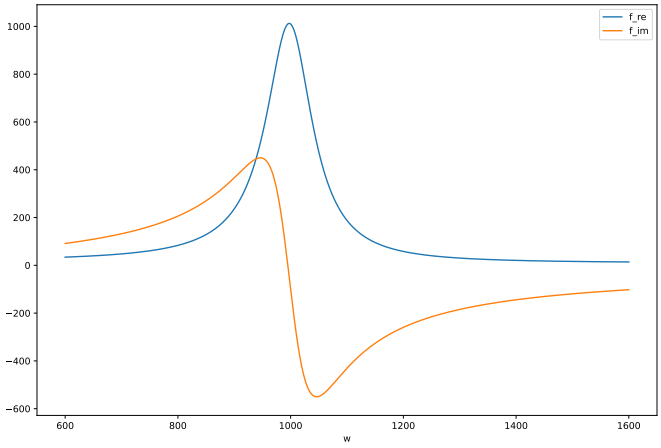


Figura 14.3: Gráfico das Partes Real e Imaginária de f(w)

```
g = dados.plot(x='w', y='f_abs')
plt.show()
```

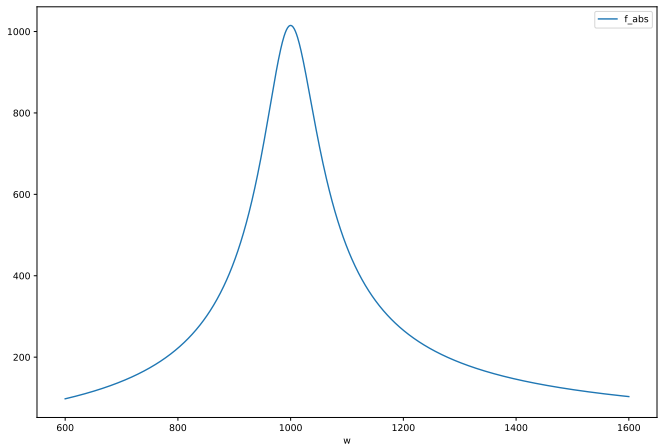


Figura 14.4: Gráfico do Módulo de f(w)

14.3 Resposta em Frequência de Um Circuito RC

Vamos agora analisar a resposta em frequência de um circuito RC em série. Deseja-se obter a relação de amplitude e fase da senoide de saída a partir da frequência da senoide de entrada. A senóide de saida é a tensão no resistor e a senóide de entrada é o próprio sinal de entrada do circuito. Para tanto aplicamos um divisor de tensão no domínio frequência para calcular a tensão V_r fazendo :

$$V_r(s) = V(s)\,\frac{R}{R + \frac{1}{s\,C}} \tag{14.8}$$

Em seguida substituindo s por $\jmath\omega$ obtemos :

$$V_r(s) = V(s)\,\frac{R}{R + \frac{1}{\jmath\omega\,C}} \tag{14.9}$$

De onde podemos obter a relação entre a tensão de saída $V_r(s)$ e a tensão de entrada $V(s)$, como :

$$T(s) = \frac{V_r(s)}{V(s)} = \frac{R}{R + \frac{1}{\jmath\omega\,C}} \tag{14.10}$$

Podemos então gerar um gráfico da função de variável complexa $T(s)$ descrita na 14.10, fazendo a mesma variar de valores muito negativos de $\jmath\omega$ até valores muito positivos de $\jmath\omega$. Vamos mostrar um gráfico obtido para os valores $R = 1,5k\Omega$ e $C = 1\mu F$, com ω variando de $0,1\,Hertz$ até $1M\,Hertz$. Como sempre utilizaremos escalas logarítmicas para o ganho ($20\log(|T(s)|)$) e para ω. A frequência de corte prevista para este circuito é dada pela expressão $f_c = \frac{1}{RC}$ o que neste caso dá $f_c = \frac{1}{10^4\,10^{-6}} = 100\,Hz$.

```
T = lambda s,R=1500,C=1e-6: R/(R + 1/(s*C))
dados = pd.DataFrame({'w':np.linspace(0.1,1e6,10000)})
dados['f_'] = list(map(T,1j*dados['w']))
dados['f_re'] = np.real(dados['f_'])
dados['f_im'] = np.imag(dados['f_'])
dados['f_abs'] = np.abs(dados['f_'])

g = dados.plot(x='w', y='f_abs', logx=True, logy=True)
plt.show()
```

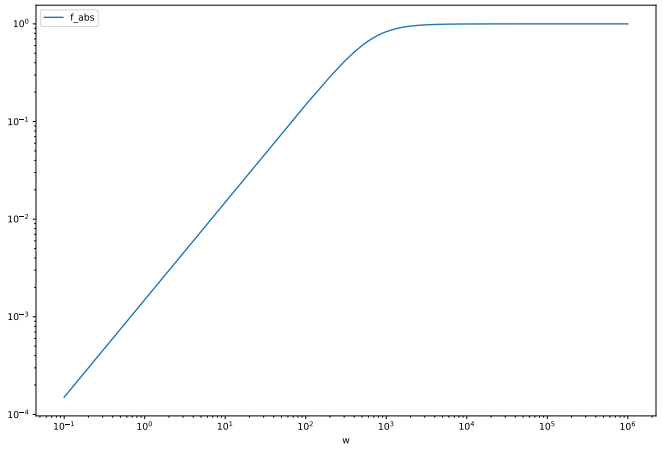


Figura 14.5: Gráfico do Módulo de T(s)

14.4 Expansão em Frações Parciais

Um problema muito comum na análise de sinais e na resolução de sistemas no domínio frequência é o da expansão de uma função racional em uma soma de frações parciais. Neste tipo de problema desejamos transformar uma expressão como :

$$\frac{s^3 + 3\,s^2 + 3}{s^5 + 2\,s^4 + 7\,\mathbf{s}^3 + 8\,s^2 + 9\,s + 4} \tag{14.11}$$

em uma série de termos do tipo :

$$\frac{A_1}{s + r_1} + \frac{A_2}{s + r_2} + \frac{A_3}{s + r_3} + \frac{A_4}{s + r_4} + \frac{A_5}{s + r_5} \tag{14.12}$$

onde os r_n são as raízes do denominador de 14.11 e os valores A_n devem ser calculados. Isto pode ser feito através da `scipy` com a função `residue` (módulo de processamento de sinais de nome `signal`) como pode ser visto a seguir:

```
from scipy.signal import residue
numer = [1,3,0,3]; denom = [1,2,7,8,9,4]
r, p, k = residue(numer, denom)
```

`r` são os resíduos em sí.

```
r
```

```
array([ 0.653+0.j   , -0.253+0.461j, -0.253-0.461j, -0.074-0.561j,
       -0.074+0.561j])
```

`p` são os pólos ordenados de forma crescente, correspondentes a cada resíduo

```
p
```

```
array([-0.625+0.j   , -0.394+1.181j, -0.394-1.181j, -0.294+2.01j ,
       -0.294-2.01j ])
```

`k` é o termo polinomial direto (caso a ordem no numerador seja maior que a do denominador). Neste caso, o termo `k` é uma lista vazia.

```
k
```

```
array([], dtype=float64)
```

Sendo assim a expansão em frações parciais deste exemplo fica igual a :

$$\begin{aligned}\frac{s^3 + 3\,s^2 + 3}{s^5 + 2\,s^4 + 7\,s^3 + 8\,s^2 + 9\,s + 4} = \frac{0,653}{s + 0,625} + \frac{-0,253 + 0,461j}{s + 0,394 - 1,181j} + ... \\ ... + \frac{-0,253 - 0,461j}{s + 0,394 + 1,181j} + \frac{-0,074 - 0,561j}{s + 0,294 - 2,010j} + \frac{-0,074 + 0,561j}{s + 0,294 + 2,010j}\end{aligned} \tag{14.13}$$

14.5 Inversão para o Domínio do Tempo

Considerando que a expansão em frações parciais já ocorreu e desejamos agora obter a função equivalente no domínio do tempo t devemos proceder com a inversão, ou anti-transformação da função em s para a função em t. Sendo assim supomos que a inversão ocorre para uma função de variável s a qual foi obtida pela Transformada de Laplace de uma função em t. Esta tarefa é feita através do par básico de Transformadas

$$f(t) = e^{at} \longleftrightarrow F(s) = \frac{1}{s-a} \tag{14.14}$$

Vamos então aplicar esta idéia básica ao caso da equação 14.13. Por inspeção, o primeiro termo da expansão ficaria :

$$\frac{0,653}{s+0,625} \longleftrightarrow 0,653e^{-0,625t} \tag{14.15}$$

Quanto aos termos com valores complexos, para tornar a inversão mais rápida podemos perceber o seguinte :

$$\frac{a+b\jmath}{s-re-im\jmath} + \frac{a-b\jmath}{s-re+im\jmath} \rightarrow (a+b\jmath).e^{re+im\jmath} + (a+b\jmath).e^{re-im\jmath} \tag{14.16}$$

Separando os termos reais (sem $\jmath$) dos termos imaginários (com $\jmath$):

$$\begin{aligned}
(a+b\jmath).e^{re+im\jmath} + (a-b\jmath).e^{re-im\jmath} = \\
2.a.e^{re}.(\frac{e^{im\jmath}+e^{-im\jmath}}{2}) + 2.b.\jmath.e^{re}.(\frac{e^{im\jmath}-e^{-im\jmath}}{2}) = \\
2.a.e^{re}.(\frac{e^{im\jmath}+e^{-im\jmath}}{2}) - 2.b.e^{re}.(\frac{e^{im\jmath}-e^{-im\jmath}}{2.\jmath}) = \\
2.a.e^{re}.\cos(im.t) - 2.b.e^{re}.\sin(im.t) = \\
2.e^{re}.[a.\cos(im.t) - b.\sin(im.t)]
\end{aligned} \tag{14.17}$$

Neste caso supusemos o resíduo $a+b\jmath$ associado com a raiz $re+im.\jmath$. Caso este resíduo esteja associado com a raiz $re-im.\jmath$, basta trocar o sinal de $-b$ para $+b$ na equação 14.17

Desta forma, os pares de transformadas serão :

$$\begin{aligned}
\frac{-0,059+0,229\jmath}{s+0,34-0,953\jmath} + \frac{-0,059-0,229\jmath}{s+0,34+0,953\jmath} \rightarrow \\
2.e^{-0,34.t}.[-0,059.\cos(0,953.t) - 0,229.\sin(0,953.t)]
\end{aligned} \tag{14.18}$$

Neste caso (equação 14.18) tivemos o sinal $-$ no denominador porque o resíduo $-0,059 + 0,229\jmath$ estava associado à raiz $-0,34 + 0,953\jmath$. No caso a seguir (equação 14.19), o resíduo $-0,144 - 0,232\jmath$ esta associado a raiz $-0,585 + 2,751\jmath$, logo o sinal do denominador será $+$. Sendo assim teremos :

$$\frac{-0,144 - 0,232\jmath}{s + 0,585 - 2,751\jmath} + \frac{-0,144 + 0,232\jmath}{s + 0,585 + 2,751\jmath} \rightarrow$$
$$2.e^{-0,585.t}.[-0,144.\cos(2,751.t) + 0,232.\sin(2,751.t)] \tag{14.19}$$

14.6 Exercícios

3. Expandir em frações parciais os seguintes quocientes polinomiais:
   (a) $\frac{2\,s^2+6\,s}{s^3+8\,s^2+10\,s+4}$
   (b) $\frac{x^2+3\,x+2}{x^3+5\,x^2+10{,}5\,x+9}$
   (c) $\frac{x^2-16}{x^3+8\,x^2+24\,x+32}$
   (d) $\frac{x+1}{x^3+6\,x^2+11\,x+6}$

Capítulo 15

Quantum Machine Learning

15.1 A Poetic Introduction

Dear coleagues, it's been a while since I started reading about Quantum Computing. In the mean time, I have also developed some courses on what is called now Machine Learning. Ten, fifteen years ago Machine Learning was called Statistics. Then, the CSVs files started becoming huge, with millions of rows and hundreds, thousands of columns. Statistics became Big Data.

Some statistics rules for selecting which column (variable) to use, like p-value became difficult to be explained to customers, specially executives, which need something that can be quickly learned, easily remembered and effortlessly applied on a daily basis.

P-value also started giving strange answers when used in these colossal tables (after all, if 5% of your data is a 100.000 rows with 100 columns, who knows what will appear in this subset of your data???).

It was at this moment that somebody "remembered" that p-values had been developed originally with way much more simpler tables, tables in which you were forced to use all the rows and columns in the development of your model. In this context p-values were a way of estimating the other data that was "out there." But with tables made of millions of rows you could just separate 500.000 of them to be the "world out there," use the 4.500.000 to develop (train) your model and afterwards see ("test") what would happen in the real world when your model was put to work. Statistics that had became Big Data was now renamed also to Machine Learning.

The algorithms were still being used to model simple phenomena. Phenomena that could be described with straight lines. Science, the scientific revolution started 500 years ago when a group of geniuses tried to apply simple rules to simple phenomena. It worked for quite a while (astonishing well), but the rule

was always: simple phenomena that can be described preferably with straight lines, at most with a parabola.

The world is incredibly complex, but nonetheless our minds are capable of dealing with all kinds of phenomena, be they describable by straight lines or not. And then somebody tried to apply a method of finding non linear patterns in the data called neural networks. Statistics, now mingled with computer science became Deep Learning.

But all this has a cost. The cost of dealing with millions of rows, thousands of columns, to train, validade and test our models. And this cost is paid in the most precious commoditty that a human being can spend: TIME! Our computers are becoming overwhelmed by this combination of Big Data, Machine Learning and Deep Learning that has accomplished a lot of things but needs an ever increasing ammount of data to be processed.

It was then that another technology appeared. A technology that is still in its infancy, but is promising us with a future when computers will be able to digest an unthinkable ammount of data in a matter of seconds, process it and tell us where the results are. This technology is called Quantum Computing.

What is Quantum Computing and how it can be used in the Machine Learning way of finding the utmost complex patterns in mountains of data is the purpose of a field now called Quantum Machine Learning.

This field blends together two of my mostly cherished areas of study: Computing and Physics (and of course, the possibility of making some money along the line, after all, bills doesn't stop arriving each month for us to have them paid...). And this group of Jupyter Notebooks is my attempt to make this area of study better know by as many people as possible.

Quantum Machine Learning. This is the Holly Grail of my knowledge dreams. Understand this dream, make it available to others, is the objective of this and other Jupyter Notebooks that I will try writing in the near future.

In this journey, Python will be our driver, and Qiskit our compass. The destiny: translating to the world of Quantum Computing the Machine Learning algorithms that have made us devise a future with Intelligent Machines at our disposal.

Hope you enjoy the ride. It will be bumpy, sometimes frustating, but if we keep our focus and always remember why we are here, I promise you that there will be a pot of gold at the end of the rainbow waiting for us.

Happy Journey! And Let Quantum Machine Learning BE!

15.2 Machine Learning

Machine learning is the process of tagging data in a data frame. The first thing that needs to be choosen is a target variable. Depending on whether you have

examples of your target variable or not, the task can fall in one of two main categories:

1. **Supervised Learning**: when you have past examples of your input and output data. Depending on whether your output variable is categorical (discrete) or continuous, the supervised learning task can be called:
   (a) **Classification**: when your output variable is discrete (categorical). Logistic regression is the main example of classification technique.
   (b) **Regression**: when your output variable is continuous. Linear regression is the main example.
2. **Unsupervised Learning**: when you don't have past examples of your output variable or not even know what it could be. In this case you want the machine to group your data so you can try finding some pattern in it. Depending on wether you want to group the rows or the columns of you data frame, this task can be called:
   (a) **Clustering**: when you want the machine to tag (group) the rows of your data frame. This tagging can become in the future your output variable to be used in a supervised learning classification problem.
   (b) **PCA**: when you want the machine to tag (group) the columns of your data frame. In this case you want a new column that will replace some of the original ones through a simple formula.

The process of solving a problem using Machine Learning usually involves five steps:

1. **Problem Definition**: decide which variable will be predicted. This will be called your output variable.
2. **Data Preparation**: decide which variables you need to try predicting your output variable. These variables will be called input variables. In this step you will also find examples of your variables and have them cleaned and ready to be used.
3. **Model Development**: Based on your output variable type availability you will decide if you're going to develop a classification, regression, clustering or PCA model.
4. **Results Analysis**: When you link the math results from trainning and testing the model developed in stage 3 with your business objectives and present your results for both a technical and a business oriented audience.
5. **Model Deployment**: When your model goes live. Here usually you deal with problems of interfacing your model with real people (managerial issues) or with other systems in your client (a task usually called data engineering).

15.3 Quantum Computing

The idea behind quantum computing is based in three concepts:

1. **Superposition of states**: the Qubit is neither in the $|0\rangle$ or $|1\rangle$ state but in a mixed state called superposition of states. Bare in mind that the notion that it is at the same time in the $|0\rangle$ and $|1\rangle$ state is wrong. The system is in a state that is formed from the basis states $|0\rangle$ and $|1\rangle$.

2. **Interference**: is the possibility of making states interact with one another (or with themselves) in such a way that the desired pattern (the one that holds the answer for our problem) is reinforced while the other patterns are cancelled out.

3. **Entanglement**: an extremely strong correlation between states that cannot happen in classical physics. Because of it, two particles when they interact with one another remain entangled ("amarradas"in portuguese) after that. This entangled state means that their states are grouped in such a way that they cannot be represented individually.

Os conceitos apresentados acima são aprofundados a seguir:

1. **Notação de Dirac**: através dela podemos representar a sobreposição de n estados e amplitudes em um sistema ψ conforme segue: $\psi = c_0|x_0\rangle + c_1|x_1\rangle + ... + \ c_{n-1}|x_{n-1}\rangle$

2. **Colapso da Função de Onda**: só podemos saber "o que se passa"com uma partícula subatômica (um fóton de luz por exemplo) se tentarmos medir o seu estado. Neste momento a partícula irá nos "responder"com qualquer dos possíveis estados x_i acima. Neste momento a partícula "interrompe"sua sobreposição e nos mostra uma fotografia momentânea de si mesma. Damos o nome a este fenômeno de colapso da função de onda associada à partícula.

3. **Amplitudes c_0 ... c_{n-1} e probabilidades de medição dos estados x_0 ... x_{n-1} de uma partícula**: Cada estado passível de ser medido x_0 ... x_{n-1} tem uma amplitude c_i associada ao mesmo. Esta amplitude pode ser um número complexo qualquer, com qualquer valor (negativo, nulo ou positivo) tanto na sua parte real quanto na parte imaginária.

A probabilidade de obtermos como resposta a uma medição do estado de uma partícula o valor x_i está associada com o produto $c_i.c_i^*$ associado a amplitude c_i do estado x_i.

Uma vez que o valor numérico c_i pode ser um número complexo qualquer, devemos lembrar que c_i^* é o seu complexo conjugado (o número obtido trocando a parte imaginária do mesmo de sinal). Por exemplo se $c_i = 3 + 4i$, seu complexo conjugado c_i^* será igual a $c_i^* = 3 - 4i$.

Porém observe que o produto $c_i.c_i^*$ sempre será um número real. Se supusermos que a soma de todos os produtos $c_i.c_i^*$ para i indo de 0 a $n-1$ seja igual a 1,

podemos dizer que cada produto $c_i.c_i^*$ representará a probabilidade de obtermos o estado x_i associado em uma medida qualquer.

Toda a "estranheza" da Mecânica Quântica decorre da possibilidade destas n amplitudes c_i serem constituidas de qualquer número possível. Podendo misturar valores positivos com negativos podemos obter como resultado para uma combinação de c_is distintos até mesmo o valor 0. Isto significa que o estado x_i associado à amplitude c_i terá probabilidade 0% de ser medido, o que significa que não vamos medi-lo. Dizemos portanto que aquele resultado "desapareceu" de nossas medidas, em outras palavras, naquela situação a partícula "sumiu." Dado que tal soma com resultado nulo só pode aparecer decorrente da interação de outros estados, dizemos que ocorreu um fenômeno de **interferência** da partícula. Porém somos levados a concluir que, dados os estados da partícula interagirem entre sí (enquanto a mesma está em sobreposição), a partícula está interindo **consigo própria**.

Do ponto de vista filosófico, isto pode gerar (alias tem gerado a mais de cem anos) uma enorme confusão, com grandes nomes da ciência tentando dar voltas de pensamento e executando verdadeiros malabarismos mentais para encontrar formas de explicar tais fenômenos. De nossa parte, o que nos interessa é que a sobreposição e suas consequências podem ser facilmente analisadas através da maquinaria dos números complexos e da álgebra linear. Seguiremos aqui a linha *filosófica* muito bem descrita por uma frase pronunciada pelo físico David Mermin a um de seus orientandos quando o mesmo começou a gerar as famosas perguntas sem resposta que todo professor de Mecânica Quântica já se acostumou em responder com um sorriso enigmático. A frase simplesmente foi (em inglês no original): *Shut Up and Calculate*.

Pelo estilo da mesma, ela foi atribuida ao físico ganhador do Nobel, Richard Feynman. Ele não pronunciou a mesma, mas parafraseando as palavras de um grande humorista brasileiro Dias Gomes, criador do inesquecível prefeito de Sucupira Odorico Paraguassu: "Se não disse, deveria ter dito."

A partir deste momento, nossas partículas elementares passarão a se constituir de entes abstratos chamados de **Qubits**. Tais elementos terão a capacidade de entrar em sobreposição de seus estados fundamentais x_i e serão retirados dela quando forem medidos, momento no qual nos apresentarão como resultado um dos x_i possíveis. A probabilidade de medirmos o resultado x_i será dada pelo produto $c_i.c_i^*$. Estaremos interessados apenas com as situações nas quais a soma de todos os produtos $c_i.c_i^*$ dará 1 (100%).

4. **NISQ: Noisy Intermediate Scale Quantum Computing**: Quantum computing today is based in a technique called **NISQ**. This technology uses both classical and quantum computers. Classical computers handle the machine learning problem the same way they usually do. The task of "crunching"the numbers (get the parameters for the machine learning model and having it tested) is handled by the quantum counterpart. The algorithms that handle the problem this way are called **Variational

15.4 Estados Quânticos

A representação da sobreposição das amplitudes e dos estados de um sistema ψ com n estados possíveis é apresentada a seguir:

$$\psi = c_0|x_0\rangle + + + c_1|x_1\rangle \ + \ +... + \ + \ + c_{n-1}|x_{n-1}\rangle$$

1. Calcule a probabilidade dos sistemas abaixo, ao serem medidos, indicarem o estado $|1\rangle$

```
import numpy as np
p_e = lambda vetor,x : (np.linalg.norm(vetor[x])/np.linalg.norm(vetor))**2

psi = [1/np.sqrt(3), np.sqrt(2/3)]
phi = [1j/2, np.sqrt(3)/2]
chi = [(1+1j)/np.sqrt(3), -1j/np.sqrt(3)]

[float(x.__format__('.3f')) for x in [p_e(psi,1), p_e(phi,1), p_e(chi,1)]]
```

```
[0.667, 0.75, 0.333]
```

2. Qual a probabilidade do sistema Phi estar no 3^o estado?

```
import numpy as np
import sympy as sp
Phi = sp.Matrix([-3 - sp.I, -2*sp.I, sp.I, 2])
num1 = Phi[2] * Phi[2].conjugate()
den1 = sp.sqrt(sum(sp.matrix_multiply_elementwise(Phi,Phi.conjugate())))
P = sp.simplify((num1/den1)**2)
P
```

3. Dadas as amplitudes $c_{up} = 3 - 4j$ e $c_{down} = 7 + 2j$ aplicadas à uma base ortonormal, calcule as probabilidades associadas a cada uma das amplitudes.

```
import numpy as np
c_up = 3-4j
c_down = 7+2j
ket = np.array([c_up,c_down]).reshape(-1,1)
p_up = np.linalg.norm(c_up)**2/np.linalg.norm(ket)**2
p_down = np.linalg.norm(c_down)**2/np.linalg.norm(ket)**2
[float(x) for x in [p_up.__format__('.3f'), p_down.__format__('.3f')]]
```

```
[0.321, 0.679]
```

4. Apresente a representação *ket* do vetor ψ com valores $2-i, 2i, 1-i, 1, -2i, 2$

```
import numpy as np
import sympy as sp
psi = sp.Matrix([2 - sp.I, 2*sp.I, 1 - sp.I, 1, -2*sp.I, 2])
expr = "| \\psi \\rangle =" + sp.latex(psi)
expr
```

```
'| \\psi \\rangle =\\left[\\begin{matrix}2 - i\\\\2 i\\\\1 - i\\\\1\\\\- 2 i
```

5. Apresente a representação *bra* do vetor ψ com valores $2-i, 2i, 1-i, 1, -2i, 2$

```
psiT = sp.conjugate(sp.Matrix.transpose(psi))
expr = "\\langle \\psi | =" + sp.latex(psiT)
expr
```

```
'\\langle \\psi | =\\left[\\begin{matrix}2 + i & - 2 i & 1 + i & 1 & 2 i & 2
```

6. Apresente a representação *ket* do vetor ψ com valores $1/2, -i/2, \sqrt{2}/2$, a representação *bra* associada e o produto da representação *bra* com a matriz A (vide abaixo) e com a representação *ket*.

```
psi_ket = sp.Matrix([[sp.Rational(1,2)],
                     [-sp.I/2],
                     [1/sp.sqrt(2)]])
psi_ket
```

```
Matrix([
[      1/2],
[     -I/2],
[sqrt(2)/2]])
```

```
psi_bra = psi_ket.adjoint()
psi_bra
```

```
Matrix([[1/2, I/2, sqrt(2)/2]])
```

```
A = sp.Matrix([[0,1/sp.sqrt(2),0],
               [1,1/sp.sqrt(2),0],
               [0,0,0]])
psi_bra, A, psi_ket, (psi_bra * A * psi_ket).expand(complex=True)
```

```
(Matrix([[1/2, I/2, sqrt(2)/2]]), Matrix([
[0, sqrt(2)/2, 0],
[1, sqrt(2)/2, 0],
[0,         0, 0]]), Matrix([
```

```
[       1/2],
[     -I/2],
[sqrt(2)/2]]), Matrix([[sqrt(2)/8 - sqrt(2)*I/8 + I/4]]))
```

7. Calcule o módulo do vetor ψ

```
expr = sp.simplify(psiT.multiply(psi))
expr
```

```
Matrix([[20]])
```

8. Prove que os vetores $||psi_1\rangle = [1+i, i]^T$ e $\psi_2\rangle = [2+4i, 3i-1]^T$ diferem pelo fator $3+i$

```
psi1 = np.array([1+1j,1j]).reshape(2,1)
psi2 = np.array([2+4j,3j-1]).reshape(2,1)
psi2/psi1
```

```
array([[3.+1.j],
       [3.+1.j]])
```

9. Os vetores $|\psi_1\rangle = [1+i, 2-i]^T$ e $|\psi_2\rangle = [2+2i, 1-2i]^T$ podem representar o mesmo estado? RESP: Não, pois suas componentes diferem por valores distintos.

```
psi1 = np.array([1+1j,2-1j]).reshape(-1,1)
psi2 = np.array([2+2j,1-2j]).reshape(-1,1)
psi2/psi1
```

```
array([[2. +0.j ],
       [0.8-0.6j]])
```

10. Qual é o comprimento do vetor $|\psi\rangle = [2-3i, 1+2i]^T$?

```
psi1 = np.array([2-3j,1+2j]).reshape(-1,1)
print(np.sqrt(psi1.conjugate().T.dot(psi1)[0,0]).real)
```

```
4.242640687119285
```

```
np.linalg.norm(psi1)
```

```
4.242640687119285
```

11. Normalize o ket $|\psi\rangle = [3-i, 2+6i, 7-8i, 6.3+4.9i, 13i, 0, 21.1]^T$

```
psi1 = np.array([3-1j, 2+6j, 7-8j, 6.3+4.9j, 13j, 0, 21.1]).reshape(-1,1)
psi1_norm = (psi1 / np.linalg.norm(psi1))
psi1_norm
```

```
array([[0.103-0.034j],
       [0.069+0.207j],
       [0.241-0.276j],
       [0.217+0.169j],
       [0.   +0.448j],
       [0.   +0.j   ],
       [0.728+0.j   ]])
```

12. Verifique que os dois estados: $x_1 = [\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]^T$ e $x_2 = [\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}]^T$ tem módulo igual a 1 e apresente o vetor que representa a sobreposição destes estados.

O vetor é: $|\psi\rangle = c_1|x_1\rangle + c_2|x_2\rangle$

```
import numpy as np
x1 = np.array([np.sqrt(2)/2,np.sqrt(2)/2]).reshape(-1,1)
x2 = np.array([np.sqrt(2)/2,-np.sqrt(2)/2]).reshape(-1,1)
print(np.linalg.norm(x1), np.linalg.norm(x2))
```

```
1.0 1.0
```

13. Apresente a matrix hermitiana de Ω apresentada a seguir

```
omega = np.array([[-1,-1j],
                  [1j, 1]])
sp.Matrix(omega)
```

```
Matrix([
[ -1.0, -1.0*I],
[1.0*I,    1.0]])
```

```
sp.Matrix(np.matrix(omega).getH())
```

```
Matrix([
[ -1.0, -1.0*I],
[1.0*I,    1.0]])
```

14. Calcule o vetor resultante da atuação do operador Ω em ψ

```
psi = np.array([-1+0j,-1-1j])
omega = np.array([[-1,-1j],
                  [1j, 1]])
```

```
expr = "\Omega =" + sp.latex(sp.Matrix(omega))
expr = expr + sp.latex('\psi =') + sp.latex(sp.Matrix(psi))
expr = expr + sp.latex('\Omega . \psi =') + sp.latex(sp.Matrix(omega.dot(psi
expr
```

```
'\\Omega =\\left[\\begin{matrix}-1.0 & - 1.0 i\\\\1.0 i & 1.0\\end{matrix}\\
```

15. Calcule as operações a seguir

```
X = sp.Matrix([[0,1],[1,0]])
psi_bra = sp.Matrix([[1/sp.sqrt(3),sp.sqrt(sp.Rational(2,3))]]) # sp.Rationa
psi_ket = psi_bra.T
psi_bra * X * psi_ket
```

```
Matrix([[2*sqrt(2)/3]])
```

```
exp_X = psi_bra * X * psi_ket
exp_X2 = psi_bra * X * X * psi_ket # X squared expectation
exp_X_2 = exp_X ** 2 # X expectation squared
desv_X = sp.sqrt(exp_X2[0] - exp_X_2[0]) # X standard deviation
exp_X2, exp_X_2, desv_X
```

```
(Matrix([[1]]), Matrix([[8/9]]), 1/3)
```

16. Using the dynamics given in Equation (3.4), determine what the state of the system would be if you start with the state $[5, 5, 0, 2, 0, 15]^T$. The Dinamics is given by matrix M below:

```
import sympy as sp
M = sp.Matrix([[0,0,0,0,0,0],
               [0,0,0,0,0,0],
               [0,1,0,0,0,1],
               [0,0,0,1,0,0],
               [0,0,1,0,0,0],
               [1,0,0,0,1,0]])
X = (sp.Matrix([5,5,0,2,0,15]))
M
```

```
Matrix([
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 1],
[0, 0, 0, 1, 0, 0],
[0, 0, 1, 0, 0, 0],
[1, 0, 0, 0, 1, 0]])
```

```
X
```

```
Matrix([
[ 5],
[ 5],
[ 0],
[ 2],
[ 0],
[15]])
```

```
M*X
```

```
Matrix([
[ 0],
[ 0],
[20],
[ 2],
[ 0],
[ 5]])
```

17. For the matrix M given in Equation (3.4), calculate M^6. If all the marbles start at vertex 2, where will all the marbles end up after 6 time steps?

```
Mn = lambda M,n : ["M"+str(n)+" =", M**n]
n = 6
expr1 = "M^{} =".format(n) + sp.latex(M**n)
expr2 = "X =" + sp.latex(X)
expr3 = "M^{}.X =".format(n) + sp.latex((M**n)*X)
expr = expr1 + expr2 + expr3
expr
```

```
'M^6 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0\\\\0 & 0 & 0 & 0 & 0 & 0\\\
```

18. Consider the following graph representing city streets. Singleheaded arrows (->) correspond to one-way streets and double-headed arrows (<–>) correspond to two-way streets. Imagine that it takes one time click to traverse an arrow. You may assume that everyone must move at every time click. If every corner starts with exactly one person, where will everyone be after one time click? After two time clicks? After four time clicks?

```
M = sp.Matrix([[0,1,0,0,0,0,0,0,0],
               [1,0,0,0,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0],
               [0,0,0,1,0,0,0,0,0],
               [0,0,0,0,0,0,0,0,0],
```

```
                [0,0,1,0,0,0,0,0,0],
                [0,0,0,0,0,0,0,0,0],
                [0,0,0,0,0,0,1,0,0],
                [0,0,0,0,0,1,0,1,1]])
X = sp.Matrix([1,1,1,1,1,1,1,1,1])
expr1 = M*X
expr2 = (M**2)*X
expr4 = (M**4)*X
expr = "X_1 =" + sp.latex(expr1) + "X_2 =" + \
       sp.latex(expr2) + "X_4 =" + sp.latex(expr4)
expr
```

```
'X_1 =\\left[\\begin{matrix}1\\\\1\\\\0\\\\1\\\\0\\\\1\\\\0\\\\1\\\\3\\end{m
```

19. Suponha a matriz de transição de estados M apresentada abaixo. Calcule o produto elemento a elemento da matriz M com sua transposta conjugada.

```
import sympy as sp
M = sp.Matrix([[           0,                    0,0,0,0,0,0,0],
               [1/sp.sqrt(2),                    0,0,0,0,0,0,0],
               [1/sp.sqrt(2),                    0,0,0,0,0,0,0],
               [           0,(-1+sp.I)/sp.sqrt(6),
0,1,0,0,0,0],
               [           0,(-1-sp.I)/sp.sqrt(6),
0,0,1,0,0,0],
               [           0,(+1-sp.I)/sp.sqrt(6),(-1+sp.I)/sp.sqrt(6),0,0,1
               [           0,                    0,(-1-sp.I)/sp.sqrt(6),0,0,0
               [           0,                    0,(+1-sp.I)/sp.sqrt(6),0,0,0

X1 = sp.Matrix([1,0,0,0,0,0,0,0])
expr = "M =" + sp.latex(M) + "X_1 =" + sp.latex(X1)
expr
```

```
'M =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\\\frac{\\sqrt{2}}
```

```
P1 = sp.simplify(sp.matrix_multiply_elementwise(M,M.conjugate()))
expr = "P_1 =" + sp.latex(P1)
expr
```

```
'P_1 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\\\frac{1}{2} & (
```

```
M2 = sp.simplify(M*M)
expr = "M_2 =" + sp.latex(M2)
expr
```

```
'M_2 =\\left[\\begin{matrix}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\\\0 & 0 & 0 & 0 &
```

15.5 Interferência e Sobreposição

If the weights are real numbers, the probabilities associated with them can only increase when they are added.

In contrast, if the weights are complex numbers, the probabilities are associated with their modulus squared. Therefore if you add two weights that are complex numbers, the probability can either increase, add up to 0 or decrease.

When two complex numbers are added and cancel each other (they add up to 0) this is referred to as **interference**.

Let the state of the system be given by $X = [c_0, c_1, ..., c_{n01}], T \in C_n$. It is incorrect to say that the probability of the photon's being in position k is $|c_k|$

Rather, to be in state X means that the particle is in some sense in all positions simultaneously. The photon passes through the top slit and the bottom slit simultaneously, and when it exits both slits, it can cancel itself out. A photon is not in a single position, rather it is in many positions, a **superposition**.

Matrix below shows the probability of starting in column j a photon will end up in row i after 2 time periods (or 2 ticks of clock).

For instance, if a photon is in state 0 (first column), after two time periods it will end up in state 3 (forth row) with probability $\frac{1}{6}$.

Now look at the element $P_2[5, 0]$. It is equal to 0. This means that the probability of a photon being in state 0 in instant 0 and in state 5 in instant 2 is 0, so there will be no photon at all there.

This is caused by **interference**.

The difficult thing to understand (to accept would be a better term) is that it is wrong to say that there's a probability for the photon to be in a state. In reality the photon is itself in all states, and each state has a definite probability associated with it.

Probability of what? Of us, when trying to measure where the photon "is" will find it in that state at that time.

BUT, when we're not looking (i.e.trying to measure it), photon is in a **superposition** of states. In other words: it is simultaneously in all states at the same time!

A classical computer can be in one state at a time (0 or 1). A quantum computer uses the superposition phenomenon to be in all classical states at the same time. In other words: a quantum computer is the ultimate idea in parallel processing.

Both matter and light manifest a particle-like and a wave-like behavior

15.6 Interconexão de Sistemas

Assembling Systems: Calcule o produto tensorial de M com N. O que ele representa em termos de circuitos quânticos?

```
import sympy as sp
from sympy.physics.quantum import TensorProduct
M = sp.Matrix([[0,                sp.Rational(1,6),sp.Rational(5,6)],
               [sp.Rational(1,3),sp.Rational(1,2),sp.Rational(1,6)],
               [sp.Rational(2,3),sp.Rational(1,3),
0]])

N = sp.Matrix([[sp.Rational(1,3),sp.Rational(2,3)],
               [sp.Rational(2,3),sp.Rational(1,3)]])
MxN = TensorProduct(M,N)
expr1 = sp.latex('M = ') + sp.latex(M)
expr2 = sp.latex('N = ') + sp.latex(N)
expr3 = sp.latex('M x N = ') + sp.latex(MxN)
expr = expr1 + expr2 + expr3
expr
```

```
'\\mathtt{\\text{M = }}\\left[\\begin{matrix}0 & \\frac{1}{6} & \\frac{5}{6}
```

15.7 O Dataset Bancalvo

A base de dados a seguir será utilizada para exemplificar a aplicação de Quantum Computing à resolução de problemas de Machine Learning

```
site = "https://github.com/"
diretorio = "gustavomirapalheta/classes_datasets/blob/master/"
arquivo = "bancalvo.xlsx?raw=true"
link = site + diretorio + arquivo
train = pd.read_excel(link)

lista = list(train['UF'].unique())
lista.remove('SP')
train['novaUF'] = train['UF'].replace(lista, 'NOSP')
train['lRENDA'] = np.log10(train['RENDA'])
train = train.drop(['IDENTIDADE','UF','RENDA'], axis=1)
train = pd.get_dummies(train, drop_first=True)
X = train.drop('STATUS_inad', axis=1).copy()
y = train['STATUS_inad'].copy()

from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, random_state = 53, tes
Xtrain.shape, Xtest.shape, ytrain.shape, ytest.shape
```

Bibliografia

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for r.* https://CRAN.R-project.org/package=rmarkdown.

Canale, Raymond P., and Steven C. Chapra. 2016. *Métodos Numéricos Para Engenharia.* 5a ed. São Paulo: McGraw-Hill.

Demidovitch, B. 1974. *Problems in Mathematical Analysis.* 2nd ed. Moscow: MIR Publishers.

Dirac, P. A. M. 1957. *The Principles of Quantum Mechanics.* 4th ed. USA: Snowball Publishing. www.snowballpublishing.com.

McMahon, David. 2007. *Quantium Computing Explaneid.* 1st ed. Wiley-IEEE Computer Society Pr. www.ieee.org.

Nielsen, Michael A., and Isaac L. Chuang. 2016. *Quantum Computation and Quantum Information.* 4th ed. Cambridge, UK: Cambridge University Press. www.cambridge.org.

Strang, Gilbert. 2009. *Computational Science and Engineering.* 1st ed. Wellesley-Cambridge Press. www.cambridge.org.

Xie, Yihui. 2021. *Knitr: A General-Purpose Package for Dynamic Report Generation in r.* https://yihui.org/knitr/.