Introdução à computação científica em economia e finanças em Python

Nelson Seixas dos Santos

Faculdade de Ciências Econômicas
Universidade Federal do Rio Grande do Sul

June 1, 2018

Sumário

Introdução

- Trata-se aqui da solução de problemas teóricos de economia e finanças cuja solução exige o emprego de recursos computacionais;
- Tais problemas normalmente exigem soluções que empregam o cálculo diferencial e integral em valores vetoriais.
- Os métodos (algoritmos) que permitem determinar as soluções para problemas matemáticos diversos que vão desde a solução de raizes de uma equação até o cálculo diferencial e integral vetorial compõem uma parte da matemática chamada cálculo numérico;

Introdução (cont.)

- Os pacotes Python mais usados na implementação dos métodos numéricos são o o math, cmath, fractions, random, statistics, numpy, matPlotLib, scipy e pandas.
- Em data science, usa-se adicionalmente, os pacotes Python statsModels e seaborn.
- Aqui trataremos apenas dos pacotes mencionados no primeiro item.

O pacote math

Para ver as funções disponibilizadas pelo pacote, clique aqui

O pacote cmath

Contém as mesmas funções que o pacote math com a peculiaridade de serem aplicáveis a números complexos. Para ver as funções disponibilizadas pelo pacote, clique aqui

O pacote fractions

Para ver as funções disponibilizadas pelo pacote, clique aqui

O pacote random

Para ver as funções disponibilizadas pelo pacote, clique aqui

O pacote statistics

Para ver as funções disponibilizadas pelo pacote, clique aqui

A api do matplotlib

Para ver as funções disponibilizadas pelo pacote, clique aqui

O pacote numpy

- O pacote numpy habilita o interpretador python com estruturas de dados e operações sobre estas que serão úteis em cálculo numérico;
- A estrutura de dados fundamental é o numpy ndarray, que basicamente implementa a noção de vetor da álgebra linear;
- Para detalhes, clique aqui.

O numpy array: sintaxe

Exemplo: vetor

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

Exemplo: matriz

```
import numpy as np
x = np.array([[1,2,3],[4,5,6]])
```

O pacote scipy

- O pacote scipy fornece ao Python módulos onde estão implementados os mais importantes métodos numéricos conhecidos na literatura de cálculo numérico;
- Os métodos implementados no scipy são aplicados sobre a estrutura de dados numpy array. Por isso, para usar o scipy, é preciso ter o numpy instalado;
- Os módulos mais importantes para nosso uso são o scipy.optimize e o scipy.linalg.

Pacotes científicos e a distribuição Anaconda Python

- Os pacotes científicos estão reunidos no pacote Scipy e podem ser obtidos com o gerenciador de pacotes do linux, porém os disponíveis no linux são desatualizados.
- Para resolver o problema de atualização, pode-se instalar o Scipy pelo gerenciador de pacotes do Python, usando a opção –user.
- Distribuição é um grande conjuntos de pacotes que são instalados simultaneamente quando da instalação da distribuição;
- A distribuição mais usada em economia e finanças é a Anaconda Python
- Aprenda a usar o Jupyter Notebook, clicando aqui

Equações não lineares: o problema

Problema

Seja f uma função não linear. Determine a solução da equação a seguir:

$$f(x) = 0$$

Algumas soluções clássicas

1. método da bissecção
2. método da iteração linear
3. método de Newton
4. método das secantes

Exemplo

Resolva a seguinte equação do segundo grau.

$$2.x^2 + 5.x + 3 = 0$$

Solução da equação não linear em Python: o módulo scipy.optimize

- O módulo optimize do pacote scipy oferece diversas funções para otimização de funções, isto é, determinação de máximos ou mínimos.
- A solução de uma equação não linear usa métodos numéricos de aproximação onde há minimização do erro. Por isso, é neste módulo que se encontram as funções de solução de equações não lineares.
- A função recomendada para determinar a raiz de uma equação não linear dado um intervalo real [a,b] é optimize.brentq().
- bretq() só encontra raízes se o valor do sinal da função nos extremo inferior do intervalo procurado for diferente do sinal no extremo superior.

Solução do exemplo em Python

```
#Importacao de pacotes
from scipy import optimize

# Entrada da funcao a ser otimizada
def funcnelquad(x):
        y = 2*x**2 + 5*x + 3
        return(y)

# Processamento
z = optimize.brentq(funcnelquad, -1.5,0)

# Saida
print(z)
```

Sistemas lineares: o problema

Determine a solução do sistema linear a seguir:

$$A.x = b \tag{1}$$

onde A é uma matriz $n \times n$ e x e b são um vetores $n \times 1$

Sistema linear: um exemplo

$$\begin{cases} x + y + z & = 2 \\ 9x + 3y + z & = 4 \\ 25x + 5y + z & = 6 \end{cases}$$

Sistemas lineares: soluções clássicas

- $x = A^{-1}.b$ - solução lenta
- triangularizar A por eliminação gaussiana e resolver o sistema de baixo para cima - solução rápida.

Solução do sistema linear em Python: o módulo scipy.linalg

O módulo linalg do pacote Scipy fornece os métodos numéricos para solução de problemas de álgebra linear. Em particular, temos que:

- A primeira solução é obtida em Python scipy.linalg.inv(A) e scipy.dot(b)
- a segunda solução é obtida usando a função linalg.solve(A,b)

Solução do do exemplo em Python

```
#Importacao de pacotes
from scipy import linalg

# Entrada
a = np.array([[1,1,1],[9,3,1],[25,5,1]])
b = np.array([2,4,6])

# Processamento
x = linalg.solve(a, b)

# Saida
print('O valor de x eh igual a', x)
```

Autovetores e autovalores

Os autovalores de um operador linear (matriz quadrada) A são as raízes do polinômio característico. Formalmente:

$$|A - \lambda.I| = 0$$

onde I é a matriz identidade de mesma ordem que A.

A função scipy.linalg.elg()

Esta função retorna diretamente os autovalores e autovetores de A.

Mínimos Quadrados Ordinários

scipy.linalg.lstsq()

Interpolação polinomial

Problema

Determinar o polinômio P(x) que passa pelos n+1 pontos $(x_0, y_0), (x_1, y_1), (x_2, y_2), ..., (x_{n-1}, y_{n-1}), (x_n, y_n)$.

Solução

Para resolver o problema posto, lembre que a equação geral de um polinômio de n-ésimo grau é dada por:

$$P(x) = a_0 + a_1.x + a_2.x^2 + ... + a_{n-1}.x^{n-1} + a_n.x^n$$

Como a equação acima tem n+1 coeficientes, basta-nos resolver o sistema linear a seguir:

1. $y_0 = P(x_0)$
2. $y_1 = P(x_1)$
3. ...
4. $y_{n-1} = P(x_{n-1})$
5. $y_n = P(x_n)$

Exemplo

Encontre o polinômio que passa pelos pontos abaixo:

1. $(x_0, y_0) = (1, 2)$
2. $(x_1, y_1) = (3, 4)$
3. $(x_2, y_2) = (5, 6)$

Solução

$$y = a.x^2 + b.x + c \tag{2}$$

Logo, substituindo as coordenadas dos pontos dados na equação (2) acima, temos:

$$\begin{cases} a + b + c & = 2 \\ 9a + 3b + c & = 4 \\ 25a + 5b + c & = 6 \end{cases}$$

Scipy

Alternativamente, podemos usar a função de interpolação do pacote Python Scipy

Exemplo: determinação da estrutura a termo das taxas de juros de uma economia

Problema

Determinar o polinômio que que liga os pontos $(t - t, r_{t,T})$ no plano tempo-taxa.