MC-102 — Aula 13
Funções III

Prof. Luiz F. Bittencourt

Turmas QR

Instituto de Computação – Unicamp

2019

Conteúdo adaptado de slides fornecidos pelo Prof. Eduardo Xavier.

Roteiro

1. Exemplo com funções: Calculadora Financeira
   - Juros de uma Compra a Prazo
   - Retorno de uma Aplicação
2. Exercícios

Calculadora Financeira

- Vamos criar um programa com algumas funções de matemática financeira.
- O programa deve ter as seguintes funcionalidades:
  - *Juros de compra a prazo:* dado o valor à vista de um produto, **vProd**, e o valor das prestações, **vPrest**, que devem ser pagas por **p** períodos, deve-se achar a taxa de juros **j** cobrada por período.
  - *Valor de uma aplicação:* dado um montante inicial **mont** aplicado em um fundo com taxa de juros **j** por período, e uma quantia **apl** aplicada em cada período subsequente, deve-se calcular o valor da aplicação após **p** períodos.

Juros de uma Compra a Prazo

- Computar a taxa de juros cobrada, quando compramos um produto cujo valor à vista é **vProd**, com prestações no valor **vPrest** que devem ser pagas em **p** períodos.
- O valor dos juros **j** cobrados satisfaz a equação abaixo:

$$f(j) = vProd \cdot (1+j)^p - vPrest \cdot \frac{(1+j)^p - 1}{j} = 0$$

- Ou seja, devemos achar o valor de **j** que é um zero da função **f(j)**.

Juros de uma Compra a Prazo

- Vamos utilizar o método de Newton para isso:
  - Dada uma função $f(x)$, podemos achar os zeros dessa função com sucessivas aproximações.
  - Seja x_0 um valor inicial que achamos estar próximo do zero da função.
  - Dada uma aproximação x_n anterior, uma próxima aproximação melhor é computada pela equação:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

No nosso caso:

$$\mathrm{f(j)}' = \mathrm{vProd} \cdot \mathrm{p} \cdot (1+\mathrm{j})^{\mathrm{p}-1} - \mathrm{vPrest} \cdot \left(\frac{\mathrm{p} \cdot (1+\mathrm{j})^{\mathrm{p}-1}}{\mathrm{j}} - \frac{(1+\mathrm{j})^{\mathrm{p}} - 1}{\mathrm{j}^2} \right)$$

Juros de uma Compra a Prazo

Criamos uma função para avaliar

$$f(j) = \mathrm{vProd} \cdot (1 + \mathrm{j})^{\mathrm{p}} - \mathrm{vPrest} \cdot \frac{(1 + \mathrm{j})^{\mathrm{p}} - 1}{\mathrm{j}}$$

```
def funcaoFj(vProd, p, vPrest, j):
  pote = math.pow(1+j, p)
  return vProd*pote - vPrest*((pote-1)/j)
```

OBS: Estamos utilizando a função **pow** da biblioteca **math** para computar potências. Portanto inclua o comando **import math** no início do seu código.

Juros de uma Compra a Prazo

Criamos uma função para avaliar

$$f(j)' = vProd \cdot p \cdot (1+j)^{p-1} - vPrest \cdot \left(\frac{p \cdot (1+j)^{p-1}}{j} - \frac{(1+j)^{p} - 1}{j^2} \right)$$

```python
def derivadaFj(vProd,  p, vPrest, j):
  pote1 = math.pow(1+j, p)
  pote2 = math.pow(1+j, p-1)
  aux = vProd*p*pote2 - vPrest*p*pote2/j + vPrest*(pote1 - 1)/(j*j)
  return aux
```

Juros de uma Compra a Prazo

- As sucessivas aproximações são computadas segundo:

$$j_{n+1} = j_n - \frac{f(j_n)}{f'(j_n)}$$

- Podemos fazer $j_0 = 1$, pois provavelmente $0 \leq \mathrm{j} \leq 1$.
- Faremos sucessivas aproximações, mas quando parar?
  - Quando acharmos **j** que faz a equação ser próxima o suficiente de zero:

$$\mathrm{f(j) = vProd \cdot (1 + j)^p - vPrest \cdot \frac{(1 + j)^p - 1}{j}} \approx 0$$

Juros de uma Compra a Prazo

- Definimos que $f(j) \approx 0$ quando $-0.000000001 \leq f(j) \leq 0.000000001$.
- Criamos uma função para computar o módulo:

```
def modulo(x):
  if(x > 0):
    return x
  return -1*x
```

Juros de uma Compra a Prazo

Com todas as funções anteriores estamos prontos para aplicar o método de Newton e achar o valor dos juros cobrados.

$$j_{n+1} = j_n - \frac{f(j_n)}{f'(j_n)}$$

- O nosso algoritmo deverá funcionar da seguinte forma:

```
j = 1.0
Enquanto j não for zero da função f(j) faça
           j = j - f(j)/f'(j)
```

Juros de uma Compra a Prazo

Agora em Python utilizando as funções anteriores:

```
def achaJ(vProd, p, vPrest):
  j=1.0
  fj = funcaoFj(vProd, p, vPrest, j)
  while( modulo(fj) > EPS ):
    dfj = derivadaFj(vProd, p, vPrest, j)
    j = j - fj/dfj
    fj = funcaoFj(vProd, p, vPrest, j)

  return j
```

OBS: **EPS** é uma constante definida após a seção de bibliotecas com o comando:

```
EPS = 0.000000001
```

Retorno de uma Aplicação

- Dado um montante inicial **mont** aplicado em um fundo com taxa de juros **j** por período, com aplicações **apl** subsequentes deve-se calcular o valor aplicado em cada um dos **p** períodos.
- O valor final **vFim** após *p* períodos é dado por:

$$\mathrm{vFim} = (1 + \mathrm{j})^{\mathbf{p}} \cdot \mathrm{mont} + \mathrm{apl} \cdot \left(\frac{(1 + \mathrm{j})^{\mathbf{p}} - 1}{j} \right)$$

Retorno de uma Aplicação

- A função deverá retornar o valor aplicado ao final de cada período em um vetor de retorno que chamaremos de **ret**.

```
def retornoApli(mont, apl, p, j):
    ret = []
    pote = 1+j

    for i in range(p):
        ret.append(pote*mont + apl*(pote-1)/j)
        pote = pote*(1+j)
    return ret
```

- Lembre-se que valor ao final de **p** períodos é dado por

$$\mathrm{vFim} = (1+\mathrm{j})^{\mathbf{p}} \cdot \mathrm{mont} + \mathrm{apl} \cdot \left(\frac{(1+\mathrm{j})^{\mathrm{p}} - 1}{j} \right)$$

Retorno de uma Aplicação

- Com a função do item anterior podemos chamá-la de uma outra que lê os dados do teclado e computa o retorno como por exemplo:

```
def retornoAplicacao():
  mont = float(input("Valor aplicado inicialmente: "))
  p = int(input("Numero de periodos: "))
  apl = float(input("Valor aplicado por periodo subsequente: "))
  j = float(input("Juros da aplicacao por periodo (em %%): "))
  j = j/100

  retorno = retornoApli(mont, apl, p, j) #chamada da fun. anterior
  for i in range(p):
    print("Montante ao final do periodo " +str(i+1)+ " %.2f" %(retorno[i]))
```

Programa Completo

Exemplo de programa completo:

```
import math
EPS = 0.00000001

def main():
  print("\t Escolha uma funcionalidade:")
  print("\t1 - Encontrar valor do juros em compra a prazo")
  print("\t2 - Encontrar valor final de uma aplicacao")
  print("\tEntre com opcao (1-2):")
  opcao = int(input())

  if(opcao == 1):
    jPrestacao()
  elif(opcao == 2):
    retornoAplicacao()


def jPrestacao():
  vProd = float(input("Valor a vista do produto: "))
  vPrest = float(input("Valor da prestacao do produto: "))
  p = int(input("Numero de prestacoes: "))
  aux = achaJ(vProd, p, vPrest)*100
  print("Valor do juros por periodo: %.2f%%" %aux)

def achaJ(vProd, p, vPrest):
  j=1.0
  fj = funcaoFj(vProd, p, vPrest, j)
  while( modulo(fj) > EPS ):
    dfj = derivadaFj(vProd, p, vPrest, j)
    j = j - fj/dfj
    fj = funcaoFj(vProd, p, vPrest, j)
  return j
```

Programa Completo

Exemplo de programa completo:

```
def modulo(x):
  if(x > 0):
    return x
  return -1*x

def funcaoFj(vProd, p, vPrest, j):
  pote = math.pow(1+j, p)
  return vProd*pote - vPrest*((pote-1)/j)


def derivadaFj(vProd, p, vPrest, j):
  pote1 = math.pow(1+j, p)
  pote2 = math.pow(1+j, p-1)
  aux = vProd*p*pote2 - vPrest*p*pote2/j + vPrest*(pote1 - 1)/(j*j)
  return aux
```

Programa Completo

Exemplo de programa completo:

```
def retornoAplicacao():
  mont = float(input("Valor aplicado inicialmente: "))
  p = int(input("Numero de periodos: "))
  apl = float(input("Valor aplicado por periodo subsequente: "))
  j = float(input("Juros da aplicacao por periodo (em %): "))
  j = j/100

  retorno = retornoApli(mont, apl, p, j)
  for i in range(p):
    print("Montante ao final do periodo " +str(i+1)+ " %.2f" %(retorno[i]))


def retornoApli(mont,  apl,  p,  j):
  ret = []
  pote = 1+j

  for i in range(p):
    ret.append(pote*mont + apl*(pote-1)/j)
    pote = pote*(1+j)
  return ret
```

Exercício

Crie uma função para a seguinte funcionalidade da nossa calculadora financeira:

- *Calculo do valor das prestações*: dado um valor à vista **vProd** de um produto, o valor **vPrest** das prestações que devem ser pagas, assumindo-se **p** períodos e taxa de juros **j** é dado por

$$\mathrm{vPrest} = \frac{(1+\mathrm{j})^{\mathrm{p}} \cdot \mathrm{vProd} \cdot \mathrm{j}}{(1+\mathrm{j})^{\mathrm{p}} - 1}$$

- Crie uma função para calcular o valor das prestações de um produto em uma compra a prazo.