MC-102 — Aula 17
Funções III

Instituto de Computação – Unicamp

13 de Outubro de 2016

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** cobradas 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:
  - Dado 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.
  - Dado 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

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

```
double funcaoFj(double vProd, int p, double vPrest, double j){
  double pote, aux =0;
  pote = pow(1+j, p);
  return vProd*pote - vPrest*((pote-1)/j);
}
```

OBS: Estamos utilizando a função **pow** da biblioteca **math.h** para computar potências.

Juros de uma Compra a Prazo

Criamos uma função para avaliar

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

```
double derivadaFj(double vProd, int p, double vPrest, double j){
  double pote1, pote2, aux;
  pote1 = pow(1+j, p);
  pote2 = 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:

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

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:

```
double modulo(double 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 C utilizando as funções anteriores:

```
double achaJ(double vProd, int p, double vPrest){
  int i;
  double j=1.0, fj, dfj;

  fj = funcaoFj(vProd, p, vPrest, j);
  while( modulo(fj) > EPS ){ //Enquanto j nao for um zero da funcao fa
      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:

```
#define 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:

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

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**.

```
void retornoApli(double mont, double apl, int p, double j, double ret[]){
  double pote = 1+j;
  int i;
  for(i=0; i < p; i++){
    ret[i] = pote*mont + apl*(pote-1)/j;
    pote = pote*(1+j);
  }
}
```

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

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

Retorno de uma Aplicação

- Com a função do item anterior podemos chamá-la de um programa como por exemplo:

```
int main(){
  double mont, apl, j, retorno[100];
  int p;
  printf("Valor aplicado inicialmente: ");
  scanf("%lf", &mont);
  printf("Numero de periodos: ");
  scanf("%d", &p);
  printf("Valor aplicado por periodo subsequente: ");
  scanf("%lf", &apl);
  printf("Juros da aplicacao por periodo (em %%): ");
  scanf("%lf", &j);
  j = j/100;

  if(p>100)
    p=100;
  retornoApli(mont, apl, p, j, retorno);
  for(int i=0; i<p; i++){
    printf("Montante ao final do periodo %d: %lf\n", i+1, retorno[i]);
  }
}
```

Programa Completo

Exemplo de programa completo:

```
#include <stdio.h>
#include <math.h>

#define EPS 0.00000001

void jPrestacao();
double achaJ(double vProd, int p, double vPrest);
double funcaoFj(double vProd, int p, double vPrest, double j);
double derivadaFj(double vProd, int p, double vPrest, double j);
double modulo(double x);

void retornoAplicacao();
void retornoApli(double mont, double apl, int p, double j, double ret[]);

int main(){
  int opcao;

  printf("\n\t Escolha uma funcionalidade:\n\n");
  printf("\t1 - Encontrar valor do juros em compra a prazo\n");
  printf("\t2 - Encontrar valor final de uma aplicacao\n");
  printf("\tEntre com opcao (1-2):");
  scanf("%d", &opcao);

  if(opcao == 1){
    jPrestacao();
  }else if(opcao == 2){
    retornoAplicacao();
  }
}
```

Programa Completo

Exemplo de programa completo:

```
void jPrestacao(){
  double vProd, vPrest;
  int p;
  printf("Valor a vista do produto: ");
  scanf("%lf", &vProd);
  printf("Valor da prestacao do produto: ");
  scanf("%lf", &vPrest);
  printf("Numero de prestacoes: ");
  scanf("%d", &p);

  printf("Valor do juros por periodo: %lf%%\n", achaJ(vProd, p, vPrest)*100 );
}

double achaJ(double vProd, int p, double vPrest){
  int i;
  double j=1.0, fj, dfj;

  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:

```
double modulo(double x){
  if(x > 0)
    return x;
  return -1*x;
}

double funcaoFj(double vProd, int p, double vPrest, double j){
  double pote, aux =0;
  pote = pow(1+j, p);
  return vProd*pote - vPrest*((pote-1)/j);
}

double derivadaFj(double vProd, int p, double vPrest, double j){
  double pote1, pote2, aux;
  pote1 = pow(1+j, p);
  pote2 = 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:

```
void retornoAplicacao(){
  double mont, apl, j, retorno[100];
  int p;
  printf("Valor aplicado inicialmente: ");
  scanf("%lf", &mont);
  printf("Numero de periodos: ");
  scanf("%d", &p);
  printf("Valor aplicado por periodo subsequente: ");
  scanf("%lf", &apl);
  printf("Juros da aplicacao por periodo (em %%): ");
  scanf("%lf", &j);
  j = j/100;

  if(p>100)
    p=100;
  retornoApli(mont, apl, p, j, retorno);
  for(int i=0; i<p; i++){
    printf("Montante ao final do periodo %d: %lf\n", i+1, retorno[i]);
  }
}


void retornoApli(double mont, double apl, int p, double j, double ret[]){
  double pote = 1+j;
  int i;
  for(i=0; i < p; i++){
    ret[i] = pote*mont + apl*(pote-1)/j;
    pote = pote*(1+j);
  }
}
```

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

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

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