Cálculo Numérico - Trabalho Prático 1
Vinícius Claudino Ferraz - 18/09/2007

(Os programas foram feitos no Turbo C da Borland.)

1

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

typedef float Matriz[101] [101];
typedef float Vetor[101];

float Sqr(float x) {
  return x * x;
}

void Transpor(int n, Matriz L) { // retornar L^t
  int i, j;
  float reg;
  for (i = 1; i <= n; i++)
    for (j = i + 1; j <= n; j++) {
      reg = L[i][j];
      L[i][j] = L[j][i];
      L[j][i] = reg;
    }
}

// Lx = c, retornar x
void Subst_Sucessivas(int n, Matriz L, Vetor x, Vetor c) {
  int i, j;
  float soma;

  x[1] = c[1] / L[1][1];
  for(i = 2; i <= n; i++) {
    soma = 0;
    for(j = 1; j <= i - 1; j++)
      soma = soma + L[i][j] * x[j];

    x[i] = (c[i] - soma) / L[i][i];
  }
}

// Ux = d, retornar x
void Subst_Retroativas(int n, Matriz U, Vetor x, Vetor d) {
  int i, j;
  float soma;

  x[n] = d[n] / U[n][n];
  for (i = n - 1; i >= 1; i--) {
    soma = 0;
    for (j = i + 1; j <= n; j++)
      soma = soma + U[i][j] * x[j];
    x[i] = (d[i] - soma) / U[i][i];
  }
}

int LDL_t(int n, Matriz A, Matriz L, Matriz D) {
  int i, j, k;
  float soma;

  for (j = 1; j <= n; j++)
    L[j][j] = 1;

  for (j = 1; j <= n; j++) {
    soma = 0;
    for (k = 1; k <= j - 1; k++)
      soma = soma + Sqr(L[j][k]) * D[k][k];

    D[j][j] = A[j][j] - soma;
    if (D[j][j] <= 0) {
      printf("A matriz nao eh definida positiva");
      return 1; // 1 para sair do bloco "main"
    }

    for (i = j + 1; i <= n; i++) {
      soma = 0;
      for (k = 1; k <= j - 1; k++)
     soma = soma + L[i][k] * D[k][k] * L[j][k];

      L[i][j] = (A[i][j] - soma) / D[j][j];
    }
  }
  return 0;
}

void main() {
  int i, j, n;
  Matriz A, L, D;
  Vetor b, t, x, y;

  clrscr();
  printf("Digite n (m ximo de 100): ");
  scanf("%d", &n);
  if (n > 100) return;
  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++) {
      printf("Digite A[%d,%d]: ", i, j);
      scanf("%f", &A[i] [j]);
      L[i][j] = 0; // inicializa L
      D[i][j] = 0; // inicializa D
    }

  for (i = 1; i <= n; i++) {
    printf("Digite b[%d]: ", i);
    scanf("%f", &b[i]);
  }

  if (LDL_t(n, A, L, D))
    return;

  Subst_Sucessivas(n, L, y, b);
//como D eh diagonal, podemos usar as subst. sucessivas abaixo:
  Subst_Sucessivas(n, D, t, y);

  Transpor(n, L);
  Subst_Retroativas(n, L, x, t); // L^t ú x = y

//exibir L
  Transpor(n, L); // (L^t)^t = L
  printf("\n\nPela decomposicao LDL_t:\n");

  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      printf("L[%d,%d] = %8.10f\n", i, j, L[i] [j]);

  getch();

//exibir D
  printf("\n\nTambem pela decomposicao LDL_t:\n");

  for (i = 1; i <= n; i++)
    for (j = 1; j <= n; j++)
      printf("D[%d,%d] = %8.10f\n", i, j, D[i] [j]);

  getch();

//exibir y, t
  printf("\nPelas substituicoes sucessivas:\n");

  for (i = 1; i <= n; i++)
    printf("y[%d] = %8.10f\n", i, y[i]);

  printf("\n");
  for (i = 1; i <= n; i++)
    printf("t[%d] = %8.10f\n", i, t[i]);

  getch();

//exibir x
  printf("\nPelas substituicoes retroativas:\n");

  for (i = 1; i <= n; i++)
    printf("x[%d] = %8.10f\n", i, x[i]);

  getch();
}
```

Resultado da Letra c

```
Pela decomposicao LDL_t:
L[1,1] = 1.0000000000
L[1,2] = 0.0000000000
L[1,3] = 0.0000000000
L[1,4] = 0.0000000000
L[2,1] = 0.3333333433
L[2,2] = 1.0000000000
L[2,3] = 0.0000000000
L[2,4] = 0.0000000000
L[3,1] = 0.1666666716
L[3,2] = 0.2000000030
L[3,3] = 1.0000000000
L[3,4] = 0.0000000000
L[4,1] = -0.1666666716
L[4,2] = 0.1000000089
L[4,3] = -0.2432432324
L[4,4] = 1.0000000000

Tambem pela decomposicao LDL_t:
D[1,1] = 6.0000000000
D[1,2] = 0.0000000000
D[1,3] = 0.0000000000
D[1,4] = 0.0000000000
D[2,1] = 0.0000000000
D[2,2] = 3.3333332539
D[2,3] = 0.0000000000
D[2,4] = 0.0000000000
D[3,1] = 0.0000000000
D[3,2] = 0.0000000000
D[3,3] = 3.7000000477
D[3,4] = 0.0000000000
D[4,1] = 0.0000000000
D[4,2] = 0.0000000000
D[4,3] = 0.0000000000
D[4,4] = 1.5810811520

Pelas substituicoes sucessivas:
y[1] = 36.3199996948
y[2] = 22.8933334351
y[3] = 56.6879997253
y[4] = 92.5529708862

t[1] = 6.0533332825
t[2] = 6.8680000305
t[3] = 15.3210811615
t[4] = 58.5377731323

Pelas substituicoes retroativas:
x[1] = 12.5155553818
x[2] = -4.8977775574
x[3] = 29.5599975586
```

```
x[4] = 58.5377731323
```

2

```
#include <math.h>

float f(float x, float y) {
  return (x * x - 3 * x - 1 - y);
}

float g(float x) {
  return (0.55 * exp(1-x) + x * x - 5 * x + 4);
}

void DOPRI(float a, float b, float m, float y0) {
  int i;
  float h, xt, yt, VetX[101], VetY[101], EG[101], x, y,
    k1, k2, k3, k4, k5, k6, k7, ErroGlobal, Erro,
  a21 = 1.0/5,
  a31 = 3.0/40,
  a32 = 9.0/40,
  a41 = 44.0/45,
  a42 = -56.0/15,
  a43 = 32.0/9,
  a51 = 19372.0/6561,
  a52 = -25360.0/2187,
  a53 = 64448.0/6561,
  a54 = -212.0/729,
  a61 = 9017.0/3168,
  a62 = -355.0/33,
  a63 = 46732.0/5247,
  a64 = 49.0/176,
  a65 = -5103.0/18656,
  a71 = 35.0/384,
  a73 = 500.0/1113,
  a74 = 125.0/192,
  a75 = -2187.0/6784,
  a76 = 11.0/84,
  c2 = 1.0/5,
  c3 = 3.0/10,
  c4 = 4.0/5,
  c5 = 8.0/9,
  c6 = 1.0,
  c7 = 1.0,
  e1 = 71.0/57600,
  e3 = -71.0/16695,
  e4 = 71.0/1920,
  e5 = -17253.0/339200,
  e6 = 22.0/525,
  e7 = -1.0/40;
  h = (b - a)/m;
  xt = a;
  yt = y0;
  VetX[1] = xt;
  VetY[1] = yt;
  EG[1] = 0;
  printf(" i   x             y              ErroGlobal
Erro\n");
  printf("%2d  %13.10f  %13.10f\n", 0, xt, yt);
  for(i = 1; i <= m; i++) {
    x = xt;
    y = yt;
    k1 = h * f(x,y);
    x = xt + c2 * h;
    y = yt + a21 * k1;
    k2 = h * f(x,y);
    x = xt + c3 * h;
    y = yt + a31 * k1 + a32 * k2;
    k3 = h * f(x,y);
    x = xt + c4 * h;
    y = yt + a41 * k1 + a42 * k2 + a43 * k3;
    k4 = h * f(x,y);
    x = xt + c5 * h;
    y = yt + a51 * k1 + a52 * k2 + a53 * k3 + a54 * k4;
    k5 = h * f(x,y);
    x = xt + c6 * h;
    y = yt + a61 * k1 + a62 * k2 + a63 * k3 + a64 * k4 + a65 * k5;
    k6 = h * f(x,y);
    x = xt + c7 * h;
    y = yt + a71 * k1 + a73 * k3 + a74 * k4 + a75 * k5 + a76 * k6;
    k7 = h * f(x,y);
    xt = a + i * h;
    yt = yt + a71 * k1 + a73 * k3 + a74 * k4 + a75 * k5 + a76 *
k6;
    ErroGlobal = e1 * k1 + e3 * k3 + e4 * k4 + e5 * k5 + e6 * k6 +
e7 * k7;
    VetX[i + 1] = xt;
    VetY[i + 1] = yt;
    EG[i + 1] = ErroGlobal;
    printf("%2d  %13.10f  %13.10f  %13.10f %13.10f\n", i, xt, yt,
ErroGlobal, abs(yt - g(xt)));
    getch();
  }
}

void main() {
  clrscr();
  DOPRI(1, 2, 50, 0.55);
}
```

Resultado da Letra c

```
 i   x              y              ErroGlobal    Erro
 0   1.0000000000   0.5500000119
 1   1.0199999809   0.4795092940   0.0000000002  0.0000000000
 2   1.0399999619   0.4100342095   0.0000000001  0.0000000000
 3   1.0599999428   0.3415705264   0.0000000002  0.0000000000
```

```
  4   1.0800000429   0.2741140127   0.0000000005  0.0000000000
  5   1.1000000238   0.2076606005   0.0000000004  0.0000000000
  6   1.1200000048   0.1422062516   0.0000000001  0.0000000000
  7   1.1399999857   0.0777470395   0.0000000000  0.0000000000
  8   1.1599999666   0.0142790973   0.0000000005  0.0000000000
  9   1.1799999475  -0.0482013710   0.0000000004  0.0000000000
 10   1.2000000477  -0.1096980721   0.0000000002  0.0000000000
 11   1.2200000286  -0.1702146530   0.0000000003  0.0000000000
 12   1.2400000095  -0.2297546715   0.0000000003  0.0000000000
 13   1.2599999905  -0.2883216143   0.0000000001  0.0000000000
 14   1.2799999714  -0.3459189236   0.0000000002  0.0000000000
 15   1.2999999523  -0.4025499523  -0.0000000000  0.0000000000
 16   1.3199999332  -0.4582180083   0.0000000001  0.0000000000
 17   1.3400000334  -0.5129262805   0.0000000002  0.0000000000
 18   1.3600000143  -0.5666779876   0.0000000003  0.0000000000
 19   1.3799999952  -0.6194761992   0.0000000002  0.0000000000
 20   1.3999999762  -0.6713239551   0.0000000003  0.0000000000
 21   1.4199999571  -0.7222242355   0.0000000001  0.0000000000
 22   1.4399999380  -0.7721799612   0.0000000002  0.0000000000
 23   1.4600000381  -0.8211939931   0.0000000002  0.0000000000
 24   1.4800000191  -0.8692691326   0.0000000002  0.0000000000
 25   1.5000000000  -0.9164081216   0.0000000003  0.0000000000
 26   1.5199999809  -0.9626137018   0.0000000002  0.0000000000
 27   1.5399999619  -1.0078884363   0.0000000002  0.0000000000
 28   1.5599999428  -1.0522350073  -0.0000000000  0.0000000000
 29   1.5800000429  -1.0956559181   0.0000000000  0.0000000000
 30   1.6000000238  -1.1381536722  -0.0000000000  0.0000000000
 31   1.6200000048  -1.1797306538   0.0000000001  0.0000000000
 32   1.6399999857  -1.2203892469   0.0000000002  0.0000000000
 33   1.6599999666  -1.2601318359   0.0000000001  0.0000000000
 34   1.6799999475  -1.2989606857   0.0000000003  0.0000000000
 35   1.6999999285  -1.3368780613   0.0000000001  0.0000000000
 36   1.7200000286  -1.3738862276   0.0000000001  0.0000000000
 37   1.7400000095  -1.4099873304   0.0000000000  0.0000000000
 38   1.7599999905  -1.4451833963   0.0000000001  0.0000000000
 39   1.7799999714  -1.4794765711   0.0000000001  0.0000000000
 40   1.7999999523  -1.5128690004   0.0000000000  0.0000000000
 41   1.8199999332  -1.5453624725   0.0000000002  0.0000000000
 42   1.8400000334  -1.5769591331   0.0000000001  0.0000000000
 43   1.8600000143  -1.6076607704   0.0000000001  0.0000000000
 44   1.8799999952  -1.6374692917   0.0000000001  0.0000000000
 45   1.8999999762  -1.6663866043   0.0000000001  0.0000000000
 46   1.9199999571  -1.6944144964   0.0000000001  0.0000000000
 47   1.9399999380  -1.7215546370   0.0000000002  0.0000000000
 48   1.9600000381  -1.7478088140   0.0000000000  0.0000000000
 49   1.9800000191  -1.7731788158   0.0000000001  0.0000000000
 50   2.0000000000  -1.7976661921   0.0000000001  0.0000000000
```