

Oct 26, 08 1:23

Crank-Nicolson.c

Page 1/3

```

/*
 * CRANK-NICOLSON ALGORITHM
 *
 * To approximate the solution of the parabolic partial-differential
 * equation subject to the boundary conditions
 *  $u(0,t) = u(1,t) = 0, 0 < t < T = \max t$ 
 * and the initial conditions
 *  $u(x,0) = F(x), 0 \leq x \leq 1$ :
 *
 * INPUT:  endpoint 1; maximum time T; constant ALPHA; integers m, N:
 *
 * OUTPUT: approximations W(I,J) to  $u(x(I),t(J))$  for each
 *         I = 1,..., m-1 and J = 1,..., N.
 */
/* Compile gcc -o cranknic Crank-Nicolson.c */

#include<stdio.h>
#include<math.h>
#define pi 4*atan(1)
#define true 1
#define false 0

double F(double X);
void INPUT(int *, double *, double *, double *, int *, int *);
void OUTPUT(double, double, int, double *, double);

main()
{
    double V[250], L[250], U[250], Z[250];
    double FT,FX,ALPHA,H,K,VV,T,X;
    int N,M,M1,M2,N1,FLAG,I1,I,J,OK;

    INPUT(&OK, &FX, &FT, &ALPHA, &N, &M);
    if (OK) {
        M1 = M - 1;
        M2 = M - 2;
        /* STEP 1 */
        H = FX / M;
        K = FT / N;
        /* VV is used for lambda */
        VV = ALPHA * ALPHA * K / ( H * H );
        /* set V(M) = 0 */
        V[M-1] = 0.0;
        /* STEP 2 */
        for (I=1; I<=M1; I++) V[I-1] = F( I * H );
        /* STEP 3 */
        /* STEPS 3 through 11 solve a tridiagonal linear system by
         LU algorithm */
        L[0] = 1.0 + VV;
        U[0] = -VV / ( 2.0 * L[0] );
        /* STEP 4 */
        for (I=2; I<=M2; I++) {
            L[I-1] = 1.0 + VV + VV * U[I-2] / 2.0;
            U[I-1] = -VV / ( 2.0 * L[I-1] );
        }
        /* STEP 5 */
        L[M1-1] = 1.0 + VV + 0.5 * VV * U[M2-1];
        /* STEP 6 */
        for (J=1; J<=N; J++) {
            /* STEP 7 */
            /* current t(j) */
            T = J * K;
            Z[0] = ((1.0-VV)*V[0]+VV*V[1])/2.0/L[0];
            /* STEP 8 */
            for (I=2; I<=M1; I++)
                Z[I-1] = ((1.0-VV)*V[I-1]+0.5*VV*(V[I]+V[I-2]+Z[I-2]))/L[I-1];
            /* STEP 9 */
            V[M1-1] = Z[M1-1];
            /* STEP 10 */

```

Oct 26, 08 1:23

Crank-Nicolson.c

Page 2/3

```

        for (I1=1; I1<=M2; I1++) {
            I = M2 - I1 + 1;
            V[I-1] = Z[I-1] - U[I-1] * V[I];
        }
        /* STEP 11 */
        OUTPUT(FT, X, M1, V, H);
    }
    /* STEP 12 */
    return 0;
}

/* Change F for a new problem */
double F(double X)
{
    double f;

    f = sin(pi * X);
    return f;
}

void INPUT(int *OK, double *FX, double *FT, double *ALPHA, int *N, int *M)
{
    int FLAG;
    char AA;

    printf("This is the Crank-Nicolson Method.\n");
    printf("Has the function F(x) been created immediately\n");
    printf("preceding the INPUT function? Answer Y or N.\n");
    scanf("%c", &AA);
    if ((AA == 'Y') || (AA == 'y')) {
        printf("The lefthand endpoint on the X-axis is 0.\n");
        *OK = false;
        while (!(*OK)) {
            printf("Input the righthand endpoint on the X-axis.\n");
            scanf("%lf", FX);
            if (*FX <= 0.0)
                printf("Must be positive number.\n");
            else *OK = true;
        }
        *OK = false;
        while (!(*OK)) {
            printf("Input the maximum value of the time variable T.\n");
            scanf("%lf", FT);
            if (*FT <= 0.0)
                printf("Must be positive number.\n");
            else *OK = true;
        }
        printf("Input the constant alpha.\n");
        scanf("%lf", ALPHA);
        *OK = false;
        while (!(*OK)) {
            printf("Input integer m = number of intervals on X-axis\n");
            printf("and N = number of time intervals - separated by a blank.\n");
            printf("Note that m must be 3 or larger.\n");
            scanf("%d %d", M, N);
            if ((*M <= 2) || (*N <= 0))
                printf("Numbers are not within correct range.\n");
            else *OK = true;
        }
    }
    else {
        printf("The program will end so that the function F can be created.\n");
        *OK = false;
    }
}

void OUTPUT(double FT, double X, int M1, double *V, double H)
{

```

Oct 26, 08 1:23

Crank-Nicolson.c

Page 3/3

```
int I, J, FLAG;
char NAME[30];
FILE *OUP;

printf("Choice of output method:\n");
printf("1. Output to screen\n");
printf("2. Output to text file\n");
printf("Please enter 1 or 2.\n");
scanf("%d", &FLAG);
if (FLAG == 2) {
    printf("Input the file name in the form -- name.ext\n");
    printf("for example: cranknic.txt\n");
    scanf("%s", NAME);
    OUP = fopen(NAME, "w");
}
else OUP = stdout;
fprintf(OUP, "CRANK-NICOLSON METHOD\n\n");
fprintf(OUP, " I      X(I)    W(X(I),%12.6e)\n", FT);
for (I=1; I<=M1; I++) {
    X = I * H;
    fprintf(OUP, "%3d %11.8f %13.8f\n", I, X, V[I-1]);
}
fclose(OUP);
}
```