

Oct 14, 08 10:23

NonLinDifFin.c

Page 1/4

```

/*
 * NONLINEAR FINITE-DIFFERENCE ALGORITHM
 *
 * To approximate the solution to the nonlinear boundary-value problem
 *  $Y'' = F(X,Y,Y')$ ,  $A \leq X \leq B$ ,  $Y(A) = \text{ALPHA}$ ,  $Y(B) = \text{BETA}$ :
 *
 * INPUT: Endpoints A,B; boundary conditions ALPHA, BETA;
 * integer N; tolerance TOL; maximum number of iterations M.
 *
 * OUTPUT: Approximations W(I) TO Y(X(I)) for each  $I=0,1,\dots,N+1$ 
 * or a message that the maximum number of iterations was
 * exceeded.
 */

#include<stdio.h>
#include<math.h>
#define true 1
#define false 0

main()
{
    int NP=100;
    double W[100], A[100], B[100], C[100], D[100], L[100], U[100], Z[100], V[100];
;
    double AA,BB,ALPHA,BETA,TOL,H,X,T,VMAX;
    int N1,NN,K,N,J,I,OK;
    FILE *OUP[1];

    void INPUT(int *, double *, double *, double *, double *, double *, int *, in
t *);
    void OUTPUT(FILE **);
    double F(double, double, double);
    double FY(double, double, double);
    double FYP(double, double, double);
    double absval(double);
    double sol(double);

    INPUT(&OK, &AA, &BB, &ALPHA, &BETA, &TOL, &NN, &N);
    if (OK) {
        OUTPUT(OUP);
        /* STEP 1 */
        N1 = N - 1;
        H = (BB - AA) / (N + 1);
        /* STEP 2 */
        for (I=1; I<=N; I++) W[I-1] = ALPHA+I*H*(BETA-ALPHA)/(BB-AA);
        /* STEP 3 */
        K = 1;
        /* STEP 4 */
        while ((K <= NN) && OK) {
            /* STEP 5 */
            X = AA + H;
            T = ( W[1] - ALPHA ) / ( 2.0 * H );
            A[0] = 2.0 + H * H * FY( X, W[0], T );
            B[0] = -1.0 + H * FYP( X, W[0], T ) / 2.0;
            D[0] = -( 2.0 * W[0] - W[1] - ALPHA + H * H * F( X, W[0], T ) );
            /* STEP 6 */
            for (I=2; I<=N1; I++) {
                X = AA + I * H;
                T = ( W[I] - W[I-2] ) / ( 2.0 * H );
                A[I-1] = 2.0 + H * H * FY( X, W[I-1], T );
                B[I-1] = -1.0 + H * FYP( X, W[I-1], T ) / 2.0;
                C[I-1] = -1.0 - H * FYP( X, W[I-1], T ) / 2.0;
                D[I-1] = -(2.0*W[I-1]-W[I]-W[I-2]+H*H*F( X, W[I-1], T ) );
            }
            /* STEP 7 */
            X = BB - H;
            T = ( BETA - W[N-2] ) / ( 2.0 * H );
            A[N-1] = 2.0 + H * H * FY( X, W[N-1], T );
            C[N-1] = -1.0 - H * FYP( X, W[N-1], T ) / 2.0;

```

Oct 14, 08 10:23

NonLinDifFin.c

Page 2/4

```

D[N-1] = -( 2.0 * W[N-1] - W[N-2] - BETA + H * H * F(X, W[N-1], T) );
/* STEP 8 */
/* STEPS 8 through 12 solve a tridiagonal linear system
using Algorithm 6.7 */
L[0] = A[0];
U[0] = B[0] / A[0];
Z[0] = D[0] / L[0];
/* STEP 9 */
for (I=2; I<=N1; I++) {
    L[I-1] = A[I-1] - C[I-1] * U[I-2];
    U[I-1] = B[I-1] / L[I-1];
    Z[I-1] = (D[I-1]-C[I-1]*Z[I-2])/L[I-1];
}
/* STEP 10 */
L[N-1] = A[N-1] - C[N-1] * U[N-2];
Z[N-1] = (D[N-1]-C[N-1]*Z[N-2])/L[N-1];
/* STEP 11 */
V[N-1] = Z[N-1];
VMAX = absval(V[N-1]);
W[N-1] = W[N-1] + V[N-1];
/* STEP 12 */
for (J=1; J<=N1; J++) {
    I = N - J;
    V[I-1] = Z[I-1] - U[I-1] * V[I];
    W[I-1] = W[I-1] + V[I-1];
    if (absval(V[I-1]) > VMAX) VMAX = absval(V[I-1]);
}
/* STEP 13 */
/* test for accuracy */
if (VMAX <= TOL) {
    I = 0;
    fprintf(*OUP, "%3d %13.8f %13.8f %13.8f\n", I, AA, ALPHA, sol(AA));
    for (I=1; I<=N; I++) {
        X = AA + I * H;
        fprintf(*OUP, "%3d %13.8f %13.8f %13.8f\n", I, X, W[I-1], sol(X));
    }
    I = N + 1;
    fprintf(*OUP, "%3d %13.8f %13.8f %13.8f\n", I, BB, BETA, sol(BB));
    OK = false;
}
else
    /* STEP 18 */
    K++;
}
/* STEP 19 */
if (K > NN)
    fprintf(*OUP, "No convergence in %d iterations\n", NN);
fclose(*OUP);
}
return 0;
}

/* Change functions F, FY and FYP for a new problem */
double F(double X, double Y, double Z)
{
    double f;

    f = (32+2*X*X*X-Y*Z)/8;
    return f;
}

/* FY REPRESENTS PARTIAL OF F WITH RESPECT TO Y */
double FY(double X, double Y, double Z)
{
    double fy;

    fy = -Z/8;
    return fy;
}

```

Oct 14, 08 10:23

NonLinDifFin.c

Page 3/4

```

/* FYP REPRESENTS PARTIAL OF F WITH RESPECT TO Y' */
double FYP(double X, double Y, double Z)
{
    double fyp;

    fyp = -Y/8;
    return fyp;
}

void INPUT(int *OK, double *AA, double *BB, double *ALPHA, double *BETA, double
*TOL, int *NN, int *N)
{
    double X;
    char AB;

    printf("This is the Nonlinear Finite-Difference Method.\n");
    *OK = false;
    printf("Have the functions F, FY (partial of F with respect to y).\n");
    printf("FYP (partial of F with respect to y') been created\n");
    printf("immediately preceding the INPUT procedure?\n");
    printf("Answer Y or N.\n");
    scanf("%c", &AB);
    *OK = false;
    if ((AB == 'Y') || (AB == 'y')) {
        *OK = false;
        while (!(*OK)) {
            printf("Input left and right endpoints separated by blank.\n");
            scanf("%lf%lf", AA, BB);
            if (*AA >= *BB)
                printf("Left endpoint must be less than right endpoint.\n");
            else *OK = true;
        }
        printf("Input Y( %.10e).\n", *AA);
        scanf("%lf", ALPHA);
        printf("Input Y( %.10e).\n", *BB);
        scanf("%lf", BETA);
        *OK = false;
        while (!(*OK)) {
            printf("Input an integer > 1 for the number of\n");
            printf("subintervals. Note that h = (b-a)/(n+1).\n");
            scanf("%d", N);
            if (*N <= 1) printf("Number must exceed 1.\n");
            else *OK = true;
        }
        *OK = false;
        while (!(*OK)) {
            printf("Input Tolerance.\n");
            scanf("%lf", TOL);
            if (*TOL <= 0.0) printf("Tolerance must be positive.\n");
            else *OK = true;
        }
        *OK = false;
        while (!(*OK)) {
            printf("Input maximum number of iterations.\n");
            scanf("%d", NN);
            if (*NN <= 0) printf("Must be positive integer.\n");
            else *OK = true;
        }
    }
    else printf("The program will end so that F, FP, FPY can be created.\n");
}

void OUTPUT(FILE **OUP)
{
    char NAME[30];
    int FLAG;

```

Oct 14, 08 10:23

NonLinDifFin.c

Page 4/4

```

    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 OUTPUT.DTA\n");
        scanf("%s", NAME);
        *OUP = fopen(NAME, "w");
    }
    else *OUP = stdout;
    fprintf(*OUP, "NONLINEAR FINITE-DIFFERENCE METHOD\n");
    fprintf(*OUP, " I      X(I)      W(I)      y(I)\n");
}

/* Absolute Value Function */
double absval(double val)
{
    if (val >= 0) return val;
    else return -val;
}

double sol(double x){
    double y=x*x+16./x;
    return y;
}

```