

Oct 14, 08 10:23

variational\_lin.c

Page 1/4

```

/*
 * PIECEWISE LINEAR RAYLEIGH-RITZ ALGORITHM 11.5
 *
 * To approximate the solution of the boundary-value problem
 *
 *      -D(P(X)Y')/DX + Q(X)Y = F(X), 0 <= X <= 1,
 *      Y(0) = Y(1) = 0,
 *
 * with a piecewise linear function:
 *
 * INPUT:  integer N; mesh points X(0) = 0 < X(1) < ...
 *         < X(N) < X(N+1) = 1
 *
 * OUTPUT: coefficients C(1),...,C(N) of the basis functions
 */

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

main()
{
    double X[27],H[27],A[25],B[25],ALPHA[25],BETA[25],ZETA[25],Z[25],C[25];
    double Q[6][26];
    double HQ,HC;
    int N1,J1,FN,N,J,OK;

    void INPUT(int *, double *, double *, int *, double *);
    void OUTPUT(int, double *, double *);
    double P(double);
    double QQ(double);
    double F(double);
    double SIMPSON(int, double, double);

    INPUT(&OK, X, H, &N, &HC);
    /* Step 1 is done within the input procedure */
    if (OK) {
        N1 = N - 1;
        /* STEP 3 */
        for (J=1; J<=N1; J++) {
            Q[0][J-1] = SIMPSON( 1, X[J], X[J+1] ) / (( H[J] ) * ( H[J] ));
            Q[1][J-1] = SIMPSON( 2, X[J-1], X[J] ) / (( H[J-1] ) * ( H[J-1] ));
            Q[2][J-1] = SIMPSON( 3, X[J], X[J+1] ) / (( H[J] ) * ( H[J] ));
            Q[3][J-1] = SIMPSON( 4, X[J-1], X[J] ) / (( H[J-1] ) * ( H[J-1] ));
            Q[4][J-1] = SIMPSON( 5, X[J-1], X[J] ) / H[J-1];
            Q[5][J-1] = SIMPSON( 6, X[J], X[J+1] ) / H[J];
        }
        Q[1][N-1] = SIMPSON( 2, X[N-1], X[N] ) / (( H[N-1] ) * ( H[N-1] ));
        Q[2][N-1] = SIMPSON( 3, X[N], X[N+1] ) / (( H[N] ) * ( H[N] ));
        Q[3][N-1] = SIMPSON( 4, X[N-1], X[N] ) / (( H[N-1] ) * ( H[N-1] ));
        Q[3][N] = SIMPSON( 4, X[N], X[N+1] ) / (( H[N] ) * ( H[N] ));
        Q[4][N-1] = SIMPSON( 5, X[N-1], X[N] ) / H[N-1];
        Q[5][N-1] = SIMPSON( 6, X[N], X[N+1] ) / H[N];
        /* STEP 4 */
        for (J=1; J<=N1; J++) {
            ALPHA[J-1] = Q[3][J-1]+Q[3][J]+Q[1][J-1]+Q[2][J-1];
            BETA[J-1] = Q[0][J-1]-Q[3][J];
            B[J-1] = Q[4][J-1]+Q[5][J-1];
        }
        /* STEP 5 */
        ALPHA[N-1] = Q[3][N-1]+Q[3][N]+Q[1][N-1]+Q[2][N-1];
        B[N-1] = Q[4][N-1]+Q[5][N-1];
        /* STEPS 6-10 solve a symmetric tridiagonal linear system using
         Algorithm 6.7. */
        /* STEP 6 */
        A[0] = ALPHA[0];
        ZETA[0] = BETA[0] / ALPHA[0];

```

Oct 14, 08 10:23

variational\_lin.c

Page 2/4

```

    Z[0] = B[0] / A[0];
    /* STEP 7 */
    for (J=2; J<=N1; J++) {
        A[J-1] = ALPHA[J-1] - BETA[J-2] * ZETA[J-2];
        ZETA[J-1] = BETA[J-1] / A[J-1];
        Z[J-1] = (B[J-1]-BETA[J-2]*Z[J-2])/A[J-1];
    }
    /* STEP 8 */
    if ( N > 1 ) {
        A[N-1] = ALPHA[N-1] - BETA[N-2] * ZETA[N-2];
        Z[N-1] = (B[N-1]-BETA[N-2]*Z[N-2])/A[N-1];
    }
    /* STEP 9 */
    C[N-1] = Z[N-1];
    for (J=1; J<=N1; J++) {
        J1 = N - J;
        C[J1-1] = Z[J1-1] - ZETA[J1-1] * C[J1];
    }
    OUTPUT(N, X, C);
}
/* STEP 10 */
return 0;

/* Change functions P, QQ and F for a new problem */
double P(double X)
{
    double p;

    p = 1.0;
    return p;
}

double QQ(double X)
{
    double qq;

    qq = pi * pi;
    return qq;
}

double F(double X)
{
    double f;

    f = 2.0 * pi * pi * sin(pi * X);
    return f;
}

void INPUT(int *OK, double *X, double *H, int *N, double *HC)
{
    int FLAG, J;
    char AA;
    char NAME[30];
    FILE *INP;

    printf("This is the Piecewise Linear Rayleigh-Ritz Method.\n");
    printf("This program requires functions P, QQ, F to be created and\n");
    printf("X(0), ..., X(N+1) to be supplied.\n");
    printf("Are the preparations complete? Answer Y or N.\n");
    scanf("%c",&AA);
    *OK = false;
    if ((AA == 'Y') || (AA == 'y')) {
        *OK = false;
        while (!(*OK)) {
            printf("Input integer N where X(0) = 0, X(N+1) = 1.\n");
            scanf("%d", N);
            if (*N < 0) printf("N must be greater than zero.\n");
            else *OK = true;
        }
    }
}

```

Oct 14, 08 10:23

variational\_lin.c

Page 3/4

```

    }
    X[0] = 0.0;
    X[*N+1] = 1.0;
    printf("Choice of method to input X(1), ..., X(N):\n");
    printf("1. Input from keyboard at the prompt\n");
    printf("2. Equally spaced nodes to be calculated\n");
    printf("3. Input from text file\n");
    printf("Please enter 1, 2, or 3.\n");
    scanf("%d", &FLAG);
    if (FLAG == 2) {
        *HC = 1.0 / (*N + 1.0);
        for (J=1; J<=*N; J++) {
            X[J] = J * *HC;
            H[J-1] = *HC;
        }
        H[*N] = *HC;
    }
    else {
        if (FLAG == 3) {
            printf("Has the input file been created? ");
            printf("Enter Y or N.\n");
            scanf("\n%c", &AA);
            if ((AA == 'Y') || (AA == 'y')) {
                printf("Enter the input file name using the format\n");
                printf("- drive:name.ext\n");
                printf("for example: A:DATA.DTA\n");
                scanf("%s", NAME);
                INP = fopen(NAME, "r");
                for (J=1; J<=*N; J++) fscanf(INP, "%lf", &X[J]);
                for (J=0; J<=*N; J++) H[J] = X[J+1] - X[J];
                fclose(INP);
            }
            else {
                printf("The program will end so that the input ");
                printf("file can be created.\n");
                *OK = false;
            }
        }
        else {
            for (J=1; J<=*N; J++) {
                printf("Input X(%d).\n", J);
                scanf("%lf", &X[J]);
                H[J-1] = X[J] - X[J-1];
            }
            H[*N] = X[*N+1] - X[*N];
        }
    }
}
else {
    printf("The program will end so that the functions\n");
    printf("can be created.\n");
    *OK = false;
}
}

double SIMPSON(int FN, double A, double B)
{
    double Z[5];
    double Y, H, simpson;
    int I;

    H = (B - A) / 4.0;
    for (I=0; I<=4; I++) {
        Y = A + I * H;
        switch (FN) {
            case 1:
                Z[I] = ( 4.0 - I ) * I * H * H * QQ( Y );
                break;
            case 2:

```

Oct 14, 08 10:23

variational\_lin.c

Page 4/4

```

                Z[I] = ( I * H ) * ( I * H ) * QQ( Y );
                break;
            case 3:
                Z[I] = ( H * ( 4.0 - I ) ) * ( H * ( 4.0 - I ) ) * QQ( Y );
                break;
            case 4:
                Z[I] = P( Y );
                break;
            case 5:
                Z[I] = I * H * F( Y );
                break;
            case 6:
                Z[I] = ( 4.0 - I ) * H * F( Y );
                break;
        }
    }
    simpson = ( Z[0] + Z[4] + 2.0 * Z[2] + 4.0 * ( Z[1] + Z[3] ) ) * H / 3.0;
    return simpson;
}

void OUTPUT(int N, double *X, double *C)
{
    char NAME[30];
    int FLAG, J;
    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 - drive:name.ext\n");
        printf("for example A:OUTPUT.DTA\n");
        scanf("%s", NAME);
        OUP = fopen(NAME, "w");
    }
    else OUP = stdout;
    fprintf(OUP, "PIECEWISE LINEAR RAYLEIGH-RITZ METHOD\n\n");
    fprintf(OUP, "I X(I-1) X(I) X(I+1) C(I)\n\n");
    for (J=1; J<=N; J++)
        fprintf(OUP, "%3d %11.8f %11.8f %11.8f %13.8f\n", J, X[J-1], X[J], X[J+1], C[J-1]);
    fclose(OUP);
}

```