



Oct 14, 08 10:23

variational\_spl.c

Page 3/9

```

    }
    /* Obtain coefficients for F, P, Q - NOTE - the fourth and fifth
    arguments in COEF are the derivatives of F, P, or Q evaluated
    at 0 and 1 respectively. */
    COEF(1, N2, N1, FPL, FPR, AF, BF, CF, DF, X, H);
    COEF(2, N2, N1, PPL, PPR, AP, BP, CP, DP, X, H);
    COEF(3, N2, N1, QPL, QPR, AQ, BQ, CQ, DQ, X, H);
    /* STEPS 5 - 9 are implemented in what follows */
    for (I=1; I<=N2; I++) {
        /* indices for limits of integration for A[I,I] and B[I] */
        J1 = MINO(I+2,N+2);
        J0 = MAXO(I-2,1);
        J2 = J1 - 1;
        /* integrate over each subinterval where phi(I) nonzero */
        for (JJ=J0; JJ<=J2; JJ++) {
            /* limits of integration for each call */
            XU = X[JJ];
            XL = X[JJ-1];
            /* coefficients of bases */
            K = INTE(I,JJ);
            A1 = DCO[I-1][K-1][0];
            B1 = DCO[I-1][K-1][1];
            C1 = DCO[I-1][K-1][2];
            D1 = 0.0;
            A2 = CO[I-1][K-1][0];
            B2 = CO[I-1][K-1][1];
            C2 = CO[I-1][K-1][2];
            D2 = CO[I-1][K-1][3];
            /* call subprogram for integrations */
            A[I-1][I-1] = A[I-1][I-1] +
                XINT(XU, XL, AP[JJ-1], BP[JJ-1], CP[JJ-1], DP[JJ-1],
                A1, B1, C1, D1, A1, B1, C1, D1) +
                XINT(XU, XL, AQ[JJ-1], BQ[JJ-1], CQ[JJ-1], DQ[JJ-1],
                A2, B2, C2, D2, A2, B2, C2, D2);
            A[I-1][N+2] = A[I-1][N+2] +
                XINT(XU, XL, AF[JJ-1], BF[JJ-1], CF[JJ-1], DF[JJ-1],
                A2, B2, C2, D2, 1.0, 0.0, 0.0, 0.0);
        }
        /* compute A[I,J] for J = I+1, ..., Min(I+3,N+2) */
        K3 = I+1;
        if (K3 <= N2) {
            K2 = MINO(I+3,N+2);
            for (J=K3; J<=K2; J++) {
                J0 = MAXO(J-2,1);
                for (JJ=J0; JJ<=J2; JJ++) {
                    XU = X[JJ];
                    XL = X[JJ-1];
                    K = INTE(I,JJ);
                    A1 = DCO[I-1][K-1][0];
                    B1 = DCO[I-1][K-1][1];
                    C1 = DCO[I-1][K-1][2];
                    D1 = 0.0;
                    A2 = CO[I-1][K-1][0];
                    B2 = CO[I-1][K-1][1];
                    C2 = CO[I-1][K-1][2];
                    D2 = CO[I-1][K-1][3];
                    K = INTE(J,JJ);
                    A3 = DCO[J-1][K-1][0];
                    B3 = DCO[J-1][K-1][1];
                    C3 = DCO[J-1][K-1][2];
                    D3 = 0.0;
                    A4 = CO[J-1][K-1][0];
                    B4 = CO[J-1][K-1][1];
                    C4 = CO[J-1][K-1][2];
                    D4 = CO[J-1][K-1][3];
                    A[I-1][J-1] = A[I-1][J-1] +
                        XINT(XU, XL, AP[JJ-1], BP[JJ-1],
                        CP[JJ-1], DP[JJ-1], A1, B1, C1, D1,

```

Oct 14, 08 10:23

variational\_spl.c

Page 4/9

```

                        A3, B3, C3, D3) +
                        XINT(XU, XL, AQ[JJ-1], BQ[JJ-1],
                        CQ[JJ-1], DQ[JJ-1], A2, B2, C2, D2,
                        A4, B4, C4, D4);
                }
                A[J-1][I-1] = A[I-1][J-1];
            }
        }
    }
    /* STEP 10 */
    for (I=1; I<=N1; I++) {
        for (J=I+1; J<=N2; J++) {
            CC = A[J-1][I-1]/A[I-1][I-1];
            for (K=I+1; K<=N3; K++) A[J-1][K-1] = A[J-1][K-1] - CC*A[I-1][K-1];
            A[J-1][I-1] = 0.0;
        }
        C[N2-1] = A[N2-1][N3-1]/A[N2-1][N2-1];
        for (I=1; I<=N1; I++) {
            J = N1-I+1;
            C[J-1] = A[J-1][N3-1];
            for (KK=J+1; KK<=N2; KK++) C[J-1] = C[J-1] - A[J-1][KK-1]*C[KK-1];
            C[J-1] = C[J-1]/A[J-1][J-1];
        }
        /* STEP 11 */
        /* output coefficients */
        fprintf(*OUP, "\nCoefficients: c(1),c(2),...,c(n)\n\n");
        for (I=1; I<=N2; I++) fprintf(*OUP, " %12.6e\n", C[I-1]);
        fprintf(*OUP, "\n");
        /* compute and output value of the approximation at the nodes */
        fprintf(*OUP, "The approximation evaluated at the nodes:\n\n");
        fprintf(*OUP, " Node      Value\n\n");
        for (I=1; I<=N2; I++) {
            S = 0.0;
            for (J=1; J<=N2; J++) {
                J0 = MAXO(J-2,1);
                J1 = MINO(J+2,N+2);
                SS = 0.0;
                if ((I < J0) || (I >= J1)) S = S + C[J-1]*SS;
                else {
                    K = INTE(J,I);
                    SS = ((CO[J-1][K-1][3]*X[I-1]+CO[J-1][K-1][2])*X[I-1]+
                        CO[J-1][K-1][1])*X[I-1]+CO[J-1][K-1][0];
                    S = S + C[J-1]*SS;
                }
            }
            fprintf(*OUP, "%12.8f%12.8f\n", X[I-1], S);
        }
        /* STEP 12 */
        fclose(*OUP);
    }
    return 0;
}

double F(double X)
{
    double fi;

    f = 2 * 4*atan(1) * 4*atan(1) * sin(4*atan(1) * (X));
    return f;
}

double P(double X)
{
    double pi;

    p = 1;
    return p;
}

```

Oct 14, 08 10:23

variational\_spl.c

Page 5/9

```

double Q(double X)
{
    double q;

    q = 4*atan(1) * 4*atan(1);
    return q;
}

void INPUT(int *OK, int *N, double *FPL, double *FPR, double *PPL, double *PPR,
double *QPL, double *QPR)
{
    double X;
    char AA;

    printf("This is the Cubic Spline Rayleigh-Ritz Method.\n");
    *OK = false;
    printf("Have functions P, Q and F been created?\n");
    printf("immediately preceding the INPUT procedure?\n");
    printf("Answer Y or N.\n");
    scanf("%c",&AA);
    if ((AA == 'Y') || (AA == 'y')) {
        while (!(*OK)) {
            printf("Input a positive integer n, where x(0) = 0, ");
            printf("..., x(n+1) = 1.\n");
            scanf("%d", N);
            if (*N <= 0) printf("Number must be a positive integer.\n");
            else *OK = true;
        }
        printf("Input the derivative of f at 0 and at 1\n");
        scanf("%lf%lf", FPL, FPR);
        printf("Input the derivative of p at 0 and at 1\n");
        scanf("%lf%lf", PPL, PPR);
        printf("Input the derivative of q at 0 and at 1\n");
        scanf("%lf%lf", QPL, QPR);
    }
    else printf("The program will end so that P, Q, F can be created.\n");
}

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

    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, "CUBIC SPLINE RAYLEIGH-RITZ METHOD\n");
}

int MINO(int I, int J)
{
    if (J < I) return J;
    else return I;
}

int MAXO(int I, int J)
{
    if (J > I) return J;
    else return I;
}

```

Oct 14, 08 10:23

variational\_spl.c

Page 6/9

```

}

int INTE(int J, int JJ)
{
    int inte;

    inte = JJ - J + 3;
    return inte;
}

void DPHICO(int I, int J, double *A, double *B, double *C, double H, int N)
{
    double EE,E, AA, BB, CC;
    int OK;

    *A = 0.0;
    *B = 0.0;
    *C = 0.0;
    E = I - 1.0;
    OK = true;
    if ( (J < I-2) || (J >= I+2) ) OK = false;
    if ( (I == 1) && (J < I) ) OK = false;
    if ( (I == 2) && (J < I-1) ) OK = false;
    if ( (I == N+1) && (J > N+1) ) OK = false;
    if ( (I == N+2) && (J >= N+2) ) OK = false;
    if (OK) {
        if (J <= I-2) {
            *A = ((E-4.0)*E+4.0)/(8.0*H);
            *B = (-E+2.0)/(4.0*H*H);
            *C = 1.0/(8.0*H*H*H);
            OK = false;
        }
        else {
            if (J > I) {
                *A = ((-E-4.0)*E-4.0)/(8.0*H);
                *B = (E+2.0)/(4.0*H*H);
                *C = -1.0/(8.0*H*H*H);
                OK = false;
            }
            else {
                if (J > I-1) {
                    *A = (3.0*E+4.0)*E/(8.0*H);
                    *B = (-3.0*E-2.0)/(4.0*H*H);
                    *C = 3.0/(8.0*H*H*H);
                    if ( (I != 1) && (I != N+1) ) OK = false;
                }
                else {
                    *A = (-3.0*E+4.0)*E/(8.0*H);
                    *B = (3.0*E-2.0)/(4.0*H*H);
                    *C = -3.0/(8.0*H*H*H);
                    if ( (I != 2) && (I != N+2) ) OK = false;
                }
            }
        }
    }
    if (OK) {
        if ( I <= 2 ) {
            AA = -1.0/(8.0*H);
            BB = 1.0/(4.0*H*H);
            CC = -1.0/(8.0*H*H*H);
            if ( I == 2 ) {
                *A = *A - AA;
                *B = *B - BB;
                *C = *C - CC;
            }
        }
        else {
            *A = *A - 4.0*AA;
            *B = *B - 4.0*BB;
            *C = *C - 4.0*CC;
        }
    }
}

```

Oct 14, 08 10:23

variational\_spl.c

Page 7/9

```

    }
    }
    else {
        EE = N+2.0;
        AA = ((EE-4.0)*EE+4.0)/(8.0*H);
        BB = (-EE+2.0)/(4.0*H*H);
        CC = 1.0/(8.0*H*H*H);
        if ( I == N+1 ) {
            *A = *A - AA;
            *B = *B - BB;
            *C = *C - CC;
        }
        else {
            *A = *A - 4.0*AA;
            *B = *B - 4.0*BB;
            *C = *C - 4.0*CC;
        }
    }
}

void PHICO(int I, int J, double *A, double *B, double *C, double *D, double H, int N)
{
    double EE,E, AA, BB, CC,DD;
    int OK;

    *A = 0.0;
    *B = 0.0;
    *C = 0.0;
    *D = 0.0;
    E = I - 1.0;
    OK = true;
    if ( (J < I-2) || (J >= I+2) ) OK = false;
    if ( (I == 1) && (J < I) ) OK = false;
    if ( (I == 2) && (J < I-1) ) OK = false;
    if ( (I == N+1) && (J > N+1) ) OK = false;
    if ( (I == N+2) && (J >= N+2) ) OK = false;
    if (OK) {
        if (J <= I-2) {
            *A = (((-E+6.0)*E-12.0)*E+8.0)/24.0;
            *B = ((E-4.0)*E+4.0)/(8.0*H);
            *C = (-E+2.0)/(8.0*H*H);
            *D = 1.0/(24.0*H*H*H);
            OK = false;
        }
        else {
            if (J > I) {
                *A = (((E+6.0)*E+12.0)*E+8.0)/24.0;
                *B = ((-E-4.0)*E-4.0)/(8.0*H);
                *C = (E+2.0)/(8.0*H*H);
                *D = -1.0/(24.0*H*H*H);
                OK = false;
            }
            else {
                if (J > I-1) {
                    *A = ((-3.0*E-6.0)*E*E+4.0)/24.0;
                    *B = (3.0*E+4.0)*E/(8.0*H);
                    *C = (-3.0*E-2.0)/(8.0*H*H);
                    *D = 1.0/(8.0*H*H*H);
                    if ( (I != 1) && (I != N+1) ) OK = false;
                }
                else {
                    *A = ((3.0*E-6.0)*E*E+4.0)/24.0;
                    *B = (-3.0*E+4.0)*E/(8.0*H);
                    *C = (3.0*E-2.0)/(8.0*H*H);
                    *D = -1.0/(8.0*H*H*H);
                    if ( (I != 2) && (I != N+2) ) OK = false;
                }
            }
        }
    }
}

```

Oct 14, 08 10:23

variational\_spl.c

Page 8/9

```

    }
    }
    if (OK) {
        if ( I <= 2 ) {
            AA = 1.0/24.0;
            BB = -1.0/(8.0*H);
            CC = 1.0/(8.0*H*H);
            DD = -1.0/(24.0*H*H*H);
            if ( I == 2 ) {
                *A = *A - AA;
                *B = *B - BB;
                *C = *C - CC;
                *D = *D - DD;
            }
            else {
                *A = *A - 4.0*AA;
                *B = *B - 4.0*BB;
                *C = *C - 4.0*CC;
                *D = *D - 4.0*DD;
            }
        }
        else {
            EE = N+2.0;
            AA = (((-EE+6.0)*EE-12.0)*EE+8.0)/24.0;
            BB = ((EE-4.0)*EE+4.0)/(8.0*H);
            CC = (-EE+2.0)/(8.0*H*H);
            DD = 1.0/(24.0*H*H*H);
            if ( I == N+1 ) {
                *A = *A - AA;
                *B = *B - BB;
                *C = *C - CC;
                *D = *D - DD;
            }
            else {
                *A = *A - 4.0*AA;
                *B = *B - 4.0*BB;
                *C = *C - 4.0*CC;
                *D = *D - 4.0*DD;
            }
        }
    }
}

double XINT(double XU, double XL, double A1, double B1, double C1, double D1, double A2, double B2, double C2, double D2, double A3, double B3, double C3, double D3)
{
    double C[20];
    double AA,BB,CC,DD,EE,FF,GG,XHIGH,XLOW,xint;
    int I;

    AA = A1*A2;
    BB = A1*B2+A2*B1;
    CC = A1*C2+B1*B2+C1*A2;
    DD = A1*D2+B1*C2+C1*B2+D1*A2;
    EE = B1*D2+C1*C2+D1*B2;
    FF = C1*D2+D1*C2;
    GG = D1*D2;
    C[9] = AA*A3;
    C[8] = (AA*B3+BB*A3)/2.0;
    C[7] = (AA*C3+BB*B3+CC*A3)/3.0;
    C[6] = (AA*D3+BB*C3+CC*B3+DD*A3)/4.0;
    C[5] = (BB*D3+CC*C3+DD*B3+EE*A3)/5.0;
    C[4] = (CC*D3+DD*C3+EE*B3+FF*A3)/6.0;
    C[3] = (DD*D3+EE*C3+FF*B3+GG*A3)/7.0;
    C[2] = (EE*D3+FF*C3+GG*B3)/8.0;
    C[1] = (FF*D3+GG*C3)/9.0;
    C[0] = (GG*D3)/10.0;
}

```

Oct 14, 08 10:23

variational\_spl.c

Page 9/9

```

    XHIGH = 0.0;
    XLOW = 0.0;
    for (I=1; I<=10; I++) {
        XHIGH = (XHIGH + C[I-1])*XU;
        XLOW = (XLOW + C[I-1])*XL;
    }
    xint = XHIGH - XLOW;
    return xint;
}

void COEF(int L, int N, int M, double FPO, double FPN, double *A, double *B, double *C, double *D, double *X, double H)
{
    double AA[25], BB[25], CC[25], DD[25], XA[25], XL[25], XU[25], XZ[25];
    int I, J;

    for (I=1; I<=N; I++)
        switch (L) {
            case 1:
                AA[I-1] = F(X[I-1]);
                break;
            case 2:
                AA[I-1] = P(X[I-1]);
                break;
            case 3:
                AA[I-1] = Q(X[I-1]);
                break;
        }
    XA[0] = 3.0*(AA[1]-AA[0])/H - 3.0*FPO;
    XA[N-1] = 3.0*FPN - 3.0*(AA[N-1]-AA[N-2])/H;
    XL[0] = 2.0*H;
    XU[0] = 0.5;
    XZ[0] = XA[0]/XL[0];
    for (I=2; I<=M; I++) {
        XA[I-1] = 3.0*(AA[I]-2.0*AA[I-1]+AA[I-2])/H;
        XL[I-1] = H*(4.0-XU[I-2]);
        XU[I-1] = H/XL[I-1];
        XZ[I-1] = (XA[I-1]-H*XZ[I-2])/XL[I-1];
    }
    XL[N-1] = H*(2.0-XU[N-2]);
    XZ[N-1] = (XA[N-1]-H*XZ[N-2])/XL[N-1];
    CC[N-1] = XZ[N-1];
    for (I=1; I<=M; I++) {
        J = N-I;
        CC[J-1] = XZ[J-1]-XU[J-1]*CC[J];
        BB[J-1] = (AA[J]-AA[J-1])/H -H*(CC[J]+2.0*CC[J-1])/3.0;
        DD[J-1] = (CC[J]-CC[J-1])/(3.0*H);
    }
    for (I=1; I<=M; I++) {
        A[I-1] = ((-DD[I-1]*X[I-1]+CC[I-1])*X[I-1]-BB[I-1])*X[I-1]+AA[I-1];
        B[I-1] = (3.0*DD[I-1]*X[I-1]-2.0*CC[I-1])*X[I-1]+BB[I-1];
        C[I-1] = CC[I-1]-3.0*DD[I-1]*X[I-1];
        D[I-1] = DD[I-1];
    }
}

```