restart;

natural_spline := proc(xx::list,y::list,S::name)
    if nops(xx)<>nops(y) then
        ERROR("Both lists must be the same size");
    fi;
    N:=nops(xx)-1;
    for I from 0 to N do
        X[I] := xx[I+1];
        A[I] := y[I+1];
    od;
    M := N - 1;

    STEP 1
    for I from 0 to M do
        H[I] := X[I+1] - X[I];
    od;

    STEP 2
    Use XA in place of ALPHA
    for I from 1 to M do
        XA[I] := 3.0*(A[I+1]*H[I-1]-A[I]*(X[I+1]-X[I-1])+A[I-1]*H[I])/
            (H[I]*H[I-1]);
    od;

    STEP 3
    STEPs 3, 4, 5 and part of 6 solve the tridiagonal system using Algorithm 6.7
    use XL, XU, XZ in place of L, MU, Z resp.
    XL[0] := 1;
    XU[0] := 0;
    XZ[0] := 0;

    STEP 4
    for I from 1 to M do
        XL[I] := 2*(X[I+1]-X[I-1])-H[I-1]*XU[I-1];
        XU[I] := H[I]/XL[I];
        XZ[I] := (XA[I]-H[I-1]*XZ[I-1])/XL[I];
    od;

    STEP 5
    XL[N] := 1;
    XZ[N] := 0;
    C[N] := XZ[N];

    STEP 6
    for I from 0 to M do
\[ J := M-I; \]
\[ C[J] := XZ[J]-XU[J]*C[J+1]; \]
\[ D[J] := (C[J+1] - C[J]) / (3.0 \times H[J]); \]
\[ od; \]

STEP 7

> printf(`\n\nThe coefficients of the natural cubic spline \(S[i]\) on the subintervals are:\n`);

> printf(`
\( a(i) \quad b(i) \quad c(i) \quad d(i) \n\)`);

> for I from 0 to M do
  > printf(` %13.8f %13.8f %13.8f %13.8f 
`, A[I], B[I], C[I], D[I]);
> od;

> printf(`
\n\n`);

> for I from 0 to M do
  > SS[I]:=A[I]+B[I]*(x-X[I])+C[I]*(x-X[I])^2+D[I]*(x-X[I])^3;
> od;

\[ L:=\{\text{seq}([x<X[I+1],SS[I]], I=0..M)\}; \]
\[ P:=\text{seq}([\text{op}(L[I]), I=1..M+1]); \]
\[ P := \text{subsop}(2*M+1=NULL,P); \]
\[ S:=\text{simplify}(\text{piecewise}(\text{op}(P))); \]
> end;

Warning, imaginary unit `I` used as a local variable in procedure natural_spline

natural_spline := proc(xx::list, y::list, S::name)
  if nops(xx) <> nops(y) then ERROR("Both lists must be the same size") end if
  N := nops(xx) - 1;
  for I from 0 to N do X[I] := xx[I + 1]; A[I] := y[I + 1] end do
  M := N - 1;
  for I from 0 to M do H[I] := X[I + 1] - X[I] end do
  for I to M do
  end do
  XL[0] := 1;
  XU[0] := 0;
  XZ[0] := 0;
  for I to M do
    XL[I] := 2*X[I+1] - 2*X[I] - H[I-1]*XU[I-1];
    XU[I] := (H[I])/(XL[I]);
    XZ[I] := (XA[I] - H[I-1]*XZ[I-1])/(XL[I])
  end do
  XL[N] := 1;
  XZ[N] := 0;
  C[N] := XZ[N];
  for I from 0 to M do
    J := M - I;
    C[J] := XZ[J] - XU[J]*C[J+1];
    D[J] := (C[J+1] - C[J])/(3.0*H[J]);
  end do
  printf(` The coefficients of the natural cubic spline \(S[i]\) on the subintervals are\);`;
end;
print(`          a(i)    b(i)    c(i)    d(i) `);
for I from 0 to M do printf(`%13.8f %13.8f %13.8f %13.8f ; ` A[I], B[I], C[I], D[I]) end do
print(``);
for I from 0 to M do SS[I] := A[I] + B[I]*(x - X[I]) + C[I]*x - X[I]^2 + D[I]*x - X[I]^3 end do
L := [seq([x < X[I] + 1], SS[I]], I = (0 .. M))];
P := [seq(op(L[I]), I = (1 .. M + 1))];
P := subsop(2*M + 1 = NULL, P);
S := simplify(piecewise(op(P)))
end proc

> natural_spline_dir:=proc()
  printf(`natural_spline returns the piecewise cubic spline.\n\n`);
  printf(`The output also includes a table of values\n`);
  printf(`for the cubic polynomials\n`);
  printf(`S[i](x)=a[i]+b[i](x-x[i])+c[i](x-x[i])^2+d[i](x-x[i])^3\n\n`);
  printf(`for i=0..n-1 making up the spline.\n\n`);
  printf(`The arguments for natural_spline are:\n`);
  printf(`(1)the list of x-values\n`);
  printf(`(2)the list of f(x) or y-values\n`);
  printf(`(3)the variable for returning the piecewise cubic spline\n\n`);
  printf(`If assigning the result to a variable, have the\n`);
  printf(`variable and the 3rd argument the same.\n`);
  printf(`If S is the variable for returning the spline\n`);
  printf(`(and has already been given a value,\n`);
  printf(`the procedure should be preceded by the statement:\n`);
  printf(`S:='S'\`);
end;

natural_spline_dir := proc()
  printf(`natural_spline returns the piecewise cubic spline. `);
  printf(`The output also includes a table of values `);
  printf(`for the cubic polynomials `);
  printf(`S[i](x)=a[i]+b[i](x-x[i])+c[i](x-x[i])^2+d[i](x-x[i])^3 `);
  printf(`for i=0..n-1 making up the spline. `);
  printf(`The arguments for natural_spline are: `);
  printf(`(1)the list of x-values `);
  printf(`(2)the list of f(x) or y-values `);
  printf(`(3)the variable for returning the piecewise cubic spline `);
  printf(`If assigning the result to a variable, have the `);
  printf(`variable and the 3rd argument the same. `);
  printf(`If S is the variable for returning the spline `);
  printf(`(and has already been given a value, `);
  printf(`the procedure should be preceded by the statement: `);
  printf(`S:='S' `);
end proc

>