MATH5004 LAB 6

FEM for 1D problems (continue)

Consider the following boundary value problem:

$$-\frac{d}{dx}\left(a(x)\frac{du}{dx}\right)+c(x)u(x)=f(x),\quad 0<x<1,$$
$$u(0)=u(1)=0.$$

In this LAB, we apply a MATLAB code based on finite element method with piecewise linear elements to above BVP, and compare the computed and exact solutions with the L2 and seminorm errors.

To compute a finite element approximation, a set of n equally spaced nodes is defined from 0.0 to 1.0, a set of piecewise linear basis functions is set up, with one basis function associated with each node, and then an integral form of the BVP is used, in which the differential equation is multiplied by each basis function, and integration by parts is used to simplify the integrand. A simple two point Gauss quadrature formula is used to estimate the resulting integrals over each interval.

```
function main()
% - uxx + u = x  for 0 < x < 1
% u(0) = u(1) = 0
% exact  = x - sinh(x) / sinh(1)
% exact' = 1 - cosh(x) / sinh(1)
  timestamp ( );
  fprintf ( 1, '\n' );
   n = 11;
  fprintf ( 1, '\n' );
  fprintf ( 1, 'FEM1D_BVP_LINEAR_TEST00\n' );
  fprintf ( 1, '  Solve -( A(x) U''(x) )'' + C(x) U(x) = F(x)\n' );
  fprintf ( 1, '  for 0 < x < 1, with U(0) = U(1) = 0.\n' );
  fprintf ( 1, '  A(X)  = 1.0\n' );
  fprintf ( 1, '  C(X)  = 1.0\n' );
  fprintf ( 1, '  F(X)  = X\n' );
  fprintf ( 1, '  U(X)  = X - SINH(X) / SINH(1)\n' );
  fprintf ( 1, '\n' );
  fprintf ( 1, '  Number of nodes = %d\n', n );
% Geometry definitions
  x_first = 0.0;
  x_last = 1.0;
  x = linspace ( x_first, x_last, n );
  x = x(:);
  u = fem1d_bvp_linear( n, @a, @c, @f, x );

  uexact = exact( x );

  fprintf ( 1, '\n' );
  fprintf ( 1, '     I    X         U         Uexact    Error\n' );
  fprintf ( 1, '\n' );

  for i = 1 : n
    fprintf ( 1, '  %4d  %8f  %8f  %8f  %8e\n', ...
      i, x(i), u(i), uexact(i), abs ( u(i) - uexact(i) ) );
  end

  e1 = l1_error ( n, x, u, @exact );
  e2 = l2_error_linear ( n, x, u, @exact );
  h1s = h1s_error_linear ( n, x, u, @exact_ux );
  mx = max_error_linear ( n, x, u, @exact );
  fprintf ( 1, '\n' );
  fprintf ( 1, '  l1 norm of error   = %g\n', e1 );
  fprintf ( 1, '  L2 norm of error   = %g\n', e2 );
  fprintf ( 1, '  Seminorm of error = %g\n', h1s );
  fprintf ( 1, '  Max norm of error = %g\n', mx );

  fprintf ( 1, '\n' );
  fprintf ( 1, '  Normal end of execution.\n' );
  fprintf ( 1, '\n' );
  timestamp ( );
end
```

In main() function, **fem1d_bvp_linear** (*n*, *@a*, *@c*, *@f*, *x*) is called. This function has five arguments, i.e.

n is the number of equally spaced nodes.
@a is the function which evaluates a(x);
@c is the function which evaluates c(x);
@f is the function which evaluates f(x).
x is the input vector of *n* nodes.

and its return is the output vector u of *n* values at the nodes, which can also be regarded as the finite element coefficients.

The functions a(x), c(x) and f(x) are defined in the following MATLAB functions.

```
function value = a( x )
%  Parameters:
%    Input, real X, the evaluation point.
%    Output, real VALUE, the value of A(X).
%
  value = 1.0;
 end
```

```
function value = c( x )
%  Parameters:
%    Input, real X, the evaluation point.
%    Output, real VALUE, the value of C(X).
  value = 1.0;
end
```

```
function value = f( x )
%  Parameters:
%    Input, real X, the evaluation point.
%    Output, real VALUE, the value of F(X).
%
  value = x;
end
```

We also compare the computed and exact solutions with the L2 and semi-norm errors. We then need the following functions.

```
function value = exact( x )
%  Parameters:
%    Input, real X, the evaluation point.
%    Output, real VALUE, the value of U(X).
%    exact  = x - sinh(x) / sinh(1)
  value = x - sinh ( x ) / sinh ( 1.0 );
end
```

```
function value = exact_ux( x )
%  Parameters:
%    Input, real X, the evaluation point.
%    Output, real VALUE, the value of dUdX(X).
%
  value = 1.0 - cosh ( x ) / sinh ( 1.0 );
end
```

```
function timestamp ( )
%% TIMESTAMP prints the current YMDHMS date as a timestamp.
%
  t = now;
  c = datevec ( t );
  s = datestr ( c, 0 );
  fprintf ( 1, '%s\n', s );
end
```

```
function h1s = h1s_error_linear ( n, x, u, exact_ux )
%  seminorm error of a finite element solution.
%  Parameters:
%    Input: integer N, the number of nodes.
%           real X(N), the mesh points.
%           real U(N), the finite element coefficients.
%           function EQ = EXACT_UX ( X ), returns the value of the exact
%                                         derivative at the point X.
%    Output: real H1S, the estimated seminorm of the error.
%  h1s = 0.0;
%  Quadrature definitions.

  quad_num = 2;
  abscissa(1)  = -0.577350269189625764509148780502;
  abscissa(2)  = +0.577350269189625764509148780502;
  weight(1)  = 1.0;
  weight(2)  = 1.0;

%  Integrate over each interval.
  for i = 1 : n - 1
    xl  = x(i);
    xr  = x(i+1);
    ul  = u(i);
    ur  = u(i+1);

     for q = 1 : quad_num
       xq = ( ( 1.0 - abscissa(q) ) * xl   ...
           + ( 1.0 + abscissa(q) ) * xr ) ...
```

```
          /   2.0;
       wq = weight(q) * ( xr - xl ) / 2.0;


%  The piecewise linear derivative is a constant in the interval.

      uxq = ( ur - ul ) / ( xr - xl );
      exq = exact_ux ( xq );
       h1s = h1s + wq * ( uxq - exq )^2;
     end
  end

   h1s = sqrt ( h1s );
end
```

```
function e1 = l1_error ( n, x, u, exact )
%  l1 error norm of a finite element solution.
%  Parameters:
%    Input, integer N, the number of nodes.
%           real X(N), the mesh points.
%           real U(N), the finite element coefficients.
%           function EQ = EXACT ( X ), returns the value of the exact
%                                   solution at the point X.
%    Output, real E1, the little l1 norm of the error.
%
  u = u(:);
  x = x(:);

  e1 = sum ( abs ( u(1:n) - exact ( x(1:n) ) ) );
  e1 = e1 / n;
end
```

```
function e2 = l2_error_linear ( n, x, u, exact )
% L2 error norm of a finite element solution.
%  Parameters:
%    Input, integer N, the number of nodes.
%           real X(N), the mesh points.
%           real U(N), the finite element coefficients.
%           function EQ = EXACT ( X ), returns the value of the exact
%                                  solution at the point X.
%    Output, real E2, the estimated L2 norm of the error.
  e2 = 0.0;
%  Quadrature definitions.
  quad_num = 2;
  abscissa(1)  = -0.577350269189625764509148780502;
  abscissa(2)  = +0.577350269189625764509148780502;
  weight(1)  = 1.0;
  weight(2)  = 1.0;
%  Integrate over each interval.
  for i = 1 : n - 1
    xl = x(i);
    xr = x(i+1);
    ul = u(i);
    ur = u(i+1);
    for q = 1 : quad_num
      xq = ( ( 1.0 - abscissa(q) ) * xl   ...
           + ( 1.0 + abscissa(q) ) * xr ) ...
           /   2.0;
      wq = weight(q) * ( xr - xl ) / 2.0;

%  Use the fact that U is a linear combination of piecewise linear.
      uq = ( ( xr - xq       ) * ul ...
           + (      xq - xl ) * ur ) ...
           / ( xr      - xl );
       eq = exact ( xq );
       e2 = e2 + wq * ( uq - eq )^2;
     end
  end
   e2 = sqrt ( e2 );
   return
end
```

```
function value = max_error_linear ( n, x, u, exact, value )
%  the max error norm of a finite element solution.
%  This function estimates the max norm of the error:
%  MAX_NORM = Integral ( A <= X <= B ) max ( abs ( U(X) - EXACT(X) ) ) dX
%  Parameters:
%    Input, integer N, the number of nodes.
%           real X(N), the mesh points.
%           real U(N), the finite element coefficients.
%           function EQ = EXACT ( X ), returns the value of the exact
%           solution at the point X.
%    Output, real VALUE, the estimated max norm of the error.

  quad_num = 8;
  value = 0.0;
  e_num = n - 1;

  for e = 1 : e_num
    l = e;
    xl = x(l);
    ul = u(l);
    r = e + 1;
    xr = x(r);
    ur = u(r);

    for q = 0 : quad_num - 1
       xq = ( ( quad_num - q ) * xl   ...
           + (            q ) * xr ) ...
         /   ( quad_num     );

%  Use the fact that U is a linear combination of piecewise linears.

      uq = ( ( xr - xq       ) * ul   ...
           + (      xq - xl ) * ur ) ...
           / ( xr      - xl );
      eq = exact ( xq );
       value = max ( value, abs ( uq - eq ) );

     end
   end

%  For completeness, check last node.
```

```
  xq = x(n);
  uq = u(n);
  eq = exact ( xq );
  value = max ( value, abs ( uq - eq ) );

%  Integral approximation requires multiplication by interval length.

  value = value * ( x(n) - x(1) );
end
```

MATLAB OUTPUT

```
12-Sep-2016 14:17:28

FEM1D_BVP_LINEAR_TEST
  MATLAB version
  Test the FEM1D_BVP_LINEAR library.


FEM1D_BVP_LINEAR_TEST01
  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)
  for 0 < x < 1, with U(0) = U(1) = 0.
  A1(X)  = 1.0
  C1(X)  = 0.0
  F1(X)  = X * ( X + 3 ) * exp ( X )
  U1(X)  = X * ( 1 - X ) * exp ( X )

  Number of nodes = 11
     I    X         U         Uexact    Error
     1  0.000000  0.000000  0.000000  2.220446e-16
     2  0.100000  0.099466  0.099465  1.334229e-07
     3  0.200000  0.195425  0.195424  2.475629e-07
     4  0.300000  0.283471  0.283470  3.394330e-07
     5  0.400000  0.358038  0.358038  4.056126e-07
     6  0.500000  0.412181  0.412180  4.421874e-07
     7  0.600000  0.437309  0.437309  4.446805e-07
     8  0.700000  0.422888  0.422888  4.079761e-07
     9  0.800000  0.356087  0.356087  3.262308e-07
    10  0.900000  0.221364  0.221364  1.927749e-07
    11  1.000000  0.000000  0.000000  0.000000e+00

  l1 norm of error  = 2.93988e-06
  L2 norm of error  = 0.00400665
  Seminorm of error = 0.138667
```

Assignment II

Question 1 (LAB-Wk6). Given coefficient functions

$$A(X) = 1.0,$$

$$C(X) = 2.0 * X,$$

external load

$$F(X) = -X * (2 * X * X - 3 * X - 3) * exp(X),$$

and exact solution

$$U(X) = X * (1 - X) * exp(X)$$

of the BVP:

$$-(A(x)\, U'(x))' + C(x)\, U(x) = F(x) \quad \text{for } 0 < x < 1,$$

$$\text{with } U(0) = U(1) = 0.$$

Implement MATLAB codes to obtain FE approximation, and determine L_1, L_2 and semi-norm errors.

Note: Assignments II (25%): Assignment Questions will be given weekly.

In this week, Questions 1 (LAB-WK6) is a part of Assignment II, please submit a document file with MATLAB code via Blackboard by the due date of Assignment I on Friday 23 October 2020