```
# include <cstdlib>
# include <iostream>
# include <iomanip>
# include <cmath>

using namespace std;

# include "fem1d_bvp_linear.hpp"

int main ( );
void test01 ( );
double a1 ( double x );
double c1 ( double x );
double exact1 ( double x );
double exact_ux1 ( double x );
double f1 ( double x );

void test02 ( );
double a2 ( double x );
double c2 ( double x );
double exact2 ( double x );
double exact_ux2 ( double x );
double f2 ( double x );

void test03 ( );
double a3 ( double x );
double c3 ( double x );
double exact3 ( double x );
double exact_ux3 ( double x );
double f3 ( double x );

void test04 ( );
double a4 ( double x );
double c4 ( double x );
double exact4 ( double x );
double exact_ux4 ( double x );
double f4 ( double x );

void test05 ( );
double a5 ( double x );
double c5 ( double x );
double exact5 ( double x );
double exact_ux5 ( double x );
double f5 ( double x );

void test06 ( );
double a6 ( double x );
double c6 ( double x );
double exact6 ( double x );
double exact_ux6 ( double x );
double f6 ( double x );

void test07 ( );
double a7 ( double x );
double c7 ( double x );
double exact7 ( double x );
double exact_ux7 ( double x );
```

```
double f7 ( double x );

void test08 ( );
double a8 ( double x );
double c8 ( double x );
double exact8 ( double x );
double exact_ux8 ( double x );
double f8 ( double x );

void test09 ( );
double a9 ( double x );
double c9 ( double x );
double exact9 ( double x );
double exact_ux9 ( double x );
double f9 ( double x );

//****************************************************************************80

int main ( )

//****************************************************************************80
//
//  Purpose:
//
//    MAIN is the main program for FEM1D_BVP_LINEAR_PRB.
//
//  Discussion:
//
//    FEM1D_BVP_LINEAR_PRB tests the FEM1D_BVP_LINEAR library.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
{
  timestamp ( );
  cout << "\n";
  cout << "FEM1D_BVP_LINEAR_PRB\n";
  cout << "  C++ version\n";
  cout << "  Test the FEM1D_BVP_LINEAR library.\n";

  test01 ( );
  test02 ( );
  test03 ( );
  test04 ( );
  test05 ( );
  test06 ( );
  test07 ( );
  test08 ( );
```

```
  test09 ( );
//
//  Terminate.
//
  cout << "\n";
  cout << "FEM1D_BVP_LINEAR_PRB\n";
  cout << "  Normal end of execution.\n";
  cout << "\n";
  timestamp ( );

  return 0;
}
//****************************************************************************80

void test01 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST01 carries out test case #1.
//
//  Discussion:
//
//    Use A1, C1, F1, EXACT1, EXACT_UX1.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;
```

```
  cout << "\n";
  cout << "TEST01\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A1(X)  = 1.0\n";
  cout << "  C1(X)  = 0.0\n";
  cout << "  F1(X)  = X * ( X + 3 ) * exp ( X )\n";
  cout << "  U1(X)  = X * ( 1 - X ) * exp ( X )\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a1, c1, f1, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact1 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact1 );
  e2 = l2_error_linear ( n, x, u, exact1 );
  h1s = h1s_error_linear ( n, x, u, exact_ux1 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a1 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A1 evaluates A function #1.
//
```

```
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A1, the value of A(X).
//
{
  double value;

  value = 1.0;

  return value;
}
//****************************************************************************80

double c1 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C1 evaluates C function #1.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C1, the value of C(X).
//
{
  double value;

  value = 0.0;
```

```
  return value;
}
//****************************************************************************80

double exact1 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT1 evaluates exact solution #1.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT1, the value of U(X).
//
{
  double value;

  value = x * ( 1.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

double exact_ux1 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX1 evaluates the derivative of exact solution #1.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
```

```
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX1, the value of dUdX(X).
//
{
  double value;

  value = ( 1.0 - x - x * x ) * exp ( x );

  return value;
}
//****************************************************************************80

double f1 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F1 evaluates right hand side function #1.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F1, the value of F(X).
//
{
  double value;

  value = x * ( x + 3.0 ) * exp ( x );

  return value;
}
//****************************************************************************80

void test02 ( )

//****************************************************************************80
//
```

```
//  Purpose:
//
//    TEST02 carries out test case #2.
//
//  Discussion:
//
//    Use A2, C2, F2, EXACT2, EXACT_UX2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST02\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A2(X)  = 1.0\n";
  cout << "  C2(X)  = 2.0\n";
  cout << "  F2(X)  = X * ( 5 - X ) * exp ( X )\n";
  cout << "  U2(X)  = X * ( 1 - X ) * exp ( X )\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );
```

```
  u = fem1d_bvp_linear ( n, a2, c2, f2, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact2 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact2 );
  e2 = l2_error_linear ( n, x, u, exact2 );
  h1s = h1s_error_linear ( n, x, u, exact_ux2 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a2 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A2 evaluates A function #2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A2, the value of A(X).
//
```

```
{
  double value;

  value = 1.0;

  return value;
}

//****************************************************************************80

double c2 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C2 evaluates C function #2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C2, the value of C(X).
//
{
  double value;

  value = 2.0;

  return value;
}
//****************************************************************************80

double exact2 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT2 evaluates exact solution #2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
```

```
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT2, the value of U(X).
//
{
  double value;

  value = x * ( 1.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

double exact_ux2 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX2 evaluates the derivative of exact solution #2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX2, the value of dUdX(X).
//
{
  double value;

  value = ( 1.0 - x - x * x ) * exp ( x );

  return value;
}
//****************************************************************************80
```

```
double f2 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F2 evaluates right hand side function #2.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F2, the value of F(X).
//
{
  double value;

  value = x * ( 5.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

void test03 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST03 carries out test case #3.
//
//  Discussion:
//
//    Use A3, C3, F3, EXACT3, EXACT_UX3.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
```

```
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST03\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A3(X)  = 1.0\n";
  cout << "  C3(X)  = 2.0 * X\n";
  cout << "  F3(X)  = - X * ( 2 * X * X - 3 * X - 3 ) * exp ( X )\n";
  cout << "  U3(X)  = X * ( 1 - X ) * exp ( X )\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a3, c3, f3, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact3 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact3 );
```

```
  e2 = l2_error_linear ( n, x, u, exact3 );
  h1s = h1s_error_linear ( n, x, u, exact_ux3 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a3 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A3 evaluates A function #3.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A3, the value of A(X).
//
{
  double value;

  value = 1.0;

  return value;
}

//****************************************************************************80

double c3 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C3 evaluates C function #3.
```

```
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C3, the value of C(X).
//
{
  double value;

  value = 2.0 * x;

  return value;
}
//****************************************************************************80

double exact3 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT3 evaluates exact solution #3.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT3, the value of U(X).
//
{
  double value;
```

```
  value = x * ( 1.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

double exact_ux3 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX3 evaluates the derivative of exact solution #3.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX3, the value of dUdX(X).
//
{
  double value;

  value = ( 1.0 - x - x * x ) * exp ( x );

  return value;
}
//****************************************************************************80

double f3 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F3 evaluates right hand side function #3.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
```

```
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F3, the value of F(X).
//
{
  double value;

  value = - x * ( 2.0 * x * x - 3.0 * x - 3.0 ) * exp ( x );

  return value;
}
//****************************************************************************80

void test04 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST04 carries out test case #4.
//
//  Discussion:
//
//    Use A4, C4, F4, EXACT4, EXACT_UX4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
```

```
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST04\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A4(X)  = 1.0 + X * X\n";
  cout << "  C4(X)  = 0.0\n";
  cout << "  F4(X)  = ( X + 3 X^2 + 5 X^3 + X^4 ) * exp ( X )\n";
  cout << "  U4(X)  = X * ( 1 - X ) * exp ( X )\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a4, c4, f4, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact4 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact4 );
  e2 = l2_error_linear ( n, x, u, exact4 );
  h1s = h1s_error_linear ( n, x, u, exact_ux4 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a4 ( double x )

//****************************************************************************80
```

```
//
//  Purpose:
//
//    A4 evaluates A function #4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A4, the value of A(X).
//
{
  double value;

  value = 1.0 + x * x;

  return value;
}
//****************************************************************************80

double c4 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C4 evaluates C function #4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C4, the value of C(X).
```

```
//
{
  double value;

  value = 0.0;

  return value;
}
//****************************************************************************80

double exact4 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT4 evaluates exact solution #4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT4, the value of U(X).
//
{
  double value;

  value = x * ( 1.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

double exact_ux4 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX4 evaluates the derivative of exact solution #4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
```

```
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX4, the value of dUdX(X).
//
{
  double value;

  value = ( 1.0 - x - x * x ) * exp ( x );

  return value;
}
//****************************************************************************80

double f4 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F4 evaluates right hand side function #4.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F4, the value of F(X).
//
{
  double value;

  value = ( x + 3.0 * x * x + 5.0 * x * x * x + x * x * x * x ) * exp ( x );

  return value;
}
//****************************************************************************80
```

```
void test05 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST05 carries out test case #5.
//
//  Discussion:
//
//    Use A5, C5, F5, EXACT5, EXACT_UX5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST05\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A5(X)  = 1.0 + X * X for X <= 1/3\n";
  cout << "         = 7/9 + X     for      1/3 < X\n";
  cout << "  C5(X)  = 0.0\n";
  cout << "  F5(X)  = ( X + 3 X^2 + 5 X^3 + X^4 ) * exp ( X )\n";
  cout << "                        for X <= 1/3\n";
  cout << "         = ( - 1 + 10/3 X + 43/9 X^2 + X^3 ) .* exp ( X )\n";
  cout << "                        for      1/3 <= X\n";
  cout << "  U5(X)  = X * ( 1 - X ) * exp ( X )\n";
```

```
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a5, c5, f5, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact5 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact5 );
  e2 = l2_error_linear ( n, x, u, exact5 );
  h1s = h1s_error_linear ( n, x, u, exact_ux5 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a5 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A5 evaluates A function #5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
```

```
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A5, the value of A(X).
//
{
  double value;

  if ( x <= 1.0 / 3.0 )
  {
    value = 1.0 + x * x;
  }
  else
  {
    value = x + 7.0 / 9.0;
  }

  return value;
}
//****************************************************************************80

double c5 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C5 evaluates C function #5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C5, the value of C(X).
//
{
  double value;

  value = 0.0;

  return value;
```

```
}
//****************************************************************************80

double exact5 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT5 evaluates exact solution #5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT5, the value of U(X).
//
{
  double value;

  value = x * ( 1.0 - x ) * exp ( x );

  return value;
}
//****************************************************************************80

double exact_ux5 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX5 evaluates the derivative of exact solution #5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
```

```
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX5, the value of dUdX(X).
//
{
  double value;

  value = ( 1.0 - x - x * x ) * exp ( x );

  return value;
}
//****************************************************************************80

double f5 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F5 evaluates right hand side function #5.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    20 August 2010
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F5, the value of F(X).
//
{
  double value;

  if ( x <= 1.0 / 3.0 )
  {
    value = ( x + 3.0 * x * x + 5.0 * x * x * x * x + x * x * x * x * x ) * exp ( x );
  }
  else
  {
    value = ( - 1.0 + ( 10.0 / 3.0 ) * x
      + ( 43.0 / 9.0 ) * x * x + x * x * x ) * exp ( x );
  }

  return value;
}
```

```
//**************************************************************************80

void test06 ( )

//**************************************************************************80
//
//  Purpose:
//
//    TEST06 does an error analysis.
//
//  Discussion:
//
//    Use A6, C6, F6, EXACT6, EXACT_UX6.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    19 February 2012
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST06\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A6(X)  = 1.0\n";
  cout << "  C6(X)  = 0.0\n";
  cout << "  F6(X)  = pi*pi*sin(pi*X)\n";
  cout << "  U6(X)  = sin(pi*x)\n";
  cout << "\n";
  cout << "  Compute L2 norm and seminorm of error for various N.\n";
  cout << "\n";
```

```
  cout << "     N        l1 error      L2 error      Seminorm error\n";
  cout << "\n";

  n = 11;
  for ( i = 0; i <= 4; i++ )
  {
//
//  Geometry definitions.
//
    x_first = 0.0;
    x_last = 1.0;
    x = r8vec_even_new ( n, x_first, x_last );

    u = fem1d_bvp_linear ( n, a6, c6, f6, x );

    e1 = l1_error ( n, x, u, exact6 );
    e2 = l2_error_linear ( n, x, u, exact6 );
    h1s = h1s_error_linear ( n, x, u, exact_ux6 );

    cout << "  " << setw(4) << n
         << "  " << setw(14) << e1
         << "  " << setw(14) << e2
         << "  " << setw(14) << h1s << "\n";

    n = 2 * ( n - 1 ) + 1;

    delete [] u;
    delete [] x;
  }

  return;
}
//****************************************************************************80

double a6 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A6 evaluates A function #6.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
```

```
//
//    Output, double A6, the value of A(X).
//
{
  double value;

  value = 1.0;

  return value;
}
//****************************************************************************80

double c6 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C6 evaluates C function #6.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C6, the value of C(X).
//
{
  double value;

  value = 0.0;

  return value;
}
//****************************************************************************80

double exact6 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT6 returns exact solution #6.
//
//  Licensing:
//
```

```
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    19 February 2012
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT6, the value of U(X).
//
{
  const double pi = 3.141592653589793;
  double value;

  value = sin ( pi * x );

  return value;
}
//****************************************************************************80

double exact_ux6 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX6 returns the derivative of exact solution #6.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    14 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX6, the value of U(X).
//
{
  const double pi = 3.141592653589793;
  double value;

  value = pi * cos ( pi * x );
```

```
  return value;
}
//****************************************************************************80

double f6 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F6 evaluates right hand side function #6.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    19 February 2012
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F6, the value of F(X).
//
{
  static double pi = 3.141592653589793;
  double value;

  value = pi * pi * sin ( pi * x );

  return value;
}
//****************************************************************************80

void test07 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST07 does an error analysis.
//
//  Discussion:
//
//    Use A7, C7, F7, EXACT7, EXACT_UX7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
```

```
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Eric Becker, Graham Carey, John Oden,
//    Finite Elements, An Introduction, Volume I,
//    Prentice-Hall, 1981, page 123-124,
//    ISBN: 0133170578,
//    LC: TA347.F5.B4.
//
{
  int i;
  int n;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST07\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  Becker/Carey/Oden example\n";
  cout << "\n";
  cout << "  Compute L2 norm and seminorm of error for various N.\n";
  cout << "\n";
  cout << "     N        l1 error      L2 error      Seminorm error\n";
  cout << "\n";

  n = 11;
  for ( i = 0; i <= 4; i++ )
  {
//
//  Geometry definitions.
//
    x_first = 0.0;
    x_last = 1.0;
    x = r8vec_even_new ( n, x_first, x_last );

    u = fem1d_bvp_linear ( n, a7, c7, f7, x );

    e1 = l1_error ( n, x, u, exact7 );
    e2 = l2_error_linear ( n, x, u, exact7 );
    h1s = h1s_error_linear ( n, x, u, exact_ux7 );

    cout << "  " << setw(4) << n
         << "  " << setw(14) << e1
```

```
          << "  " << setw(14) << e2
          << "  " << setw(14) << h1s << "\n";

    n = 2 * ( n - 1 ) + 1;

    delete [] u;
    delete [] x;
  }

  return;
}
//****************************************************************************80

double a7 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A7 evaluates A function #7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A7, the value of A(X).
//
{
  double alpha;
  double value;
  double x0;

  alpha = 30.0;
  x0 = 1.0 / 3.0;
  value = 1.0 / alpha + alpha * pow ( x - x0, 2 );

  return value;
}
//****************************************************************************80

double c7 ( double x )

//****************************************************************************80
//
//  Purpose:
```

```
//
//    C7 evaluates C function #7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C7, the value of C(X).
//
{
  double value;

  value = 0.0;

  return value;
}
//****************************************************************************80

double exact7 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT7 returns exact solution #7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT7, the value of U(X).
//
{
```

```
  double alpha;
  double value;
  double x0;

  alpha = 30.0;
  x0 = 1.0 / 3.0;
  value = ( 1.0 - x )
    * ( atan ( alpha * ( x - x0 ) ) + atan ( alpha * x0 ) );

  return value;
}
//****************************************************************************80

double exact_ux7 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX7 returns the derivative of exact solution #7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX7, the value of U(X).
//
{
  double alpha;
  double value;
  double x0;

  alpha = 30.0;
  x0 = 1.0 / 3.0;
  value = - atan ( alpha * ( x - x0 ) ) - atan ( alpha * x0 )
    + ( 1.0 - x ) * alpha / ( 1.0 + alpha * alpha * pow ( x - x0, 2 ) );


  return value;
}
//****************************************************************************80

double f7 ( double x )

//****************************************************************************80
```

```
//
//  Purpose:
//
//    F7 evaluates right hand side function #7.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    09 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F7, the value of F(X).
//
{
  double alpha;
  double value;
  double x0;

  alpha = 30.0;
  x0 = 1.0 / 3.0;
  value = 2.0 * ( 1.0 + alpha * ( x - x0 ) *
    ( atan ( alpha * ( x - x0 ) ) + atan ( alpha * x0 ) ) );

  return value;
}
//****************************************************************************80

void test08 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST08 carries out test case #8.
//
//  Discussion:
//
//    Use A8, C8, F8, EXACT8, EXACT_UX8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
```

```
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST08\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A8(X)  = 1.0\n";
  cout << "  C8(X)  = 0.0\n";
  cout << "  F8(X) = X * ( X + 3 ) * exp ( X ),   X <= 2/3\n";
  cout << "        = 2 * exp ( 2/3),                  2/3 < X\n";
  cout << "  U8(X) = X * ( 1 - X ) * exp ( X ),   X <= 2/3\n";
  cout << "        = X * ( 1 - X ) * exp ( 2/3 ),      2/3 < X\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a8, c8, f8, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact8 ( x[i] );
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
```

```
  }

  e1 = l1_error ( n, x, u, exact8 );
  e2 = l2_error_linear ( n, x, u, exact8 );
  h1s = h1s_error_linear ( n, x, u, exact_ux8 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a8 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A8 evaluates A function #8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A8, the value of A(X).
//
{
  double value;

  value = 1.0;

  return value;
}
//****************************************************************************80

double c8 ( double x )

//****************************************************************************80
//
//  Purpose:
```

```
//
//    C8 evaluates C function #8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C8, the value of C(X).
//
{
  double value;

  value = 0.0;

  return value;
}
//****************************************************************************80

double exact8 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT8 evaluates exact solution #8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT8, the value of U(X).
//
{
```

```
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = x * ( 1.0 - x ) * exp ( x );
  }
  else
  {
    value = x * ( 1.0 - x ) * exp ( 2.0 / 3.0 );
  }

  return value;
}
//****************************************************************************80

double exact_ux8 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX8 evaluates the derivative of exact solution #8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX8, the value of dUdX(X).
//
{
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = ( 1.0 - x - x * x ) * exp ( x );
  }
  else
  {
    value = ( 1.0 - 2.0 * x ) * exp ( 2.0 / 3.0 );
  }

  return value;
}
//****************************************************************************80
```

```
double f8 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F8 evaluates right hand side function #8.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F8, the value of F(X).
//
{
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = x * ( x + 3.0 ) * exp ( x );
  }
  else
  {
    value = 2.0 * exp ( 2.0 / 3.0 );
  }

  return value;
}
//****************************************************************************80

void test09 ( )

//****************************************************************************80
//
//  Purpose:
//
//    TEST09 carries out test case #9.
//
//  Discussion:
//
//    Use A9, C9, F9, EXACT9, EXACT_UX9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
```

```
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Reference:
//
//    Dianne O'Leary,
//    Scientific Computing with Case Studies,
//    SIAM, 2008,
//    ISBN13: 978-0-898716-66-5,
//    LC: QA401.O44.
//
{
  int i;
  int n = 11;
  double e1;
  double e2;
  double h1s;
  double *u;
  double uexact;
  double *x;
  double x_first;
  double x_last;

  cout << "\n";
  cout << "TEST09\n";
  cout << "  Solve -( A(x) U'(x) )' + C(x) U(x) = F(x)\n";
  cout << "  for 0 < x < 1, with U(0) = U(1) = 0.\n";
  cout << "  A9(X)  = 1.0\n";
  cout << "  C9(X)  = 0.0\n";
  cout << "  F9(X) = X * ( X + 3 ) * exp ( X ),   X <= 2/3\n";
  cout << "        = 2 * exp ( 2/3),                  2/3 < X\n";
  cout << "  U9(X) = X * ( 1 - X ) * exp ( X ),   X <= 2/3\n";
  cout << "        = X * ( 1 - X ) * exp ( 2/3 ),      2/3 < X\n";
  cout << "\n";
  cout << "  Number of nodes = " << n << "\n";
//
//  Geometry definitions.
//
  x_first = 0.0;
  x_last = 1.0;
  x = r8vec_even_new ( n, x_first, x_last );

  u = fem1d_bvp_linear ( n, a9, c9, f9, x );

  cout << "\n";
  cout << "     I    X         U         Uexact    Error\n";
  cout << "\n";

  for ( i = 0; i < n; i++ )
  {
    uexact = exact9 ( x[i] );
```

```
    cout << "  " << setw(4) << i
         << "  " << setw(8) << x[i]
         << "  " << setw(14) << u[i]
         << "  " << setw(14) << uexact
         << "  " << setw(14) << fabs ( u[i] - uexact ) << "\n";
  }

  e1 = l1_error ( n, x, u, exact9 );
  e2 = l2_error_linear ( n, x, u, exact9 );
  h1s = h1s_error_linear ( n, x, u, exact_ux9 );
  cout << "\n";
  cout << "  l1 norm of error  = " << e1 << "\n";
  cout << "  L2 norm of error  = " << e2 << "\n";
  cout << "  Seminorm of error = " << h1s << "\n";

  delete [] u;
  delete [] x;

  return;
}
//****************************************************************************80

double a9 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    A9 evaluates A function #9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double A8, the value of A(X).
//
{
  double value;

  value = 1.0;

  return value;
}
//****************************************************************************80
```

```
double c9 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    C9 evaluates C function #9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double C9, the value of C(X).
//
{
  double value;

  value = 0.0;

  return value;
}
//****************************************************************************80

double exact9 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT9 evaluates exact solution #9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
```

```
//    Input, double X, the evaluation point.
//
//    Output, double EXACT9, the value of U(X).
//
{
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = x * ( 1.0 - x ) * exp ( x );
  }
  else
  {
    value = x * ( 1.0 - x );
  }

  return value;
}
//****************************************************************************80

double exact_ux9 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    EXACT_UX9 evaluates the derivative of exact solution #9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double EXACT_UX9, the value of dUdX(X).
//
{
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = ( 1.0 - x - x * x ) * exp ( x );
  }
  else
  {
    value = ( 1.0 - 2.0 * x );
  }
```

```
  return value;
}
//****************************************************************************80

double f9 ( double x )

//****************************************************************************80
//
//  Purpose:
//
//    F9 evaluates right hand side function #9.
//
//  Licensing:
//
//    This code is distributed under the GNU LGPL license.
//
//  Modified:
//
//    16 June 2014
//
//  Author:
//
//    John Burkardt
//
//  Parameters:
//
//    Input, double X, the evaluation point.
//
//    Output, double F9, the value of F(X).
//
{
  double value;

  if ( x <= 2.0 / 3.0 )
  {
    value = x * ( x + 3.0 ) * exp ( x );
  }
  else
  {
    value = 2.0;
  }

  return value;
}
```