```
function fe2dx_n_fast_test ( )
%*****************************************************************************80
%
%% FE2DX_N_FAST_TEST tests the FE2DX_N_FAST code.
%
%  Discussion:
%
%    This function sets all parameter values and initial condition information
%    necessary to execute the "fast" version of the fe2dx_n algorithm.
%
%  Licensing:
%
%    Copyright (C) 2014 Marcus R. Garvie.
%    See 'mycopyright.txt' for details.
%
%  Modified:
%
%    28 April 2014
%
%  Author:
%
%    Marcus R. Garvie.
%
%  Reference:
%
%    Marcus R Garvie, John Burkardt, Jeff Morgan,
%    Simple Finite Element Methods for Approximating Predator-Prey Dynamics
%    in Two Dimensions using MATLAB,
%    Submitted to Bulletin of Mathematical Biology, 2014.
%
  timestamp ( );
  fprintf ( 1, '\n' );
  fprintf ( 1, 'FE2DX_N_FAST_TEST:\n' );
  fprintf ( 1, '  Test the FE2DX_N_FAST function\n' );
  fprintf ( 1, '  which applies Neumann boundary conditions as it\n' );
  fprintf ( 1, '  approximates a solution to a predator-prey system.\n' );
%
%  Set the parameters.
%
  alpha = 0.4;
  beta = 2.0;
  gamma = 0.6;
  delta = 1.0;
%
%  Use T=150.0 for normal run.
%  Use T=0.50 for a "quick" run that might take 15 minutes of computing.
%
  T = 0.50;
  delt = 1.0 / 384.0;
  t = tic;
  fe2dx_n_fast ( alpha, beta, gamma, delta, T, delt, @u0f, @v0f, @guf, @gvf );
  t = toc ( t );
  fprintf ( 1, '  Execution took %10.2g minutes \n', t / 60.0 );
%
%  Terminate.
```

```
%
  fprintf ( 1, '\n' );
  fprintf ( 1, 'FE2DX_N_FAST_TEST:\n' );
  fprintf ( 1, '  Normal end of execution.\n' );
  fprintf ( 1, '\n' );
  timestamp ( );
  return
end
function value = u0f ( x, y )
%*****************************************************************************80
%
%% U0F evaluates the initial condition for U.
%
%  Licensing:
%
%    Copyright (C) 2014 Marcus R. Garvie.
%    See 'mycopyright.txt' for details.
%
%  Modified:
%
%    26 April 2014
%
%  Author:
%
%    Marcus R. Garvie.
%
%  Parameters:
%
%    Input, real X, Y, a location in the region.
%
%    Output, real VALUE, the initial condition for U at (X,Y).
%
  value = 6.0 / 35.0 - 2.0E-07 * ( x - 0.1 * y - 225.0 ) * ( x - 0.1 * y - 675.0 );
  return
  return
end
function value = v0f ( x, y )
%*****************************************************************************80
%
%% V0F evaluates the initial condition for V.
%
%  Licensing:
%
%    Copyright (C) 2014 Marcus R. Garvie.
%    See 'mycopyright.txt' for details.
%
%  Modified:
%
%    26 April 2014
%
%  Author:
%
%    Marcus R. Garvie.
%
%  Parameters:
%
%    Input, real X, Y, a location in the region.
```

```
%
%    Output, real VALUE, the initial condition for V at (X,Y).
%
  value = 116.0 / 245.0 - 3.0E-05 * ( x - 450.0 ) - 1.2E-04 * ( y - 150.0 );
  return
end
function value = guf ( x, y, t )
%*****************************************************************************80
%
%% GUF evaluates the Neumann boundary condition for U.
%
%  Licensing:
%
%    Copyright (C) 2014 Marcus R. Garvie.
%    See 'mycopyright.txt' for details.
%
%  Modified:
%
%    28 April 2014
%
%  Author:
%
%    Marcus R. Garvie.
%
%  Parameters:
%
%    Input, real X, Y, a location on the boundary.
%
%    Input, real T, the time.
%
%    Output, real VALUE, the prescribed value of dU/dn at (X,Y,T).
%
  value = 0.0;
  return
end
function value = gvf ( x, y, t )
%*****************************************************************************80
%
%% GVF evaluates the Neumann boundary condition for V.
%
%  Licensing:
%
%    Copyright (C) 2014 Marcus R. Garvie.
%    See 'mycopyright.txt' for details.
%
%  Modified:
%
%    28 April 2014
%
%  Author:
%
%    Marcus R. Garvie.
%
%  Parameters:
%
%    Input, real X, Y, a location on the boundary.
%
```

```
%    Input, real T, the time.
%
%    Output, real VALUE, the prescribed value of dV/dn at (X,Y,T).
%
  value = 0.0;
  return
end
```

Published with MATLAB® R2013b