The PDE Toolbox

Minicourse

Mihai Dorobantu and Magnus Ringh
Computer Solutions Europe AB, Sweden*

This course is an introduction to the Partial Differential Equations (PDE) Toolbox for MATLAB. In order to follow it, a minimal knowledge of MATLAB is necessary. In the introduction you learn about the problem class solved by the toolbox. Three examples guide you through the process of defining, solving and visualizing the results of interesting engineering and scientific problems.

Introduction

The PDE Toolbox solves Partial Differential Equations (PDE) that arise in engineering or scientific applications. These problems are posed on domains with complicated geometries which can be discretized only on unstructured meshes. The manipulation of unstructured meshes and the discretization of differential equations on such meshes are by no means trivial.
With the PDE Toolbox you solve the equation (in the unknown u):

$$-\nabla \cdot (c\nabla u) + au = f,$$

on a bounded two-dimensional domain. The boundary conditions are prescribed values of the solution u and its derivatives. The toolbox also supports many generalizations of the basic equation: systems of equations, matrix valued c, time-dependent equations, nonlinear coefficients, and eigenvalue problems. These equations are general enough to model many physical phenomena, such as: structural mechanics, electrostatics, magnetostatics, AC power electromagnetics, DC conductive media, heat transfer, diffusion, wave propagation, etc.

 Contact: Magnus Ringh, Computer Solutions Europe AB, Björnnäsvägen 21, S-113 47, STOCKHOLM, Sweden, Tel: +46-8-15 30 22, FAX: +46-8-15 76 35, E-mail: magnus@comsol.se.

The logic of the PDE Toolbox follows the natural steps for solving equations:

- Draw the domain
- Set up the boundary conditions
- Define the equations
- Generate the mesh
- Discretize the equations and solve on the mesh
- Plot the results

The PDE Toolbox lets you navigate easily through these steps. You are at liberty to return any number of steps back to e.g., redefine a boundary condition, change the mesh, etc.

The Graphical User Interface (GUI) provides an advanced drawing tool and a menu-driven interface that assists you in all the steps. Using set operations you can build the domain out of simple pieces. Complex boundary conditions can be defined. Several application modes help you define the problem in the specific terms of application. A wide array of plotting facilities allows you to postprocess, visualize and extract interesting information from your solution.

At any point you can leave the GUI and enter the workspace with your results. There you have the full flexibility of the MATLAB computing environment and access to the functions forming the PDE Toolbox in order to further process your results. The effect is that you do not "see" the tedious and very technical details of unstructured mesh manipulation, but rather enjoy its benefits in a comfortable MATLAB environment.

Example 1: The Wrench

The aim of this exercise is to familiarize you with the GUI. It also demonstrates a specific application mode, a system of two equations, plots, and adaptive refinement.

You are supposed to compute the displacements and stresses in a simplified, 2-dimensional wrench that is fixed around a bolt and is loaded with a certain force in in the handle, in the same plane. The unknowns u and v are displacements, in the x and y directions. Also assume that all the stresses lie in the same plane as the wrench.

The wrench is modeled by an elliptic system and the boundary conditions specify displacements (Dirichlet) or surface tractions (Neumann).

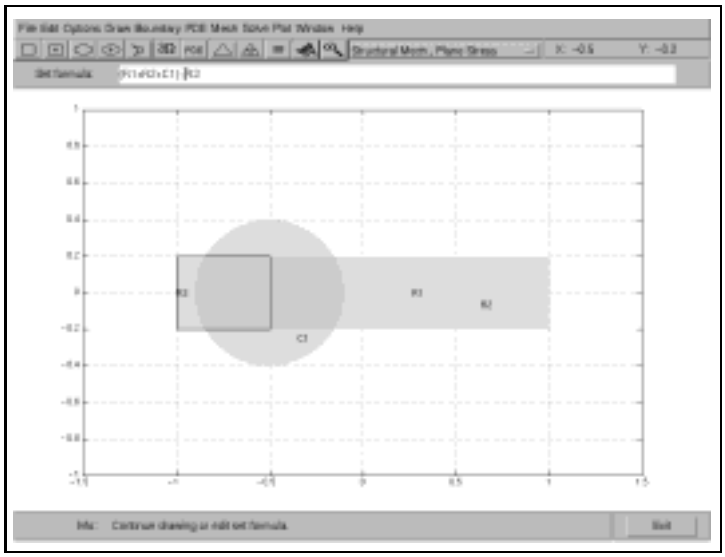
Start with some preliminaries.

- Start by invoking the GUI via the command `pdetool`. You are in the Generic Scalar applications mode (at the right end of the Toolbar). A click on this pop-up menu displays the ten applications mode out of which you choose Structural Mech., Plane Stress.
- From the **Options** menu, select the Grid and Snap menu items.

The wrench is represented by the union of a rectangle (body), a square (handle) and a circle (head). The "claws" are obtained by cutting out a square.

- Press the button marked with a rectangle (left-most on the Toolbar). Create the wrench body by clicking the corner (-0.5 , -0.2) and then dragging to (0.5 , 0.2). This object is automatically labeled R1.
- Create the handle by pressing the left-most button on the Toolbar (marked with a rectangle), clicking at (0.5 , -0.2), and dragging to (1 , 0.2). The handle is marked R2.
- To create the wrench head, press the button marked with a centered ellipse. Using the right mouse button, click at (-0.5 , 0), and then drag away until you create a circle that touches the height 0.4 (on the vertical axis). The circle is marked C1.
- Complete the drawing by creating a rectangle with the corners (-1 , -0.2) and (-0.5 , 0.2), marked R3.

Your have three rectangles, R1, R2, R3, and a circle C1. The PDE Toolbox figure looks like the following:



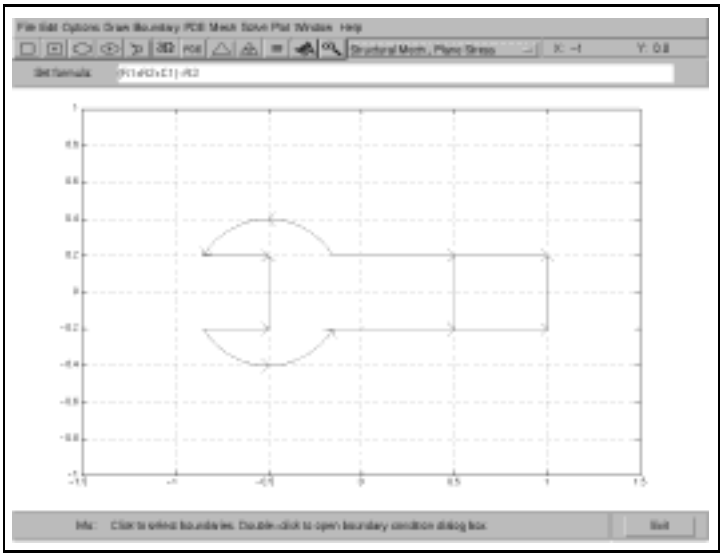
Next use set algebra to cut out the square R3 from the union of R1, R2 and C1.

- Edit the set formula to: (R1+R2+C1)-R3, followed by pressing the Enter key. The effect is cutting out the R3 block from the union of the first three blocks.

A click on the button marked $\partial\Omega$ sets you in the Boundary mode. The geometrical objects are replaced by their boundaries. Note that there are subdomain borders (drawn gray). The subdomain border separating the handle from the body is useful and you should not remove it. The other three subdomain borders (drawn in gray at the head) have no significance for the wrench and you should remove them.

- Click on one of the subdomain borders in the head. After its color changes to black, pressing the Shift key, click on the other two subdomain border segments. Then choose the Remove Subdomain Border option of the **Boundary** menu.

  You obtain the following PDE Toolbox window:



- Choose the Specify Boundary Conditions option from the **Boundary** menu. The Boundary Condition dialog box pops up. Select Neumann conditions and let all the coefficients be zero (their default value). Press the OK button to confirm.

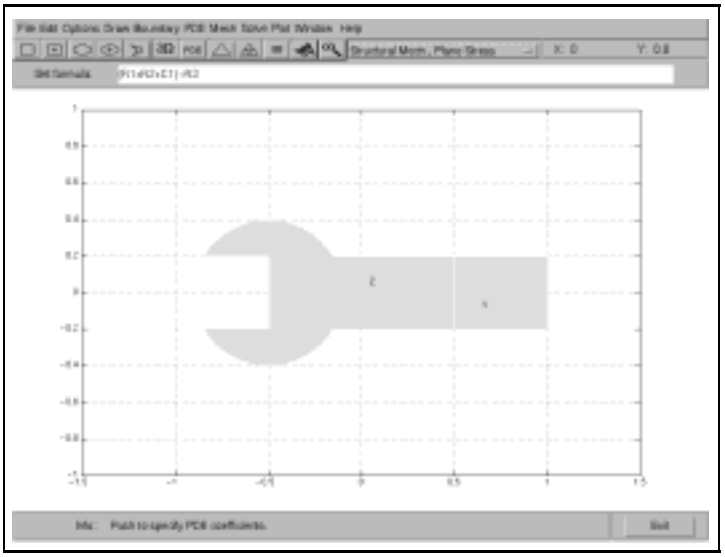
- Click on one of the three edges defining the wrench "claws", i.e., an edge of the rectangle R3 cut out using the set formula. Pressing the Shift key,

click on the other two edges and then choose again the Specify Boundary Conditions option from the **Boundary** menu. In the Boundary Condition dialog box, set Dirichlet boundary conditions with default values (no displacements), as that part of the boundary is fixed. Press OK to confirm.

Next you specify the equation coefficients: Young's modulus, Poisson ratio, loads etc.

- From the **PDE** menu, first choose PDE mode and then check the Show Subdomain Labels options.

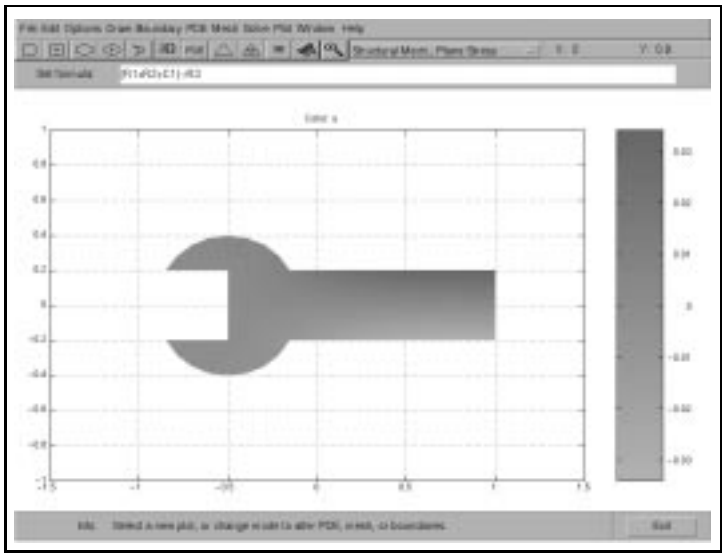
  Your PDE Toolbox figure will look like the following, perhaps with reversed subdomain numbering:



- Double-click in the handle (region 1 in our figure). The PDE Specifications dialog box appears. Edit the Ky field (Volume force, y-direction) and set it to -10. We are content with the default material qualities of the wrench.

- Press the = button to perform three tasks: to initialize the mesh, compute the solution and plot.

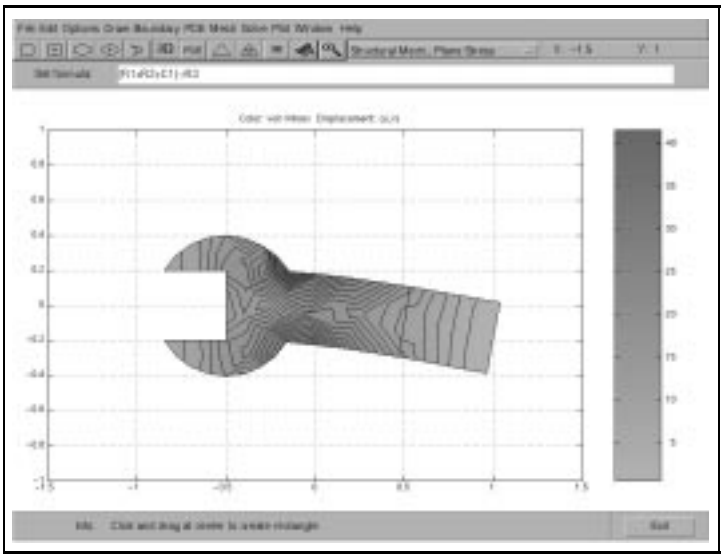
The colors represent the displacements in the x direction. This representation is not very relevant.



The information is graphically enhanced by the right plot style. In this case, the size of the displacements, in both directions, are easier to visualize in a deformed mesh plot.

- From the **Plot** menu choose the Parameters... option. In the Plot Selection window, choose the Deformed mesh Plot type to represent deformations in both directions, and the Color and Contour Plot types to represent some other interesting property of the solution. Choose the von Mises stresses from the Property pop-up menu corresponding to the Color Plot type.

  Pressing the Plot button produces the following figure:



This plot is much more informative than the previous one. Note that the displacements are enhanced, their real size is to be read from the previous plot. The shape of the level lines in the von Mises stress suggests that the resolution of the discretization is not fine enough.

- Refine the mesh by pressing the uniform refinement button (between the Δ and the $=$ buttons). Refine once again and then solve the equations using the $=$ button of the Toolbar.

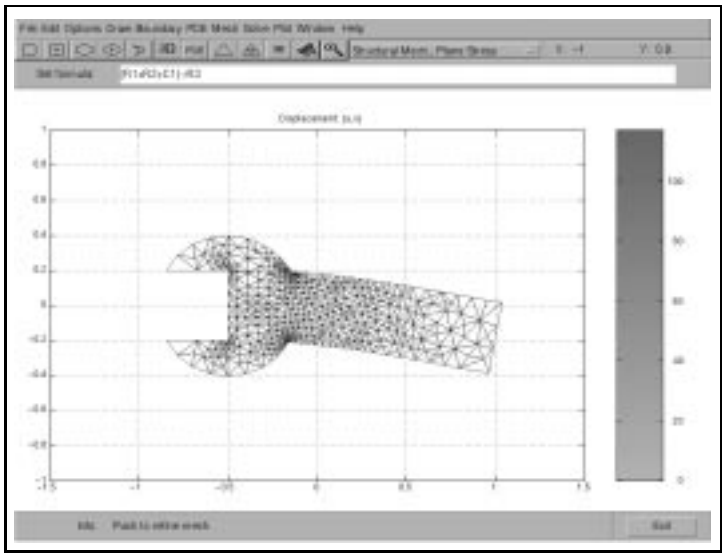
The results seem more trustworthy, but the computing time is rather large. Further refine-and-solve iterations seems an expensive strategy.

Now try an adaptive mesh generation.

- Initialize the mesh by pressing the Δ button on the Toolbar.
- In the **Solve** menu, choose the Parameters... option which opens the Solve Parameters dialog box. Check the Adaptive mode option and press the OK button.

- Solve again and plot the results pressing the = button in the Toolbar.

To visualize the mesh, choose Parameters... in the **Plot** menu and then turn off the Color and Contour Plot types in the Plot Selection dialog box. Press Plot to obtain the following figure:



Note that the mesh is strongly refined in those regions where the stresses are large. This yields the same accuracy as a fine-resolution mesh on the whole domain at a fraction of the computational costs. Indeed, there are 1,030 triangles, some at the eighth level of refinement. Uniform refinement to the same level produces 1,605,632 triangles!

The exact solution has singularities at two of the "corners" and thus point-wise convergence cannot be obtained. These singularities induce an error in the solution on the neighboring triangles which is detected by an error-estimator. The adaptive algorithm will refine only those triangles with large errors, producing very fine triangles around the corners. A solution within a given tolerance at all nodes can never be produced. The adaptive algorithm tries to localize the large errors in a microscopic region around the singularities. At all the other nodes, the solution comes within the prescribed tolerance.

A realistic wrench is implemented in the model `wrench`. You can run this model using the Open option in the **File** menu and then selecting `wrench.m`. Selecting Draw Mode shows how the realistic wrench is defined.

Example 2: The Magnet

The second example gives you an insight into nonlinear problems, more complex plots, and a more advanced usage of adaptivity.

A simple electro-magnet is a metal core (a ferro-magnetic material such as iron), surrounded by an electrical current (usually traveling through a copper coil).

Consider a long core and neglect the end effects. Thus you can restrict the study of the magnet to a plane section, governed by the equation for the magnetic potential A:

$$\nabla \cdot \left(\frac{1}{\mu} \nabla A \right) + J = 0,$$

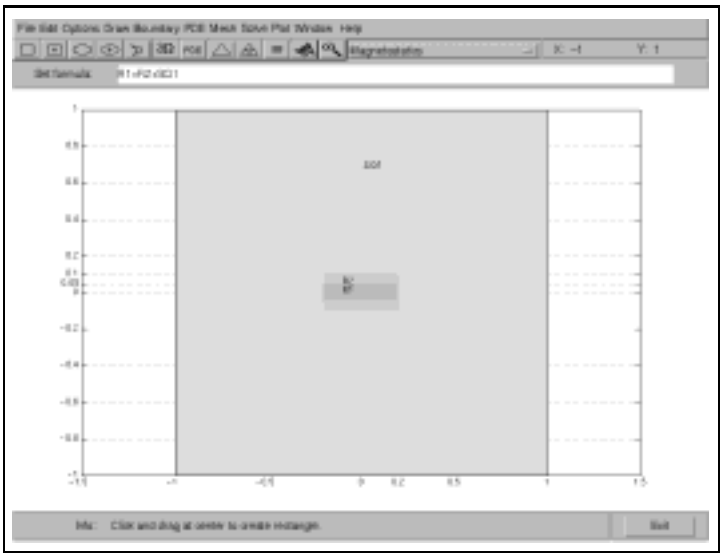
where J is the current density, and μ is the magnetic permeability (a nonlinear function of ∇A in the case of ferro-magnetic materials, otherwise a constant).

You are requested to find the magnetic potential level lines and check that reversing the current in the coil reverses the magnetic field.

The computational domain is a simple, unit square region. The core and the coil are small rectangles. These define subdomains, used to specify the data μ and J.

- From the **File** menu, choose the New option.
- Turn to the Magnetostatics application mode.
- In the **Options** menu, choose the Grid Spacing option. In the dialog box, turn off both Auto buttons. The electro-magnet is much smaller than the whole domain, so you need some finer detail to draw it. Edit the X-axis extra ticks by writing 0.2. Then edit the Y-axis extra ticks by adding 0.1 0.05. Press the Apply and Done buttons and remark the extra grid-lines on the drawing surface.

- Define two rectangles, R1 and R2, and a square SQ1 as in the following figure:

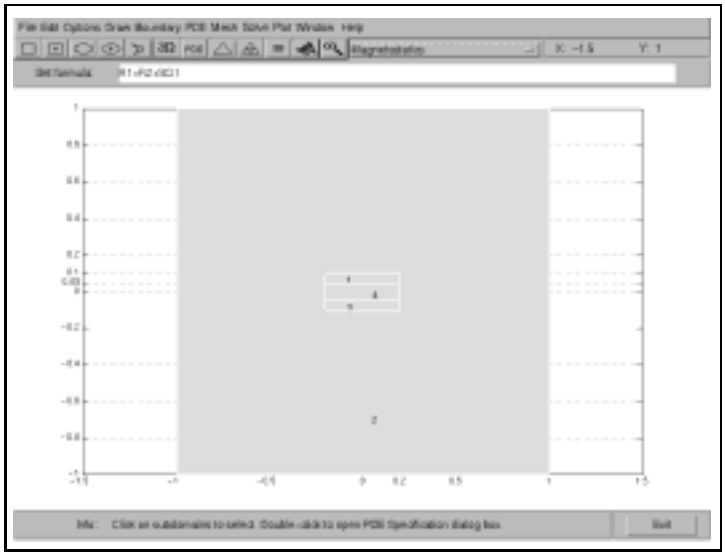


Use the button marked with a centered rectangle (second from the left on the Toolbar). Click (with the left button) in the origin (0 0) and drag towards (0.2 0.05) to create R1. Repeat the procedure dragging up to (0.2 0.1) to create R2. Finally, using the right mouse button, click in the origin and drag towards (1 1) to create SQ1.

- Dirichlet boundary conditions (i.e., the far away magnetic potential is zero) are default, so you can enter PDE Mode:

- From the **PDE** menu, first choose the PDE Mode option and then check the Show Subdomain Labels option.

Apart from a possible different numbering of the subdomains, the PDE Toolbox figure will look like the following:



Region 2 is air, region 4 is the ferro-magnetic core, and regions 1 and 3 are the parts of the coil.

- Double-click in each region to pop-up the corresponding PDE Specification dialog box. Set the current J to zero in air and core (regions 2 and 4), and +1 and -1 in the two parts of the coil (regions 1 and 3). The magnetic permeability μ (`mu`) is unit-size except for the core (region 4), where it has a slightly non-linear expression:

$$\mu = 200 + \frac{5000}{1 + 0.05|\nabla A|^2}$$

  In the PDE Toolbox notation, the components of ∇A are `ux` and `uy`. Thus you complete the `mu` field with:

```
200+5000./(1+0.05*(ux.^ 2+uy.^ 2))
```

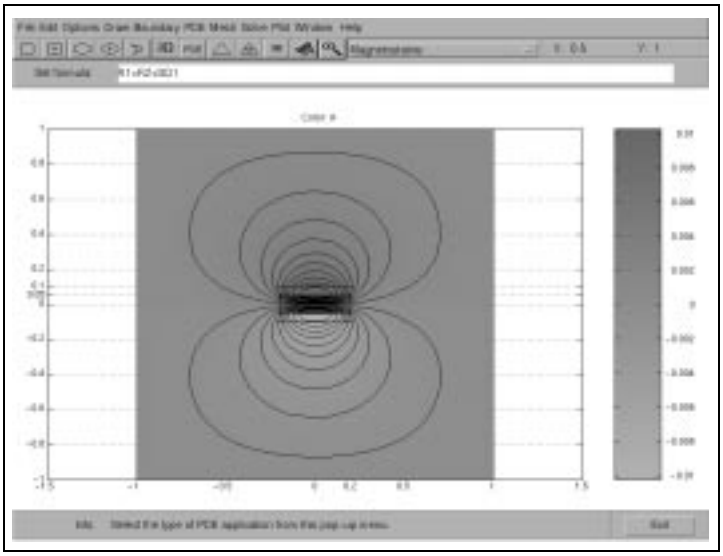
  The dialog boxes are closed pressing the OK button.

- In the **Plot** menu choose the Parameters... option and in the Plot Parameters dialog box check the contour Plot type. Press the button marked Done.

- Initialize the mesh by pressing the Δ button on the Toolbar.

- In the **Solve** menu choose the Parameters... option. In the Solve Parameters dialog box, turn on the adaptive and nonlinear modes. Set the maximum number of triangles to 750 and then press the OK button.

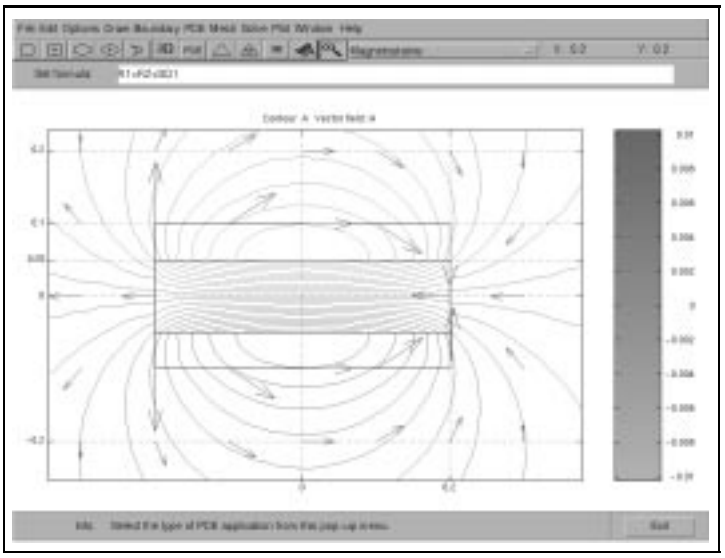
- Press the = button in the Toolbar to solve the equation and plot the results. Note that the level lines are not really smooth in the regions where the triangles are large. This is due to the linear interpolation in each triangle.
- Turn off the adaptive mode (Solve Parameters box from the Parameters... option of the **Solve** menu)
- Do a uniform refinement by pressing the appropriate button in the Toolbar (between the Δ and = buttons).
- Press the = button for another solve. A pleasing and well-known result is obtained:



Finally check that reversing the current in the coil switches the magnet's poles. You need an arrow plot of the magnetic field.

- Check the Arrows Plot type in the Plot Selection window and set the corresponding Property to magnetic field. Deselect the Color Plot type.
- Press the Plot button and note that the magnetic field is oriented from left to right in the core.
- Choose the PDE Mode option of the **PDE** menu. Double-click in regions 3 and 4 and reverse the signs of J.
- Solve by pressing the = button and note if the magnetic field is reversed.

- Press the Zoom button (marked by a magnifying glass) in the Toolbar and select a small region around the magnet.



For a more advanced modeling example, run `magnet`. It describes the magnetic field inside an electric motor.

Example 3: The Drums

The purpose of this exercise is to demonstrate the eigenvalue solver, and some possibilities for exporting data to the MATLAB environment (your workspace) and doing further post-processing.

The model of a drum is an oscillating surface. The tones (frequencies) emitted by the drum are the eigenvalues of the Laplace operator $\Delta u = \lambda u$. Note that both the displacements u and the eigenvalues λ are unknown. The surface has no displacement at the edge, so all drums have homogeneous Dirichlet boundary conditions. If two different drums should sound the same, they must have identical sets of eigenvalues λ, hence the question: Can one hear the shape of a drum? (posed in 1966 by Mark Kac and answered for the first time in 1992.) The answer is negative since there are examples of so-called isospectral regions, i.e., distinct surfaces such that the Laplace operators have precisely the same eigenvalues.

The purpose of this exercise is to study numerically two drums that sound the same. You compute the first six eigenvalues on a coarse, a fine, and an even finer mesh. Richardson extrapolation will give corrected values, which you compare.

Two such drums are found in the model files `drum1.m` and `drum2.m`. Both these files comes with the PDE Toolbox.

- From the **File** menu, select the New Option. Then type the command `drum1` at the MATLAB command prompt.
- Open the Plot Selection dialog box (Parameters... option in the **Plot** menu) and deselect the Contour Plot type.
- Copy the solution to your workspace. In the **Solve** menu, choose the Export Solution option. In the Export dialog box, edit the variable names `u l` to `u l11`. The first 6 eigenvalues are now contained in the vector `l11` in your workspace.
- Refine, solve and export the new solution under the names `u l12`.
- Refine once more, solve and export the new solution under the names `u l13`.
- From the command line, compare the eigenvalues corresponding to the three meshes. The numerical eigenvalue λ_h differs from the exact eigenvalue λ like

  $$\lambda_h \approx \lambda + c_1 h^{4/3} + c_2 h^2 + \cdots,$$

  where h is the mesh parameter describing the length of the typical triangles. Since we have three meshes with the parameters h, $h/2$, and $h/4$, and three corresponding numerical eigenvalues λ_h, $\lambda_{h/2}$, and $\lambda_{h/4}$, we can solve for the "unknowns" λ, c_1, and c_2. This procedure yields a linear system

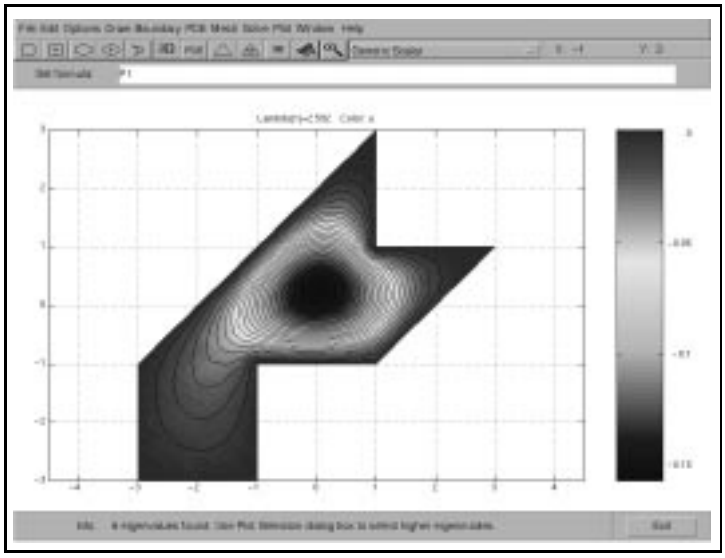
  $$\begin{bmatrix} 1 & 1 & 1 \\ 1 & (1/2)^{4/3} & (1/2)^2 \\ 1 & (1/4)^{4/3} & (1/4)^2 \end{bmatrix} \begin{bmatrix} \lambda \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \lambda_h \\ \lambda_{h/2} \\ \lambda_{h/4} \end{bmatrix}.$$

In your command window you should have:

```
>> l1new=[1 0 0]*inv(A)*[l11 l12 l13]';
>> [l11 l12 l13 l1new']
ans =
    2.5918    2.5552    2.5437    2.5379
    3.7361    3.6798    3.6632    3.6554
    5.3435    5.2250    5.1909    5.1757
    6.7310    6.5880    6.5509    6.5373
    7.5131    7.3210    7.2691    7.2483
    9.5732    9.3034    9.2341    9.2093
```

where `A` is the Richardson extrapolation matrix.

The solution corresponding to the first eigenvalue (fundamental note) is given in the plot below:

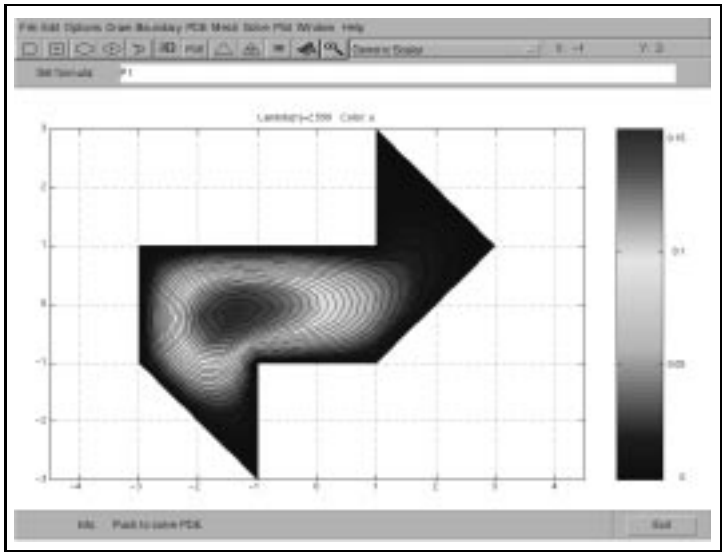


Now you draw a second drum, not similar to the first one.

- From the **File** menu, select the New Option. Then type the command `drum2` at the MATLAB command prompt.
- Deselect the contour lines plotting option.
- Export the solution under the names `u l21`.

- Refine, solve again, and export the new result as `u l22`.
- Finally, refine, solve and export the solution as `u l23`.

The first eigenmode of the second drum is plotted in the next figure:



Finally, compare the Richardson extrapolation results for the first six eigenvalues of the two domains and note that they agree to three-four decimal places.

```
>> l2new=[1 0 0]*inv(A)*[l21 l22 l23]';
>> [l1new ; l2new ]
ans =
    2.5379    3.6554    5.1757    6.5373    7.2483    9.2093
    2.5379    3.6555    5.1755    6.5372    7.2482    9.2093
```

The fact that the corrected values are closer to each other confirm both the dependence of the error on the mesh size and the fact that the two drums sound the same! In fact, the eigenvalues are known to 10 decimal places (T. Driscoll, Eigenmodes of isospectral drums, 1995). Compared with these, we have the following relative error in the direct computations on the three meshes and the improved, extrapolated results: