PDE2D GUI Example 1

Problem solved is eigenvalue problem

$$Q_{xx} + Q_{yy} + Q_{zz} - Q/\sqrt{x^2 + y^2 + z^2} \;=\; \lambda Q$$

in half of a torus, with $Q = 0$ on the curved surface of the torus and at one flat end of the torus, and $dQ/dn+Q = 0$, where dQ/dn is the normal derivative of Q, at the other flat end.

We ask to find the eigenvalue closest to -1.15.

The parametric equations for the torus used are the usual toroidal equations:

$$\begin{aligned} X &= (R0 + P3 * COS(P2)) * COS(P1) \\ Y &= (R0 + P3 * COS(P2)) * SIN(P1) \\ Z &= P3 * SIN(P2) \end{aligned}$$

where $P1$ is the toroidal angle, $P2$ is the poloidal angle and $P3$ is the radius (distance from centerline of torus).

Welcome to the PDE2D 9.4 GUI

Continue

The PDE2D GUI can be used to access the PDE2D collocation methods, which use cubic finite elements. These methods can handle 0D and 1D problems, and 2D and 3D problems in "a wide range of simple regions." For problems in complex 2D regions you should use the PDE2D Galerkin method, which is accessible only through the PDE2D Interactive Driver (type "pde2d [progname]").

If you want to see one of three prepared examples, select the example number below. If you ask to see an example, do not enter any data, just press the "Continue" buttons and look at the prepared input.

Show example number
(0 if none) 1

****Example 1. A 3D eigenvalue problem

Qxx + Qyy + Qzz - Q/sqrt(x^2+y^2+z^2) = lambda*Q

in half of a torus, with Q=0 on the curved surface of the torus and on one flat end, and dQ/dn+Q=0 (dQ/dn=normal derivative) on the other flat end. We will look for the eigenvalue closest to -1.15.

****Example 2. A 2D nonlinear steady-state problem

Uxx+Uyy = V

Vxx+Vyy = 1 + b*U^2

in an hour-glass shaped region. This is a fourth order elastic plate problem, with nonlinear loading, but reduced here to a system of two second order equations. On the left and right parabolic boundaries there are clamped boundary conditions, U=dU/dn=0, and on the flat top and bottom there are simply supported conditions, U=V=0.

****Example 3. A 1D time-dependent problem

Ut = V

Vt = -A*V + C^2*Uxx

with initial conditions U(x,0) = max(0,1-|x-5|), V(x,0)=0 and boundary conditions Ux=Vx=0 at x=0 and x=10. This is the damped wave equation Utt + A*Ut = C^2*Uxx, reduced to a system of two equations by defining V=Ut.

Continue

Begin Describing your PDE Problem

Dimension: 3d

Problem type: eigenvalue

Number of PDEs (NEQN): 1

Precision: double (Double strongly recommended)

Linear? (If unsure, assume nonlinear)

Continue

Fortran expressions

You will be asked to input "Fortran expressions" to define your PDE coefficients, boundary conditions (BC) and other parameters. Fortran expressions are almost the same as MATLAB expressions, which use +, -, *, / to represent the four arithmetic operators, and recognize functions such as sin,cos,log,exp,sqrt,abs,max... One important difference is that A to the B power is represented as A**B rather than A^B. Note also that Fortran is not case-sensitive.

These expressions may reference the independent variables T,X,Y,Z,P1,P2,P3 (depending on the dimension of the problem). Some may also reference the unknowns and their first and second spatial (not time) derivatives; these are written using the names you assign, for example if you name an unknown "PHI", refer to it and its x-derivatives as PHI,PHIx,PHIxx. Your expressions may all reference the "parameters" you define, and also the parameter "pi" (3.14159...).

Expressions which cannot be written on a single line may reference Fortran functions, which you may define in a separate file. You will be prompted for the file name at the end of this session.

Once you have clicked "Continue" on a screen, you cannot return to correct or modify your expressions. However, you will find it easy to make such changes in the Fortran program created by the PDE2D GUI, which is well-documented. There you will also be able to modify the default values set by the GUI for numerous other options, or add further Fortran code.

Define parameters (constants)

Parameter names are 1-6 alphanumeric characters beginning with a letter. Names beginning with I,J,K,L,M,N must be integers. Parameters are constants, not functions of T,X,... Parameter definitions may reference previously-defined parameters.

Continue

Parameter Name		Value (Constant or Fortran expression, up to 65 characters)
R0	=	5.0 ! (comment) Toroidal (major) radius
R1	=	4.0 ! Poloidal (minor) radius
	=	
	=	
	=	
	=	
	=	
	=	
	=	
	=	

Continue

Define region using parametric equations

You can solve PDEs in a region using this GUI only if you can describe it as the set of points
(X(p1,p2,p3) , Y(p1,p2,p3), Z(p1,p2,p3))
where A1 < P1 < B1, A2 < P2 < B2, A3 < P3 < B3, that is, if you can parameterize it (smoothly) using parameters with constants limits. For example, for a rectangle simply take X=P1, Y=P2, Z=P3, and for a cylinder you might take X=P1*cos(P2), Y=P1*sin(P2), Z=P3, where the parameters are cylindrical coordinates.

Define X(p1,p2,p3), Y(p1,p2,p3), Z(p1,p2,p3) below. Then if you assign, for example, a name "U" to an unknown, in future Fortran expressions reference its derivatives as Ux,Uy,Uz,Uxx,Uyy, Uzz,Uxy,Uxz,Uyz (Ux = dU/dX, etc.). You may also reference X,Y,Z,P1,P2,P3 and the derivatives of U with respect to the parameters as U1,U2,U3,U11,U22,U33,U12,U13,U23 (U1=dU/dP1, etc). In the boundary conditions you can also reference the normal derivative as Unorm.

Fortran expressions, up to 65 characters

X	(R0 + P3*COS(P2))*COS(P1) ! P1 = toroidal angle
Y	(R0 + P3*COS(P2))*SIN(P1) ! P2 = poloidal angle
Z	P3*SIN(P2) ! P3 = distance from centerline

Define P1, P2 and P3 grids. A1 < P1 < B1, A2 < P2 < B2, A3 < P3 < B3.

Continue

Default grid is uniform. A non-uniform grid can be specified in the Fortran program.

Number of P1-grid points (NP1GRID) 9

Fortran expressions, up to 65 characters

A1 0

B1 pi

Number of P2-grid points (NP2GRID) 9

A2 0

B2 2*pi

Number of P3-grid points (NP3GRID) 9

A3 0

B3 R1

Continue

Find (real) eigenvalue nearest EV0R

Fortran expression, up to 65 characters

EV0R -1.15

3D eigenvalue problem

Continue

F1 = lambda*RHO11*U_1
F2 = lambda*RHO22*U_2
:

where the RHOii are functions of X,Y,Z,P1,P2,P3 and the Fi are functions of X,Y,Z,P1,P2,P3 and the unknowns and their first and second derivatives.

Boundary conditions at P1=A1,B1 and P2=A2,B2 and P3=A3,B3 are:

G1 = 0
G2 = 0
:

where the Gi are functions of X,Y,Z,P1,P2,P3 and the unknowns and their first derivatives; they may also reference the components NORMx,NORMy,NORMz of the unit outward normal. The PDEs and boundary conditions must be linear and homogeneous, however.

Continue

Name the unknowns

Names are 1-3 alphanumeric characters each, beginning with a letter A-H or O-Z.

U_1 name Q

Use these names in all future input expressions.

For example, if U_1 is named "U", then refer to this unknown and its derivatives as U,Ux,Uxx... in future Fortran expressions.

U_2 name

U_3 name

U_4 name

U_5 name

U_6 name

U_7 name

U_8 name

Integrals

Define any functions whose integrals you want to compute. They may be functions of the unknowns and their first and second derivatives, plus X,Y,Z,P1,P2,P3,T.

Continue

Number of integrals desired (NINT) 0

Fortran expressions, up to 65 characters

Integral 1

Integral 2

Integral 3

Integral 4

Integral 5

Integral 6

Integral 7

Integral 8

Fortran expressions, up to 65 characters

Continue

RHO11	1
RHO22	
RHO33	
RHO44	
RHO55	
RHO66	
RHO77	
RHO88	

F1	Qxx+Qyy+Qzz - Q/sqrt(x**2+y**2+z**2)
F2	
F3	
F4	
F5	
F6	
F7	
F8	

Periodic BC?

For "no" BC, enter: None.

Continue

Define BCs at P1=A1

Fortran expressions, up to 65 characters

G1	-Qy+Q ! At flat end P1=0, dQ/dn can be written as Qnorm or -Qy
G2	
G3	
G4	
G5	
G6	
G7	
G8	

Define BCs at P1=B1

G1	Q ! Q=0 on other flat end, P1=pi
G2	
G3	
G4	
G5	
G6	
G7	
G8	

Periodic BC?

For "no" BC, enter: None.

Continue

Define BCs at P2=A2

Fortran expressions, up to 65 characters

G1

G2

G3

G4

G5

G6

G7

G8

Define BCs at P2=B2

G1

G2

G3

G4

G5

G6

G7

G8

Periodic BC?

For "no" BC, enter: None.

Continue

Define BCs at P3=A3

Fortran expressions, up to 65 characters

G1	none ! No boundary at P3=0, thus no BC
G2	
G3	
G4	
G5	
G6	
G7	
G8	

Define BCs at P3=B3

G1	Q ! Q=0 on curved surface, P3=R1
G2	
G3	
G4	
G5	
G6	
G7	
G8	

Continue

Plots of each unknown will be made at P1,P2 or P3=constant cross-sections.

Indicate below which cross-sections you want, how many, and the axes A1,A2 you want on the contour plots. For example, if your region is a cylinder (X=P1*cos(P2), Y=P1*sin(P2), Z=P3), and you request plots at P3=constant cross-sections, you will probably want to set A1=X, A2=Y (or A1=P1*cos(P2), A2=P1*sin(P2)). You may reference X,Y,Z,P1,P2,P3 in defining A1,A2. Surface plots will also be made, but with rectangular cross-sections, for example at P3=constant cross-sections, the axes will be P1 and P2.

Cross sections at: P1=constant

Number of cross-sections: 6

Fortran expressions, up to 65 characters

A1: sqrt(X**2+Y**2) ! R vs Z plots at cross-sections

A2: Z

Continue

Fortran function file

If you referenced your own Fortran functions in any of your Fortran expressions, you may now supply the name of a file where these functions are defined (or will be when RUNPDE2D is executed). Remember to type functions and their arguments as "double precision" if they are double precision in the main program, and fixed-format must be used, so start lines in column 7 or later.

File name (if any)

Example Fortran function. The function below could be called (it would be referenced simply as "GB(P2,U,Unorm)") to specify boundary conditions on the perimeter (P1=B1) of a disk, if polar coordinates (X=p1*cos(p2), Y=p1*sin(p2)) are used for a 2D problem:

```
      function GB(P2,U,Unorm)
      double precision GB,P2,U,Unorm,pi
C234567 (parameters not available within functions)
      pi = 3.141592654d0
      if (P2 .le. pi/2) then
C                 for P2  < pi/2, specify BC U=1
        GB = U-1
      else
C                 for P2  > pi/2, specify BC Unorm=0
        GB = Unorm
      endif
      return
      end
```

Finish

You have finished defining your PDE problem

The PDE2D Interactive Driver will now be run using as input the "pde2d.in" file created by the PDE2D GUI, and a Fortran program will be created, which can be executed using RUNPDE2D. This program contains comments which make it easy for you to make minor corrections and modifications to your model--the input you provided during the GUI session will be clearly marked by comments "INPUT FROM GUI". There you will also be able to select many options not documented in the GUI, for example, it will be easy to modify this program to request nonuniform grids, to compute all eigenvalues of an eigenvalue problem, to compute boundary integrals of functions of the solution and its derivatives, or to add off-diagonal terms to your "C" matrix (for time-dependent problems) or "RHO" matrix (for eigenvalue problems).

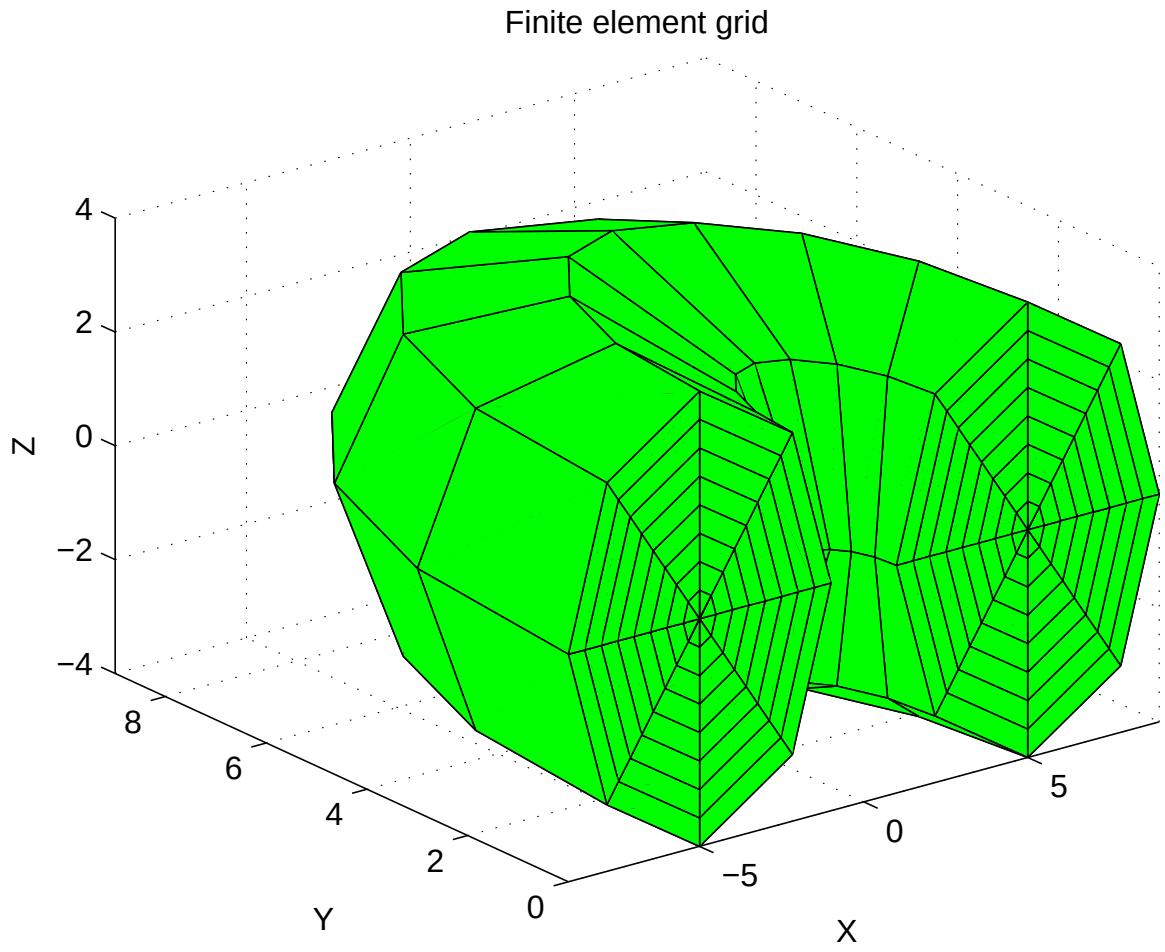
You do NOT need to be a Fortran programmer to make these modifications, and you do not need to work through another GUI session unless you have to change one of the options on the second GUI page. However, if you want to solve problems in complex 2D regions, or if you have more than eight PDEs, you will need to construct your program using the PDE2D Interactive Driver.

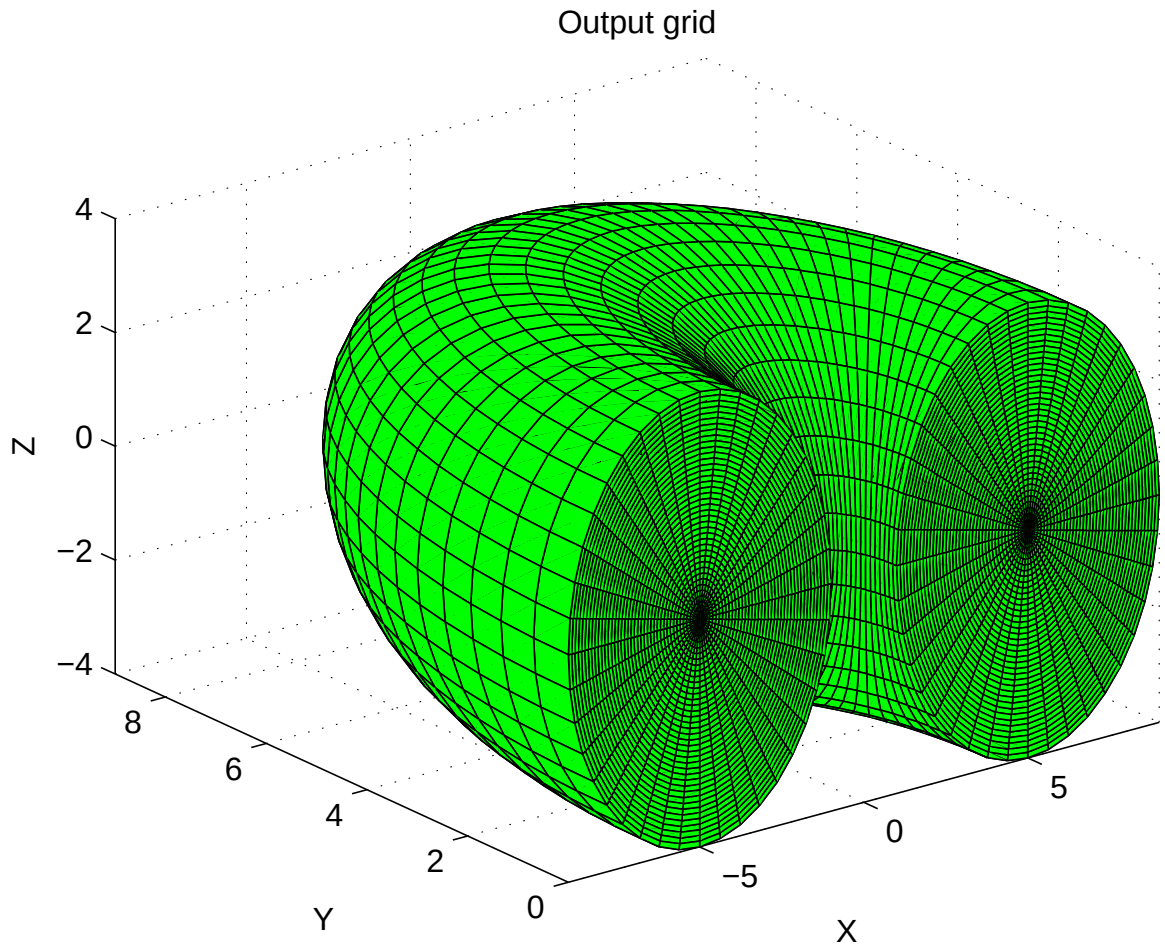
When you execute RUNPDE2D, plots of all unknowns will be made, but you can easily modify the program to postprocess the solution in many other ways, including using MATLAB plots (see subroutine POSTPR).

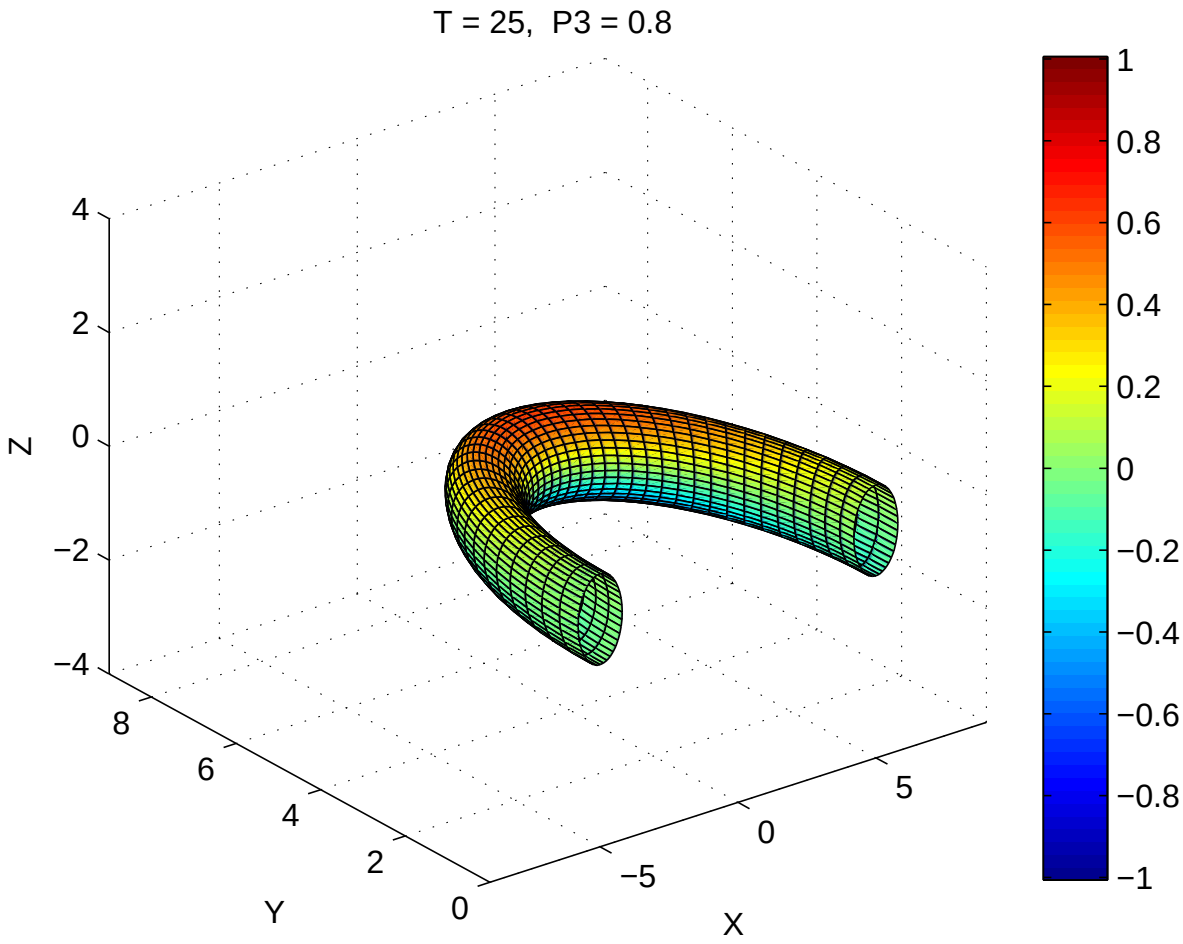
More information about PDE2D is contained in the book, "The Numerical Solution of Ordinary and Partial Differential Equations, second edition," Granville Sewell, John Wiley & Sons, 2005.

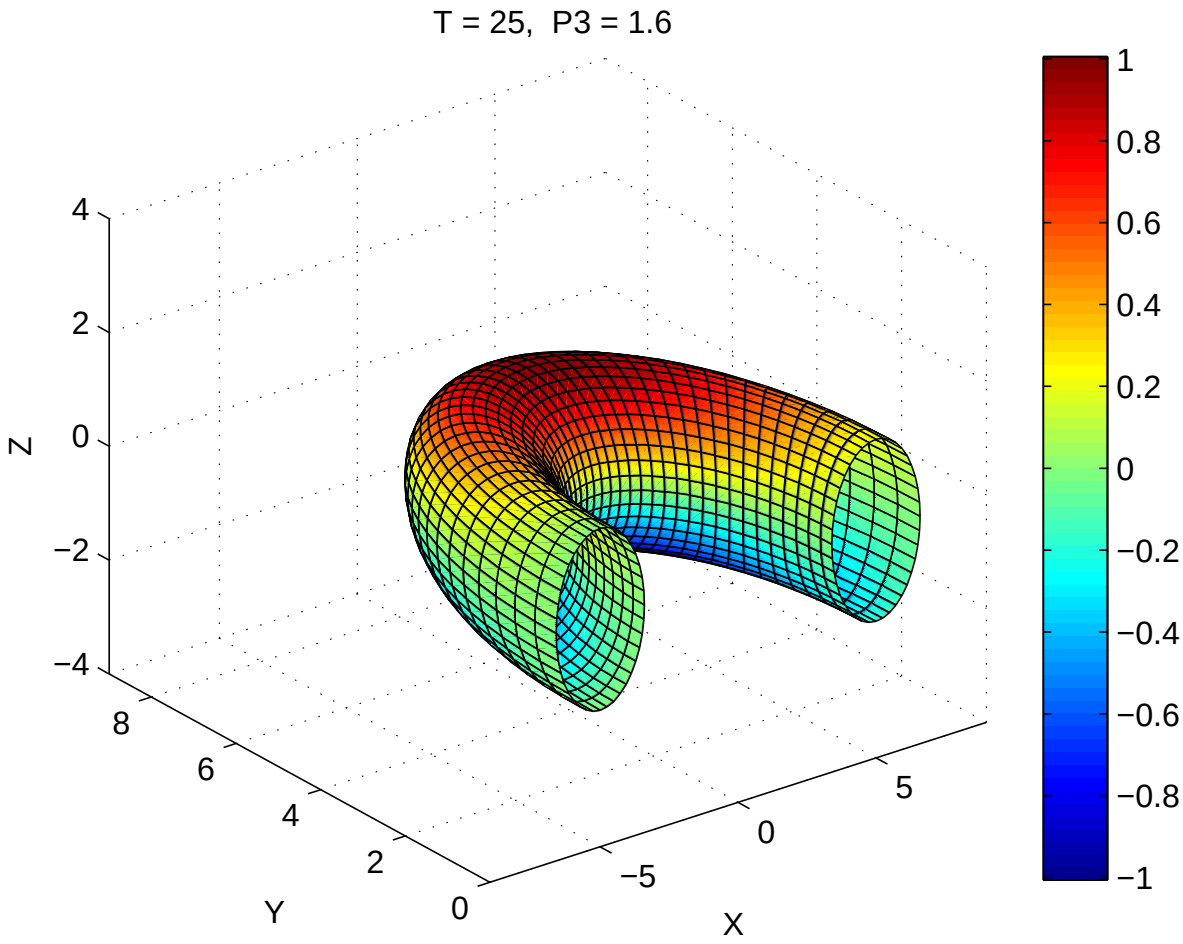
Eigenvalue output was $\lambda = -1.1375$

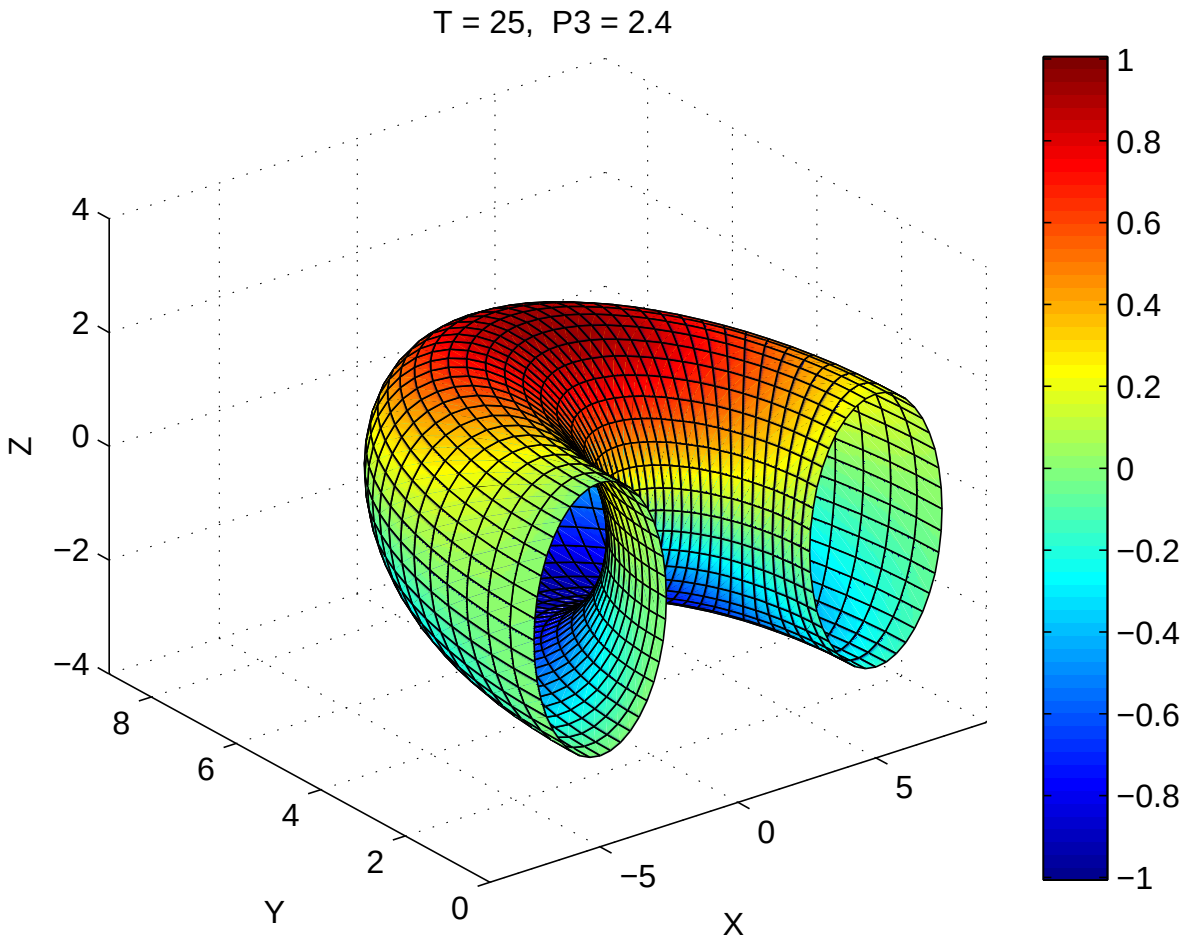
MATLAB plots of FEM and output grids are drawn below, then MATLAB graphs (of eigenfunction) at several P3=constant cross-sections are shown on the following pages. Finally PDE2D graphs at the first two P1=constant cross-sections are shown.

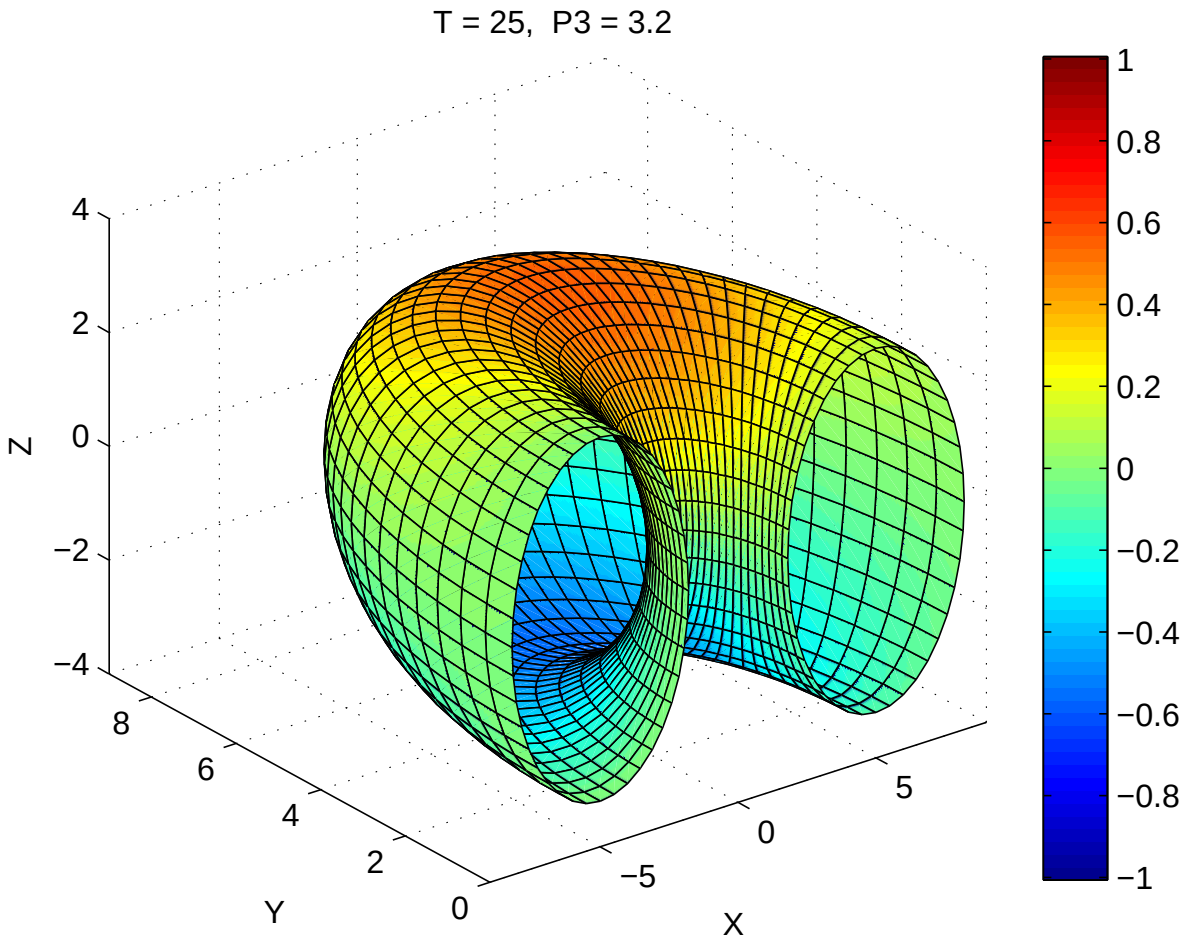


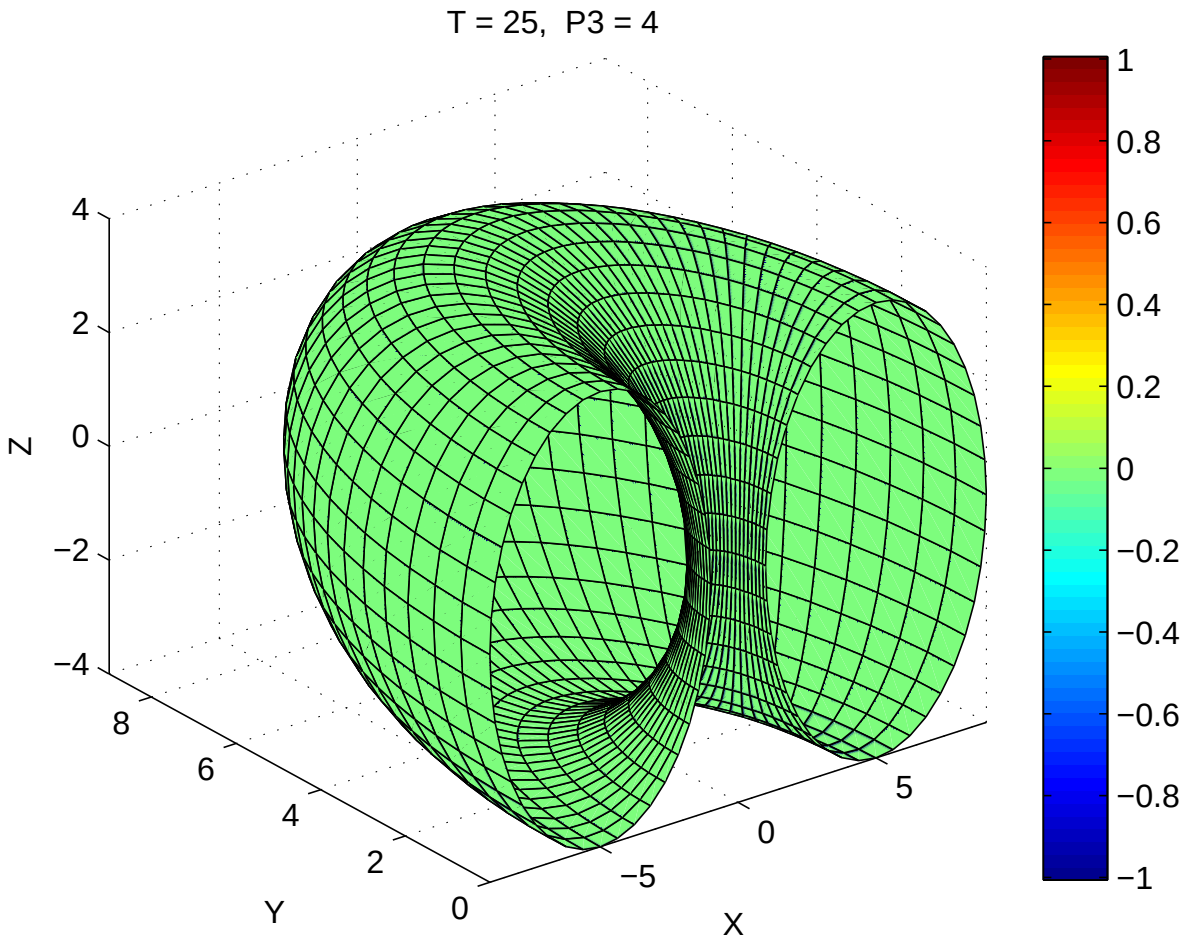

Output grid
4
2
0
−2
−4
Z
8
6
4
2
0
Y
−5
0
5
X

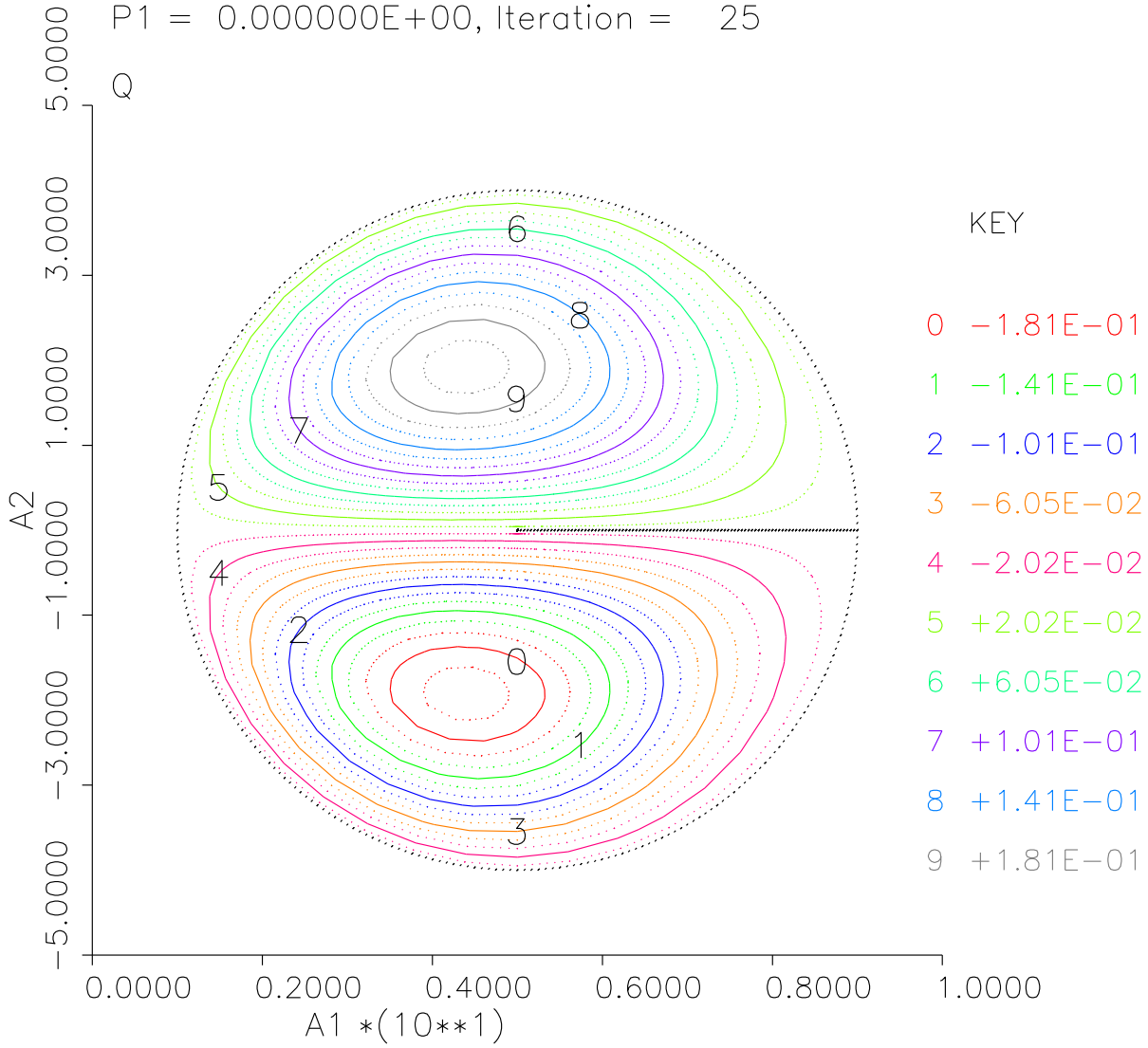

T = 25, P3 = 0.8
Z
4
2
0
−2
−4
Y
8
6
4
2
0
X
−5
0
5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

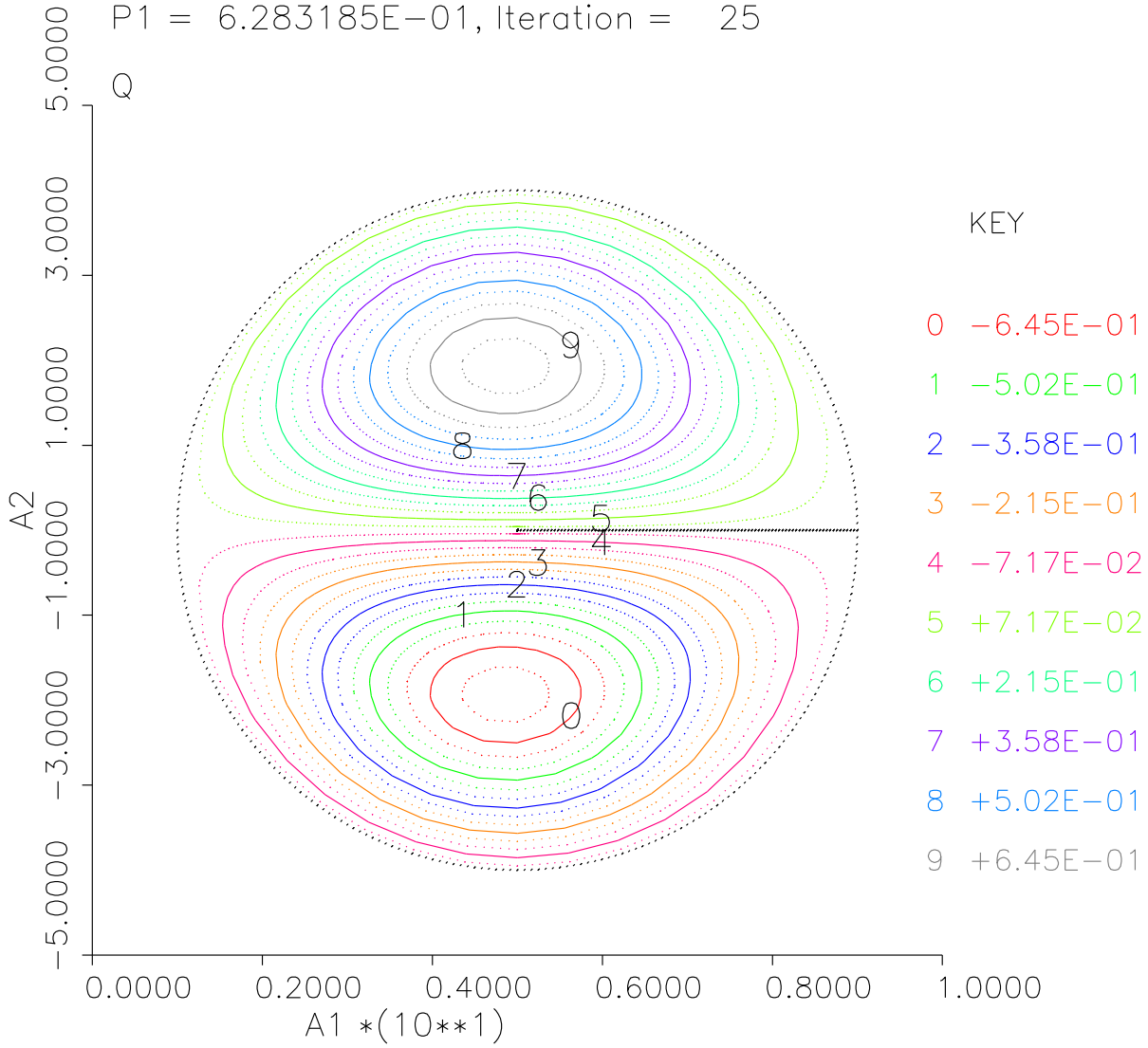

T = 25, P3 = 1.6
Z
4
2
0
−2
−4
Y
8
6
4
2
0
X
−5
0
5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


T = 25, P3 = 2.4
Z
4
2
0
−2
−4
Y
8
6
4
2
0
X
−5
0
5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


T = 25, P3 = 3.2
Z
4
2
0
−2
−4
Y
8
6
4
2
0
X
−5
0
5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


T = 25, P3 = 4
Z
4
2
0
−2
−4
Y
8
6
4
2
0
X
−5
0
5
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1


P1 = 0.000000E+00, Iteration = 25
Q
A2
5.0000
3.0000
1.0000
−1.0000
−3.0000
−5.0000
0.0000
0.2000
0.4000
0.6000
0.8000
1.0000
A1 *(10**1)
KEY
0 −1.81E−01
1 −1.41E−01
2 −1.01E−01
3 −6.05E−02
4 −2.02E−02
5 +2.02E−02
6 +6.05E−02
7 +1.01E−01
8 +1.41E−01
9 +1.81E−01


P1 = 6.283185E−01, Iteration = 25
Q
A2
5.0000
3.0000
1.0000
−1.0000
−3.0000
−5.0000
0.0000
0.2000
0.4000
0.6000
0.8000
1.0000
A1 *(10**1)
KEY
0 −6.45E−01
1 −5.02E−01
2 −3.58E−01
3 −2.15E−01
4 −7.17E−02
5 +7.17E−02
6 +2.15E−01
7 +3.58E−01
8 +5.02E−01
9 +6.45E−01