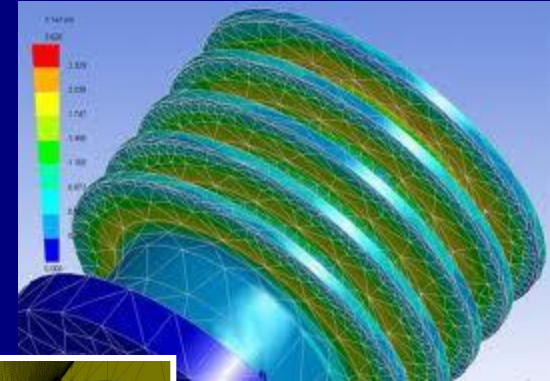
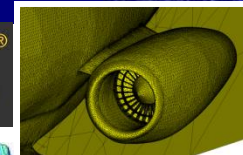


**I.I. Vorovich
Institute
Mathematics
Mechanics
Computer Science**

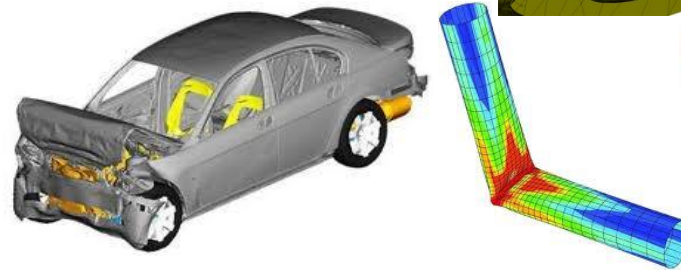
Using FlexPDE finite element program



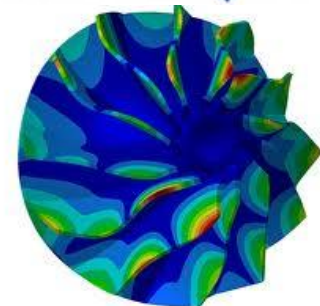
ANSYS®



Dr. Anna A. Nasedkina
Southern Federal University



ABAQUS





Outline

- What is FlexPDE, what it can do and who can use it
- FlexPDE script
- Boundary conditions
- Basic steps to solve a problem in FlexPDE
- Variables and equations
- Domain description: regions, boundary paths (lines, splines, arcs)
- Finite element mesh generation
- Defining material properties
- Graphical output

What is FlexPDE?

- FlexPDE is a "scripted finite element model builder and numerical solver". This means that from a script written by the user, FlexPDE performs the operations necessary to turn a description of a partial differential equations system into a finite element model, solve the system, and present graphical output of the results.
- FlexPDE is also a "problem solving environment", because it performs the entire range of functions necessary to solve partial differential equation systems: an editor for preparing scripts, a mesh generator for building finite element meshes, a finite element solver to find solutions, and a graphics system to plot results.
- FlexPDE has no pre-wired problem domain or equation list. The choice of partial differential equations is totally up to the user.

What can FlexPDE do?

- FlexPDE can solve systems of *first- or second-order partial differential equations* in Cartesian or axi-symmetric two-dimensional geometry or in three-dimensional Cartesian geometry.
- The system may be steady-state or time-dependent, or alternatively FlexPDE can solve eigenvalue problems.
- Any number of simultaneous equations can be solved, subject to the limitations of the computer on which FlexPDE is run.
- The equations can be linear or nonlinear. Nonlinear systems are solved by applying a modified Newton-Raphson iteration process.
- Any number of regions of different material properties may be defined. Modeled variables are assumed to be continuous across material interfaces.

Who can use FlexPDE?

- Researchers in many fields can use FlexPDE to model their experiments or apparatus, make predictions or test the importance of various effects. Parameter variations or dependencies are not limited by a library of forms, but can be programmed at will.
- Engineers can use FlexPDE to do design optimization studies, feasibility studies and conceptual analyses. The same software can be used to model all aspects of a design; no need for a separate tool for each effect.
- Educators can use FlexPDE to teach physics or engineering. A single software tool can be used to examine the full range of systems of interest in a discipline. Students see the actual equations, and can experiment interactively with the effects of modifying terms or domains.

What does a script look like?

- TITLE – a descriptive label for the output.
- SELECT – user controls over the default behavior of FlexPDE
- VARIABLES – here the dependent variables are named.
- DEFINITIONS – useful parameters, relationships or functions are defined.
- EQUATIONS – each variable is associated with a partial differential equation.
- INITIALVALUES – starting values for nonlinear or time-dependent problems.
- BOUNDARIES – the geometry is described by walking the perimeter of the domain, stringing together line or arc segments to bound the figure.
- PLOTS – the desired graphical output is listed. Plots may be any combination of CONTOUR, SURFACE, ELEVATION or VECTOR plots.

Notation

- ❑ **Comments** can be placed anywhere in a script.
 { Anything inside curly brackets is a comment. }
 from an exclamation to the end of the line is a comment.
- ❑ Differentiation, such as du/dx , is denoted by the form $dx(u)$. All active coordinate names are recognized, as are second derivatives like $dxx(u)$ and differential operators Div, Grad and Curl.
- ❑ Names are NOT case sensitive. "F" is the same as "f".

Example of a script

```
TITLE 'Simple diffusion equation'  
{ this problem lacks sources and boundary conditions }  
VARIABLES  
phi  
DEFINITIONS  
k=3 { conductivity }  
EQUATIONS  
div(-k*grad(phi)) =0  
BOUNDARIES  
REGION 1  
START(0,0)  
LINE TO (1,0)  
TO (1,1)  
TO (0,1)  
TO CLOSE  
PLOTS  
CONTOUR(phi)  
VECTOR(-k*grad(phi))  
END
```


Basic steps to solve a problem in FlexPDE

- Define the variables and equations
- Define the domain
- Define the material parameters
- Define the boundary conditions
- Specify the graphical output.

Variables and equations

- what are the variables that you want to analyze?
- what are the partial differential equations that define them?

The **VARIABLES** and **EQUATIONS** sections of a problem script supply this information. The two are closely linked, since you must have one equation for each variable in a properly posed system.

VARIABLES

Phi

EQUATIONS

$\text{Div}(\text{grad}(\text{Phi})) = 0$

VARIABLES

A,B

EQUATIONS

A: $\text{Div}(\text{grad}(A)) = 0$

B: $\text{Div}(\text{grad}(B)) = 0$

Mapping the domain: 2D domain description

- A two-dimensional problem domain is described in the **BOUNDARIES** section.
- A **REGION** specification begins with the statement **REGION** <number> (or **REGION** "name") and all loops following the header are included in the region.
- The first **REGION** should contain the entire domain. This is an unenforced convention that makes the attachment of boundary conditions easier.
- Region shapes are described by walking the perimeter, stepping from one joint to another with **LINE**, **SPLINE** or **ARC** segments. Each segment assumes that it will continue from the end of the previous segment, and the **START** clause gets things rolling. You can make a segment return to the beginning with the word **CLOSE** (or **TO CLOSE**).

Boundary paths

- A boundary path has the general form

START(a,b) segment TO (c,d) ...

where **(a,b)** and **(c,d)** are the physical coordinates of the ends of the segment, and **segment** is either **LINE**, **SPLINE** or **ARC**.

- The path continues with a connected series of segments, each of which moves the segment to a new point. The end point of one segment becomes the start point of the next segment.
- A path ends whenever the next input item cannot be construed as a segment, or when it is closed by returning to the start point

... segment TO CLOSE.

or

... segment CLOSE.

Boundary paths: line and spline segments

- Line segments take the form

LINE TO (x,y)

When successive **LINE** segments are used, the reserved word **LINE** does not have to be repeated, as in the following:

LINE TO (x1,y1) TO (x2,y2) TO (x3,y3) TO ...

- Spline segments are syntactically similar to Line segments

SPLINE TO (x,y) TO (x2,y2) TO (x3,y3) TO ...

Boundary paths: arc segments

- Arc segments create either circular or elliptical arcs, and take one of the following the forms:

ARC TO (x1,y1) to (x2,y2)

ARC (RADIUS = R) to (x,y)

ARC (CENTER = x1,y1) to (x2,y2)

ARC (CENTER = x1,y1) ANGLE=angle

- Here **angle** is an angle *measured in degrees*, following the standard convention that positive angles rotate counter-clockwise and negative angles rotate clockwise.
- When the form **ARC (CENTER=x1,y1) to (x2,y2)** is used and the center **(x1,y1)** is not equidistant from the start and end points, an elliptical arc segment is generated with major and minor axes along the X and Y coordinate directions.

Some examples of boundary paths

- Rectangular region

START(x1,y1)

LINE TO(x2,y1)

TO (x2,y2)

TO (x1,y2)

TO CLOSE

- Circle

START(r,0)

ARC(CENTER=0,0) ANGLE=360

- 90 degree circular arc

START(r,0)

ARC(CENTER=0,0) TO (0,r)

- 90 degree elliptical arc

START(x0+a,y0)

ARC(CENTER=x0,y0) TO (x0,y0+b)

Prototype form for boundaries

- Different regions: can be used for defining different materials, for example to model inclusion: material 2 is inside material 1
BOUNDARIES
REGION 1 {Material 1 with its set of material properties}
START (X1,Y1) <closed loops around the domain> TO CLOSE
REGION 2 {Material 2 with its set of material properties, can be located inside material 1}
START (X2,Y2) <closed loops around overlays of the second material>
TO CLOSE
- One region: when contains several starting points, the resulting domain will consist of the union of subdomains minus their intersections, can be used to model a hole
BOUNDARIES
REGION 1 {Material 1 with its set of material properties}
START (X1,Y1) <closed loops around the domain> TO CLOSE
START (X2,Y2) <closed loops around overlays of the second material>
TO CLOSE

Example of domain: inclusion

DEFINITIONS

$R=1/2$ {ring radius}

BOUNDARIES

REGION 1 'box' { the bounding box }

START(-1,-1)

LINE TO(1,-1)

TO (1,1)

TO (-1,1)

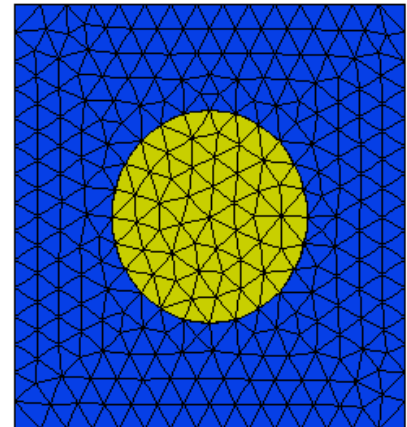
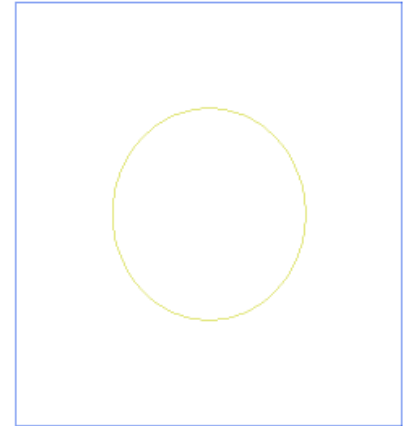
TO CLOSE

REGION 2 'blob' { the embedded circular 'blob' }

START 'ring' (R,0)

ARC(CENTER=0,0) ANGLE=360 TO CLOSE

❑ Different regions are shown by different colors



Example of domain: hole

DEFINITIONS

$R=1/2$ {ring radius}

BOUNDARIES

REGION 1 'box' { the bounding box }

START(-1,-1)

LINE TO(1,-1)

TO (1,1)

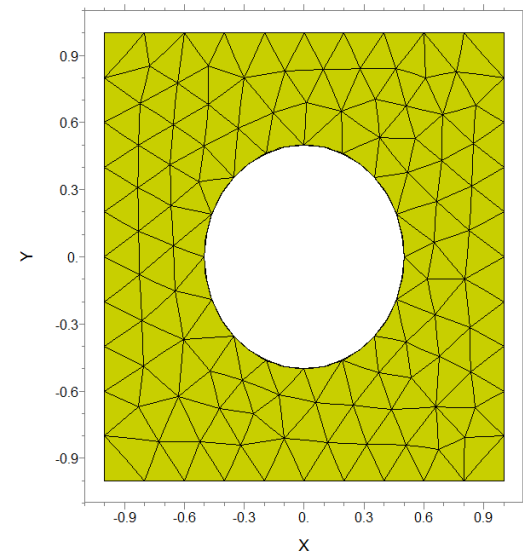
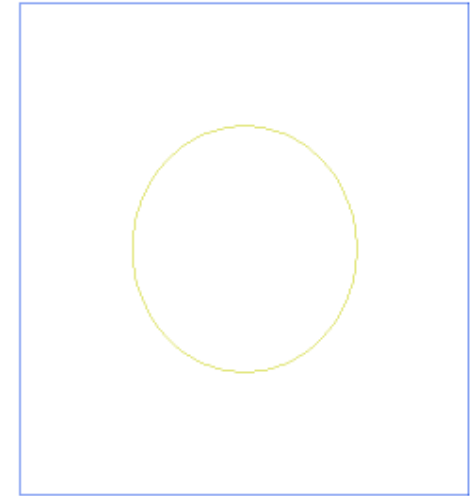
TO (-1,1)

TO CLOSE

{ circular hole' }

START 'ring' (R,0)

ARC(CENTER=0,0) ANGLE=360 TO
CLOSE



Generating FE mesh

- When you select "Run Script" from the Controls menu (or the button), FlexPDE will begin execution by automatically creating a finite element mesh to fit the domain you have described. In the automatic mesh, cell sizes will be determined by the spacing between explicit points in the domain boundary, by the curvature of arcs, or by explicit user density controls.
- There are several controls that the user can apply to change the behavior of the automatic mesh, such as MESH_DENSITY and MESH_SPACING
- However, in most cases, it is not necessary to intervene in the mesh generation, FlexPDE will adaptively refine the mesh wherever it detects that there are strong curvatures in the solution.

User density control parameter for a region or the whole domain: **MESH_DENSITY**

- **MESH_DENSITY**: sets minimal number of cells for a unit length

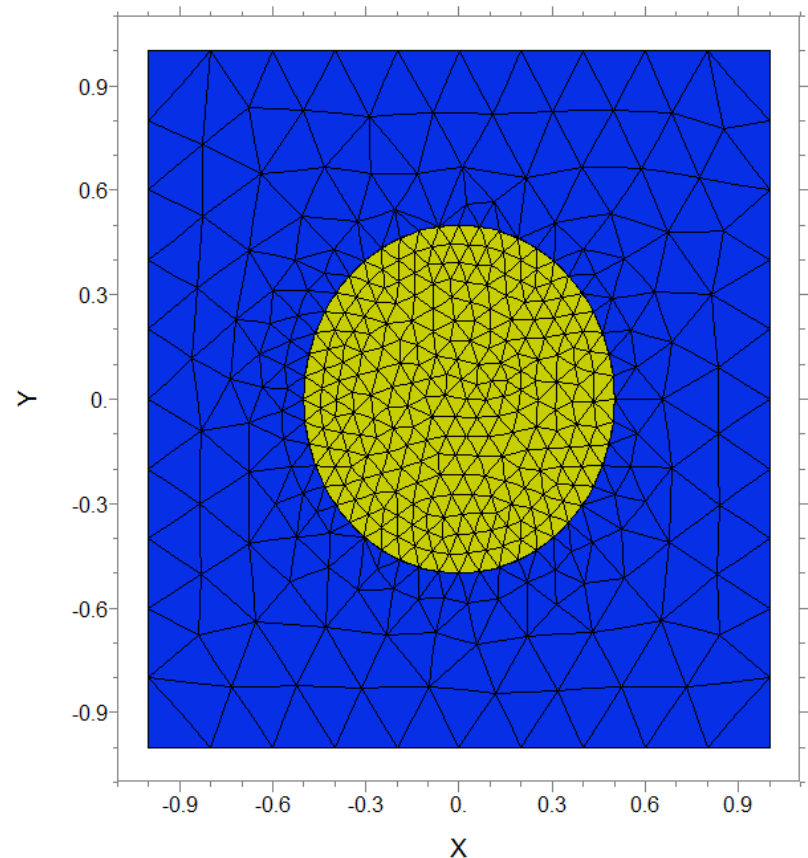
REGION 2 'blob'

MESH_DENSITY = 15

START(1/2,0)

ARC(CENTER=0,0) ANGLE=360

TO CLOSE



User density control parameter for a region or the whole domain : **MESH_SPACING**

- MESH_SPACING: sets maximal cell size

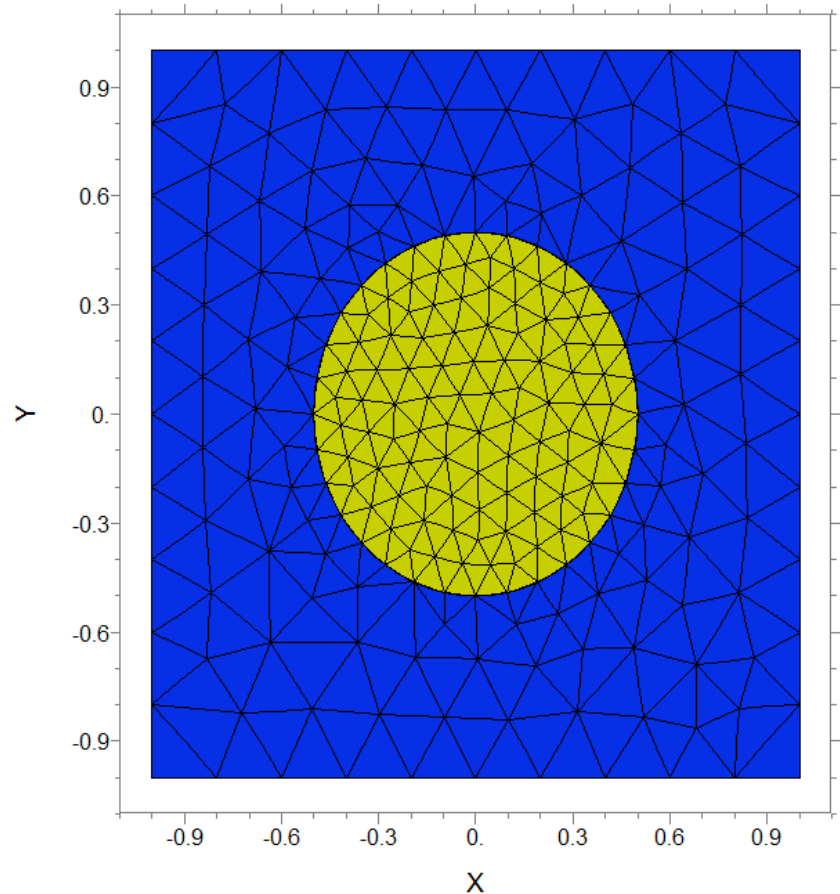
REGION 2 'blob'

MESH_SPACING = 0.1

START(1/2,0)

ARC(CENTER=0,0) ANGLE=360

TO CLOSE



Defining material parameters

- This is handled in FlexPDE by two facilities. First, the material parameters are given names and default values in the **DEFINITIONS** section. Second, the material parameters are given regional values within the domain **REGIONS**.

DEFINITIONS

k = 3

- This default value will be used as the value of "k" in every REGION of the problem, unless specifically redefined in each region, for example:

REGION 2 'blob' { the embedded blob }

k = 0.001

START(1/2,0)

ARC(CENTER=0,0) ANGLE=360

Setting boundary conditions

- Each boundary condition statement takes as an argument the name of a variable. This name associates the boundary condition with one of the listed equations, for it is in reality the equation that is modified by the boundary condition.
- **Main types of boundary conditions:**
- The VALUE (or Dirichlet) boundary condition specifies the value that a variable must take on at the boundary of the domain. Values may be any legal arithmetic expression, including nonlinear dependences on variables.
- The NATURAL (or LOAD) boundary condition specifies a flux at the boundary of the domain. Definitions may be any legal arithmetic expression, including nonlinear dependence on variables. With Laplace's equation, the NATURAL boundary condition is equivalent to the Neumann or normal derivative boundary condition. The precise meaning of the NATURAL depends on the PDE for which the BC is specified.

Setting boundary conditions: examples

- Each boundary condition statement declares a condition which will apply to the subsequent boundary segments until the declaration is changed.
- Note: the segment shape must be restated after every change of boundary condition

```
REGION 1 'box' { the bounding box }  
START(-1,-1)  
{ Phi=0 on base line: }  
VALUE(Phi)=0 LINE TO(1,-1)  
{ normal derivative =0 on right side: }  
NATURAL(Phi)=0 LINE TO (1,1)  
{ Phi = 1 on top: }  
VALUE(Phi)=1 LINE TO (-1,1)  
{ normal derivative =0 on left side: }  
NATURAL(Phi)=0 LINE TO CLOSE
```


Setting boundary conditions in the elasticity problem

variables

U, V {displacements $U=U_X$ and $V=U_Y$ are the variables}

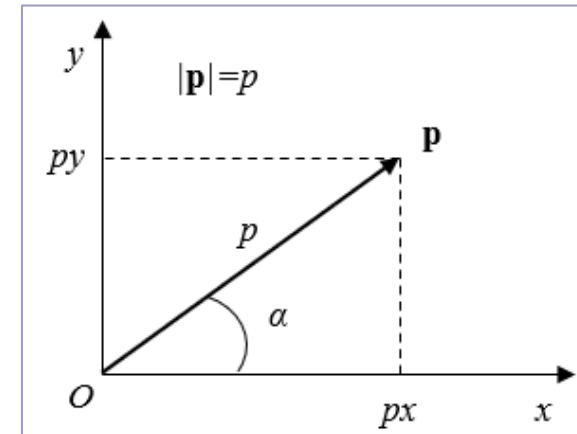
equations {differential equilibrium equations for a plane elastic problem}

$$U: dx(T_{xx}) + dy(T_{xy}) = 0$$

$$V: dx(T_{xy}) + dy(T_{yy}) = 0$$

- Rigidly fixed boundary:
value(U)=0
value(V)=0
- Symmetry along the axis OX :
load(U)=0
value(V)=0
- Symmetry along the axis OY :
value(U)=0
load(V)=0

- Pressure on segment (calculate coordinates of the pressure vector $p=\{p_x, p_y\}=\{p \cdot \cos \alpha, p \cdot \sin \alpha\}$
load(U)= p_x
load(V)= p_y
- Free boundary
load(U)=0
load(V)=0



Graphical output

- The **MONITORS** and **PLOTS** sections contain requests for graphical output. A record of all PLOTS is written in a file with suffix .PG6 (or other number of FlexPDE version)
 - **MONITORS** are used to get ongoing information about the progress of the solution.
 - **PLOTS** are used to specify final output, and these graphics will be saved in a disk file for later viewing.
- **SUMMARY** section creates a page for the placement of numerous REPORTs (text output)
- **Output commands:**
 - CONTOUR** - a plot of contours of the argument; it may be color-filled (when followed by the option PAINTED)
 - SURFACE** - a 3D surface of the argument
 - VECTOR** - a field of arrows
 - ELEVATION** - a "lineout" along a defined path
 - REPORT** - text-only reports

Output commands

■ PLOTS and SUMMARY sections

PLOTS

CONTOUR(Phi) {isolines only}

CONTOUR(Phi) painted {color filling}

VECTOR(-k*grad(Phi))

ELEVATION(Phi) FROM (0,-1) to (0,1)

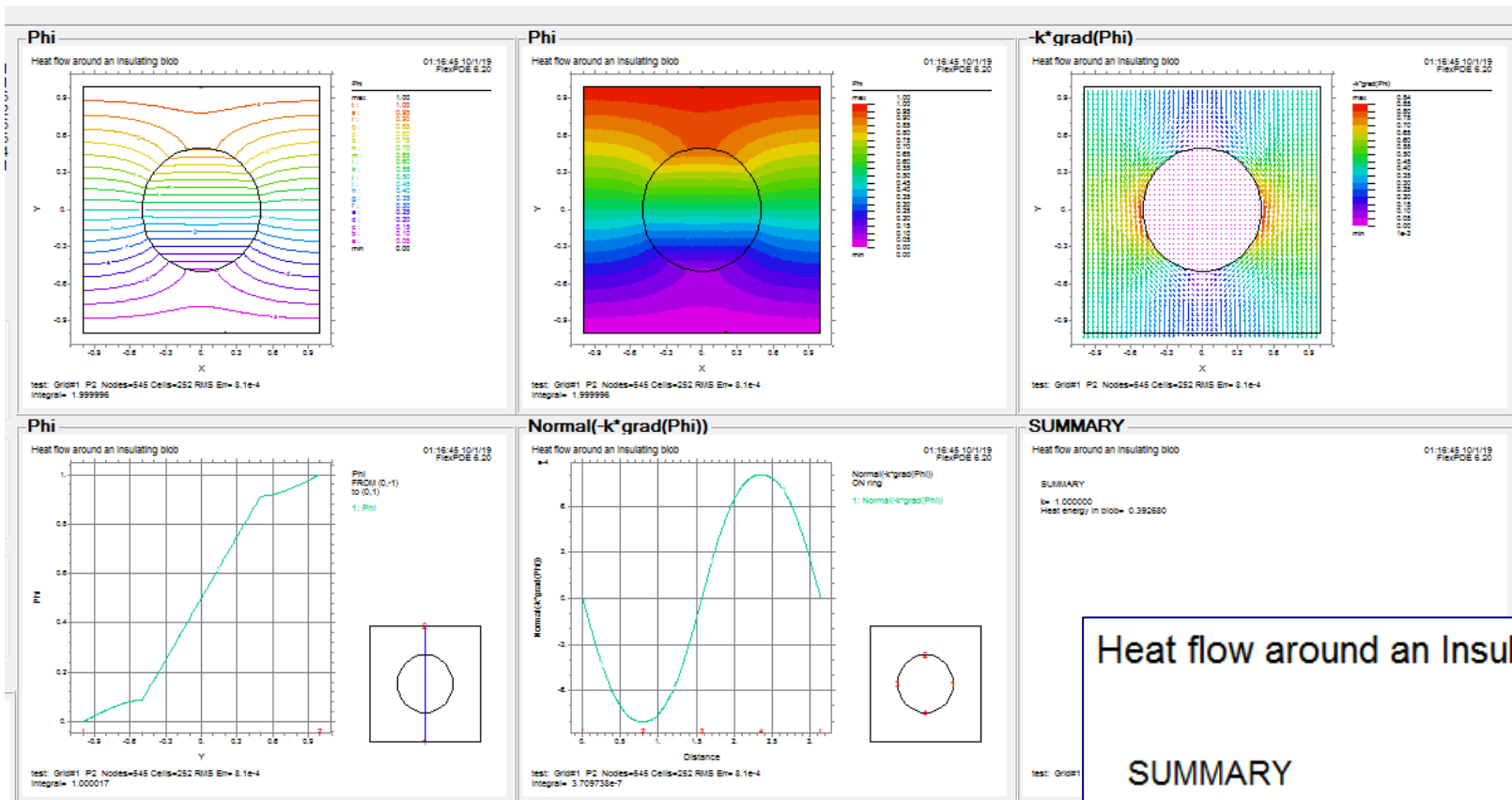
ELEVATION(Normal(-k*grad(Phi))) ON 'ring'

SUMMARY

REPORT(k)

REPORT(INTEGRAL(Phi,'blob')) as "Heat energy in blob"

Graphical output example with PLOTS and SUMMARY



Heat flow around an Insulating blob

SUMMARY

$k = 1.000000$

Heat energy in blob = 0.392680