

A GRAPHICAL PREPROCESSING INTERFACE  
FOR NON-CONFORMING SPECTRAL ELEMENT SOLVERS

A Thesis

by

BO HUNG KIM

Submitted to the Office of Graduate Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Mechanical Engineering

A GRAPHICAL PREPROCESSING INTERFACE  
FOR NON-CONFORMING SPECTRAL ELEMENT SOLVERS

A Thesis

by

BO HUNG KIM

Submitted to the Office of Graduate Studies of  
Texas A&M University  
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Ali Beskok
Committee Members,	Obdulia Ley Tahir Cagin
Head of Department,	Dennis O'Neal

August 2006

Major Subject: Mechanical Engineering

## ABSTRACT

A Graphical Preprocessing Interface  
for Non-conforming Spectral Element Solvers. (August 2006)

Bo Hung Kim, B.S., Yonsei University







Chair of Advisory Committee: Dr. Ali Beskok

A graphical preprocessor for Spectral Element Method (SEM) is developed with an emphasis on user friendly graphical interface and instructive element construction. The interface of the preprocessor helps users with every step during mesh generation, aiding their understanding of SEM. This preprocessor's Graphical User Interface (GUI) and help system are comparable to other commercial tools. Moreover, this preprocessor is designed for educational purposes, and prior knowledge of Spectral Element formulation is not required to use this tool. The information window in the preprocessor shows step-by-step instructions for the user. The preprocessor provides a graphical interface which enables visualization while the mesh is being constructed, so that the entire domain can be discretized easily. In addition, by following informative steps during the mesh construction, the user can gain knowledge about the intricate details of computational fluid dynamics.

This preprocessor provides a convenient way to implement h/p type nonconforming interfaces between elements. This aids the user in learning advanced numerical discretization techniques, such as the h/p nonconforming SEM. Using the preprocessor

facilitates enhanced understanding of SEM, isoparametric mapping, h and p type nonconforming interfaces, and spectral convergence. For advanced users, this preprocessor provides a proficient and convenient graphical interface independent of the solvers. Any spectral element solver can utilize this preprocessor, by reading the format of the output file from the preprocessor. Given these features, this preprocessor is useful both for novice and advanced users.

## TABLE OF CONTENTS

	Page
ABSTRACT .....	iii
TABLE OF CONTENTS .....	v
LIST OF TABLES .....	vii
LIST OF FIGURES.....	viii
CHAPTER	
I INTRODUCTION .....	1
Spectral Discretizations Using Method of Weight Residuals (MWR) .....	5
Choice of Approximation Functions .....	6
Spectral Method, Finite Element Method, Spectral Element Method .....	7
II SPECTRAL ELEMENT METHOD FORMULATION .....	10
Element Mesh in the Preprocessor: Gauss Lobatto-Legendre Interpolants.....	10
Various Elements in the Preprocessor: Isoparametric Mapping .....	13
Convergence.....	18
Nonconforming Interfaces.....	21
III PREPROCESSOR.....	23
Nomenclature for the Work Space .....	23
Output File.....	24
Drawing Elements .....	35
Modify Elements .....	36
Edit Mode [  ].....	37
Splitting Elements [  ,  ,  ].....	39
Export Output File [  ] .....	40
Drawing Curved Elements [  ].....	42
IV APPLICATIONS .....	51
Kovaszny Flow .....	51
Backward Facing Step Flow .....	53

CHAPTER	Page
Y Channel Flow .....	56
Flow Over a Stack of Cylinders .....	58
V CONCLUSIONS .....	62
REFERENCES .....	63
APPENDIX A .....	66
APPENDIX B .....	69
VITA .....	73

LIST OF TABLES

TABLE		Page
1	Description of parameters in the preprocessor.....	25

## LIST OF FIGURES

FIGURE	Page
1	Left: Discretization of a square using $50 \times 50$ 1 <sup>st</sup> order elements by FEM. Right: Discretizations of a square using 50 <sup>th</sup> -order single domain with SM..... 3
2	The discretization of global domain with Finite Element Method, Spectral Method, Spectral Element Method [4]..... 4
3	The Gauss Lobatto Legendre points and interpolants ( $N = 3, 4, 5, 6$ ). ..... 11
4	Gauss-Lobatto-Legendre(GLL) points for $6 \times 6$ quadrilateral elements (left). The corresponding 2D shape function at GLL point (2, 3) (Sert and Beskok, 2005) (right). ..... 12
5	Various types of elements in the preprocessor..... 13
6	Isoparametric mapping between global (x,y) coordinates and local (r,s) coordinates in the preprocessor..... 14
7	(a) h-type refinement for the Kovasznay flow geometry using: 8, 32, 64 elements with 3 <sup>rd</sup> order discretization per direction. (b) p-type refinement using: 8 elements with 6, 12, 16 <sup>th</sup> order approximation per direction..... 19
8	Convergence obtained from Kovasznay flow under: p-type and h-type refinements. .... 20
9	The p-type (upper) and h-type (lower) nonconforming interface. .... 22
10	Nomenclature of work space..... 23
11	Parameter input dialog. .... 26
12	Boundary conditions, initial conditions, forcing function and exact solutions in the BCs input dialog. .... 26
13	Numbering scheme of vertices and sides. .... 27
14	Coordinate input interface in the element dialog with rectangular (top) and non-rectangular (bottom) elements..... 28



FIGURE	Page
15	The curvature input dialog and assignment of curvature on the side..... 29
16	The output format for curvature in CURVATURE section. .... 29
17	The assigned curvature information in the output text file. .... 30
18	Connectivity and boundary conditions in the element dialog. .... 31
19	The boundary and connectivity information in the CONNECTIVITY section..... 31
20	Click the mesh button to start the drawing mode..... 33
21	Press ‘OK’ in the parameters dialog box to start mesh mode. .... 34
22	Boundary conditions and initial conditions. Press ‘OK’ to proceed. .... 35
23	Draw an element: selecting upper right vertices of an element. .... 35
24	The default values of grid size and the line thickness in the attribute dialog..... 36
25	Element dialog box..... 36
26	Set a curvature to the bottom side of the element. .... 37
27	Select a target element by dragging a mouse (left) over the target element. .... 38
28	After selecting an element, the element can be modified in the element dialog box..... 38
29	Select a target element and performing 4 split and horizontal split. .... 39
30	View Output-inf.txt file generated from the preprocessor..... 40
31	Solving problem with the default solver. .... 41
32	Result screen from post the processor - Kovasznay flow. .... 41
33	Parameters in SOLVERScalar section. .... 42

FIGURE	Page
34	Curvature dialog box..... 42
35	Modifying drawn element with non-rectangular option. .... 43
36	Define curvature at each side. .... 43
37	Curved elements and mesh refinement using splitting functions: h- type non-conforming interfaces. .... 44
38	1×4 size element before splitting. .... 45
39	Perform vertical split of element into four pieces. .... 45
40	Horizontal split of elements into eight pieces. .... 45
41	Splitting bottom elements..... 46
42	Refine the element mesh using splitting. .... 46
43	Add groove elements 4, 8, 13, 17, 22, 26, 31..... 46
44	Modifying a previously drawn geometry..... 47
45	Square element with vertices $[(-1, -1), (1, 1)]$ ..... 47
46	Set the radius of a hole in the element. .... 48
47	A circular hole with radius 0.5..... 48
48	Distance calculator in the curvature dialog box..... 49
49	Define the second curvature..... 49
50	Set outside curvature to make annulus geometry..... 50
51	Heat equation with constant wall temperature ( $T_1=100$ , $T_2=50$ ) in an annulus using curved elements..... 50
52	Drawing the mesh for Kovaszny problem in the preprocessor. .... 51
53	Kovaszny flow mesh. Eight 6×6 conforming elements are used (left). Stream wise velocity contours (right). .... 52

FIGURE	Page
54	Kovaszny flow p-type nonconforming mesh: Same eight element mesh is used. Elements 1,2,3,4 have expansions orders $N$ , where elements 2,4,6,8 have expansion orders $N-2$ (left). Kovaszny flow with h-type nonconforming elements: Six element mesh is used. All elements have expansion order $N$ , except elements 3 and 6 has expansion orders $N+4$ in the y-direction (right). ..... 52
55	Convergence results for Kovaszny flow obtained using conforming, and h- and p-type non-conforming meshes. $N_{df}$ is the number of nodes (degrees of freedom). Adopted from Sert and Beskok 2005 [1]. ..... 53
56	Build a global geometry using splitting element and modify drawn element. .... 54
57	Perform h-type refinements in the inlet channel. .... 54
58	Perform horizontal split in the outlet channel. .... 55
59	H-type refinement in the flow region. .... 55
60	Draw an element and change the coordinates of four vertices. .... 56
61	Draw the elements at inlet and outlet channel. .... 56
62	Add an element in the joint region. .... 57
63	Perform refinement in the flow region. .... 57
64	Draw three long elements and perform p-type refinement. .... 58
65	Vertical split of the elements (left). Put a hole in the element 5 (right). .... 59
66	Refine the mesh around the cylinder using 4-split function. .... 59
67	Refine the mesh at the flow region affected by the cylinder (left). Series of cylinders could be added in a similar method in Fig. 66. (right) .. 60
68	The number of elements and DOF for conforming case and non-conforming case; $N_{DOF} = 3744$ (left, conforming case), $N_{DOF} = 8640$ (right, non-conforming case). .... 61

## CHAPTER I

### INTRODUCTION

The procedure for analyzing engineering problems often includes approximation, finding adequate models and solving the necessary governing equations in the model. Due to the rapid development of computer technology, most of the problems can be solved and analyzed utilizing numerical techniques, where complex geometries are discretized into a finite number of small, geometrically simple elements. In order to correctly represent the discretized geometry, the elements may have rather complex shapes. Under such conditions, isoparametric mapping can be used to adapt elemental discretization schemes to complex elements by using a master element and isoparametric mapping that correctly represent the shape of the elements. Calculation of discretized local geometry is relatively simple compared to one global domain. Therefore, computer simply repeats the calculations for every discretized element. Benefits of this kind of method are more pronounced with the advent of faster and more advanced computer processors.

An additional benefit of using numerical schemes with a computer is not only the efficiency of the calculation, but also the graphical modeling, which enhances the engineer's capability of constructing and understanding complex problems. For example, the visualization of analytical and modeling results is very helpful to understand engineering problems like fluid flow and heat transfer in complex engineering geometries. With that in mind, the preprocessor is developed to help users construct and

---

This thesis follows the style of *IEEE Transactions on Automatic Control*.

discretize the whole engineering domain to generate input files for the developed solver [1]. The most important advantage of the preprocessor lies in its efficiency and intuitiveness of discretizing the engineering domain. Visual process of mesh generation is more efficient than the models that are described merely by text or numbers. However, it is inevitable that the discretized models have numerical errors due to the approximation. To reduce the numerical error caused by approximation, it is either necessary to increase the number of discretized elements or increase the order of approximation function.

With rapid advancements in computer systems, it is possible to increase the number of discretized elements by using very small elements with low order approximations, so that the error of the discretized models compared with the exact solution is minimized. The most representative numerical technique of this kind is the Finite Element Method (FEM) [2]. It uses small finite number of elements with low order approximation functions (Fig. 1 left). When engineering models have complex geometry, FEM discretizes the geometry (e.g. circle) into many smaller elements so that the error is minimized. Since the low order FEM uses the 1<sup>st</sup> or 2<sup>nd</sup> order approximation functions, the error of approximation may be significant since the number of discretized elements may not be enough to reduce the error below a desired level with the entire domain.

However, Spectral Methods (SM) utilize high order approximation functions [3]. Hence, SM can maintain the desired accuracy in discretized elements. In most cases, SM utilizes global approximations in a large domain, and they are not suitable for discretization of complex geometry.

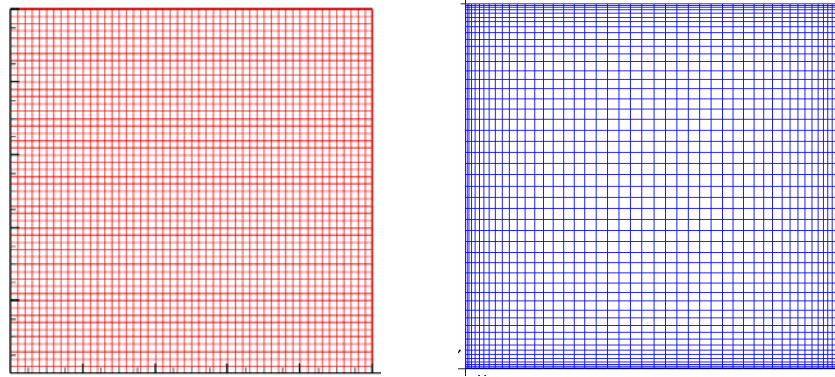


Fig. 1 Left: Discretization of a square using  $50 \times 50$  1<sup>st</sup> order elements by FEM. Right: Discretizations of a square using 50<sup>th</sup>-order single domain with SM.

An alternative solution is to develop multi-domain Spectral Methods, which can utilize rather large, yet high order discretization of complex geometries. This method is also known as the Spectral Element Method (SEM) (Fig. 2), which is a hybrid method that utilizes the common foundation and exhibits the competitive advantages of FEM and SM. For FEM, SM and SEM, discretization of elements to model the geometry could be assisted by a computer program, which provides GUI capability. Therefore, the preprocessor can provide an efficient way for discretizing complex domains by utilizing Spectral Element Method discretizations.

In this work, we concentrate on quadrilateral elements only. The concept of dividing complex domains into a collection of simpler domains is one of the biggest advantages of the Finite Element Method compared to Spectral or Finite Difference Methods. However, in Spectral Methods, the whole problem domain is represented by one element, which is discretized with global eigen-function expansions (Fig. 1 right). This method converges faster than any other method. However, it is not general enough to be applied to problems with complex geometries. Finite difference formulations use

simple, local discretizations. In case of un-equally distributed Finite Difference Methods exhibit several difficulties.

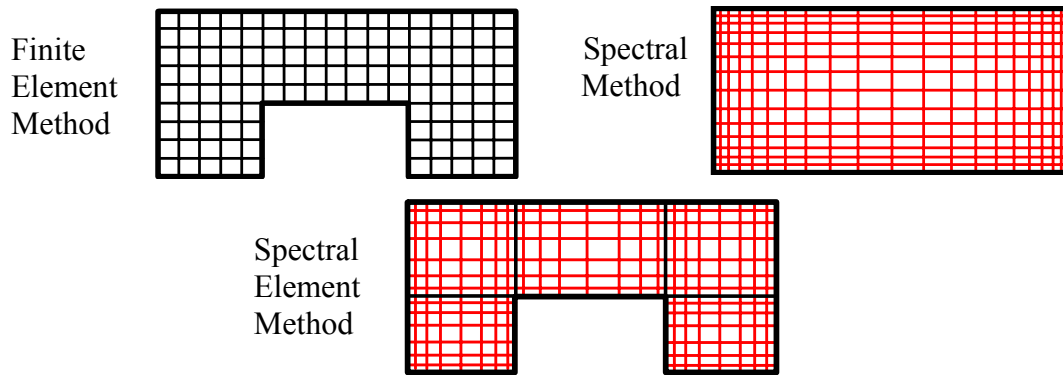


Fig. 2 The discretization of global domain with Finite Element Method, Spectral Method, Spectral Element Method [4].

The Finite Element Method is classified as a discretization scheme for differential equations, known as the method of weight residuals. In the Finite Element Method, the computational domain  $\Omega$  is divided into finite number of ( $K$ ) elements that are often mapped to quadrilateral elements isoparametrically in two dimensions. This provides a straight forward numerical integration and differentiation. To introduce the generalized scheme, the unknown ( $u$ ), geometry ( $\Omega_k$ ) and the numerical results are approximated using  $N^{th}$  order Lagrange polynomials. The default solver utilizes the Galerkin formulation. Therefore the test function ( $v$ ) is also expanded in terms of  $N^{th}$  order Lagrange polynomials for each element. In the case of FEM where low order polynomial expansions are used, the algebra becomes simple and straight forward, and numerical integration of the resulting equations can be done analytically. However, Spectral Method uses general  $N^{th}$  order eigen-function expansions. Despite its mathematical

complexity, SM exhibits exponential convergence, which is much faster than the algebraic convergence of FEM [5]. However, the Spectral Element Method is a hybrid method that utilizes the common foundations of FEM and SM. SEM uses high-order Lagrange interpolants based on Jacobi polynomials and still exhibit geometric flexibility of FEM.

### **Spectral Discretizations Using Method of Weight Residuals (MWR)**

Spectral Methods provide global approximations. Based on the weight functions selected, we can end up with three commonly used techniques: collocation, Galerkin and Tau methods. In the collocation (pseudo spectral) method, weight functions are the Dirac delta functions based on a set of collocation points. Collocation method is sometimes referred to as the nodal method, because it calculates the unknown variables at the nodes of the local domain. Collocation method can formulate non-linearities very easily but it has difficulties due to aliasing errors [6].

In the Galerkin formulation, the weight functions are selected to be the same as the approximation functions. Tau method is slightly different than the Galerkin method, where the weight functions do not necessary satisfy the boundary conditions. Instead, boundary conditions are enforced by a separate set of constraints. In Galerkin and Tau methods the unknowns are just the coefficients of a series expansion that are not in the physical space, and therefore these methods are sometimes referred to as modal methods. Typical MWR formulations for solving a partial differential equation (PDE) start with representing the partial differential equation in a residual form and equating it to zero

$$\mathbf{R} = 0. \tag{1}$$



Then the numerical equations with the unknowns are set up using a truncated series expansion. For a one-dimensional problem, this is given as

$$u(x) \approx u_N(x) = \sum_{i=0}^N \hat{u}_i h_i(x), \quad (2)$$

where  $N$  is the order of the approximation function,  $h_i$  are the approximation (trial, basis, shape) functions and  $\hat{u}_i$  are the unknown coefficients. Substitution of this approximation to the original PDE yields an approximate residual, which is weighted by the test functions ( $v$ ) and integrated over the domain. The weighted residual is then set to zero

$$\int_{x_0}^{x_1} R_N v dx = 0. \quad (3)$$

By selecting different weight functions ( $v$ ) for each unknown, we form  $(N+1)^2$  algebraic equations and solve for the unknown coefficients  $\hat{u}_i$ . This truncated series of the set of equations includes the whole problem domain.

### **Choice of Approximation Functions**

Spectral Methods can also be classified based on the approximation functions used. Chebyshev and Legendre polynomials are the most commonly used approximation functions [5]. In the Spectral Method, the use of trigonometric functions is known as the Fourier Spectral Method [7]. It is mostly used with periodic boundary conditions, such as simulation of three-dimensional homogeneous turbulence in simple domains. Chebyshev and Legendre polynomials belong to the family of Jacobi polynomials, which are the eigen-functions of singular Sturm-Liouville problems [8]. Chebyshev polynomials can be also classified as a special form of Fourier cosine expansion with a

change of variable [5]. All these polynomials which are used as approximation functions provide spectral accuracy (exponential convergence) for smooth ( $C^\infty$ ) solutions. Therefore, SM is the preferred solution technique for problems where high resolution is required [5]. Chebyshev and Legendre polynomials can be used as approximation functions through modal expansions. However, solving nonlinear problems with modal expansion is quite challenging. Therefore, SEM utilizes the Gauss Lobatto Legendre (GLL) points and nodal expansions to create grid points in the element mesh. Legendre polynomials provide more accurate numerical quadratures than Chebyshev polynomials [5]. Detailed explanations about the GLL points will be presented in Chapter II.

### **Spectral Method, Finite Element Method, Spectral Element Method**

The most important advantage of high order methods (SM and SEM), compared to low order ones (FEM) is the high order accuracy. Another advantage of high-order methods is that they are memory minimizing [7], which is the reason why they are preferred for computationally demanding meteorology problems. However, compared to low-order methods, they require more computations per degree of freedom. Also they suffer more from the geometric singularities such as corners, or discontinuities inherent in the solution such as shock waves. One big disadvantage of SM is that they provide global approximations and therefore are not suitable for complex domains. Problems with arbitrarily shaped boundaries can be solved with domain discretization methods, such as the Finite Element Method (FEM) [2]. FEM introduces a new domain discretization step in addition to the MWR formulation. In this step, the problem domain is subdivided into simple elements (sub-domains), such as triangles, quadrilaterals,

tetrahedral, etc. Complexity of the domain is no longer a problem, because these elements of different shapes and sizes can be arranged in any desired way. Unlike SMs, test and trial functions used in FEM are local, i.e. they are defined on each element separately. Another step used in FEM is the global assembly, where the local set of equations, written individually for each element are assembled by a direct-stiffness-summation procedure. The locality of the test and trial functions results in a sparse global system of equations, which is advantageous with regard to computational resources.

FEM is first designed and used as a low-order approximation for the analysis of structural problems. Still today, many FEM codes use first or second order polynomial approximations. For many fluid flow problems, which require high resolution and accuracy, this is a major limitation.

Spectral Element Method (SEM) combines the competitive advantages of Spectral and Finite Element Methods [4]. Similar to FEM, it discretizes the domain into elements but not as many or as fine as that utilized in a typical finite element mesh. On these small numbers of elements, SEM uses high order Chebyshev or Legendre polynomials to achieve spectral accuracy. In SEM codes, polynomial orders of 6-12 are typically used. SEM is first designed for the solution of incompressible Navier-Stokes equations. Similar to Spectral Methods, SEM applications are still mostly fluid flow oriented [9], [10], [11]. A review of fluid flow applications using Spectral Methods can be found in [12]. Orszag and Gottlieb were among the first researchers to use Galerkin and Spectral collocation methods [13], [14], [3]. Canuto et al [8] later used SM to focus on fluid

mechanics applications. Spectral Methods were implemented in practical applications for the first time in the early 1970s. Patera [15] presented the SEM for solution of the Navier-Stokes equations. The foundation of iso-parametric SEM was provided by Korczak and Patera [16]. Ronquist and Patera [17] provided Spectral element multi-grid formulations. Karniadakis and Henderson [18] provided a recent review of incompressible flow applications. Karniadakis and Sherwin [5] present spectral element formulations for unstructured elements, and provide many large scale applications for solution of the Navier-Stokes equations.

## CHAPTER II

## SPECTRAL ELEMENT METHOD FORMULATION

**Element Mesh in the Preprocessor: Gauss Lobatto-Legendre Interpolants**

The primary objective of the preprocessor is to divide the designed domain into elements and discretize each element using appropriate order of interpolation functions. Therefore, understanding the interpolation functions, utilized by the Spectral Method is essential. In the preprocessor, each element is further discretized using the Gauss Lobatto Legendre (GLL) interpolants. The interpolation function is independent of all the partial differential equations, or their weak forms. However, we'll utilize the Poisson equation to introduce use of the GLL interpolants [4]. The weak variational form of the Poisson equation

$$-\nabla^2 u = f \quad (4)$$

in domain  $\Omega$  can be represented as

$$\int_{\Omega} (\nabla v \cdot \nabla u) d\Omega = -\int_{\Omega} v f d\Omega, \quad \forall v \in C^0, \quad (5)$$

where  $u$  is the trial function,  $v$  is the test function,  $f$  is the forcing function and uniform material constant is assumed in the Poisson equation (for simplicity). Let us divide the domain  $\Omega$  into  $K$  disjoint quadrilateral elements

$$\Omega = \bigcup_{k=1}^K \Omega_k. \quad (6)$$

It is necessary to choose the polynomial basis for approximations. Analytical integration of  $N^{th}$  order polynomial expansions is complicated. Integrations are done numerically using Gauss quadrature rules. We choose the Lagrange interpolation

polynomial based on the Gauss Lobatto Legendre points, which coincides with the corresponding Gauss Lobatto Legendre quadrature points. These  $N+1$  Gauss Lobatto Legendre points are represented as

$$L'_N(z_i)=0 \quad i=1,\dots,N-1 \quad (7)$$

$$z_N=1, \quad (8)$$

where  $(1-z_i^2)L'_N(z_i)=0$  corresponds to the points  $(z_i)$  where the first derivative of  $N^{\text{th}}$  order Legendre polynomial is zero (i.e., extremum of the  $N^{\text{th}}$  order Legendre polynomial). With this choice of collocation points, the Lagrange interpolants (Fig. 3),  $h_j$  can be written as

$$h_j(z)=\frac{-(1-z^2)L'_N(z)}{N(N+1)L'_N(z_j)(z-z_j)}. \quad (9)$$

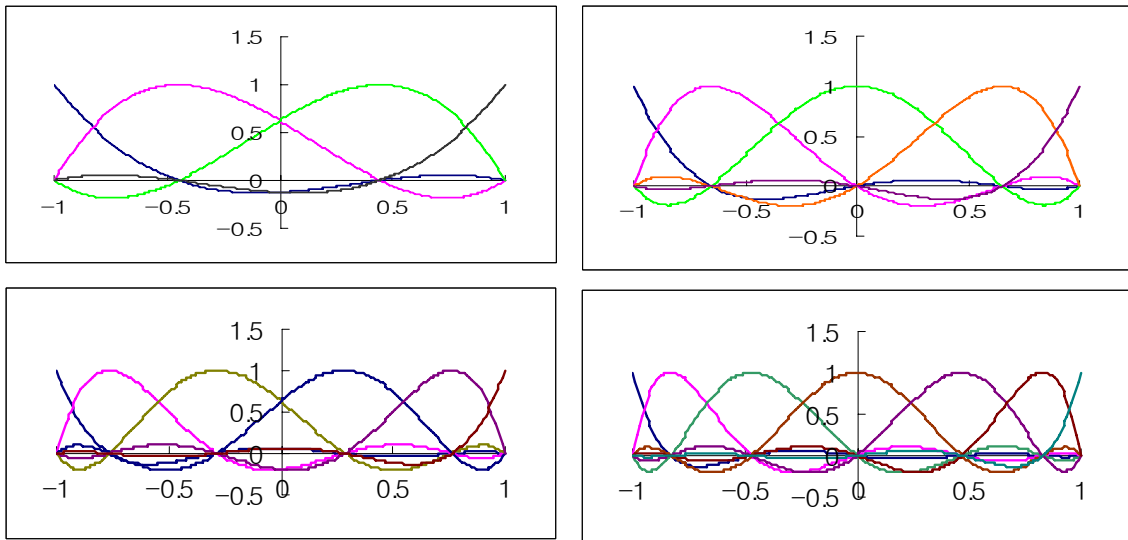


Fig. 3 The Gauss Lobatto Legendre points and interpolants ( $N = 3, 4, 5, 6$ ).

The trial function  $u$ , the test function  $v$ , and the forcing function  $f$  are expanded in terms of the tensor product form of the Gauss Lobatto Legendre interpolants as follows:

$$u^k(r, s) = \sum_{m=0}^N \sum_{n=0}^N u_{mn}^k h_m(r) h_n(s), \quad (10)$$

$$v^k(r, s) = \sum_{\alpha=0}^N \sum_{\beta=0}^N v_{\alpha\beta}^k h_\alpha(r) h_\beta(s), \quad (11)$$

$$f^k(r, s) = \sum_{m=0}^N \sum_{n=0}^N f_{mn}^k h_m(r) h_n(s), \quad (12)$$

where  $h_i$ 's are the one-dimensional Gauss-Lobatto Legendre interpolants. The approximation is called Galerkin, because the same order polynomial is used for both the trial function  $u$  and test function  $v$ . Fig. 4 illustrates (GLL) points for 2-D quadrilateral elements from the preprocessor ( $4 \times 5$  and  $30 \times 30$ ).

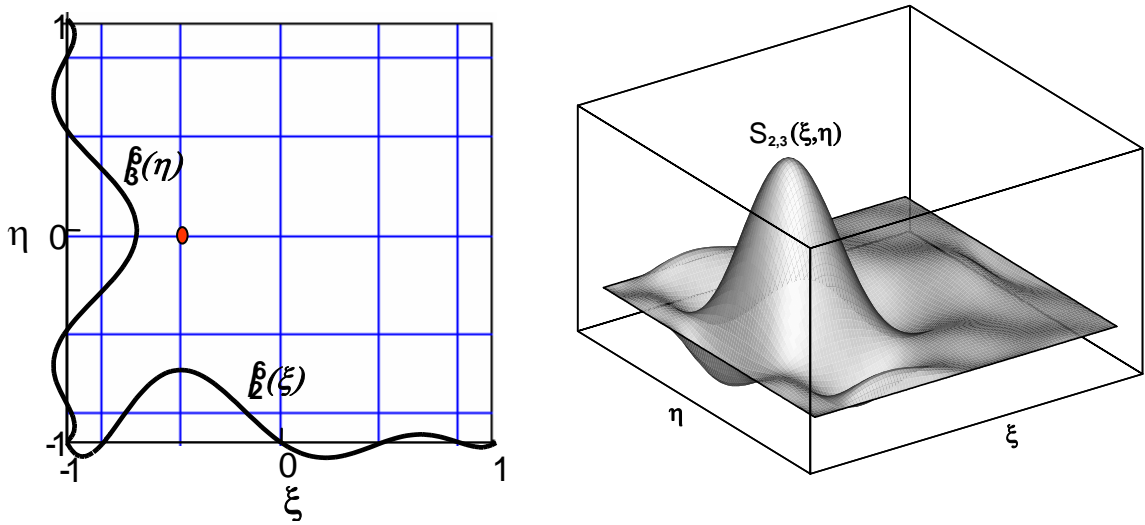


Fig. 4 Gauss-Lobatto-Legendre (GLL) points for  $6 \times 6$  quadrilateral elements (left). The corresponding 2-D shape function at GLL point (2, 3) (Sert and Beskok, 2005) (right).

Now the equivalent variational statement will be written as the sum of contributions from all the elements  $\Omega_k$ ,

$$\sum_{k=1}^{K'} \int_{\Omega_k} (\nabla v \cdot \nabla u) d\Omega_k = - \sum_{k=1}^{K'} \int_{\Omega_k} v f d\Omega_k, \quad (13)$$

where  $K'$  denotes direct stiffness summation (summation of contributions from neighboring elements at common sides and corners during the assembly of spectral element matrices).

### Various Elements in the Preprocessor: Isoparametric Mapping

The Preprocessor is capable of many types of non-rectangular and curved elements. Some of these cases are shown in Fig. 5. The GLL points in non-rectangular or curved elements are transformed using isoparametric mapping [4].

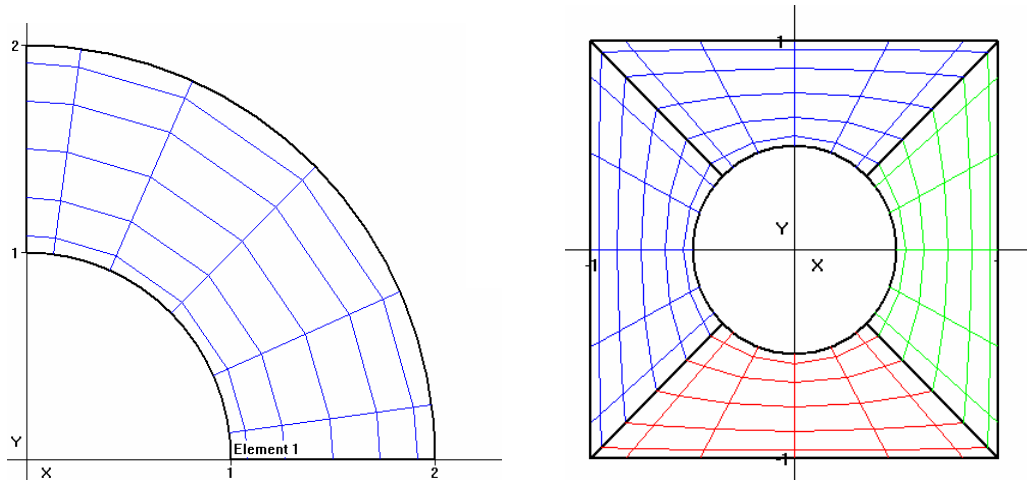


Fig. 5 Various types of elements in the preprocessor.

The elemental integrals in equation (13) are mapped into the local  $(r,s)$  coordinate system:  $(x,y) \in \Omega_k \rightarrow (r,s) \in \Lambda \times \Lambda$ ,  $\Lambda=(-1,1)$  (See Fig. 6). The local coordinate system



has the advantage of simplifying all the derivatives and integrals of the test and trial functions.

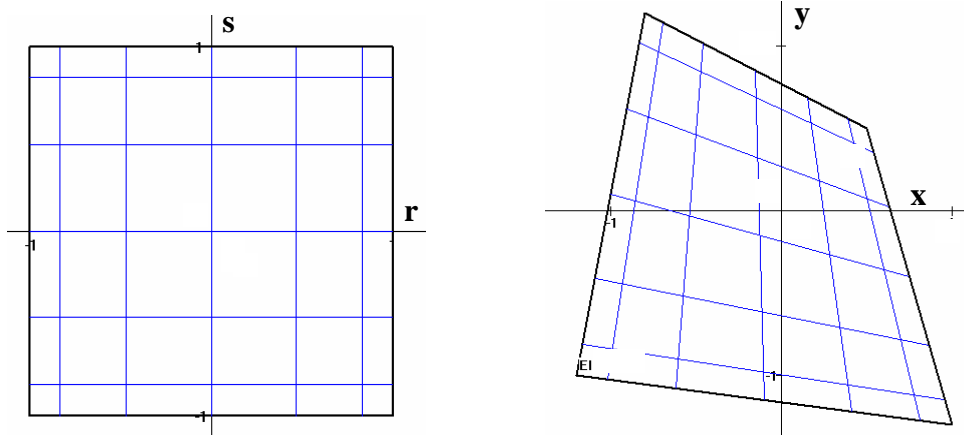


Fig. 6 Isoparametric mapping between global (x,y) coordinates and local (r,s) coordinates in the preprocessor.

For complex geometries, mapping between the global (x,y) and local (r,s) coordinates is not known. Hence, the mapping is also expanded in terms of the tensor-product form of the one-dimensional Gauss Lobatto Legendre interpolants

$$x^k(r,s) = \sum_{m=0}^N \sum_{n=0}^N x_{mn}^k h_m(r) h_n(s), \quad (14)$$

$$y^k(r,s) = \sum_{i=0}^N \sum_{j=0}^N y_{ij}^k h_i(r) h_j(s). \quad (15)$$

The mapping is called isoparametric because it utilizes the same order of approximation as the functional approximation for  $u$  and  $v$ . Now all the derivatives and integrals will be transformed to local coordinates. The transformed gradient operator arising in the variational statement is represented as

$$\tilde{\nabla} = (r_{,x}^k \partial_r + s_{,x}^k \partial_s) e_x + (r_{,y}^k \partial_r + s_{,y}^k \partial_s) e_y, \quad (16)$$

Since  $(x, y)$  is expanded in terms of  $(r, s)$  and not vice versa, it is convenient to rewrite

$\tilde{\nabla}$  as

$$\tilde{\nabla} = \frac{1}{J^k} (y_{,x}^k \partial_r - y_{,x}^k \partial_s) e_x + \frac{1}{J^k} (-x_{,x}^k \partial_r + x_{,x}^k \partial_s) e_y. \quad (17)$$

where the  $J^k$  is the Jacobian of the mapping from  $(r,s)$  to  $(x,y)$  defined as

$$J^k = x_{,x}^k y_{,s}^k + x_{,s}^k y_{,r}^k. \quad (18)$$

Now, let us transfer the integral to local coordinates

$$\int_{\Omega_k} (\nabla v \cdot \nabla u) d\Omega = \int_{\Omega_k} (\nabla v \cdot \nabla u) dx dy = - \sum_{k=1}^{K'} \int_{-1}^{+1} \int_{-1}^{+1} (\tilde{\nabla} v \cdot \tilde{\nabla} u) J^k dr ds. \quad (19)$$

The integration is replaced by the quadrature rules at the GLL points and associated numerical differentiations are performed. As a result, the weak form can be rewritten as

$$\begin{aligned} & \sum_{k=1}^{K'} \sum_{p,q=0}^N \rho_p \rho_q \left( \frac{(x_{,s}^k)_{pq}^2 + (y_{,s}^k)_{pq}^2}{J_{pq}^k} \right) \partial_r v \partial_r u + \rho_p \rho_q \left( \frac{(x_{,r}^k)_{pq}^2 + (y_{,r}^k)_{pq}^2}{J_{pq}^k} \right) \partial_s v \partial_s u \\ & - \rho_p \rho_q \left( \frac{(x_{,r}^k)_{pq} (x_{,s}^k)_{pq} + (y_{,r}^k)_{pq} (y_{,s}^k)_{pq}}{J_{pq}^k} \right) (\partial_r v \partial_s u + \partial_s v \partial_r u) \\ & = \sum_{k=1}^{K'} \sum_{p,q=0}^N \rho_p \rho_q v f_{pq}^k dr ds. \end{aligned} \quad (20)$$

Since numerical integration is performed by using the Gauss Lobatto quadrature rules,  $N^{th}$  order quadrature will be able to integrate  $(2N-1)^{th}$  order polynomial expression exactly [6]. Therefore, the numerical integration does not create significant errors.

The  $r$  and  $s$  derivatives of the tensor product expansions for  $u$  and  $v$  can be written as

$$\partial_{,r} u(r_p, s_q) = \sum_{p,q=0}^N u_{mn} h'_m(r_p) h_n(s_q) = \sum_{p,q=0}^N D_{pm} u_{mn} \delta_{nq}, \quad (21a)$$

$$\partial_{,s} u(r_p, s_q) = \sum_{p,q=0}^N D_{qn} u_{mn} \delta_{mq}, \quad (21b)$$

$$\partial_{,r} v(r_p, s_q) = \sum_{p,q=0}^N D_{p\alpha} v_{\alpha\beta} \delta_{q\beta}, \quad (21c)$$

$$\partial_{,s} v(r_p, s_q) = \sum_{p,q=0}^N D_{\alpha\beta} v_{\alpha\beta} \delta_{p\alpha}, \quad (21d)$$

For example, in equation (21a) we evaluate the derivative of  $u(r, s)$  at point  $(p, q)$  with respect to  $r$ . Since there are  $N+1$  Lagrange polynomials ( $h(r)$ ), we sum the derivatives of each polynomial at point  $(p, q)$ , and use the fact that  $h_n(s_q) = \delta_{nq}$ . The term  $D_{ij}$  is called the derivative matrix, and its components are given as

$$D_{ij} = \frac{L_N(z_i)}{L_N(z_j)(z_i - z_j)} \quad i \neq j,$$

$$D_{ii} = 0, \quad i \neq 0, N$$

$$D_{00} = -N(N+1)/4,$$

$$D_{NN} = N(N+1)/4. \quad (22)$$

Now, by substituting tensor-product expansion for the derivatives into the variational statement, we obtain a system of algebraic equations of the form

$$\sum_{k=1}^{K'} (A_{\alpha\beta mn}^k) u_{mn}^k = \sum_{k=1}^{K'} B_{\alpha\beta mn}^k f_{mn}^k, \quad (23)$$

where  $u_{mn}^k$  is the unknown and  $f_{mn}^k$  is the forcing function,

$$\begin{aligned}
A_{\alpha\beta mn}^k &= (\mathbf{g}_{,s})_{pq}^K D_{p\alpha} D_{pm} \delta_{qn} \delta_{q\beta} v_{\alpha\beta} + (\mathbf{g}_{,s})_{pq}^K D_{qn} D_{q\beta} \delta_{pm} \delta_{p\alpha} v_{\alpha\beta} \\
&+ (\mathbf{g}_{,rs})_{pq}^K D_{pm} D_{q\beta} \delta_{qn} \delta_{p\alpha} v_{\alpha\beta} + (\mathbf{g}_{,rs})_{pq}^K D_{p\alpha} D_{qn} \delta_{q\beta} \delta_{pm} v_{\alpha\beta}, \\
B_{\alpha\beta mn}^k &= \rho_p \rho_q J_{pq}^K \delta_{p\alpha} \delta_{q\beta} \delta_{pm} \delta_{qn} v_{\alpha\beta}, \\
(\mathbf{g}_{,s})_{pq}^k &= \frac{\rho_p \rho_q}{J_{pq}^k} \left[ (x_{,s}^k)_{pq}^2 + (y_{,s}^k)_{pq}^2 \right], \\
(\mathbf{g}_{,r})_{pq}^k &= \frac{\rho_p \rho_q}{J_{pq}^k} \left[ (x_{,r}^k)_{pq}^2 + (y_{,r}^k)_{pq}^2 \right], \\
(\mathbf{g}_{,rs})_{pq}^k &= -\frac{\rho_p \rho_q}{J_{pq}^k} \left[ (y_{,r}^k)_{pq} (y_{,s}^k)_{pq} + (x_{,r}^k)_{pq} (x_{,s}^k)_{pq} \right]. \tag{24}
\end{aligned}$$

The products in  $A_{\alpha\beta mn}^K$  and  $B_{\alpha\beta mn}^K$  can be collapsed by eliminating the Kronecker deltas. The test function  $v_{\alpha\beta}$  is common in both  $A_{\alpha\beta mn}^K$  and  $B_{\alpha\beta mn}^K$  terms in equation (23). Thus the test function eventually drops from the spectral element equations [4].

Equation (23) represents a system of linear equations, and its solution provides values of  $u_{mn}$  at discrete points  $(m, n)$ . Since the field variable of Poisson equation is a continuous variable  $(u(\mathbf{x}))$ , we can reconstruct the approximation of  $u(\mathbf{x})$  by using the eigen function expansions for  $u(\mathbf{x})$  given in equation (10) and the values of  $u_{mn}$ . This formulation uses the Galerkin approach which minimizes error in the whole solution domain. Upon convergence, our results are accurate everywhere in the domain.

The solution of system of linear equations, given by equation (23) can be achieved by direct methods such as LU decomposition, or iterative methods like conjugate

gradient methods. Efficient solutions of a system of linear equations are important for fast solution of larger problems.

### **Convergence**

It is important to discuss the convergence of h and p type refinements of the SEM before demonstrating other numerical examples. In order to numerically resolve the problem (i.e., to converge to the analytical solution), two basic methodologies can be used. The first method is to progressively increase the number of elements, by keeping the order of each element the same. Since the order of element is fixed, the numerical solution converges algebraically until the error is small enough to be neglected. This type of mesh refinement is called h-type refinement (Fig. 7. (a)). For example, if first order polynomial expansions are used, the method will be second order accurate in space. Hence doubling the number of elements per direction will reduce the error by a factor of four.

The second alternative is to keep the number of elements the same while increasing the order of each element. This type of mesh refinement is called p-type refinement (Fig. 7. (b)). Convergence of the method is exponential provided that the solution is analytic ( $C^\infty$ ), and there are no singularities in the geometry. Otherwise, convergence will be algebraic, depending on the regularity of the solution. For example, if the order of polynomial expansions is increased by factor of two, the error will be reduced by two orders of magnitudes. This type of convergence is called “spectral” convergence (Fig. 8).

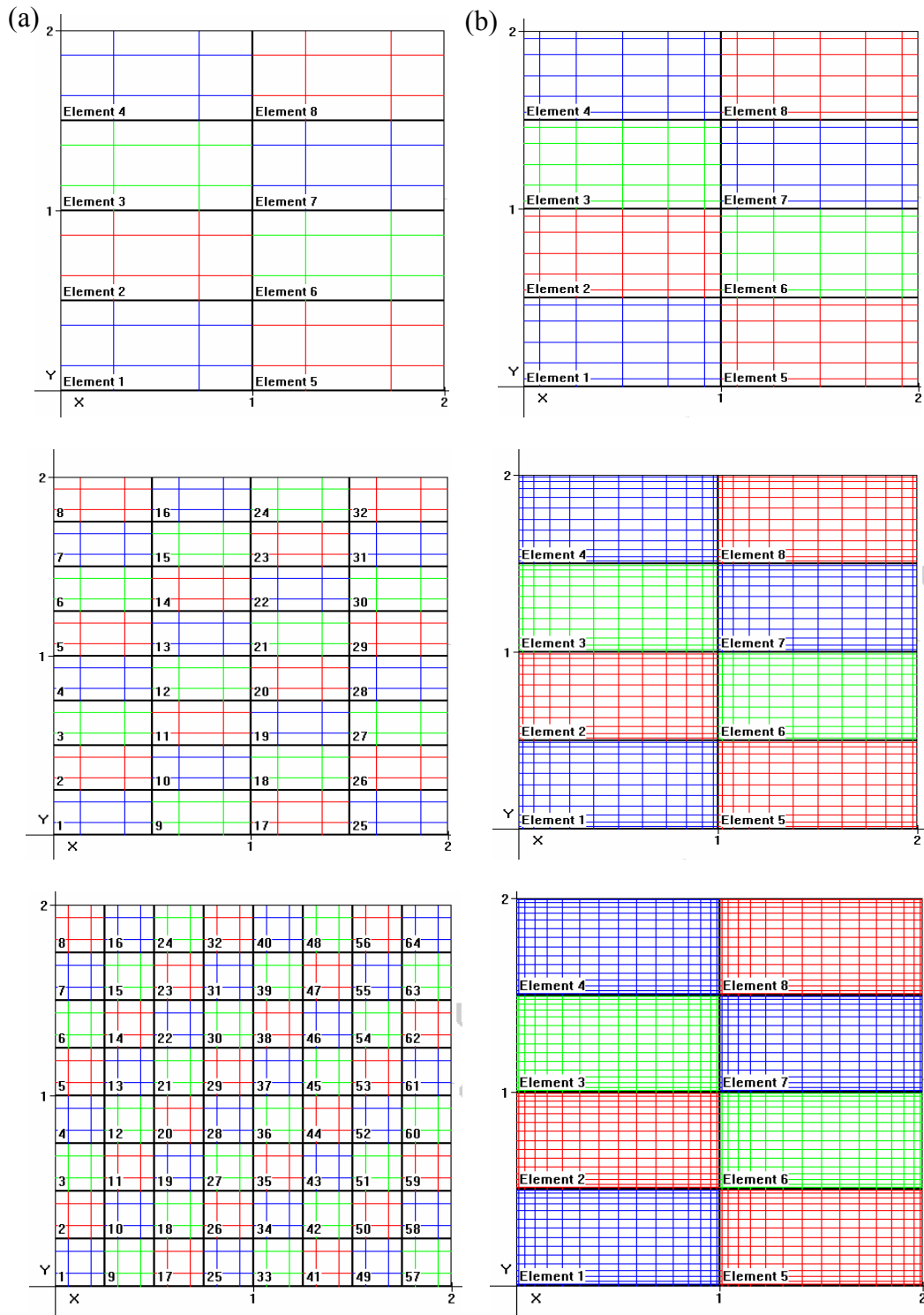


Fig. 7 (a) h-type refinement for the Kovaszny flow geometry using: 8, 32, 64 elements with 3<sup>rd</sup> order discretization per direction. (b) p-type refinement using: 8 elements with 6, 12, 16<sup>th</sup> order approximation per direction.

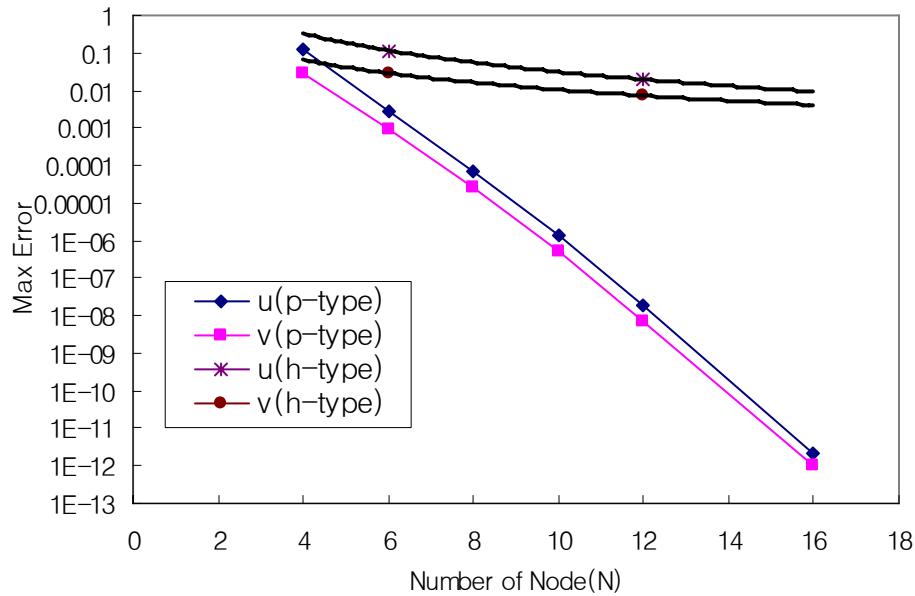


Fig. 8 Convergence obtained from Kovasznay flow under: p-type and h-type refinements.

An  $N^{\text{th}}$  order Spectral Element Method requires only  $N$  times more work compared with the conventional h-type techniques using the same number of degrees of freedom. Since the Spectral Element Method achieves a high level of accuracy exponentially, the total computational work in SEM will be less than those of low order FEM, when the result of both methods are compared with each other at sufficiently low error tolerances.

The Spectral Element Method is usually compared with p-type Finite Element Method (pFEM), which is an extension to the classical FEM, providing p-type refinements [19]. The main difference between pFEM and SEM lies in the choice of the trial and test functions. While pFEM utilizes hierarchical functions, approximation functions in SEM are based on the Jacobi polynomials.

## **Nonconforming Interfaces**

Local mesh refinement is very significant in numerical simulations. In the global domain, some areas require more refined meshes than the others. For example, in the typical test case of flow over a cylinder, the thin boundary layer around the cylinder and the wake behind the cylinder are the most important parts of the system where most of the flow physics happens. Another example is the lid-driven cavity problem where mesh refinements are required near the corners to confine the geometric singularities. Therefore these regions need to be refined better than others, like the inflow or far field regions. Often the mesh needs to be refined continuously during the solution process based on some error estimations.

The most basic technique of acquiring high resolution is to use smaller size elements in the regions where higher resolution is required. This is called h-type refinement because it changes the element sizes, which are usually denoted by 'h'. In h-type refinement case, the number of elements is increased. This can be avoided using h-type (geometrically) non-conforming elements. The geometrical nonconformity means that the intersections of neighboring elements are not matched with a whole face or a vertex. We use term h-type nonconformity for the geometrical nonconformity because it is usually caused by an h-type refinement.

The second type of refinement is the p-type refinement. In this case the number of elements and their sizes are kept the same but the order of approximation function is increased where higher resolution is required. This type of refinement is called p-type nonconformity because the interfaces of the neighboring elements match geometrically.



However, the unknowns defined on those faces do not match, since the neighboring element has different order of approximation.

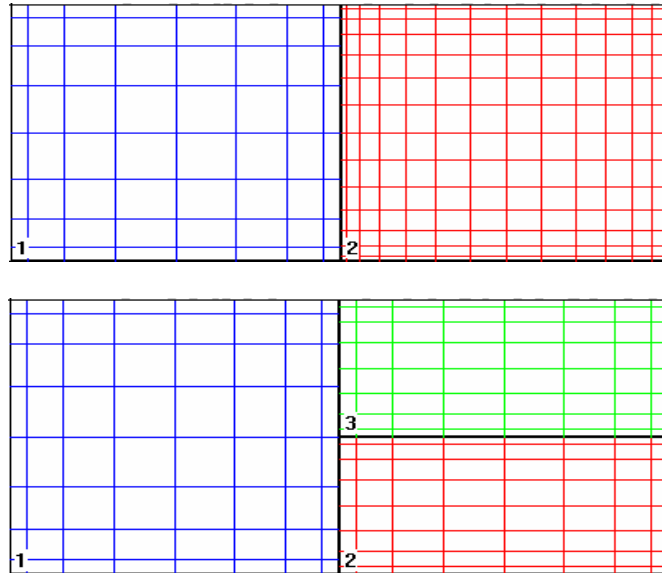


Fig. 9 The p-type (upper) and h-type (lower) nonconforming interface.

Both h- and p-type nonconformities (Fig. 9) require special treatments, when the equations from neighboring elements are assembled. There are several techniques to formulate these nonconformities. Constrained Approximation Method (CAM) is one of the early methods developed and is very simple in theory [20]. CAM is based on the interpolation of unknowns at nonconforming interfaces and provides a point wise projection. CAM is popular in the Finite Element Method. Mortar Element Method (MEM) is another technique, in which the jump across nonconforming interfaces are minimized in a weighted integral sense. It is first developed by the Spectral Element community, used mostly in spectral solvers [21].

## CHAPTER III

### PREPROCESSOR

#### Nomenclature for the Work Space

The main screen of the preprocessor consists of the information and mesh windows. The information window shows properties of the element, coordinates of selected point, and instructions for the next step. The mesh window has two modes, one is the mesh mode and other is the edit mode. The mesh mode is for drawing mesh domain with various types of elements, while the edit mode is for setting up boundary conditions and parameters, exporting files, element modification, etc. Every command on the menu bar is implemented as a tool bar for user convenience. Moreover, a help message is shown to the user, when the cursor is held over the tool bar button. And the information window shows the current coordinates of the cursor, element information, and ‘To do’ instructions. The preprocessor instructs and assists users with messages in the information window for each step of the mesh generation. Therefore, it is possible to generate the mesh in a few steps. Figure 10 shows the nomenclature of the workspace.

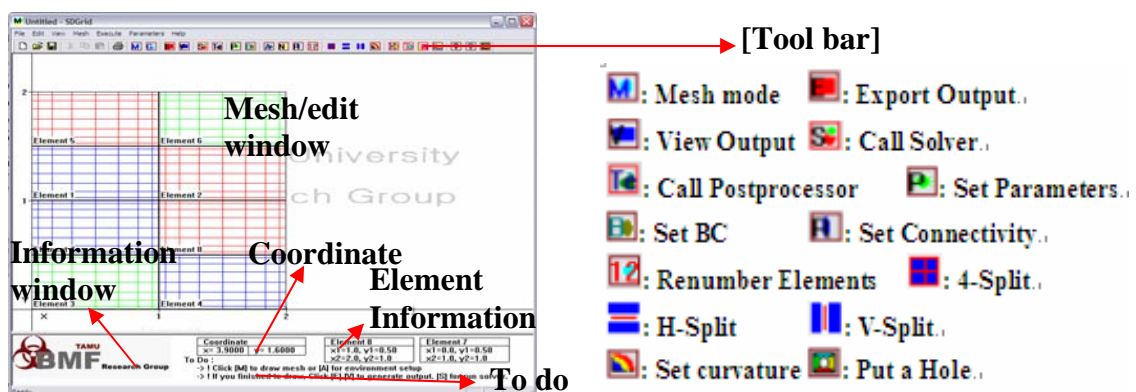


Fig. 10 Nomenclature of work space

## Output File

Since the preprocessor provides the text output file which can be utilized by the solver, it is very important to understand the output file. The output file has all the information necessary to solve the designated problem. There is a single output file, which is called Output-inp.txt. Any solver can be modified to read the Output-int.txt at the start of the solution step. The output file is generated by the preprocessor. However, it also can be typed in directly using any text editor. The latter requires tremendous work. The format of the output file has 8 sections. Each section starts with a heading.

SECTION: PARAMETERS

SECTION: FACE BOUNDARY CONDITIONS

SECTION: INITIAL CONDITIONS

SECTION: FORCING

SECTION: EXACT SOLUTION

SECTION: COORDINATES

SECTION: CURVATURE

SECTION: CONNECTIVITY


When the output file is generated without the preprocessor, these headings should be typed exactly as shown above. There should not be empty lines between a section heading and the section itself because the processor will be unable to read the output file if the format is not appropriate.

PARAMETERS [

**Table. 1** Description of parameters in the preprocessor

<b>Ndf</b>	Number of degrees of freedom. Set it to 1 for Diffusion problem and 2 for Navier-Stokes problems.
<b>Ne</b>	Number of elements.
<b>Steady</b>	Set it to 1 for steady problems and 0 for unsteady problems
<b>nts</b>	Number of time steps for unsteady problems.
<b>ndump</b>	An output file will be generated at every ndump <sup>th</sup> time step.
<b>dt</b>	Time step for unsteady problems. Its value is a real number.
<b>method</b>	Set it to 1 to use MEM, and 2 to use CAM.
<b>minRule</b>	Set it to 1 for minimum rule at p-type interfaces, and 0 for maximum rule.
<b>exact</b>	Set it to 1 if the exact solution is known, and 0 if not.
<b>diff</b>	Diffusivity. Diffusion problems only.
<b>ConstVel1</b>	Constant u-velocity for convective problems
<b>ConstVel2</b>	Constant v-velocity for convective problems.
<b>visc</b>	Kinematic viscosity. Note that density is always 1.0.
<b>stokes</b>	Set it to 1 to solve Stokes equations, and to 0 to solve Navier-Stokes problems.
<b>maxIter</b>	Number of iterations to implicitly treat the convective terms in the equation.

Such tedious input structure can be easily handled using the input dialog sections of the parameters section in the preprocessor, as shown in Fig. 11.

**BOUNDARY CONDITIONS** : The preprocessor generates the number of different boundary conditions (BCs) and their types and values. In the preprocessor, the BCs can be applied on elements coinciding with domain boundaries. First line is for the number of different BCs typed in the Boundary condition dialog. It is an integer value. Following lines give the type of BC and the numerical values at each boundary. First value is the BC number. Second value is set to 1 for essential (Dirichlet) and 2 for natural (Neumann) BCs, as shown in Fig. 12. The values of the boundary conditions can be either numbers or equations with variables. The text format of FORTRAN code is

recommended for equations. The assigned number of each boundary condition can be referenced in the connectivity section in the element dialog box.

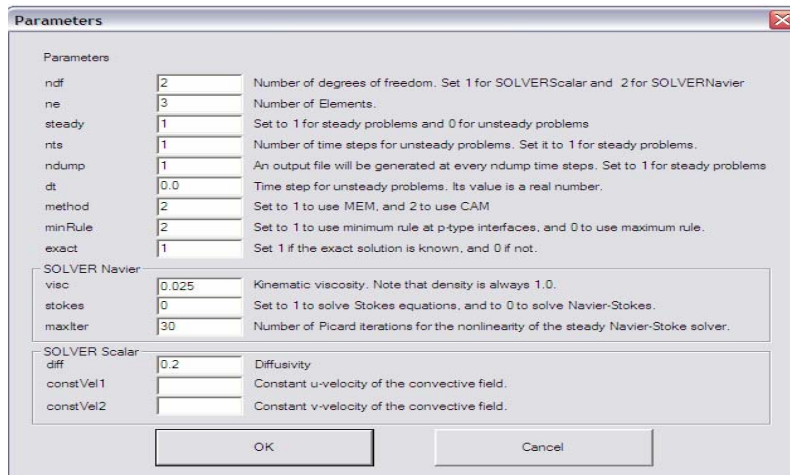


Fig. 11 Parameter input dialog

For Navier-Stokes problem, two BCs at each boundary, one for u-velocity and one for v-velocity (no BC is necessary for the pressure) are assigned. Therefore for each boundary there will be two lines of information as shown in Fig. 12.

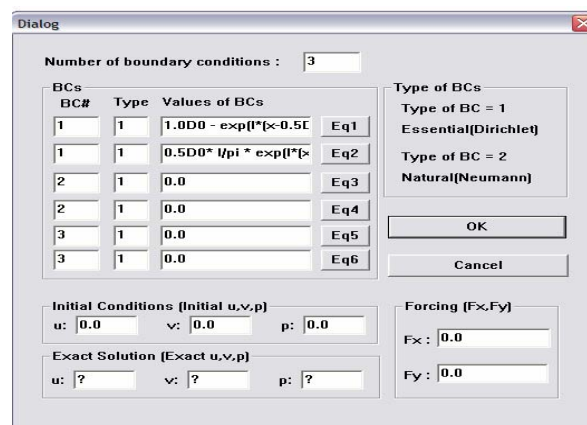


Fig. 12 Boundary conditions, initial conditions, forcing function and exact solutions in the BCs input dialog.

**INITIAL CONDITIONS:** This provides mathematical functions for initial conditions, which can be typed in the dialog box at the bottom part of BCs input dialog.

**FORCING:** A known forcing value with in the element can be typed at the bottom right part of BCs input dialog.

**EXACT SOLUTION:** If known, an exact solution can be typed at the bottom part of the BCs input dialog. This option is used to verify the code and its convergence using problems with analytic solutions.

**COORDINATES:** Element numbering of vertices and sides has fundamental importance in the preprocessor. All the connectivity, boundary conditions and curvature is defined by the elemental numbering scheme. The numbering scheme for vertices and sides of the elements are shown in Fig. 13. The preprocessor print out the coordinates of the corner points of each element as shown in Fig. 14. The preprocessor uses only quadrilateral elements with four corner points. Corners are numbered Counter Clock Wise (CCW) as shown in Fig. 13. In the output file, the element number comes first. This is line followed by four lines of x and y coordinates for corners of the element.

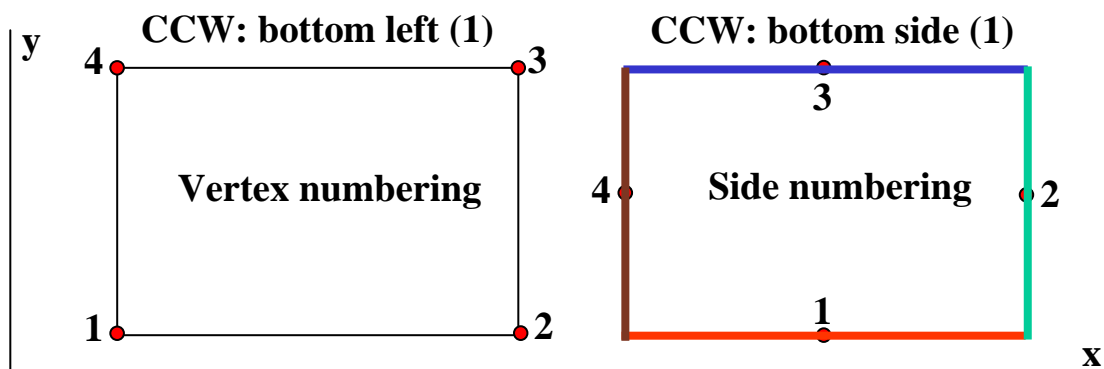


Fig. 13 Numbering scheme of vertices and sides.

The figure displays two screenshots of a coordinate input interface for an element dialog box. The top screenshot shows the 'Non-Rectangular Element' checkbox unchecked, and the bottom screenshot shows it checked. Both screenshots show input fields for element number, expansion orders, and coordinates.


**Top Screenshot (Rectangular Element):**

- Element Number:** Current Number: 1, Change to: 1
- Expansion Orders:** Order X: 6, Order Y: 6
- Element Coordinates:** x1: 0., y1: 0., x2: 2., y2: 1.55
- Non-Rectangular Element:**
- Element Coordinates (CCW Order from lower-left):** x1: 0., y1: 0., x2: 2., y2: 0., x3: 2., y3: 1.55, x4: 0., y4: 1.55
- Curvature & Curve #:** Bottom Side: 0, Right Side: 0, Top Side: 0, Left Side: 0

**Bottom Screenshot (Non-Rectangular Element):**

- Element Number:** Current Number: 1, Change to: 1
- Expansion Orders:** Order X: 6, Order Y: 6
- Element Coordinates:** x1: 0., y1: 0., x2: 2., y2: 1.55
- Non-Rectangular Element:**
- Element Coordinates (CCW Order from lower-left):** x1: 0., y1: 0., x2: 2., y2: 0., x3: 2., y3: 1.55, x4: 0., y4: 1.55
- Curvature & Curve #:** Bottom Side: 0, Right Side: 0, Top Side: 0, Left Side: 0

Fig. 14 Coordinate input interface in the element dialog box with rectangular (top) and non-rectangular elements (bottom).

**CURVATURE** [  ]: In the preprocessor, only circular curvature is supported. Fig. 15 shows the curvature input dialog box. The curvature defined here can be used when the non-rectangular element assigns the curvature number on its side. Up to 9 curvatures can be assigned using the dialog box. The preprocessor prints out the curvature information and the number of curved side of the element in the output file.

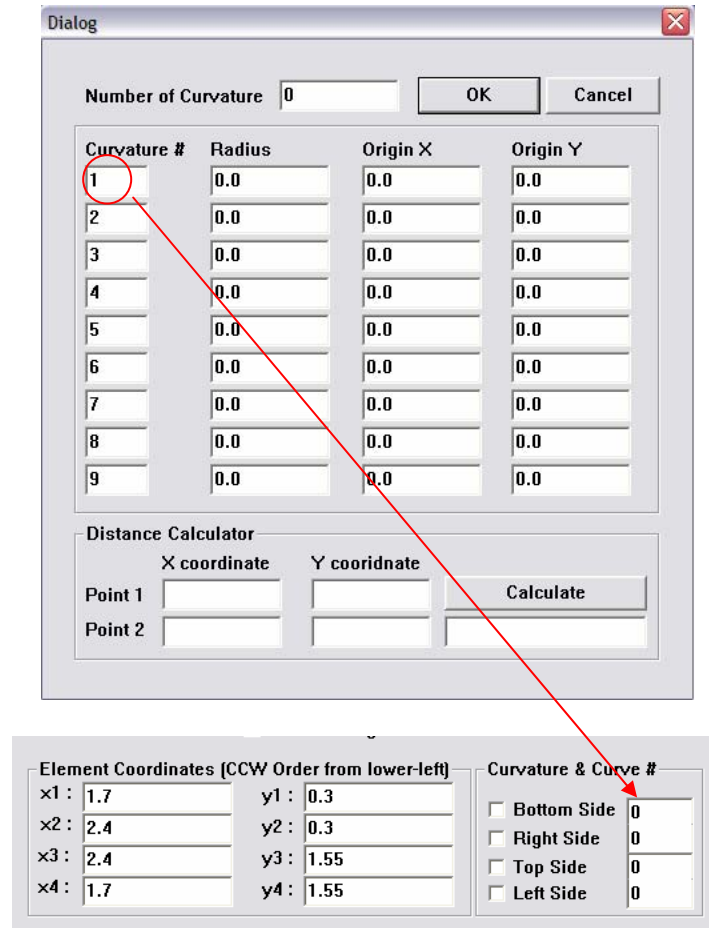


Fig. 15 The curvature input dialog and assignment of curvature on the side.

When the preprocessor extracts the output file, the curvature section looks as follows:

```

3                ! number of curvatures
1 1.0 0.0 0.0  ! curvature no, radius, origin x, origin y
2 2.0 0.0 0.0
3 3.0 0.0 0.0

```

Fig. 16 The output format for curvature in CURVATURE section.

In Figure 16, the first number, 3, tells that there are three types of curvatures in this mesh. The next three lines define these curvatures. Let's examine the first curvature



defined by “1 1.0 0.0 0.0”. First number “1” is the curvature number, “1.0” is the radius of the curvature, and last two numbers are the x and y-coordinates of the origin of the circle that defines the curvature. After the definitions of curvatures, information about the curved element faces that utilize the particular curvature information comes.


```

6          ! number of curved faces
1 1 1     ! element no, face no, curvature no
1 3 2
2 2 2
2 4 1
3 2 3
3 4 2

```

Fig. 17 The assigned curvature information in the output text file.

In figure 17, the first number “6” tells us that there are 6 curved faces. The next six lines define these faces. For example the last curved face is defined by “3 4 2”, where “3” and “4” indicate the fourth face of the third element. The last number “1” indicates that this face utilizes the first curvature defined above. If a curved face is shared by more than one element, all of them must be listed separately.

CONNECTIVITY [  ]: This section specifies the order of expansion of each element, as well as the face connectivity data, which describes how the elements are connected to each other. In the preprocessor, this information is generated automatically and can be displayed by a dialogue box shown in Fig. 18.

Connectivity					
	TYPE	Condition	Connection		Connect to (extension)
1. Bottom side	1	B	1	0	
2. Right side	1	B	1	0	
3. Top side	1	E	6	1	
4. Left side	2	E	2	2	E 1 2

E#    Side#    "E\_\_" + "E#\_" + "Side#" ...

Fig. 18 Connectivity and boundary conditions in the element dialog.

In the output text file, element order and connectivity are represented as


```

3 6 8 0
1 B 1 0
1 B 1 0
1 E 6 1
2 E 2 2 E 1 2

```

Fig. 19 The boundary and connectivity information in the CONNECTIVITY section.

In Fig. 19, the first number ‘3’ is the element number. Following “6” and “8” are the expansion orders in x and y directions, respectively. As seen here, elemental expansion orders in x and y direction can be different, which is called anisotropic expansion. Next “0” is reserved for adaptive refinement purposes, and it is not used currently.

The next four lines give the connectivity data of the four faces of this element. Faces are numbered CCW, first face being the one between the first and second corners (see Fig. 13). However, the preprocessor automatically sets up the connectivity when the user presses the connectivity button .

The bottom side face (face 1 in Fig. 13) which has the information “1 B 1 0”. The first number, 1, indicates that this face is of type 1. B indicates that this face lies on a

real boundary (not connected to another element). The next number, 1, shows the boundary condition on this face. Next “0” is a dummy parameter, and it is not used if the face is located on a real boundary face (face 3 in Fig. 13).

The top side, which has the data “1 E 6 1”. The first number 1 indicates that this face is of type 1. Next “E” indicates that this face is connected to another element. Next “6” and “1” indicate that this element is connected to the first face of element 6. Finally let’s study the connectivity data of the fourth face given by “2 E 2 2 E 1 2”. First “2” indicates that this face is of type 2. Type 2 faces are the long faces at h-type nonconforming interfaces (see Fig. 9). This long face is connected to two short faces. First connection is with face 2 of element 2 and second connection is with face 2 of element 1.

Following is a summary of the face connectivity data:

Face types:

- 1: Face on a boundary, or face connected to another face that is geometrically matching.
- 2: Long face at an h-type nonconforming interface (see Fig. 9).
- 3, 4: Short faces at an h-type nonconforming interface (see Fig. 9). When the neighbors of the corresponding long face in CCW direction is listed, the first short face encountered is of type 3 and the second short face encountered is of type 4.

Location types:


B: Face is located on a real boundary.

E: Face is connected to another element.

P: Same as E type, but used for faces located on periodic boundaries.

O: Faces located on out flow boundaries.

### Start drawing mode

Drawing a mesh begins with the drawing mode. In order to build a global mesh of the whole domain, draw a few large elements to represent the lay out of the whole domain, and use the function of ‘splitting’ and ‘edit element’ to refine the mesh in edit mode. The following detailed steps explain the whole procedure to discretize a simple problem. First, click the  button to start the drawing mode. This changes the preprocessor from the ‘edit mode’ to ‘drawing mode’ (Fig. 20).

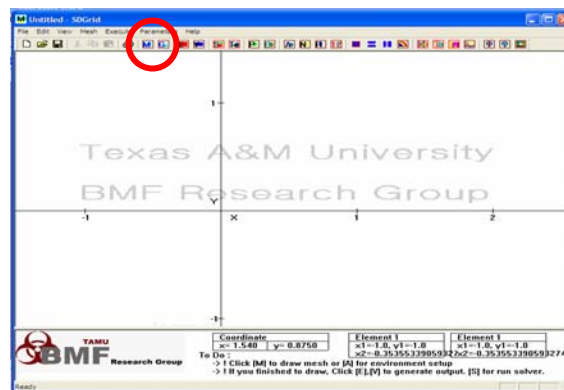



Fig. 20 Click the mesh button to start the drawing mode.

Before starting the drawing mode, the preprocessor asks to set up the parameters and boundary conditions of the designed problem. These parameters and boundary conditions are already explained in the previous section. Press ‘OK’ after to set up the parameters are finished. This dialog box can be called later by pressing  or selecting ‘parameters’ from menu to change the parameters of the problem (Fig. 21).

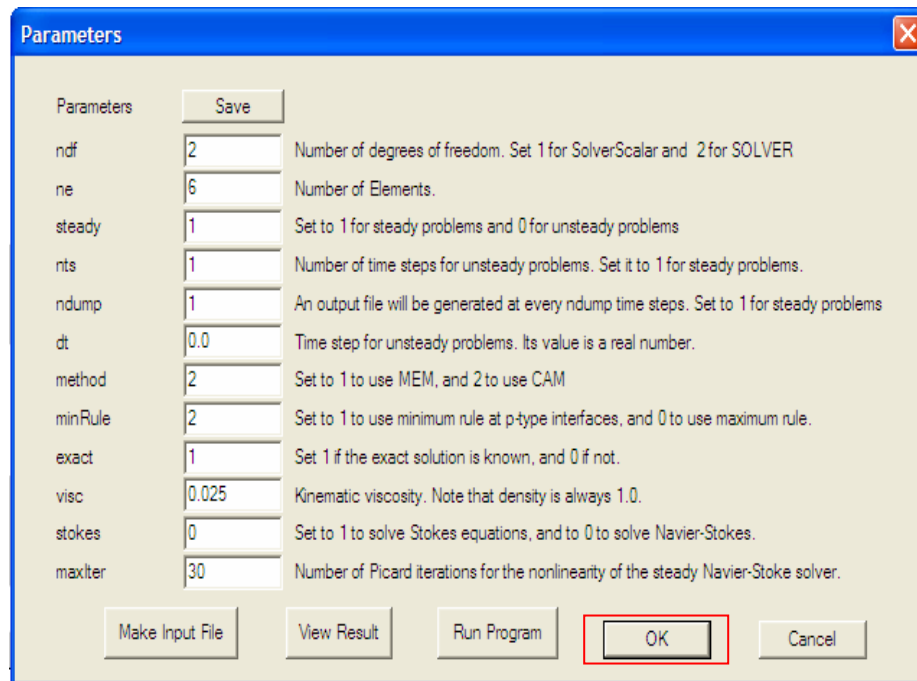



Fig. 21 Press ‘OK’ in the parameters dialog box to start mesh mode.

The parameter ‘ndf’ which shows the number of elements cannot be modified by users. Each time when the user finished drawing an element, the number of elements ‘ndf’ will be automatically updated to the current number of elements when the parameters dialog box is called. After setting up the parameters, the boundary conditions and initial conditions are input in the dialog box that appears as shown in the previous section. Press ‘OK’ to proceed to drawing elements (Fig. 22). This dialog box can be recalled later with the  button.

Number of boundary conditions : 3

BCs	Type of BCs	Values of BCs
1	1	0.0
1	1	0.0
2	1	0.0
2	1	0.0
3	1	0.0
3	1	0.0

Type of BCs  
 Type of BC = 1  
 Essential(Dirichlet)  
 Type of BC = 2  
 Natural(Neumann)

OK

Cancel


Initial Conditions (Initial u,v,p)  
 u: 0.0 v: 0.0 p: 0.0

Exact Solution (Exact u,v,p)  
 u: ? v: ? p: ?

Forcing (Fx,Fy)  
 Fx: 0.0  
 Fy: 0.0

Fig. 22 Boundary conditions and initial conditions. Press 'OK' to proceed.

### Drawing Elements

This section describes the procedure for drawing elements. Lower-right corner of the element is already selected to be (0, 0). To practice drawing an element, (2, 2) coordinate is clicked for the right-top point of element 1 (Fig. 23). The pointer is adjusted to fit in an invisible grid which sets the point to the grid coordinates. This makes it easy for the users to select specific points. The size of the grid can be adjusted in the attribute dialog box by pressing [] button (Fig. 24). The element dialog box appears after clicking the upper left corner, as shown in Fig. 23.

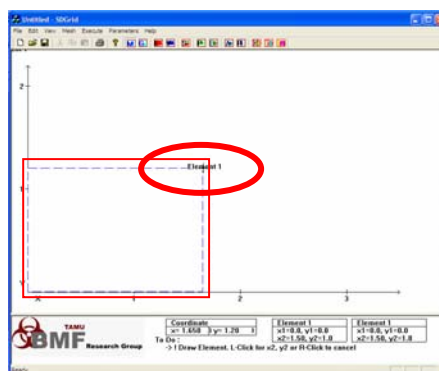


Fig. 23 Draw an element: selecting upper right vertices of an element.

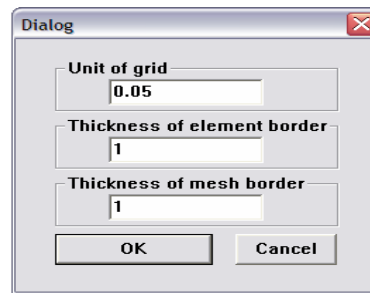


Fig. 24 The default values of grid size and the line thickness in the attribute dialog.

### Modify Elements.

When you click for the upper right corner of the element, a dialog box appears and the information (order, coordinate, connectivity) of the element is shown in this dialog box (Fig. 25). Also, the information of the drawn element can be modified using this dialog box. Basically, the drawn element is a rectangular element. This means that the element can be described by two points (lower-left and upper-right vertices).

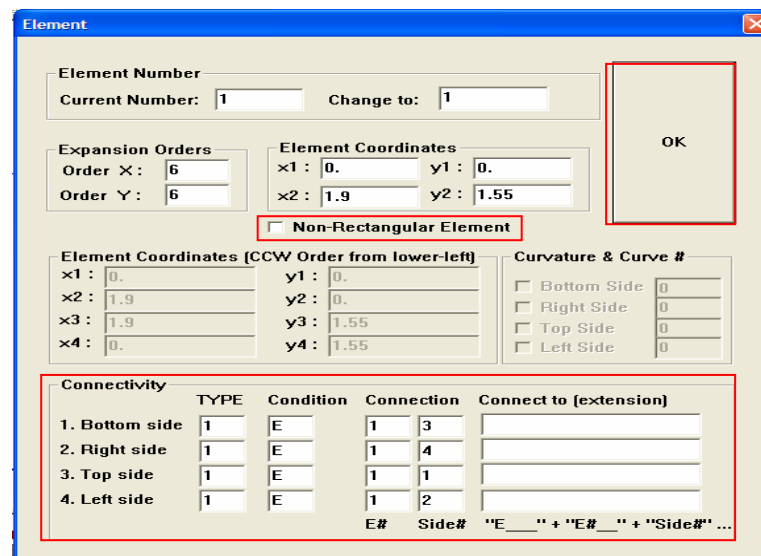


Fig. 25 Element dialog box.

In order to draw non-rectangular elements, click the check box next to ‘Non-Rectangular Element’ (Fig. 25). By checking this box, the element can be shaped in any arbitrary geometry with four vertices. When using the curved element on the specific side, the check box of the required side is checked and the assigned curvature number specified (Fig. 26).

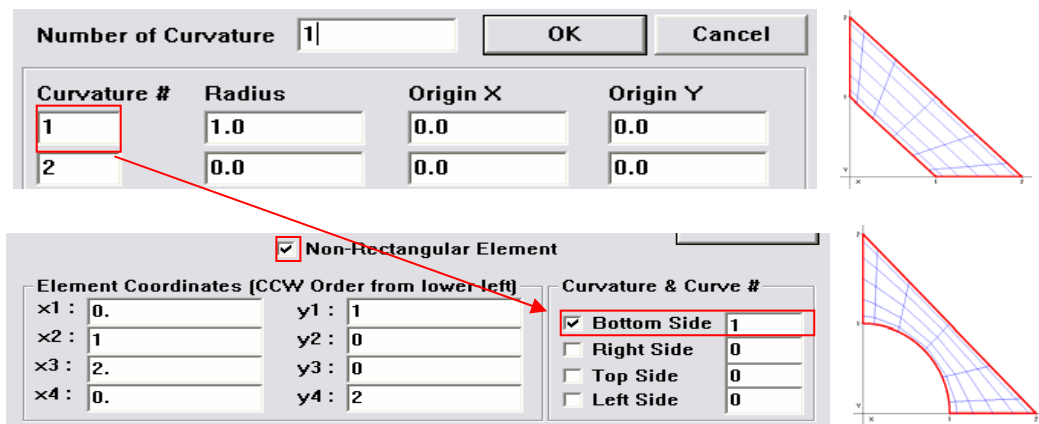




Fig. 26 Set a curvature to the bottom side of the element.

The connectivity is automatically set up by pressing the  button in the tool bar. However, the boundary conditions needed to be set up by the user after the connectivity set up. The drawn element can be canceled (erased) with the right-click of the mouse button.

### Edit Mode

After drawing the rectangular element with  $[(0, 0), (2, 2)]$  vertices, Press  to finish drawing mode and return to edit mode. In the edit mode, the user can perform tasks like splitting elements or modifying drawn elements. To modify drawn elements, the user needs to select a target element.



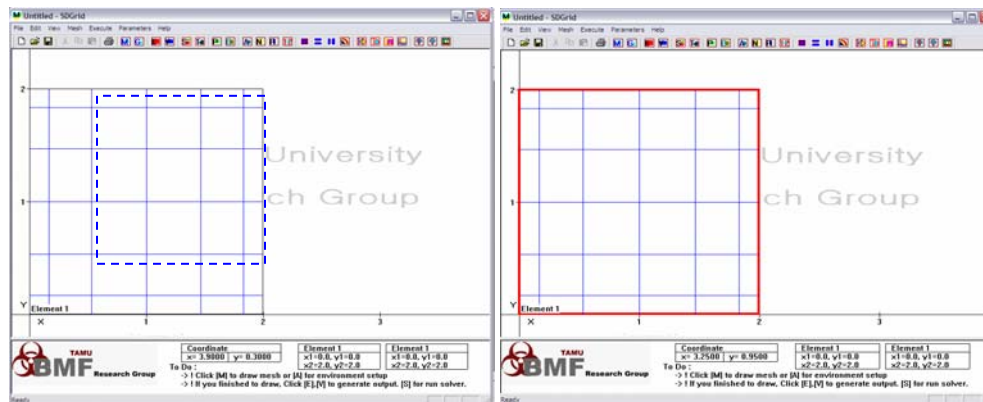


Fig. 27 Select a target element by dragging a mouse (left) over the target element.

An element can be selected by dragging the mouse over the target element. When the element is selected, the border line of the element changes to thick red line (Fig. 27). After selecting an element, the user can perform splitting or call the element dialog box to modify the element. To call the element dialog box, the right button of mouse is used (Fig. 28).

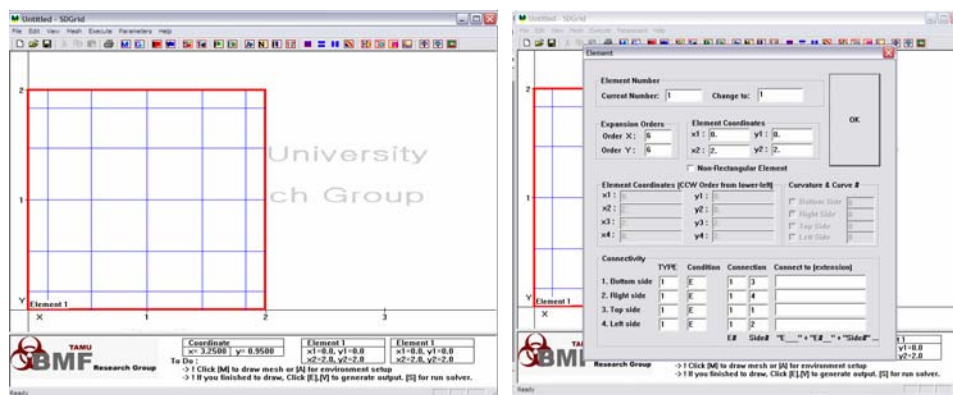


Fig. 28 After selecting an element, the element can be modified in the element dialog box.

## Splitting Elements [ , , ]

Element splitting is one of the most important functions to describe a large domain in the preprocessor. Usually, in the case of the global domains, consisting of many numbers of elements, it is unnecessary to draw each element individually. After drawing few elements which represent the layout of the whole domain, the element is discretized using the function of splitting. For drawing an element which represent the whole domain, draw an element with  $[(0, 0), (2, 2)]$  vertices.

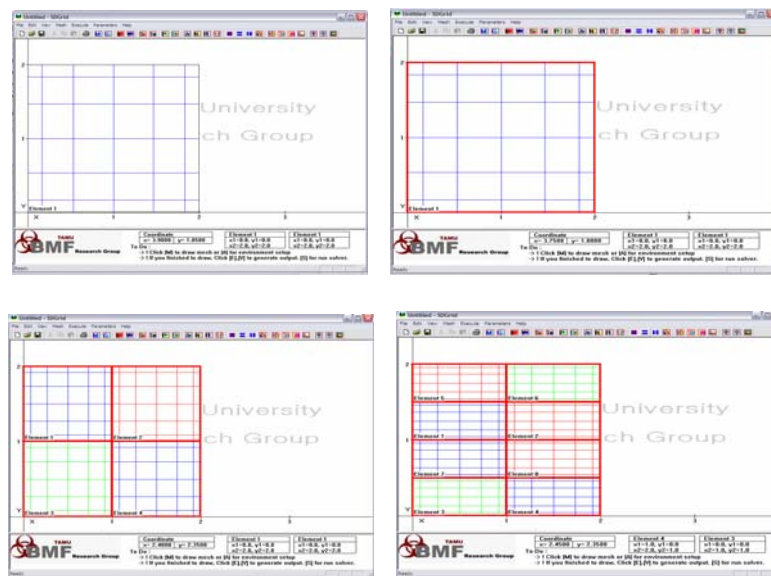








Fig. 29 Select a target element and performing 4 split and horizontal split.

Then, first press 4-split button [  ] and then, press the horizontal split button [  ] to split a target element into 8 elements (Fig. 29). The elemental numbering changes after splitting. To change the numbering of the element from left to right and bottom to top, press the renumbering button [  ]. After that, the connectivity of the element needs to be set up. For the connectivity, press the connectivity button [  ]. Now, the element

numbering is changed from left to right and bottom to top, and the connectivity of every element is set up. After setting up the connectivity, the user can specify the boundary conditions in the connectivity section in the element dialog. Finally, the parameters and boundary conditions are checked before running the solver. Since all the default parameters and boundary conditions are set as for the Kovaszny flow (see chapter IV), the processor is now ready to solve the Kovaszny flow example.

### Export Output File

In order to generate the output file [output-inp.txt] for the solver, press . Output-inp.txt will be generated from the preprocessor. The solver will read this file to solve the problem and generate the result file. Any type of solver which can read this Output-inp.txt file would solve the same problem. For this reason, this preprocessor can be utilized by any solver with Spectral Element Methods. Press  to view the generated output file (Fig. 30).

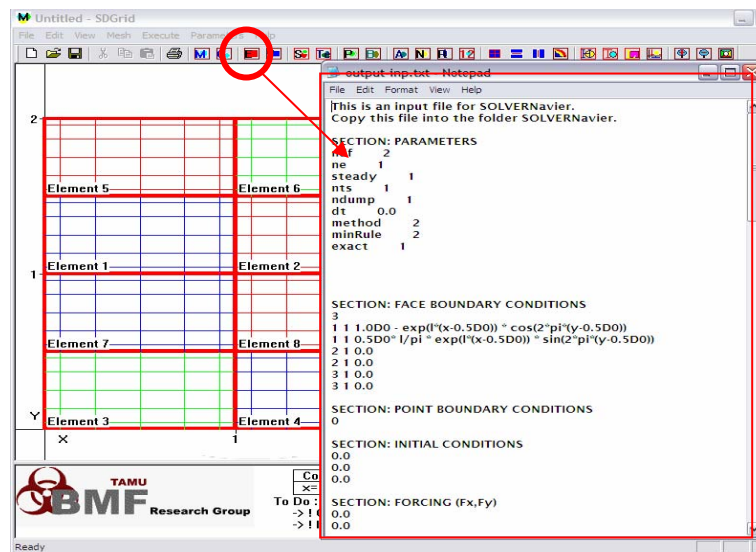



Fig. 30 View Output-inp.txt file generated from the preprocessor.

Finally, press  to run the default solver [1]. The preprocessor calls default solver to solve the problem with the generated output file (Fig. 31).

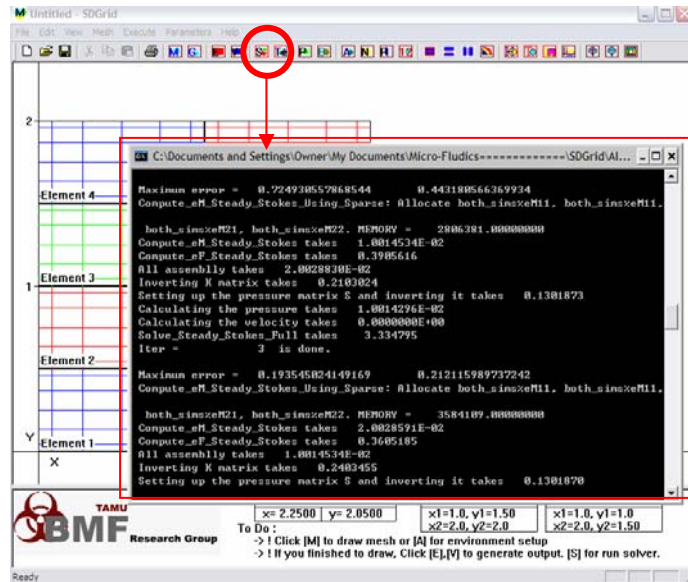



Fig. 31 Solving problem with the default solver.

After running, the result file is generated from the default solver. To see the result, press  to open (\*.dat) file with default postprocessor (Tecplot<sup>TM</sup>) as shown in Fig. 32.

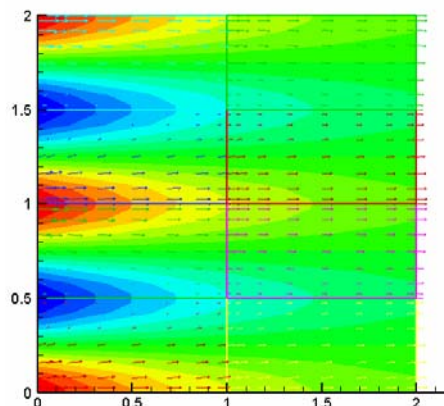



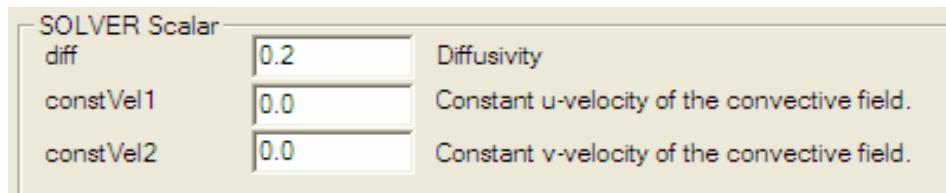


Fig. 32 Result screen from post the processor - Kovaszny flow.


To solve the diffusion equation, call parameter dialog with  and set “ndf” to 1. Then the preprocessor calls the Poisson equation solver. Finally set the parameters in the SOLVERScalar section as shown in Fig. 33. Problem solving procedure is the same with previous example, draw mesh and set connectivity , set boundary condition  and specify the boundary condition on each boundary side of the element.

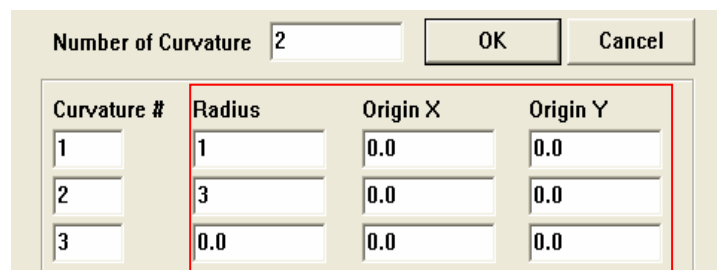


SOLVER Scalar	
diff	0.2 Diffusivity
constVel1	0.0 Constant u-velocity of the convective field.
constVel2	0.0 Constant v-velocity of the convective field.

Fig. 33 Parameters in SOLVERScalar section.

### Drawing Curved Elements

Curved element is another type of a non-rectangular element. An element can have curved side by assigning the curvature which is defined at the curvature dialog box. To define a curvature, call curvature dialog by pressing the  button and set two curvatures with radius 1, 3 with center point (0, 0) as shown in Fig. 34.



Curvature #	Radius	Origin X	Origin Y
1	1	0.0	0.0
2	3	0.0	0.0
3	0.0	0.0	0.0

Fig. 34 Curvature dialog box.

Draw a non-rectangular element with vertices  $[(1, 0), (0, 1), (3, 0), (0, 3)]$ . Then, go to the edit mode and select the element with a right click to open the element dialog box as shown in Fig. 35.

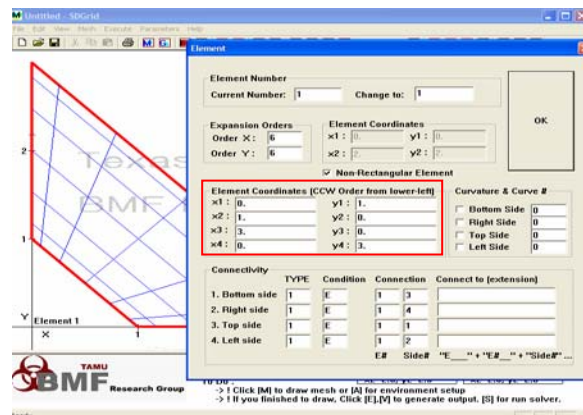


Fig. 35 Modifying drawn element with non-rectangular option.

Click the curvature check box on the “Left Side” and set the curve number to ‘1’. Then, click the curvature checkbox at “Right Side” and set the curve number to ‘2’. This means that the bottom side and top side have the curvatures, defined in the curvature dialog box (see Fig. 36).

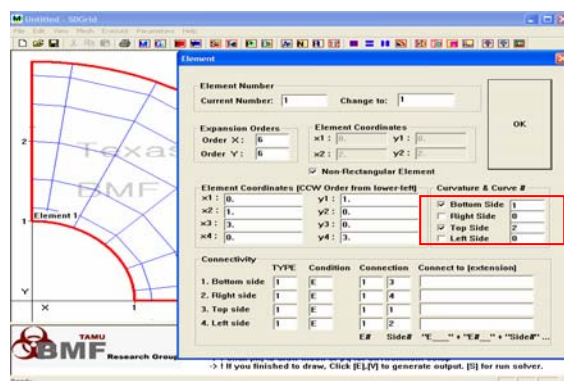


Fig. 36 Define curvature at each side.

To split the curved element, select the element by dragging and pressing the horizontal split [≡] button to split the element into two pieces. Click the inner element and press the [||] button to split the element into two pieces. Finally, press the button [↔] and [↔] buttons for renumbering and connectivity of the elements respectively. Then set up the boundary conditions and export the output file. Fig. 37 illustrates the procedure for drawing curved elements, which demonstrates convenience of the preprocessor for h-type refinements and increasing the orders of the elements.

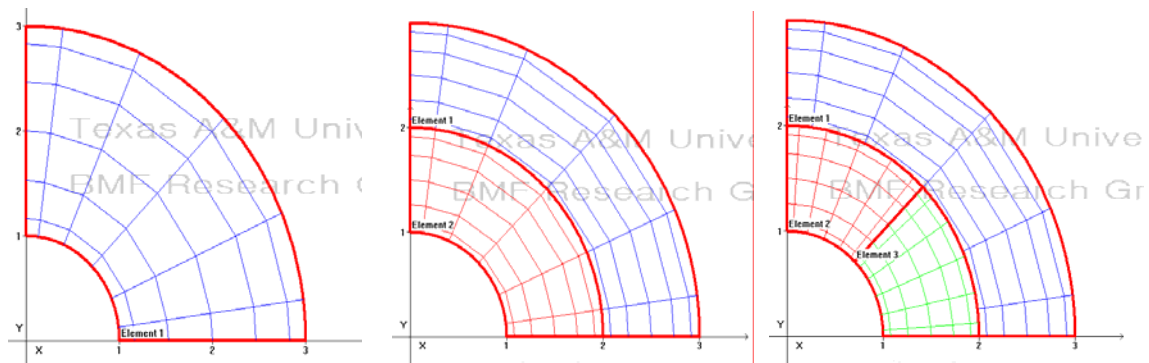


Fig. 37 Curved elements and mesh refinement using splitting functions: h- type non-conforming interfaces.

### Drawing complex geometry using splitting element [||], [≡]

The procedure for drawing complex geometries and splitting elements is described below. First,  $1 \times 4$  size element is drawn (Fig. 38) and split it into 4 pieces using the vertical split method [||] (Fig. 39). Then a horizontal split is performed to split the geometry into a total of 8 elements [≡] (Fig. 40).

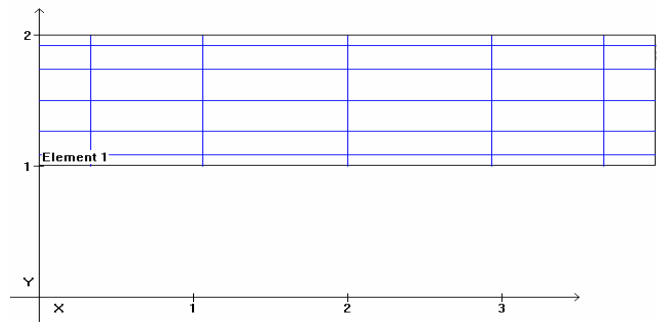


Fig. 38  $1 \times 4$  size element before splitting.

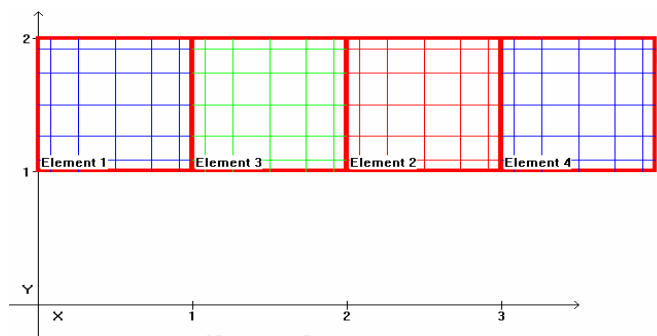


Fig. 39 Perform vertical split of element into four pieces.

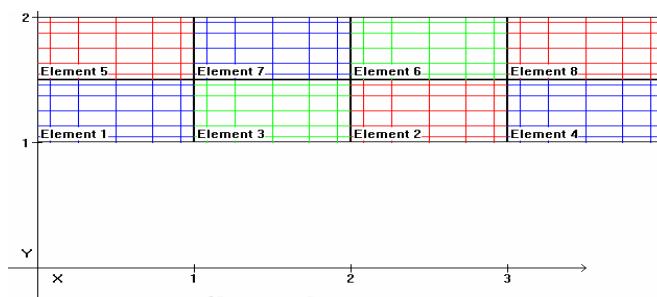



Fig. 40 Horizontal split of elements into eight pieces.

After this, select elements 1, 3, 2 and 4 in Fig. 40 by dragging them. This splits them horizontally to refine the element mesh [  ] (Fig. 41).



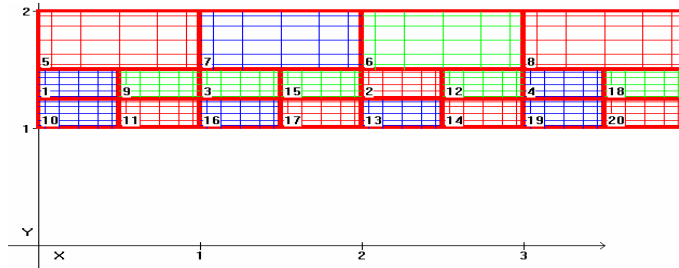


Fig. 41 Splitting bottom elements.

This procedure of splitting elements can be repeated until the bottom elements are small enough to produce a series of grooved geometries (Fig. 42).

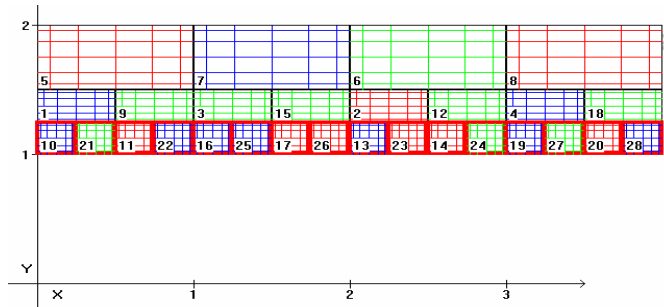




Fig. 42 Refine the element mesh using splitting.

After this, add small groove elements (Fig. 43), and set the connectivity [  ] and boundary conditions [  ] prior to solving.

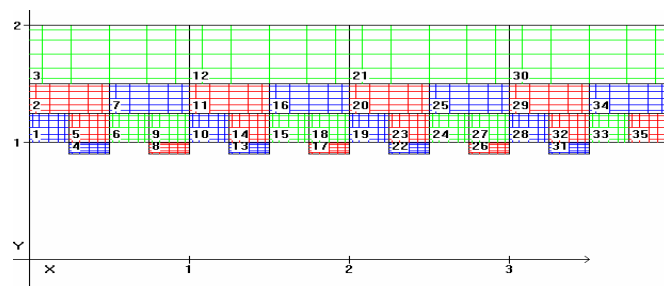


Fig. 43 Add groove elements 4, 8, 13, 17, 22, 26, 31.

Fig. 43 illustrates the mesh generated for a channel with grooved walls. Such intricate mesh generation can be easily implemented using the element splitting functions in the preprocessor. The figure also shows h/p type non-conforming interfaces along the refined grooved wall. In order to modify the drawn geometry, the upper 4 elements are chosen and their  $y_3$  and  $y_4$  coordinates are changed to 4. Then the order of the upper element is increased to 12 (Fig. 44).

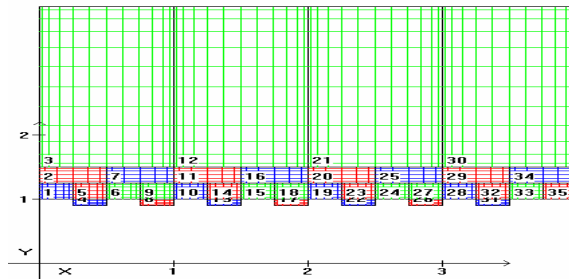


Fig. 44 Modifying a previously drawn geometry.

### Putting a circular hole in the element

Flow around a cylinder is a classical fluid mechanics problem. In order to include a circular geometry in the domain, a rectangular element is first drawn with  $[(-1, -1), (1, 1)]$  vertices (Fig. 45).

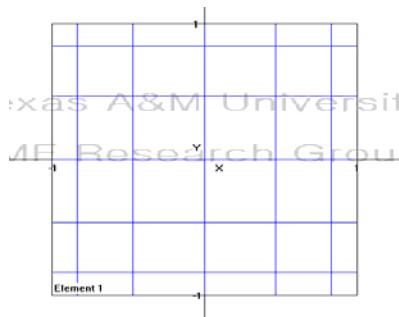



Fig. 45 Square element with vertices  $[(-1, -1), (1, 1)]$ .

Following this, the element is selected by dragging it and  icon is clicked to create a hole in the element. The radius of the circle is entered in the input dialog box that appears after this step (Fig. 46).

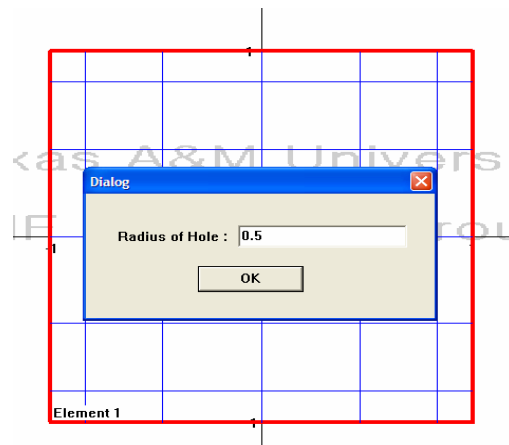


Fig. 46 Set the radius of a hole in the element.

For example, if the diameter is set to 0.5, a square element is divided into 4 curved elements (Fig. 47). The curvature used for the hole is automatically added to the curvature dialog box.

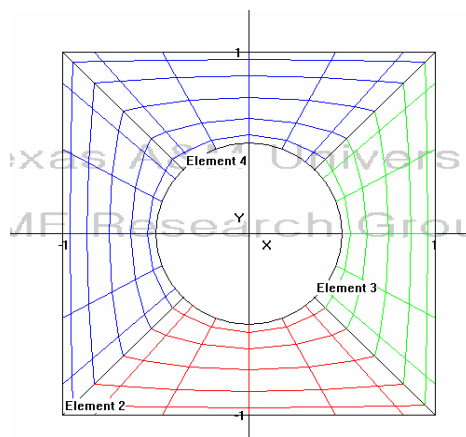


Fig. 47 A circular hole with radius 0.5.

We can add another curvature which fits outside of the mesh in order to make annular geometry. To do this, add one more curvature using radius calculator from (0, 0) to (1, 1) (Fig. 48). Then copy this value the curvature dialog box as another type of radius of curvature (Fig. 49).

Distance Calculator			
	X coordinate	Y coordinate	
Point 1	0	0	Calculate
Point 2	1	1	

Fig. 48 Distance calculator in the curvature dialog box.

Number of Curvature: 2			
Curvature #	Radius	Origin X	Origin Y
1	0.5	0.	0.
2	1.414213562	0.0	0.0
3	0.0	0.0	0.0

Fig. 49 Define the second curvature.

Every element's outer surfaces are changed to curvature number '2'. Following this, a p-type refinement is performed and boundary conditions for the inner side and outer side are set up prior to solving (Fig. 50).

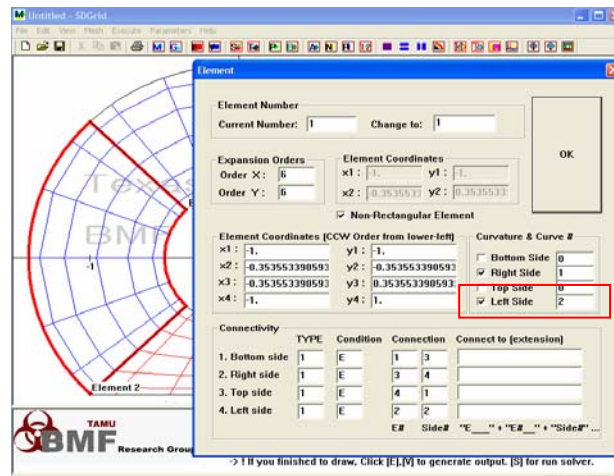


Fig. 50 Set outside curvature to make annulus geometry

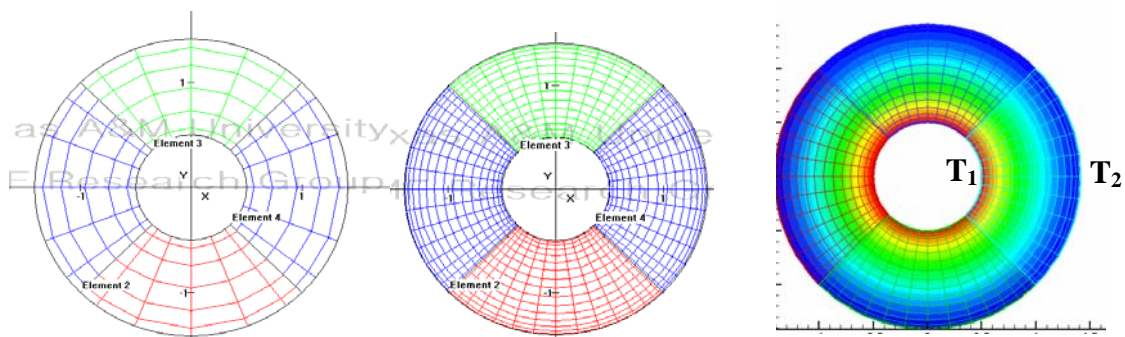


Fig. 51 Heat equation with constant wall temperature ( $T_1=100$ ,  $T_2=50$ ) in an annulus using curved elements.

Fig. 51 illustrates the SEM discretization of an annulus and the result of solving the heat equation. The mesh has four curved elements with two curved sides. Typically in h-type FEM, a large number of elements along the curved sides are necessary. However, in SEM, it is simple to implement high order curved elements. The complex geometry transforms into local coordinates by isoparametric mapping.

CHAPTER IV  
APPLICATIONS

**Kovaszny Flow**

Kovaszny flow (1948) is a common test problem for high-order methods [5]. The Kovaszny flow has a flow pattern similar to viscous flow past an array of cylinders. The flow is laminar and the analytical solution of the Navier-Stokes equations leads to

$$u = 1 - e^{\lambda x} \cos(2\pi y) \quad (25)$$

$$v = \lambda / 2 e^{\lambda x} \sin(2\pi y) \quad (26)$$

$$p = 0.5(1 - e^{\lambda x}) \quad (27)$$

where  $\lambda = 0.5 / \nu - (0.25 / \nu^2 + 4\pi^2)^{0.5}$ . The preprocessor can generate a mesh with, p-type and h-type nonconforming solutions, with the exact solution as the boundary conditions. The boundary conditions for the above problem have to follow the format described below. The first boundary condition is represented as, “1.0D0 - exp(L\*(x-0.5D0)) \* cos(2\*pi\*(y-0.5D0))”, and the second boundary condition is represented as “0.5D0\* L/pi \* exp(L\*(x-0.5D0)) \* sin(2\*pi\*(y-0.5D0))”. Where pi ( $\pi$ ) and L ( $\lambda$ ) are the coefficients already declared in the default solver.

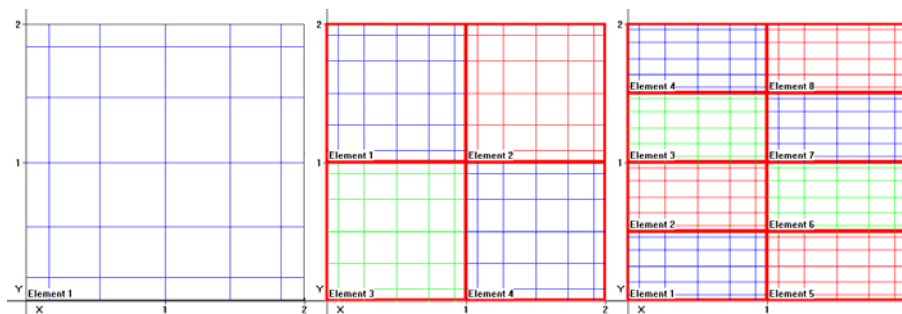


Fig. 52 Drawing the mesh for Kovaszny problem in the preprocessor.

First, an element is drawn with  $(0, 0)$  and  $(2, 2)$ . After this, the element is selected and split into four smaller elements. Following this, all four elements are selected and split horizontally once (Fig. 52). The element numbering, connectivity, boundary conditions and parameters should be checked prior to solving the model. The results of the simulation can be checked using the post processor (Fig. 53, 54).

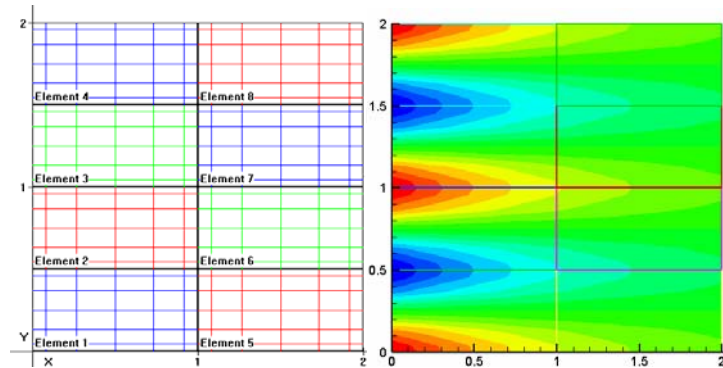


Fig. 53 Kozaszny flow mesh. Eight  $6 \times 6$  conforming elements are used (left). Stream wise velocity contours (right).

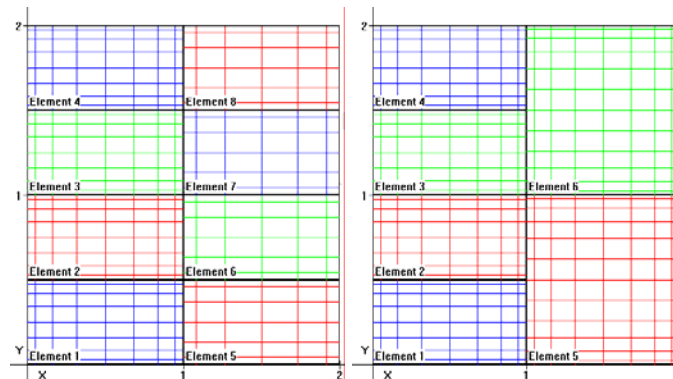


Fig. 54 Kozaszny flow p-type nonconforming mesh: Same eight element mesh is used. Elements 1,2,3,4 have expansion orders  $N$ , where elements 2,4,6,8 have expansion orders  $N-2$  (left). Kozaszny flow with h-type nonconforming elements: Six element mesh is used. All elements have expansion order  $N$ , except elements 3 and 6 has expansion orders  $N+4$  in the y-direction (right).

Fig. 55 shows convergence for each from the Navier-Stokes solver. The maximum error is plotted with the number of nodes in the element. Using the square root of the number of nodes per direction, exponential convergence is demonstrated. All these cases show spectral convergence. MEM shows better convergence than CAM. For the h-type nonconformity case, CAM and MEM results are almost the same [1]. For further discussions about the solver and CAM and MEM methods, the user is referred to [1].

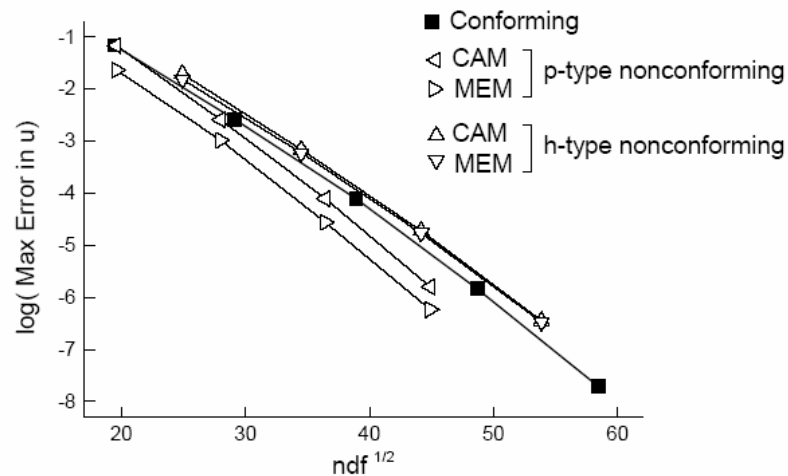


Fig. 55 Convergence results for Kovasznay flow obtained using conforming, and h- and p-type non-conforming meshes. Ndf is the number of nodes (degrees of freedom). Adopted from Sert and Beskok 2005 [1].

### Backward Facing Step Flow

The backward facing step example illustrates the method to build geometries in long channels. This is a popular test case to simulate flow separation and reattachment. In the preprocessor, we start with two elements and use the element dialog to change the element's coordinates.



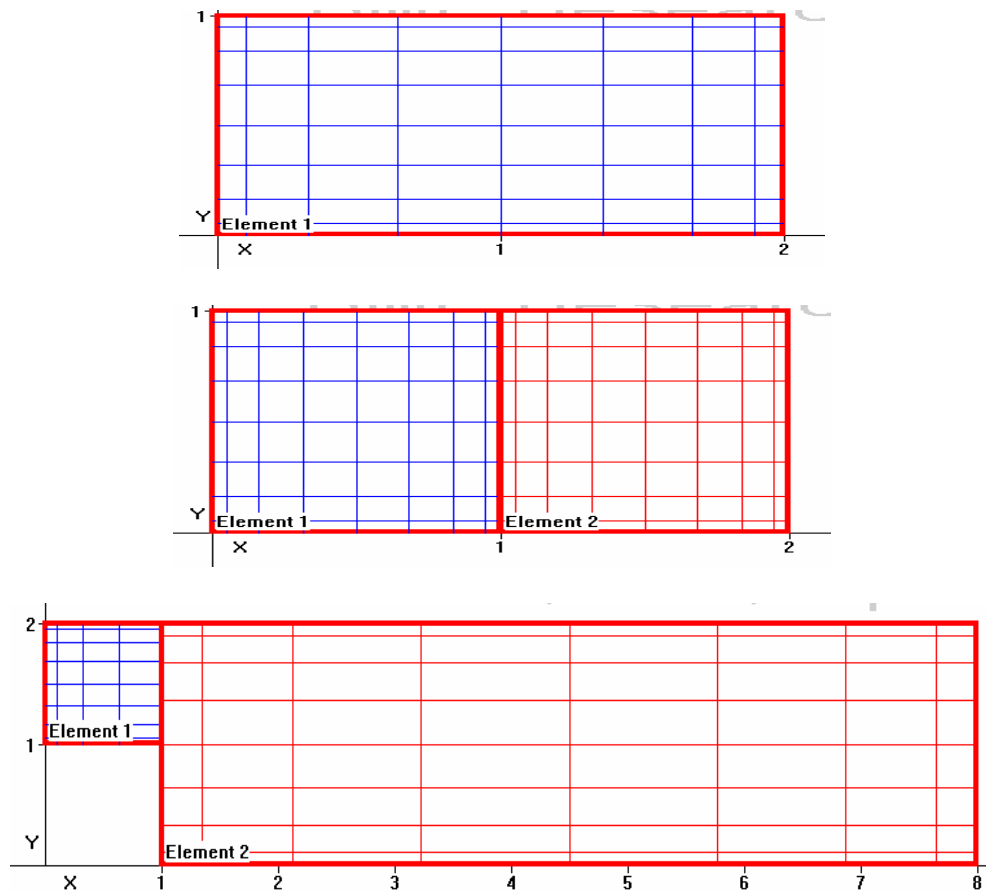


Fig. 56 Build a global geometry using splitting element and modify the drawn element.

After splitting the first element, the first element becomes inlet of the channel and the second element will result in the long channel (Fig. 56).

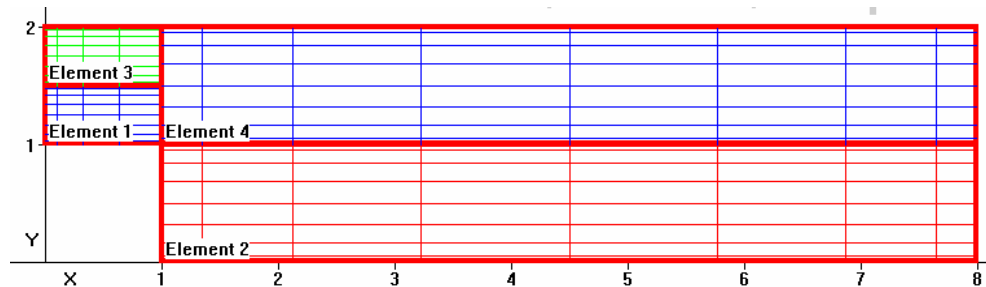


Fig. 57 Perform h-type refinements in the inlet channel.

Prior to starting element discretization, the first horizontal split needs to be performed.

Select all elements by dragging the whole domain (Fig. 57).

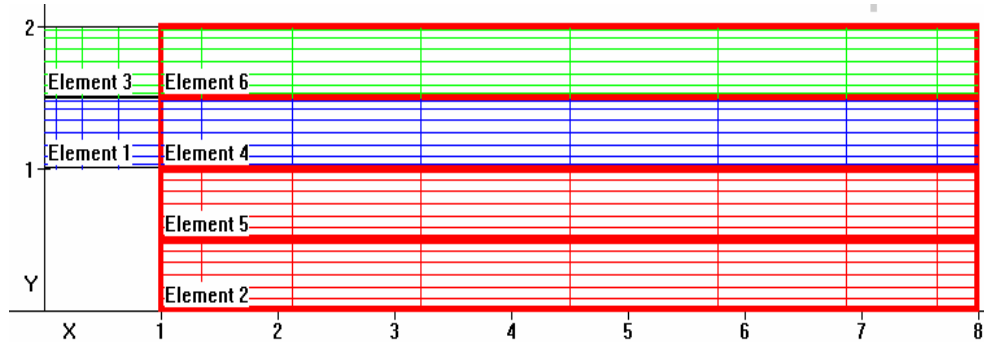


Fig. 58 Perform horizontal split in the outlet channel.

The order of elements is increased to 8 (Fig. 58), and vertical split for all channel elements is performed three times (Fig. 59).

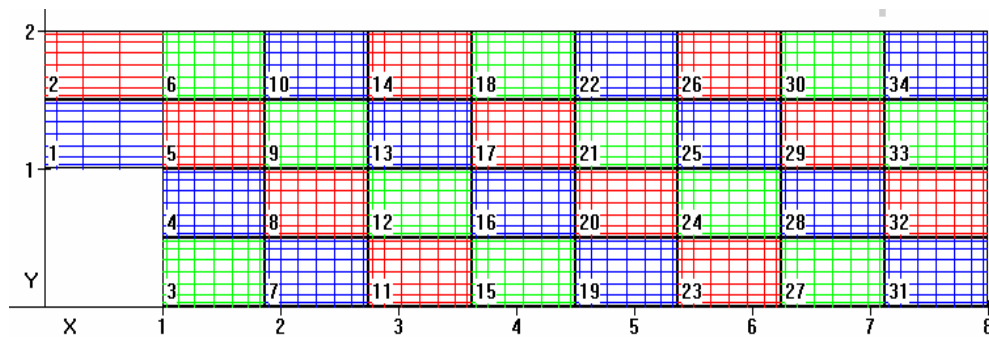


Fig. 59 H-type refinement in the flow region.

Now, the preprocessor is ready to call the solver. After checking numerical results, the h or p type refinements can be performed to obtain better numerical results.

## Y Channel Flow

Flow in Y channel is a good example to illustrate the procedure to draw arbitrary shaped non-rectangular elements and complex geometries. The detailed procedure for building the Y split channel is described below.

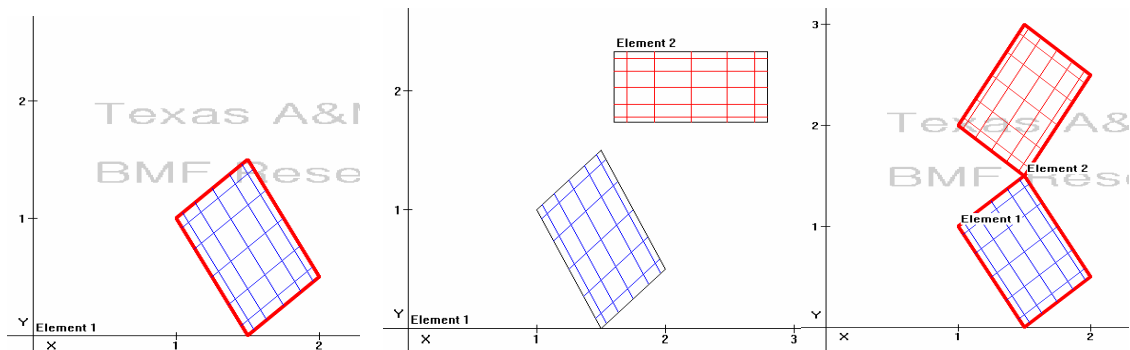


Fig. 60 Draw an element and change the coordinates of four vertices.

First, a rectangular element is drawn and then changed to non-rectangular element with vertices  $[(1.5, 0), (2, 0.5), (1.5, 1.5), (1, 1)]$ . Following this, another element is built to create another split channel and changed to non rectangular elements with symmetric vertices with the first element (Fig. 60).

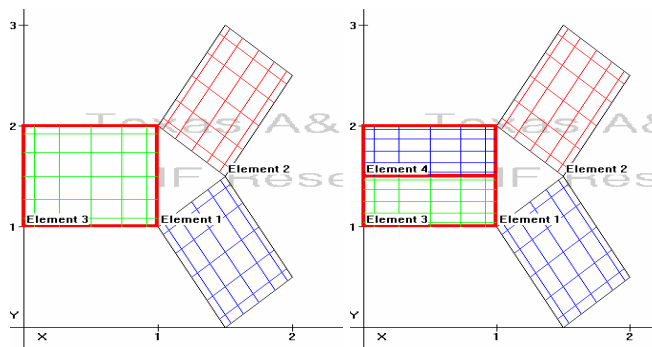


Fig. 61 Draw the elements at inlet and outlet channel.

To draw an inlet for the channel, a rectangular element with vertices  $(1, 0)$ ,  $(1,1)$ ,  $(1,2)$ ,  $(0,2)$  is drawn. Following this, one horizontal split is performed for the inlet element to fill up the joint area with additional elements (Fig. 61). After the horizontal split, the elements are changed to non-rectangular elements and the x coordinate of 3<sup>rd</sup> and 2<sup>nd</sup> vertices is changed to  $-0.5$  offset (Fig. 62).

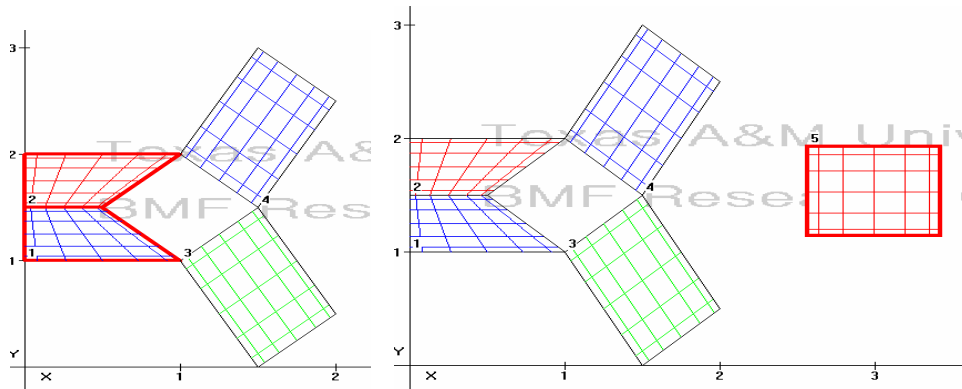


Fig. 62 Add an element in the joint region.

Another element is drawn and modified to non-rectangular element and the coordinates of its vertices are changed to fill up the region joining the Y channel with the straight channel (Fig 63, left).

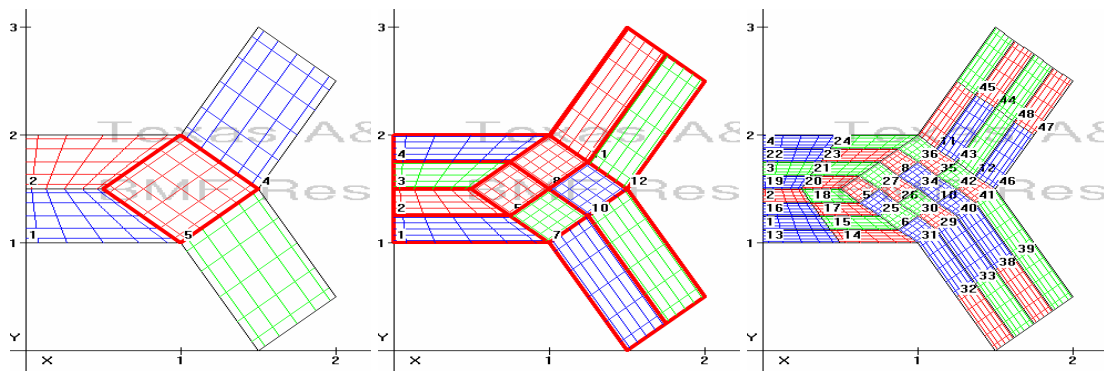


Fig. 63 Perform refinement in the flow region.

A horizontal split is performed to discretize the whole domain. After the first run, the user can perform further refinements to obtain finer mesh resolution, if needed. (Fig. 63 middle and right)

### Flow Over a Stack of Cylinders

Solving the flow around the cylinder is illustrative for studying the pressure drag, wake, boundary layer separation. Moreover, it is visually appealing. However, solving the problem as well as meshing the domain properly is rather cumbersome. In order to draw a domain to model the flow around a stack of cylinders, we start with 3 long rectangular elements (Fig. 64).

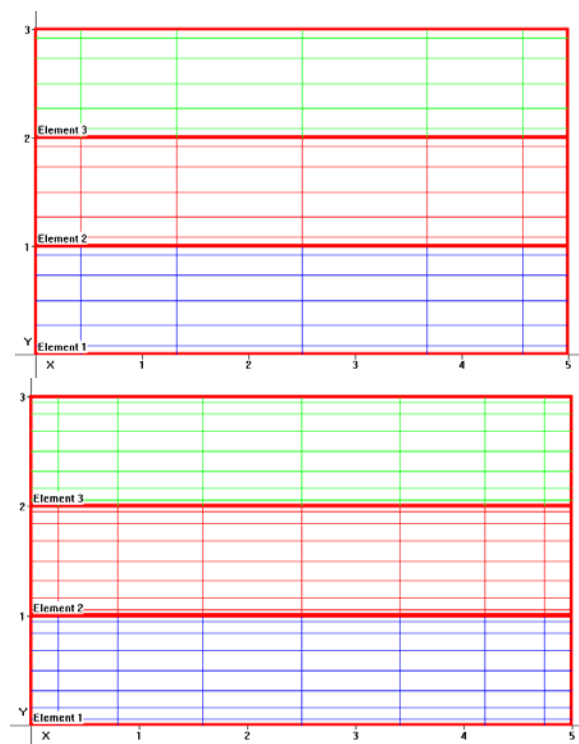


Fig. 64 Draw three long elements and perform p-type refinement.

After this a p-type refinement is performed and horizontal split is performed twice to create 12 elements (Fig. 65).

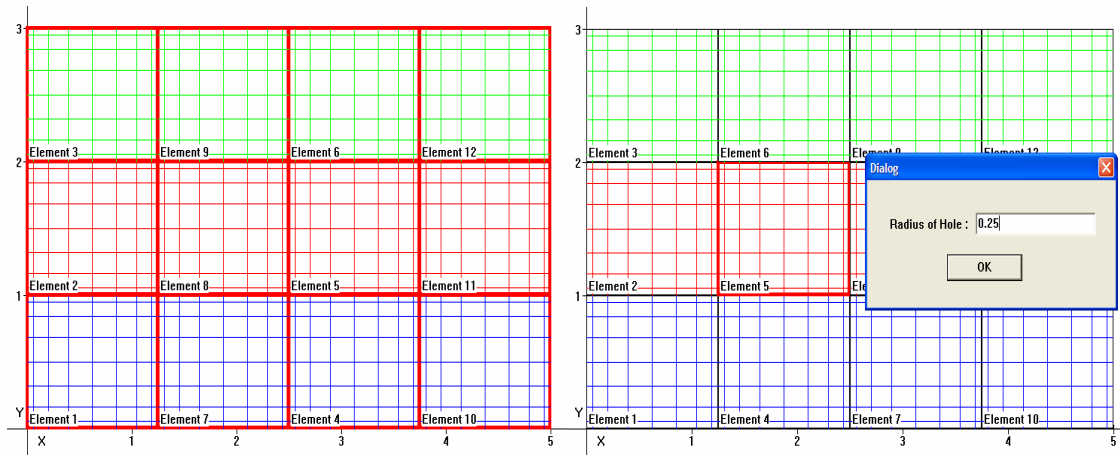


Fig. 65 Vertical split of the elements (left). Put a hole in the element 5 (right).

Following this, we create a hole in the 5<sup>th</sup> element with radius 0.25. The circle has no-slip boundary (Fig. 66).

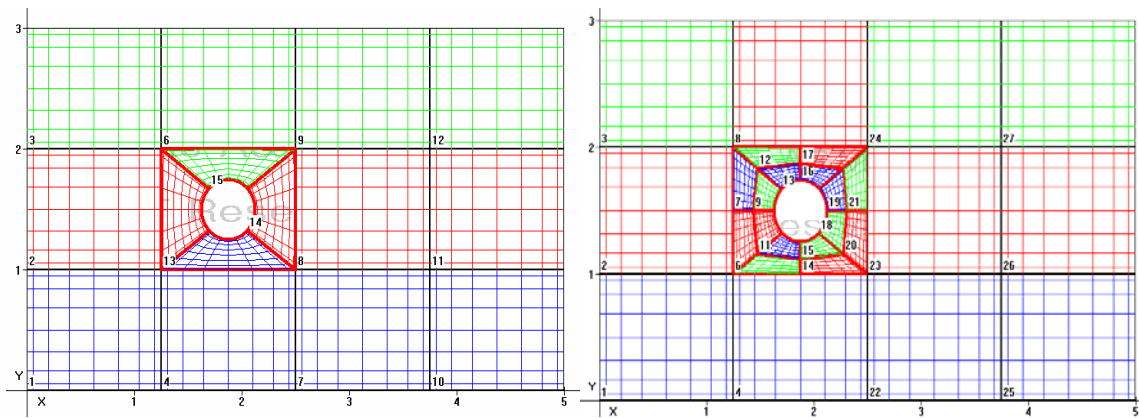



Fig. 66 Refine the mesh around the cylinder using 4-split function.

Refinement of element mesh around the cylinder is critical because this is the area of interest. Therefore, performing the mesh refinement around the cylinder is necessary using the 4 split function [  ]. Also, increase the order of elements to 12<sup>th</sup> by edit element. (Fig. 66, right)

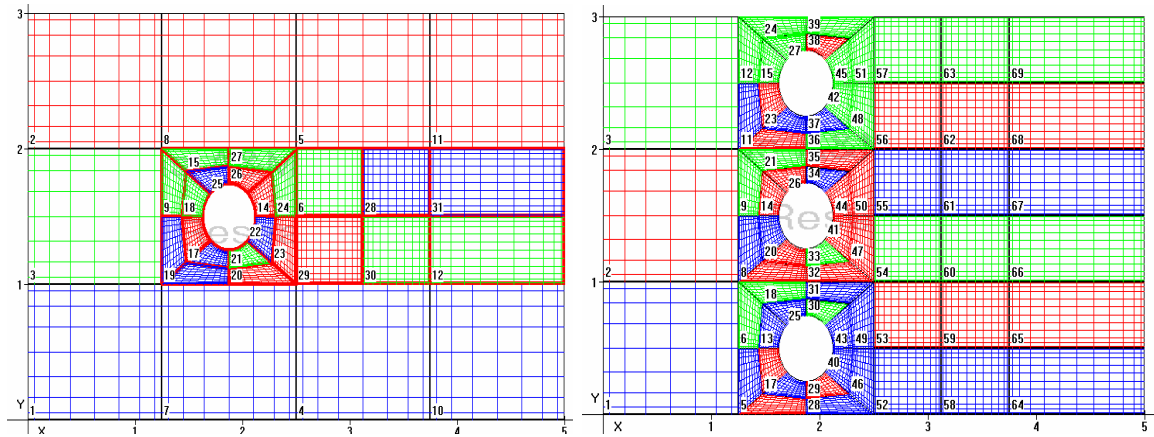


Fig. 67 Refine the mesh at the flow region affected by the cylinder (left). Series of cylinders could be added in a similar method in Fig. 66 (right).

After performing the refinement around the cylinder, another refinement is performed in the regions where the flow will be affected by the cylinder (Fig. 67, left). A stack of cylinder can be added by repeating the same process to the remaining elements (Fig. 67, right).

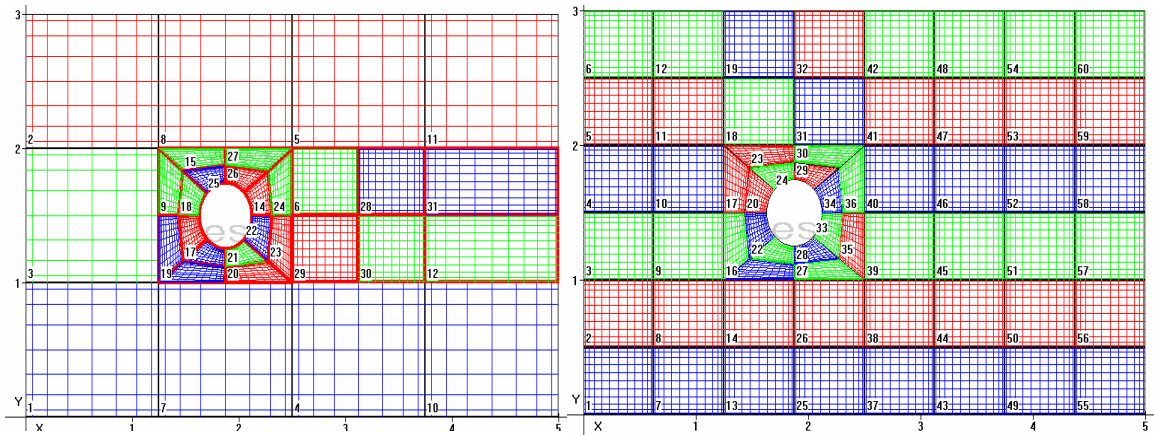


Fig. 68 The number of elements and DOF for conforming case and non-conforming case; NDOF =3744 (left, conforming case), NDOF =8640 (right, non-conforming case).

Fig. 68 shows the same geometry with conforming mesh (right) and a non-conforming mesh (left). The conforming case requires almost twice the number of degrees of freedom (NDOF), compared to the non-conforming case. If the model needs larger external flow regions in order to prevent the influence from the side boundaries, the conforming SEM mesh would require a lot more degrees of freedom compared to the non-conforming mesh. Therefore, it is clear that the non-conforming mesh is more efficient for SEM.



## CHAPTER V

### CONCLUSIONS

A GUI based preprocessor is developed for non-conforming SEM solvers. Each step of mesh generation and h-type and p-type refinement are explained with practical examples. No prior knowledge of the SEM is required to use the preprocessor, since the tool helps users with mesh generation in every step of the procedure. Classical fluid mechanics and heat transfer problems are solved and spectral convergence is demonstrated. The preprocessor is comparable with other commercial preprocessors available. It also provides a graphical interface which enables visualization while the mesh is being constructed, so that the entire domain can be discretized easily using functions like ‘splitting elements’, ‘edit elements’, and by putting circular holes in the element. This preprocessor provides a convenient way to implement h/p type nonconforming interfaces between the elements. It aids the user in learning about advanced numerical discretization techniques. Utilizing the preprocessor facilitates enhanced understanding of SEM, isoparametric mapping, h and p type nonconforming interfaces, and spectral convergence. Also, preprocessor provides a proficient and convenient graphical interface, independent of the solvers.

## REFERENCES

- [1] C. Sert, and A. Beskok, "Spectral Element Formulations on Non-conforming Grids: A Comparative Study of Pointwise Matching and Integral Projection Methods," *J. Comput. Phys.* 211, No.1, 300-325 (2006).
- [2] J. N. Reddy, *An Introduction to the Finite Element Method*, 2<sup>nd</sup>. Ed., New York: McGraw-Hill, 1993.
- [3] D. Gottlieb, S. A. Orszag, *Numerical Analysis of Spectral Method: Theory and Applications*, Philadelphia: SIAM, 1977.
- [4] A. Beskok, "A Two-Dimensional Spectral Element Formulation for Poisson Equation", MEEN679 class notes, Texas A&M University, 2005.
- [5] G. E. Karniadakis, S. J. Sherwin, *Spectral /hp Element Method for CFD*, New York: Oxford University Press, 1999.
- [6] D. Gottlieb, J. S. Hesthaven, *Spectral Methods for Time-Dependent Problems*, Mineola, NY: Cambridge University Press, 2000.
- [7] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2<sup>nd</sup> ed., Mineola, NY: Dover Publications, 2000.
- [8] C. Canuto, M. Y. Hussaini, A. Quarterino, T.A. Zang, *Spectral Methods in Fluid Dynamics*, New York: Springer-Verlag, 1988.
- [9] C. Sert, A. Beskok, "Oscillatory Flow Forced Convection in Micro Heat Spreaders," *Number. Heat Trans. Part A*, vol. 42, no. 7, pp. 131-137, 2002
- [10] P. Dutta, A. Beskok, "Time Periodic Electroosmotic Flows: Analogies to Stokes' Second Problem," *Analytical Chemistry*, vol. 73, no. 21, pp. 5097-5102, 2001

- [11] A. Beskok, T. C. Warburton, "An Unstructured h/p Finite Element Scheme for Fluid Flow and Heat Transport in Moving Domains," *J. Comp. Phys.*, vol. 174, pp. 492-509, 2001
- [12] M. Y. Hussaini, T. A. Zang, "Spectral Methods in Fluid Dynamics," *Ann. Rev. Fluid Mech.*, vol. 19, pp. 119-367, 1987.
- [13] S. A. Orszag, "Numerical Methods for the Simulation of Turbulence," *Phys. Fluids*, vol. 12, pp. 250-257, 1969.
- [14] S. A. Orszag, "Comparison of Pseudospectral and Spectral Approximations," *Stud. Appl. Math.*, vol. 51, pp. 253-259, 1972.
- [15] A. T. Patera, "A Spectral Element Method for Fluid Dynamics, Laminar Flow in a Channel Expansion," *Journal of Computational Physics*, vol. 54, pp. 468-488, 1984
- [16] K. Z. Korczak, A. T. Patera, "An Isoparametric Spectral Element Method for Solution of the Navier-Stokes Equations in Complex Geometry," *J. Comput. Physics*, vol. 62, pp. 361-382, 1986.
- [17] E. M. Ronquist, A. T. Patera, "Spectral Element Multigrid. I. Formulation and Numerical Results," *J. Sci. Comput.*, vol 2. pp. 389-406, 1987.
- [18] G. E. Karniadakis, R. D. Henderson, "Spectral Element Methods for Incompressible Flows," in R. W. Johnson (ed.), *Handbook of Fluid Dynamics*, Boca Raton, FL: CRC Press, pp. 29.1-29.41., 1998.
- [19] I. Babsuka, M. Suri, "The p and h-p Versions of the Finite Element Method, Basic Principles and Properties," *SIAM Rev*, vol. 36, no. 4, pp.578-632, 1994.

- [20] L. Demkowicz, J. T. Oden, W. Rachowicz, O. Hardy, "Toward a Universal h-p Adaptive Finite Element Strategy: Part I: Constrained Approximation and Data Structure," *Comput. Methods Appl. Mech. Engrg.*, vol. 77, pp. 79-112, 1989.
- [21] C. Bernardi, Y. Maday, C. Mavriplis, A. T. Patera, "The Mortar Element Method Applied to Spectral Discretizations," in T. J. Chung, G. R. Karr (eds.) *Finite Element Analysis in Fluids, Seventh Int. Conf. on Finite Element Methods in Flow Problems*, Huntsville, AL: UAH Press, 1989.

## APPENDIX A

Output file from preprocessor for the heat transfer example in annulus, shown in Fig. 47

---

---

Output-inf.txt

This is an input file for SOLVERScalar.  
Copy this file into the folder SOLVERScalar.

SECTION: PARAMETERS

ndf 1  
ne 4  
steady 1  
nts 1  
ndump 1  
dt 0.0  
method 2  
minRule 2  
exact 1  
diff 0.2  
constVel1 0.0  
constVel2 0.0

SECTION: FACE BOUNDARY CONDITIONS

2  
1 1 100  
2 1 50  
2 1 0.0  
2 1 0.0  
3 1 0.0  
3 1 0.0

SECTION: POINT BOUNDARY CONDITIONS

0

SECTION: INITIAL CONDITIONS

0.0  
0.0  
0.0

SECTION: FORCING (Fx,Fy)

0.0  
0.0

SECTION: EXACT SOLUTION (Exact u,v,p)

?  
?  
?

SECTION: SECTION: COORDINATES

1  
-1. -1.  
-0.353553390593274 -0.353553390593274  
-0.353553390593274 0.353553390593274  
-1. 1.  
2  
-1. -1.  
1. -1.  
0.353553390593274 -0.353553390593274  
-0.353553390593274 -0.353553390593274  
3  
-0.353553390593274 0.353553390593274  
0.353553390593274 0.353553390593274  
1. 1.  
-1. 1.  
4  
0.353553390593274 -0.353553390593274  
1. -1.  
1. 1.  
0.353553390593274 0.353553390593274

SECTION: CURVATURE

2  
1 0.5 0. 0.  
2 1.4142135623731 0.0 0.0  
8  
1 2 1  
1 4 2  
2 1 2  
2 3 1  
3 1 1  
3 3 2  
4 2 2  
4 4 1

SECTION: CONNECTIVITY

1 16 16 0  
1 E 2 4

1 B 1 0  
1 E 3 4  
1 B 2 0

2 16 16 0  
1 B 2 0  
1 E 4 1  
1 B 1 0  
1 E 1 1

3 16 16 0  
1 B 1 0  
1 E 4 3  
1 B 2 0  
1 E 1 3

4 16 16 0  
1 E 2 2  
1 B 2 0  
1 E 3 2  
1 B 1 0

## APPENDIX B

Output file from the preprocessor for Kovaszny flow example, shown in Fig.49

Output-inf.txt

---

This is an input file for SOLVERNavier.  
Copy this file into the folder SOLVERNavier.

## SECTION: PARAMETERS

```

ndf      2
ne       8
steady   1
nts      1
ndump    1
dt       0.0
method   2
minRule  2
exact    1
visc     0.025
stokes   0
maxIter  30

```

## SECTION: FACE BOUNDARY CONDITIONS

```

3
1 1 1.0D0 - exp(1*(x-0.5D0)) * cos(2*pi*(y-0.5D0))
1 1 0.5D0* 1/pi * exp(1*(x-0.5D0)) * sin(2*pi*(y-0.5D0))
2 1 0.0
2 1 0.0
3 1 0.0
3 1 0.0

```

## SECTION: POINT BOUNDARY CONDITIONS

```
0
```

## SECTION: INITIAL CONDITIONS

```
0.0
0.0
0.0
```

## SECTION: FORCING (Fx,Fy)



0.0  
0.0

SECTION: EXACT SOLUTION (Exact u,v,p)

?  
?  
?

SECTION: SECTION: COORDINATES

1  
0. 0.  
1. 0.  
1. 0.5  
0. 0.5  
2  
0. 0.5  
1. 0.5  
1. 1.  
0. 1.  
3  
0. 1.  
1. 1.  
1. 1.5  
0. 1.5  
4  
0. 1.5  
1. 1.5  
1. 2.  
0. 2.  
5  
1. 0.  
2. 0.  
2. 0.5  
1. 0.5  
6  
1. 0.5  
2. 0.5  
2. 1.  
1. 1.  
7  
1. 1.  
2. 1.  
2. 1.5  
1. 1.5

- 8
- 1. 1.5
- 2. 1.5
- 2. 2.
- 1. 2.

## SECTION: CURVATURE

0

## SECTION: CONNECTIVITY

1 6 6 0  
 1 B 1 0  
 1 E 5 4  
 1 E 2 1  
 1 B 1 0

2 6 6 0  
 1 E 1 3  
 1 E 6 4  
 1 E 3 1  
 1 B 1 0

3 6 6 0  
 1 E 2 3  
 1 E 7 4  
 1 E 4 1  
 1 B 1 0

4 6 6 0  
 1 E 3 3  
 1 E 8 4  
 1 B 1 0  
 1 B 1 0

5 6 6 0  
 1 B 1 0  
 1 B 1 0  
 1 E 6 1  
 1 E 1 2

6 6 6 0  
 1 E 5 3  
 1 B 1 0  
 1 E 7 1

1 E 2 2

7 6 6 0

1 E 6 3

1 B 1 0

1 E 8 1

1 E 3 2

8 6 6 0

1 E 7 3

1 B 1 0

1 B 1 0

1 E 4 2

## VITA

Bo Hung Kim graduated with a B.S. degree in mechanical engineering from Yonsei University, Seoul, South Korea in February 2002. In August 2004, he joined the Department of Mechanical Engineering at Texas A&M University and graduated in 2006.

Bo Hung Kim may be contacted through Dr. Ali Beskok at the Mechanical Engineering Department, Texas A&M University, College Station, TX 77843-3123