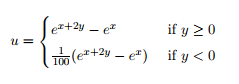
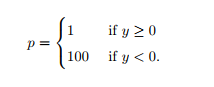
유한요소 Discontinuous Garlerkin Method

The problem is to solve the Dirichlet problem in the domain Ω =[-1 1]\*[-1 1] of an elliptic equation given by −∇ · p∇u = f in Ω by using Discontinuous Garlerkin method.

Here the exact solution u and p, f are given as follow:



Since I have only one solver CG which is only for positive and symmetric matrices, I use SIPG whose variational formula is symmetric.

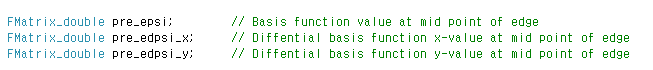
1. About Code

I save the stiffness matrix as a sparse matrix using t\_sparse.h. For generating the stiffness matrix mA, the procedure of the integration on each element is same as that of P1 conforming. Also the basis functions are same in both case. The different from P1 conforming to generate mA is line integrations on each edge. Here, I used one midpoint of each domain for integration

Thus I added some variables to evaluate line integrations.

First, **indexL** is a matrix saving some information about each edge. More precisely, for each edge, say edge, indexL[edge][0] and indexL[edge][1] are the local numbers of the adjacent element e and eN respectively. Here e and eN are notation in the lecture note and I fix the direction of normal vector between two adjacent elements by starting from an element of lower global index to the other of higher global index. If the edge is on the boundary, then eN is saved by -1. Finally, indexL[edge][2] is the local number of the edge.

For calculating of jump and average of test functions, I added following variables:



Since the variable **pre\_psi** and **pre\_dpsi** save the basis functions values and the differential basis functions values at midpoint of the reference element which are not on the edge and the edges are only considered for jump and average, the values on the edge are needed. Thus I added **pre\_epsi** for saving basis function values at midpoint of each edge and **pre\_edpsi\_x**¸ **pre\_edpsi\_y** for saving x-value and y-value of differential basis function at midpoint of edges respectively. More precisely, pre\_epsi[local edge][basis function] and pre\_edpsi\_x[local edge][basis function].



Note that each edge has relation with 6 degree of freedom. Thus **vDOF** is a vector of length 6 and it saves global indexes of degree of freedom in the elements e and eN. The variables **PSI\_JUMP** and **DPSI\_AVE** are used to calculate following line integration:



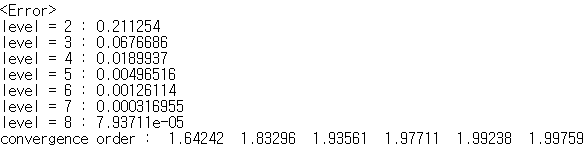
Here, values of the gradient of test functions and values of test functions appear only in the average and jump term respectively. Thus DPSI\_AVE and PSI\_JUMP save the average values of p\*basis functions and the jump values of degree of freedom in the elements e and eN. To save data in DPSI\_AVE and PSI\_JUMP, I added another variables.



**PSI** and **DPSI** express basis functions and these derivatives obtained by the mapping **Make\_realP\_and\_DF** with respect to the element e. Similarly, **PSI\_N** and **DPSI\_N** are obtained by the mapping **Make\_realP\_and\_DF** with respect to the element eN.

1. Result

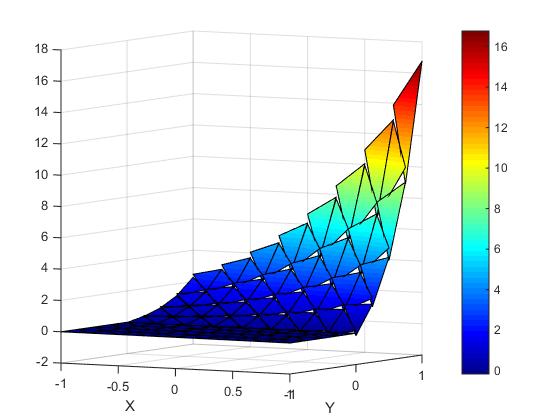
First, I set =1000, called gamma in my code. The errors and convergence orders for each level are as follow:

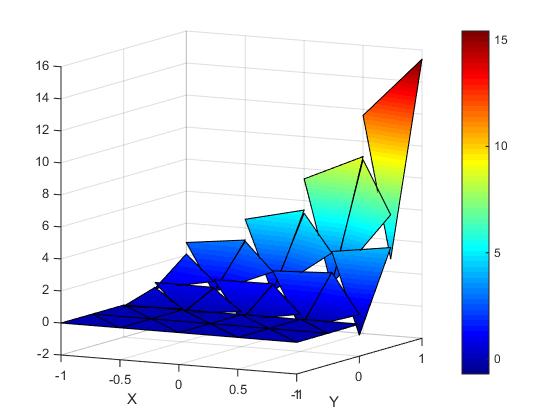


I tested 7 different levels from 2 to 8. As levels increase, the convergence order goes to 2.

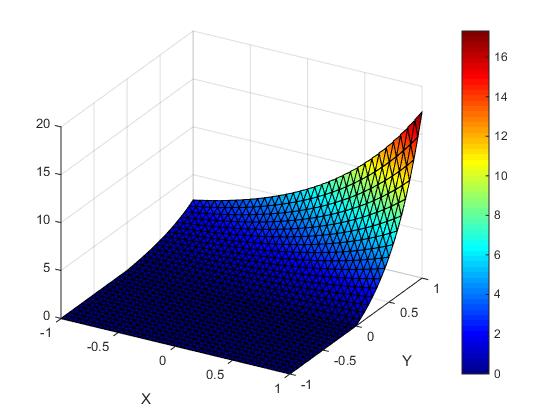
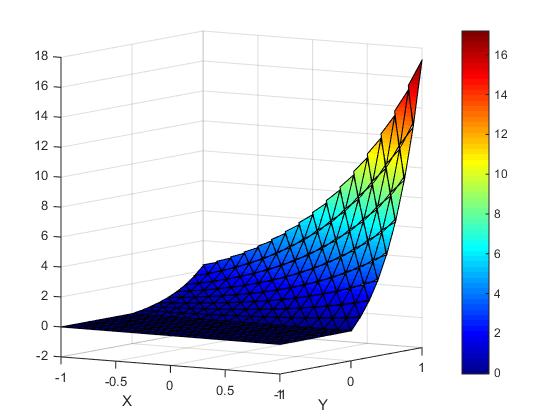
1. Graph

I saved the approximated solution **vCP** as m files to draw the graphs of approximated solutions. The following figures are the graph of approximated solutions for levels 2,3,4 and 5..





[ level = 2 ] [ level = 3 ]

[ level =4 ] [ level = 5 ]