

Appendix

Python code

Task 1b)

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3  from dolfin import * #FEniCS
4  from math import *
5
6
7  for i in [1,2]:
8
9      for N in [8,16,32,64]: # h=(b-a)/N=1/N
10         mesh = UnitSquareMesh(N,N)
11         V = FunctionSpace(mesh, "Lagrange", i)
12         V2 = FunctionSpace(mesh, "Lagrange", i+2)
13
14         u = TrialFunction(V)
15         v = TestFunction(V)
16
17         bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'), DirichletBC(V, Constant(0),
18         'x[0] > 1- DOLFIN_EPS')]
19
20         for k in [1,10]:
21             f = Expression('2*pi*pi*k*k*sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree=i)
22             uh = Function(V)
23
24             a=inner(grad(u),grad(v))*dx
25             L=f*v*dx
26
27             solve(a == L, uh, bcs)
28
29             uexp= Expression('sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree=1)
30             u.Exact= interpolate(uexp, V2)
31
32             Error=abs(u.Exact-uh)
33
34             L2_norm = errornorm(u.Exact, uh, 'l2', degree_rise=3)
35             print 'i=%g ,1/h=%g ,k=%g' % (i, N, k)
36             print 'L2 = %g' % (L2_norm)
37
38             H1_norm = errornorm(u.Exact, uh, 'H1', degree_rise=3)
39             print 'H1 = %g' % (H1_norm)
40             print '\n'
```

Task 1c)

```
1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  from dolfin import * #FEniCS
5  import matplotlib.pyplot as plt
6  from numpy import *
7  from termcolor import colored
8  set_log_active(False)
9
10 # least squares method:          y =      A      *      x      + b
11 #                               ln(Error) = ln(h) *      k      + ln(C_k)
12 for Error in ['l2', 'H1']:
13     print '\n'
14     print 'ErrorNorm = %s' % Error
15     for i in [1, 2]:
16         print '\n'
17         print '_____',
18         print 'order = %g' % i
19         for k in [1, 10]:
20             X=[]
21             Y=[]
22             print '_____',
23             print 'k=%g' % k
```

```

24     for N in [8, 16, 32, 64]: # h = (b-a)/N = 1/N
25         print 'Number of mesh points = %g' % N
26
27         mesh = UnitSquareMesh(N,N)
28
29         V = FunctionSpace(mesh, "Lagrange", i)
30         V2 = FunctionSpace(mesh, "Lagrange", i+2)
31
32         h = 1./N
33         print 'h = %g' % h
34
35         u = TrialFunction(V)
36         v = TestFunction(V)
37
38         bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'),
39                 DirichletBC(V, Constant(0), 'x[0] > 1- DOLFIN_EPS')]
40
41
42         f = Expression('2*pi*pi*k*k*sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree=i)
43         uh = Function(V)
44
45         a=inner(grad(u),grad(v))*dx
46         L=f*v*dx
47
48         solve(a == L, uh, bcs)
49
50         uexp= Expression('sin(pi*k*x[0])*cos(pi*k*x[1])', k=k, degree=1)
51         u_Exact= interpolate(uexp, V2)
52
53         Error_Norm = errornorm(u_Exact, uh, Error, degree-rise=3)
54         print 'Error Norm = %g' % Error_Norm
55
56         X.append(log(h))
57         Y.append(log(Error_Norm))
58
59         if N != 8:
60             Xx = array(X)
61             Yy = array(Y)
62
63             A = vstack([Xx, ones(len(Xx))]).T
64
65             Conv_rate, lnC = linalg.lstsq(A, Yy)[0]
66             print 'Convergence Rate=%g, C=%g' % (Conv_rate, exp(lnC))
67             if Error_Norm > (exp(lnC))*h**Conv_rate:
68                 print colored('This Error Estimate is —> NOT
69                 <— valid! Err.Norm > C.k*h^k', 'red')
70
71
72         f = lambda x: lnC + Conv_rate*x
73         x = linspace(X[0], X[-1], 101)
74
75
76         plt.plot(x, f(x), label='Checking k dependency of C.k*h^k')
77         plt.legend(loc='upper left')
78         plt.grid('on')
79         plt.xlabel('log(h)')
80         plt.ylabel('log(||u - uh||)')
81         filename = "ErrNorm_%s_PolDegree_%d.png" % (Error, i)
82         plt.savefig(filename)
83         plt.show()

```

Task 2b)

```

1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  from dolfin import * #FEniCS
5  from math import *
6  import numpy as np
7
8  for Error in ['l2', 'H1']:
9      print '\n'
10     print 'Error_Type = %s' % Error

```

```

11 for i in [1, 2]:
12     print '\n'
13     print '_____',
14     print 'Polynomial_order = %g' % i
15     for N in [8, 16, 32, 64]: # h = (b-a)/N = 1/N
16         print '_____ ',
17         print 'Number of mesh points = %g' % N
18
19     mesh = UnitSquareMesh(N,N)
20
21     V = FunctionSpace(mesh, "Lagrange", i)
22     V2 = FunctionSpace(mesh, "Lagrange", i+2)
23
24
25     u = TrialFunction(V)
26     v = TestFunction(V)
27
28     bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'), DirichletBC(V, Constant(1),
29         'x[0] > 1- DOLFIN_EPS')]
30
31
32     f = Constant(0.0)
33     uh = Function(V)
34
35     for mu in [1, 0.1, 0.01]:
36         print 'Mu value = %s' % mu
37
38         a=mu*inner(grad(u),grad(v))*dx + u.dx(0)*v*dx
39         L=f*v*dx
40
41         solve(a == L, uh, bcs)
42
43         uexp= Expression('(exp(x[0]/mu)-1)/(exp(1./mu)-1)', mu=mu, degree=1)
44         u_Exact= interpolate(uexp, V2)
45
46         Error_Norm = errornorm(u_Exact, uh, Error, degree_rise=3)
47         print 'Error Norm = %.3E' % Error_Norm
48         print '\n'

```

Task 2c)

```

1 #!/usr/bin/env python2
2 # -*- coding: utf-8 -*-
3 from dolfin import * #FEniCS
4 from math import *
5 from numpy import *
6 from termcolor import colored
7 set_log_active(False)
8
9 for Error in ['l2', 'H1']:
10     print '\n'
11     print 'Error_Type = %s' % Error
12     for i in [1, 2]:
13         X=[]
14         Y=[]
15         print '\n'
16         print '_____ ',
17         print 'Polynomial_order = %g' % i
18         for N in [8, 16, 32, 64]: # h = (b-a)/N = 1/N
19             print '_____ ',
20             print 'Number of mesh points = %g' % N
21             print '\n'
22
23         mesh = UnitSquareMesh(N,N)
24         h = 1./N
25
26         V = FunctionSpace(mesh, "Lagrange", i)
27         V2 = FunctionSpace(mesh, "Lagrange", i+2)
28
29
30         u = TrialFunction(V)
31         v = TestFunction(V)
32

```

```

33     bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'), DirichletBC(V, Constant(1),
34         'x[0] > 1- DOLFIN_EPS')]
35
36
37     f = Constant(0.0)
38     uh = Function(V)
39
40     for mu in [1, 0.1, 0.01]:
41         print 'Mu value = %s' % mu
42
43         a=mu*inner(grad(u),grad(v))*dx + u.dx(0)*v*dx
44         L=f*v*dx
45
46         solve(a == L, uh, bcs)
47
48         uexp= Expression('(exp(x[0]/mu)-1)/(exp(1./mu)-1)', mu=mu, degree=1)
49         u.Exact= interpolate(uexp, V2)
50
51         Error_Norm = errornorm(u.Exact, uh, Error, degree_rise=3)
52         print 'Error Norm = %.3E' % Error_Norm
53
54         X.append(log(h))
55         Y.append(log(Error_Norm))
56
57         if N != 8:
58             Xx = array(X)
59             Yy = array(Y)
60
61             A = vstack([Xx, ones(len(Xx))]).T
62
63             Conv_rate, lnC = linalg.lstsq(A, Yy)[0]
64             print 'Convergence Rate=%g, C=%g' %(Conv_rate, exp(lnC))
65             if Error_Norm > (exp(lnC))*h**Conv_rate:
66                 print colored('This Error Estimate is —> NOT <— valid! Err.Norm > C.k*h
^k', 'red')
67                 print '\n'
68             else:
69                 print colored('This Error Estimate is valid!', 'green')
70                 print '\n'

```

Task 2d)

```

1  #!/usr/bin/env python2
2  # -*- coding: utf-8 -*-
3
4  #Oblig 2d
5
6  from dolfin import * #FEniCS
7  from math import *
8  from numpy import *
9  from termcolor import colored
10 set_log_active(False)
11
12
13 for i in [1, 2]:
14     X=[]
15     Y=[]
16     print '\n'
17     print '_____',
18     print 'Polynomial_order = %g' % i
19     for N in [8, 16, 32, 64]: # h = (b-a)/N = 1/N
20         print '_____',
21         print 'Number of mesh points = %g' % N
22         print '\n'
23
24         mesh = UnitSquareMesh(N,N)
25         h=mesh.hmin()
26         Beta = 0.5*h
27
28         V = FunctionSpace(mesh, "Lagrange", i)
29         V2 = FunctionSpace(mesh, "Lagrange", i+3)
30
31

```

```

32     u = TrialFunction(V)
33     v = TestFunction(V)
34     w = v+Beta*v.dx(0)
35
36     bcs = [DirichletBC(V, Constant(0), 'x[0] < DOLFIN_EPS'), DirichletBC(V, Constant(1), 'x
[0] > 1-DOLFIN_EPS')]
37
38
39     f = Constant(0.0)
40     uh = Function(V)
41
42     for mu in [1, 0.1, 0.01]:
43         print 'Mu value = %s' % mu
44
45         a=mu*inner(grad(u),grad(w))*dx + u.dx(0)*w*dx
46         L=f*v*dx
47
48         solve(a == L, uh, bcs)
49
50         uexp= Expression('(exp(x[0]/mu)-1)/(exp(1./mu)-1)', mu=mu, degree=i)
51         u.Exact= interpolate(uexp, V2)
52
53         e1 = (uh-u.Exact).dx(0)
54         e2 = (uh-u.Exact).dx(1)
55         Error_Norm = sqrt(mesh.hmin()*assemble(e1**2*dx)+mu*assemble((e1+e2)**2*dx))
56         print 'Error Norm = %.3E' % Error_Norm
57
58         deltau = uh.dx(0).dx(0)+uh.dx(1).dx(1)
59         nablau = uh.dx(0)+uh.dx(1)
60         u2 = (assemble(uh**2*dx+nablau**2*dx+deltau**2*dx))
61
62         X.append(log(h))
63         Y.append(log(Error_Norm/u2))
64
65         if N != 8:
66
67             Xx = array(X)
68             Yy = array(Y)
69
70             A = vstack([Xx, ones(len(Xx))]).T
71
72             Conv_rate, lnC = linalg.lstsq(A, Yy)[0]
73             print 'Convergence Rate=%g , C=%g' %(Conv_rate ,exp(lnC))
74
75             if Error_Norm/u2 > (exp(lnC))*h**(3/2):
76                 print colored('This Error Estimate is —> NOT <— valid! Err_Norm > C_k*h^k',
'red')
77                 print '\n'
78             else:
79                 print colored('This Error Estimate is valid!', 'green')
80                 print '\n'

```