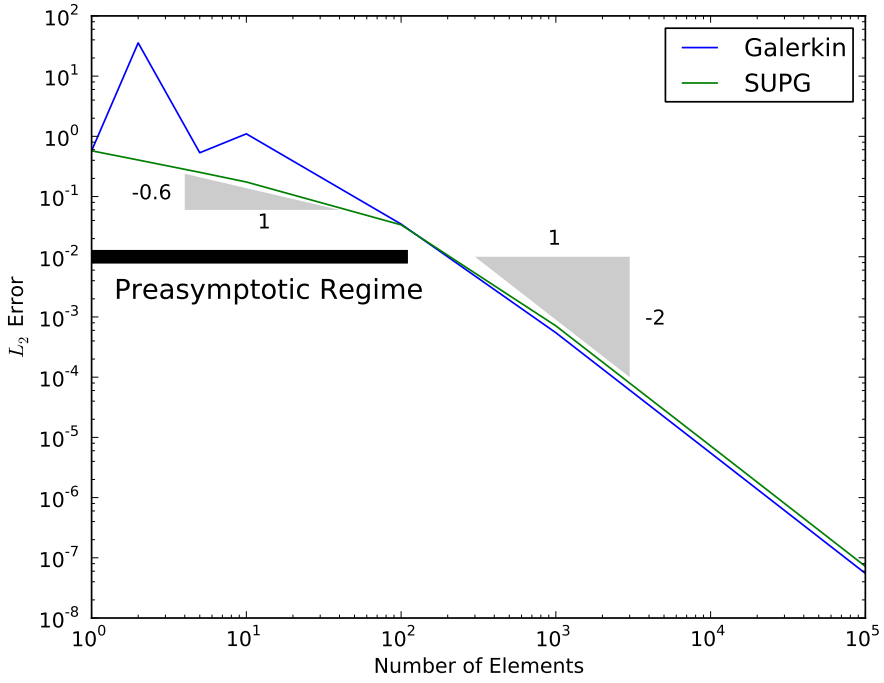
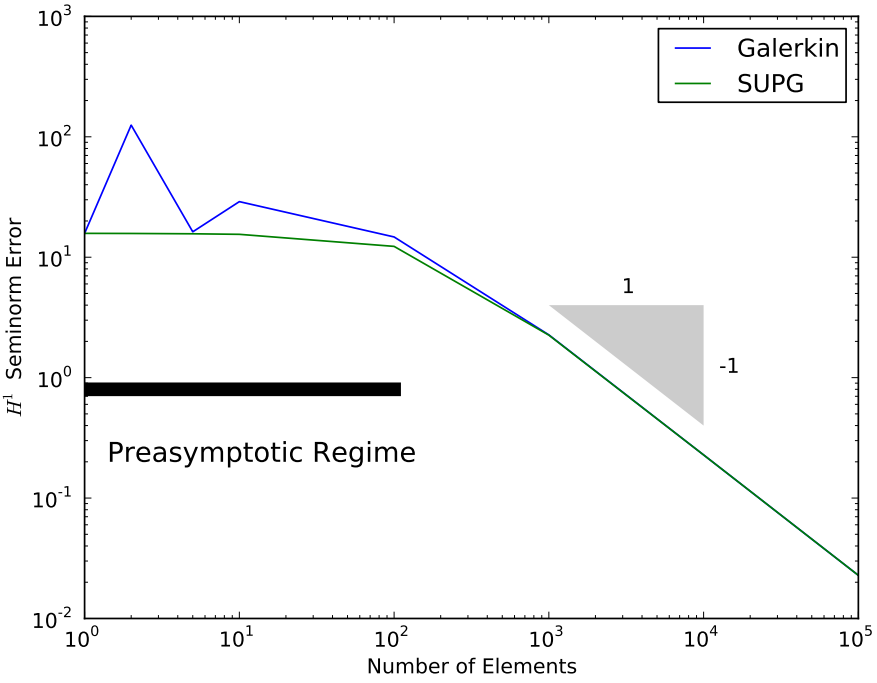
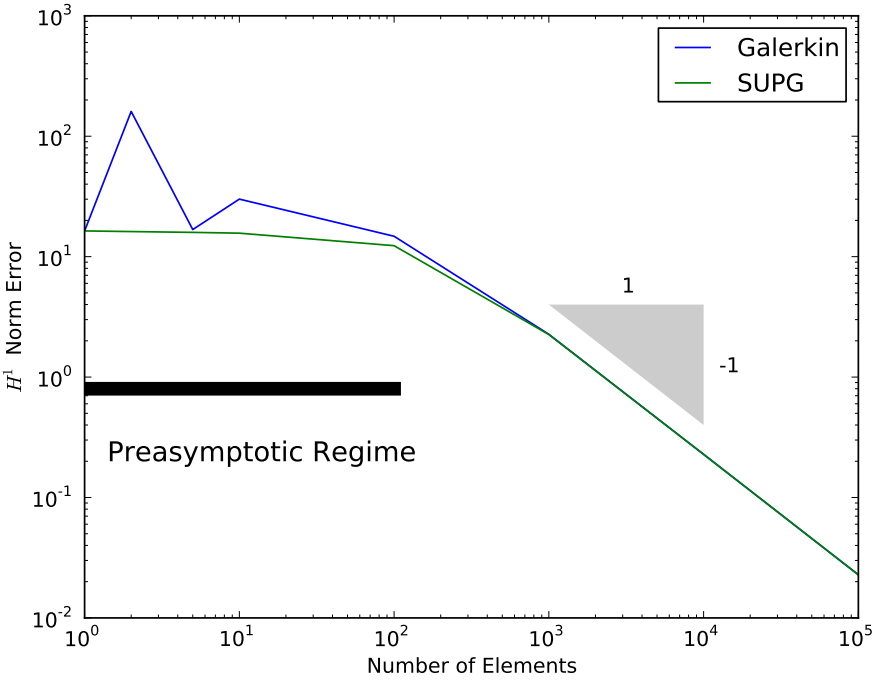
Problem 2.8



The biggest difference between the Galerkin and SUPG convergence plots is that SUPG has monotone error convergence in both the L_2 and H^1 sense, whereas the Galerkin error, like the solution, oscillates badly in the preasymptotic regime. Eventually, the Galerkin method begins to work as the element Peclet number goes down, but it is pretty oscillatory until then. The two methods eventually converge with the same rate, but the absolute value of error in the preasymptotic regime is much better for SUPG. The L_2 error of the SUPG solution starts slowly with a fractional slope of approximately 0.6 before it accelerates to a convergence rate of 2.0. The convergence rate for the solution derivative on the other hand starts out nearly stagnant before jumping to a rate of 1.0 outside of the preasymptotic regime. The Galerkin error is too oscillatory in the coarse meshes to identify a well-defined convergence rate, but it too reaches 2.0 on finer meshes. Similarly, the derivative error oscillates preasymptotically before reaching a rate of 1.0. The H^1 error is dominated by the H^1 seminorm error and we thus get nearly identical plots for both SUPG and Galerkin.


Galerkin
SUPG
H^1 Seminorm Error
Number of Elements
1
-1
Preasymptotic Regime


Galerkin
SUPG
H^1 Norm Error
Number of Elements
1
-1
Preasymptotic Regime

Problem 2.8 Code

```
from pylab import *
from scipy.integrate import quad
from slope_marker import slope_marker

close('all')

Pe = 500

num_elems = [1, 2, 5, 10, 100, 1000, 10000, 100000]
# num_elems = [1, 2, 5, 10, 100, 1000, ]

def exact(x):
  return (1-exp(Pe*x))/(1-exp(Pe))

def exact_derivative(x):
  return -Pe*exp(Pe*x)/(1-exp(Pe))

def supg(x, x1, x2):
  return exact(x1)+(x-x1)/(x2-x1)*(exact(x2)-exact(x1))

def supg_derivative(x, x1, x2):
  return (supg(x2, x1, x2) - supg(x1, x1, x2))/(x2-x1)

def galerkin(x, x1, x2, n, i):
  alpha = Pe*(x2-x1)/2
  ca = (1+alpha)/(1-alpha)
  f1 = (1-ca**i)/(1-ca**n)
  f2 = (1-ca**(i+1))/(1-ca**n)
  return f1+(x-x1)/(x2-x1)*(f2-f1)

def galerkin_derivative(x, x1, x2, n, i):
  return (galerkin(x2, x1, x2, n, i) - galerkin(x1, x1, x2, n, i))/(x2-x1)

def supg_diff_sq(x, x1, x2):
  return (exact(x)-supg(x, x1, x2))**2

def supg_derivative_diff_sq(x, x1, x2):
  return (exact_derivative(x)-supg_derivative(x, x1, x2))**2

def galerkin_diff_sq(x, x1, x2, n, i):
  return (exact(x)-galerkin(x, x1, x2, n, i))**2

def galerkin_derivative_diff_sq(x, x1, x2, n, i):
  return (exact_derivative(x)-galerkin_derivative(x, x1, x2, n, i))**2

supg_err = zeros(len(num_elems))
supg_derivative_err = zeros(len(num_elems))
galerkin_err = zeros(len(num_elems))
galerkin_derivative_err = zeros(len(num_elems))
c = 0
for n in num_elems:
  x = linspace(0, 1, n+1)

  supg_l2_err = zeros(n)
  supg_h1_err = zeros(n)
  galerkin_l2_err = zeros(n)
  galerkin_h1_err = zeros(n)

  for i in range(0,n):
    x1 = x[i]
    x2 = x[i+1]

    supg_l2_err[i], _ = quad(supg_diff_sq, x1, x2, args=(x1, x2), epsrel = 1e-2, )
```

```
    supg_err[c] = supg_err[c] + supg_l2_err[i]
    supg_h1_err[i], _ = quad(supg_derivative_diff_sq, x1, x2, args=(x1, x2), epsrel = 1e-2, )
    supg_derivative_err[c] = supg_derivative_err[c] + supg_h1_err[i]

    galerkin_l2_err[i], _ = quad(galerkin_diff_sq, x1, x2, args=(x1, x2, n, i), epsrel = 1e-2, )
    galerkin_err[c] = galerkin_err[c] + galerkin_l2_err[i]
    galerkin_h1_err[i], _ = quad(galerkin_derivative_diff_sq, x1, x2, args=(x1, x2, n, i), epsrel = 1e-2, )
    galerkin_derivative_err[c] = galerkin_derivative_err[c] + galerkin_h1_err[i]

  supg_err[c] = sqrt(supg_err[c])
  supg_derivative_err[c] = sqrt(supg_derivative_err[c])
  galerkin_err[c] = sqrt(galerkin_err[c])
  galerkin_derivative_err[c] = sqrt(galerkin_derivative_err[c])
  c = c+1

figure(1)
loglog(num_elems, galerkin_err, linewidth=1, label='Galerkin')
loglog(num_elems, supg_err, linewidth=1, label='SUPG')
slope_marker((40, .06), (-0.6, 1), .2, invert = True)
# slope_marker((70, .03), (-0.7, 1), .2, invert = True)
slope_marker((300, .01), (-2, 1), .2)
text(1.4, .002, "Preasymptotic Regime", fontsize=16)
plot([1, 100],[.01, .01], color='k', linewidth=8)
xlabel('Number of Elements')
ylabel(r'$L_2$ Error')
legend()

figure(2)
loglog(num_elems, galerkin_derivative_err, linewidth=1, label='Galerkin')
loglog(num_elems, supg_derivative_err, linewidth=1, label='SUPG')
slope_marker((1000, 4), (-1, 1), .2)
text(1.4, .2, "Preasymptotic Regime", fontsize=16)
plot([1, 100],[.8, .8], color='k', linewidth=8)
xlabel('Number of Elements')
ylabel(r'$H^1$ Seminorm Error')
legend()

figure(3)
loglog(num_elems, galerkin_err + galerkin_derivative_err, linewidth=1, label='Galerkin')
loglog(num_elems, supg_err + supg_derivative_err, linewidth=1, label='SUPG')
slope_marker((1000, 4), (-1, 1), .2)
text(1.4, .2, "Preasymptotic Regime", fontsize=16)
plot([1, 100],[.8, .8], color='k', linewidth=8)
xlabel('Number of Elements')
ylabel(r'$H^1$ Norm Error')
legend()

show()
```