HW05_prob3

November 26, 2015

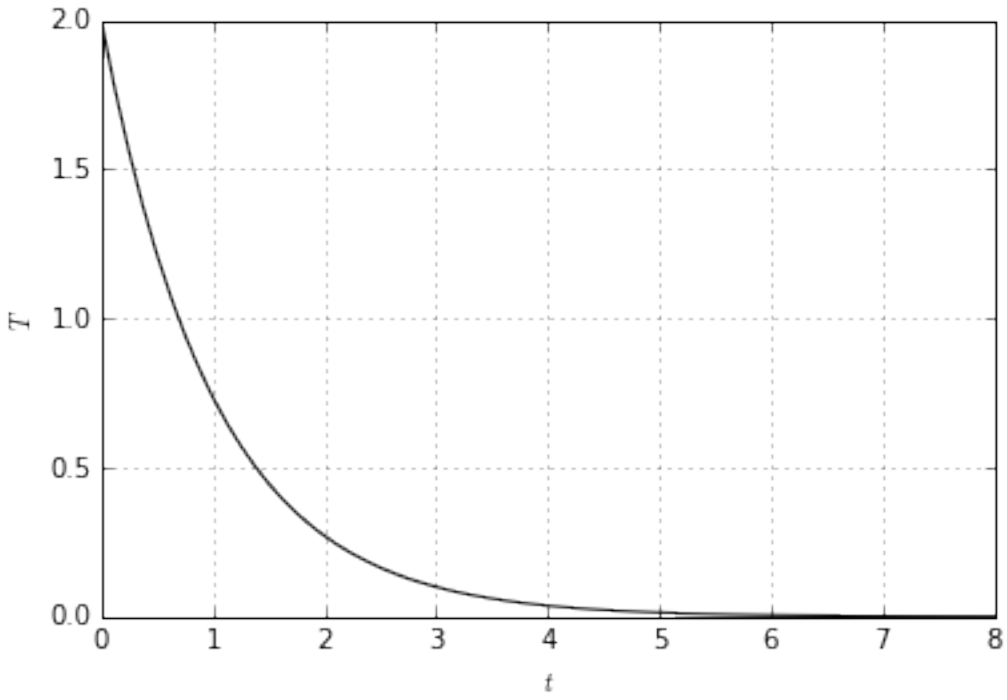
```
In [1]: import numpy as np
        import math
        import scipy.linalg
        from scipy.sparse import diags
        %matplotlib inline
        import matplotlib.pyplot as plt
```

Define the domain

```
In [2]: s_arr = np.array([4, 2, 1, 0.5, 0.25, 0.125])
        t0 = 0
        tf = 8
```

Analytical Solution

```
In [3]: lmbda = 1
        t_ann = np.linspace(t0, tf, (tf-t0)/s_arr[5] + 1)

        T_an = map(lambda t: 2 * np.exp(-lmbda * t), t_ann)
        plt.plot(t_ann, T_an, 'k-');

        plt.xlabel('$t$')
        plt.ylabel('$T$')
        plt.grid()
```



1-D System Integrator

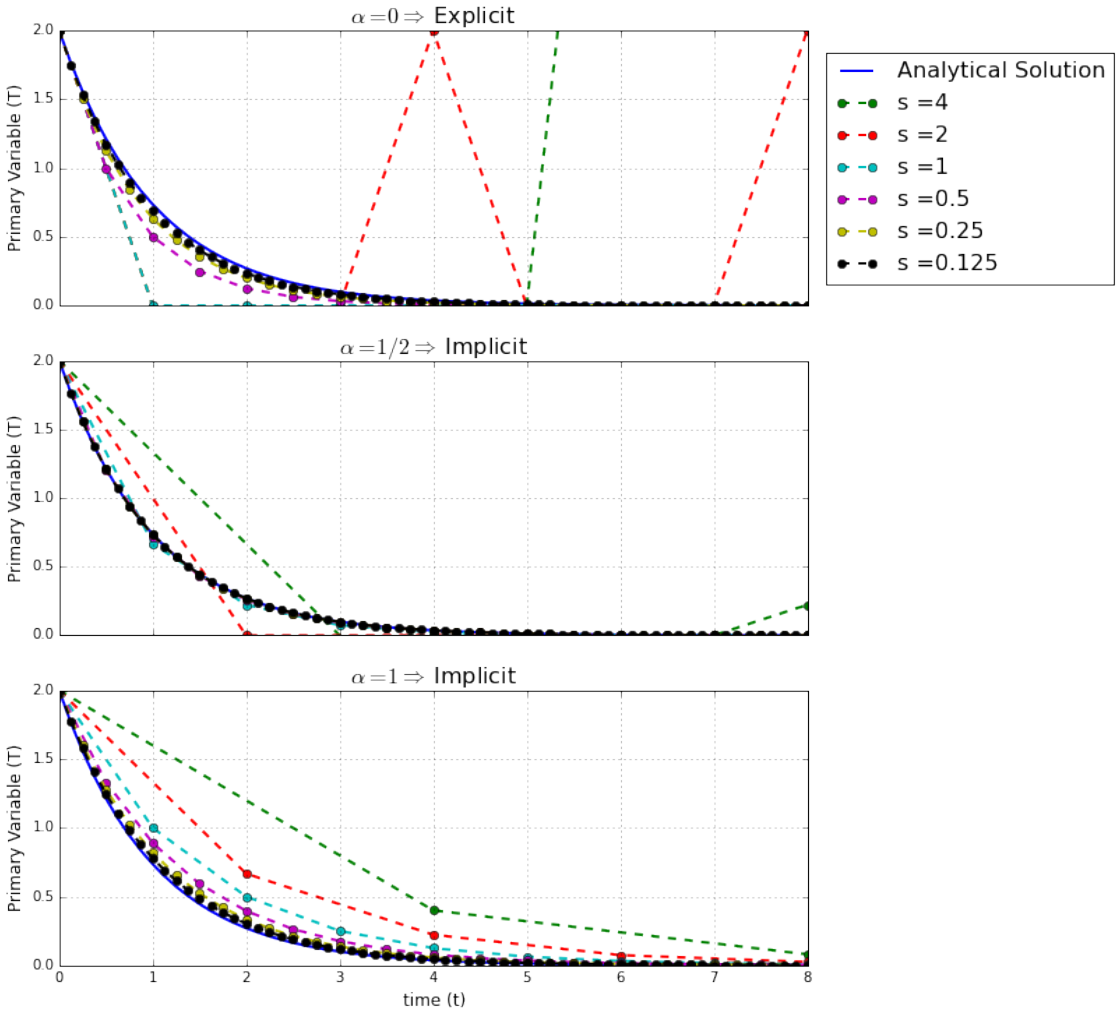
In [7]:
```
# integration methods
alpha_arr = np.array([0, 0.5, 1.0])

#define time step
s_arr = np.array([4., 2., 1., 0.5, 0.25, 0.125])

#domain
t0 = 0
tf = 8
x_num = np.zeros(1)

#coeficient functions
C = 1. #specific heat
K = 1. #conductance

#forcing function
f = lambda t: 0*t

# intial conditions
T0 = 2

# array for plotting results
T_plot = np.zeros(((tf-t0)/min(s_arr) + 1, len(s_arr), len(alpha_arr)))
t_plot = np.zeros(((tf-t0)/min(s_arr) + 1, len(s_arr), len(alpha_arr)))
t_idx = np.zeros(len(s_arr)) # for plotting
```

```
j = 0 # index for time step size (s)
k = 0 # index for alpha value
for alpha in alpha_arr: # loop on integration type
#     print alpha
    for s in s_arr: # loop on time step size
        t_num = np.linspace(t0, tf, (tf-t0)/s + 1) # temporal discretization
        # setup initial array for solutions and apply initial conditions
        T_num = np.zeros(len(t_num))
        T_num[0] = T0

        # decompose [A] - assumes constant c and k
        A_not_decomp = diags([(C + alpha*s*K)],[0],shape = (len(x_num), len(x_num))).toarray()
        Aq, Ar = scipy.linalg.qr(A_not_decomp) #Aq -> orthogonal, Ar -> upper diagonal

        # calculate [B]
        B = diags([C - (1 - alpha) * s *K],[0],shape = (len(x_num), len(x_num))).toarray()

        for i in xrange(len(T_num)-1): # loop through time steps (main program)
            t = t_num[i+1] # current time
            F = (1-alpha)*s*f(t) + alpha*s*f(t)
            b = B.dot(T_num[i]) + F
            b_hat = np.transpose(Aq).dot(b)
            T_num[i+1] = scipy.linalg.solve_triangular(Ar, b_hat)

        T_plot[0:len(T_num), j, k] = T_num
        t_plot[0:len(T_num), j, k] = t_num
        t_idx[j] = len(t_num)
        j = j+1
    j = 0
    k = k+1
```

Plot Convergence Analysis

In [9]:
```
# create labels
label_s = ['4', '2', '1', '0.5', '0.25', '0.125']
label_alpha = ['0', '0.5', '1.0']

lbl = [None] * 3*6
k = 0
for i in xrange(len(label_s)):
    lbl[k] = 's ='+ label_s[int(i)]
    k = k + 1

# figure 1
fig_3plot, (ax1, ax2, ax3) = plt.subplots(3, sharex=True, figsize = (12,12))
ax1.plot(t_ann, T_an, '-', markersize=10, lw = 2, label='Analytical Solution')
ax1.plot(t_plot[0:t_idx[0],0,0], T_plot[0:t_idx[0],0,0], 'o--', markersize=6, lw = 2, label=lbl
ax1.plot(t_plot[0:t_idx[1],1,0], T_plot[0:t_idx[1],1,0], 'o--', markersize=6, lw = 2, label=lbl
ax1.plot(t_plot[0:t_idx[2],2,0], T_plot[0:t_idx[2],2,0], 'o--', markersize=6, lw = 2, label=lbl
ax1.plot(t_plot[0:t_idx[3],3,0], T_plot[0:t_idx[3],3,0], 'o--', markersize=6, lw = 2, label=lbl
ax1.plot(t_plot[0:t_idx[4],4,0], T_plot[0:t_idx[4],4,0], 'o--', markersize=6, lw = 2, label=lbl
ax1.plot(t_plot[0:t_idx[5],5,0], T_plot[0:t_idx[5],5,0], 'o--', markersize=6, lw = 2, label=lbl

ax1.set_ylabel('Primary Variable (T)', fontsize = 12)
```

```
ax1.set_title(''r'$\alpha = 0 \Rightarrow$ Explicit' , fontsize = 16)
ax1.grid(b = True, which = 'major')
ax1.grid(b = True, which = 'major')
ax1.set_ylim(0, 2)

box1 = ax1.get_position()
ax1.set_position([box1.x0, box1.y0, box1.width * 0.8, box1.height])
ax1.legend(loc='center left', bbox_to_anchor=(1.01, 0.5),fontsize=16)

# # figure 2
ax2.plot(t_ann, T_an, '-', markersize=10, lw = 2, label='Analytical Solution')
ax2.plot(t_plot[0:t_idx[0],0,1], T_plot[0:t_idx[0],0,1], 'o--', markersize=6, lw = 2, label=lbl
ax2.plot(t_plot[0:t_idx[1],1,1], T_plot[0:t_idx[1],1,1], 'o--', markersize=6, lw = 2, label=lbl
ax2.plot(t_plot[0:t_idx[2],2,1], T_plot[0:t_idx[2],2,1], 'o--', markersize=6, lw = 2, label=lbl
ax2.plot(t_plot[0:t_idx[3],3,1], T_plot[0:t_idx[3],3,1], 'o--', markersize=6, lw = 2, label=lbl
ax2.plot(t_plot[0:t_idx[4],4,1], T_plot[0:t_idx[4],4,1], 'o--', markersize=6, lw = 2, label=lbl
ax2.plot(t_plot[0:t_idx[5],5,1], T_plot[0:t_idx[5],5,1], 'o--', markersize=6, lw = 2, label=lbl

ax2.set_ylabel('Primary Variable (T)', fontsize = 12)
ax2.set_title(''r'$\alpha = 1/2 \Rightarrow$ Implicit' , fontsize = 16)
ax2.grid(b = True, which = 'major')
ax2.grid(b = True, which = 'major')
ax2.set_ylim(0, 2)
box2 = ax2.get_position()
ax2.set_position([box2.x0, box2.y0, box2.width * 0.8, box2.height])

# # figure 3
ax3.plot(t_ann, T_an, '-', markersize=10, lw = 2, label='Analytical Solution')
ax3.plot(t_plot[0:t_idx[0],0,2], T_plot[0:t_idx[0],0,2], 'o--', markersize=6, lw = 2, label=lbl
ax3.plot(t_plot[0:t_idx[1],1,2], T_plot[0:t_idx[1],1,2], 'o--', markersize=6, lw = 2, label=lbl
ax3.plot(t_plot[0:t_idx[2],2,2], T_plot[0:t_idx[2],2,2], 'o--', markersize=6, lw = 2, label=lbl
ax3.plot(t_plot[0:t_idx[3],3,2], T_plot[0:t_idx[3],3,2], 'o--', markersize=6, lw = 2, label=lbl
ax3.plot(t_plot[0:t_idx[4],4,2], T_plot[0:t_idx[4],4,2], 'o--', markersize=6, lw = 2, label=lbl
ax3.plot(t_plot[0:t_idx[5],5,2], T_plot[0:t_idx[5],5,2], 'o--', markersize=6, lw = 2, label=lbl

ax3.set_xlabel('time (t)', fontsize = 12)
ax3.set_ylabel('Primary Variable (T)', fontsize = 12)
ax3.set_title(''r'$\alpha = 1 \Rightarrow$ Implicit' , fontsize = 16)
ax3.grid(b = True, which = 'major')
ax3.grid(b = True, which = 'major')
ax3.set_ylim(0, 2)

box3 = ax3.get_position()
ax3.set_position([box3.x0, box3.y0, box3.width * 0.8, box3.height])

fig_name = '3a_all.pdf'
path = '/Users/Lampe/Documents/UNM_Courses/ME-500/HW05/'
fig_3plot.savefig(path + fig_name)
```



0.0.1 3.b

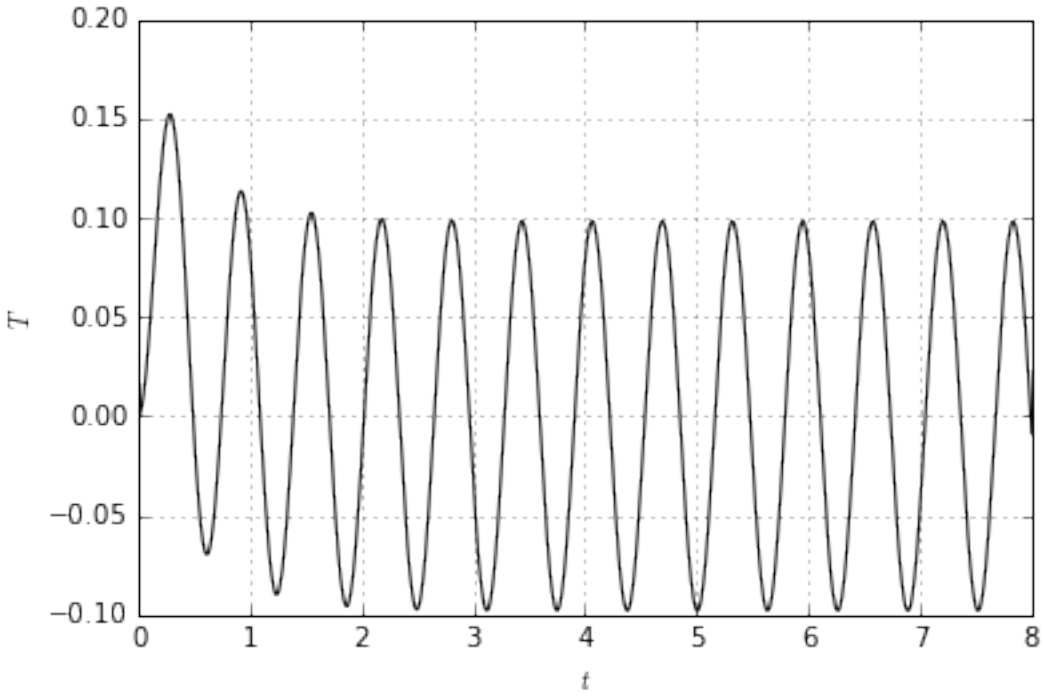
Set up transient problem Calculate the critical time step for Explicit analysis $\alpha = 0$

```
In [10]: #coeficient functions
         C = 1. #specific heat
         K = 2. #conductance
         s_cr = 1
         omega = 10
         t_p = 2*np.pi/omega #period of forcing function
         print s_cr
         print t_p
```

```
1
0.628318530718
```

```
In [11]: lmbda = 1
         t_ann = np.linspace(t0, tf, (tf-t0)/0.01 + 1)
         t_ann_tp = np.linspace(t0, tf, (tf-t0)/(t_p/4.) + 1)
```

```
T_an = map(lambda t: 1./52*(5*np.exp(-2*t)+np.sin(10*t)-5*np.cos(10*t)), t_ann)
T_an_tp = map(lambda t: 1./52*(5*np.exp(-2*t)+np.sin(10*t)-5*np.cos(10*t)), t_ann_tp)
plt.plot(t_ann, T_an, 'k-');

plt.xlabel('$t$')
plt.ylabel('$T$')
plt.grid()
```



In [36]:
```
#define time step
# s_arr = np.linspace(s_cr, t_p/4.0, 3)
s_arr = np.linspace(.25, t_p/4.0, 3)
print s_arr

# integration methods
alpha_arr = np.array([0.5, 0.75, 1.0])

#domain
t0 = 0
tf = 8
x_num = np.zeros(1)

#forcing function
f = lambda t: np.sin(omega*t)

# intial conditions
T0 = 0
```

```
        # array for plotting results
        T_plot = np.zeros(((tf-t0)/min(s_arr) + 1, len(s_arr), len(alpha_arr)))
        t_plot = np.zeros(((tf-t0)/min(s_arr) + 1, len(s_arr), len(alpha_arr)))
        t_idx = np.zeros(len(s_arr)) # for plotting

        j = 0 # index for time step size (s)
        k = 0 # index for alpha value
        for alpha in alpha_arr: # loop on integration type
            for s in s_arr: # loop on time step size
                t_num = np.linspace(t0, tf, (tf-t0)/s + 1) # temporal discretization
                # setup initial array for solutions and apply initial conditions
                T_num = np.zeros(len(t_num))
                T_num[0] = T0

                # decompose [A] - assumes constant c and k
                A_not_decomp = diags([(C + alpha*s*K)],[0],shape = (len(x_num), len(x_num))).toarray()
                Aq, Ar = scipy.linalg.qr(A_not_decomp) #Aq -> orthogonal, Ar -> upper diagonal

                # calculate [B]
                B = diags([C - (1 - alpha) * s *K],[0],shape = (len(x_num), len(x_num))).toarray()

                for i in xrange(len(T_num)-1): # loop through time steps (main program)
                    t = t_num[i+1] # current time
                    F = (1-alpha)*s*f(t) + alpha*s*f(t)
                    b = B.dot(T_num[i]) + F
                    b_hat = np.transpose(Aq).dot(b)
                    T_num[i+1] = scipy.linalg.solve_triangular(Ar, b_hat)

                T_plot[0:len(T_num), j, k] = T_num
                t_plot[0:len(T_num), j, k] = t_num
                t_idx[j] = len(t_num)
                j = j+1
            j = 0
            k = k+1
```

```
[ 0.25        0.20353982  0.15707963]
```

Calculate error norms

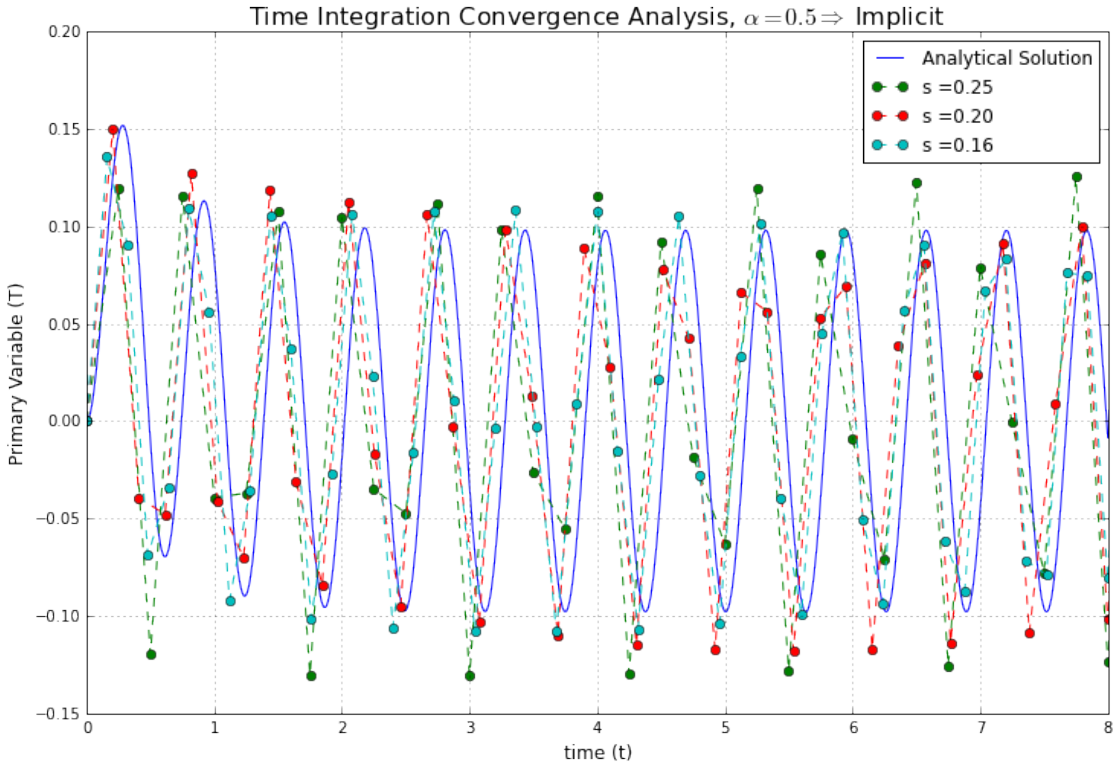
```
In [37]: e_05 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,0])**2)**0.5
         e_075 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,1])**2)**0.5
         e_1 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,2])**2)**0.5
         error_scr = np.array([e_05, e_075, e_1])
         print error_scr

         # len(T_plot[0:t_idx[2],2,0])

         e_05 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an_tp - T_plot[0:t_idx[2],2,0])**2)**0.5
         e_075 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an_tp - T_plot[0:t_idx[2],2,1])**2)**0.5
         e_1 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an_tp - T_plot[0:t_idx[2],2,2])**2)**0.5
         error_tp = np.array([e_05, e_075, e_1])
         error_tp
```

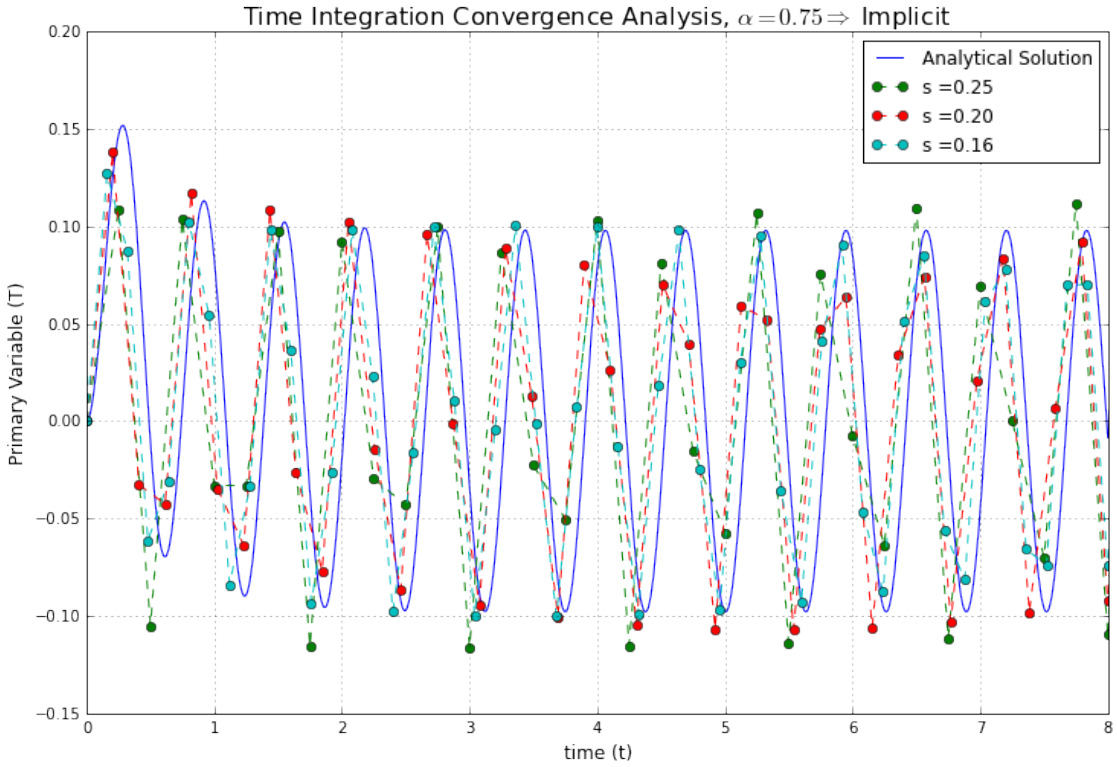
```
ValueError                                Traceback (most recent call last)

<ipython-input-37-0e8f478a743e> in <module>()
----> 1 e_05 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,0])**2)**0.5
      2 e_075 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,1])**2)**0.5
      3 e_1 = 1./len(T_an)**(0.5) * np.sum(np.abs(T_an - T_plot[0:t_idx[0],0,2])**2)**0.5
      4 error_scr = np.array([e_05, e_075, e_1])
      5 print error_scr


ValueError: operands could not be broadcast together with shapes (801,) (33,)
```

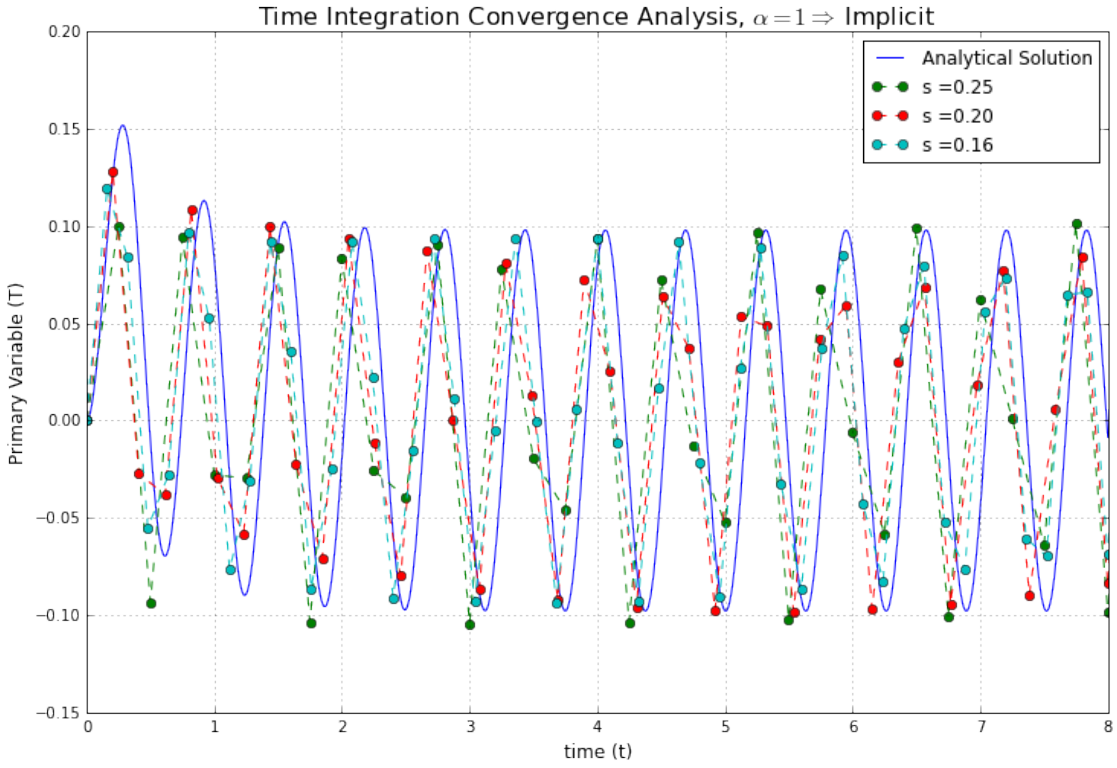
Create Plots

In [39]:
```
# create labels
label_s = ['0.25', '0.20', '0.16']

lbl = [None] * 3
k = 0
for i in xrange(len(label_s)):
    lbl[k] = 's ='+ label_s[int(i)]
    k = k + 1

# figure 4
fig_1DOF_4, ax = plt.subplots(figsize = (12,8))
ax.plot(t_ann, T_an, '-', markersize=6, lw = 1, label='Analytical Solution')
ax.plot(t_plot[0:t_idx[0],0,0], T_plot[0:t_idx[0],0,0], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[1],1,0], T_plot[0:t_idx[1],1,0], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[2],2,0], T_plot[0:t_idx[2],2,0], 'o--', markersize=6, lw = 1, label=lbl

ax.set_xlabel('time (t)', fontsize = 12)
ax.set_ylabel('Primary Variable (T)', fontsize = 12)
ax.set_title(''r'Time Integration Convergence Analysis, $\alpha = 0.5 \Rightarrow$ Implicit' ,
ax.grid(b = True, which = 'major')
ax.grid(b = True, which = 'major')
# ax.set_ylim(0, 2)
ax.legend(loc=0, framealpha = 1)

fig_name = '3b_1.pdf'
path = '/Users/Lampe/Documents/UNM_Courses/ME-500/HW05/'
fig_1DOF_4.savefig(path + fig_name)


# figure 5
fig_1DOF_5, ax = plt.subplots(figsize = (12,8))
ax.plot(t_ann, T_an, '-', markersize=6, lw = 1, label='Analytical Solution')
ax.plot(t_plot[0:t_idx[0],0,1], T_plot[0:t_idx[0],0,1], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[1],1,1], T_plot[0:t_idx[1],1,1], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[2],2,1], T_plot[0:t_idx[2],2,1], 'o--', markersize=6, lw = 1, label=lbl

ax.set_xlabel('time (t)', fontsize = 12)
ax.set_ylabel('Primary Variable (T)', fontsize = 12)
```

```
ax.set_title(''r'Time Integration Convergence Analysis, $\alpha = 0.75 \Rightarrow$ Implicit'
ax.grid(b = True, which = 'major')
ax.grid(b = True, which = 'major')
# ax.set_ylim(0, 2)
ax.legend(loc=0, framealpha = 1)

fig_name = '3b_2.pdf'
path = '/Users/Lampe/Documents/UNM_Courses/ME-500/HW05/'
fig_1DOF_5.savefig(path + fig_name)

# figure 6
fig_1DOF_6, ax = plt.subplots(figsize = (12,8))
ax.plot(t_ann, T_an, '-', markersize=6, lw = 1, label='Analytical Solution')
ax.plot(t_plot[0:t_idx[0],0,2], T_plot[0:t_idx[0],0,2], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[1],1,2], T_plot[0:t_idx[1],1,2], 'o--', markersize=6, lw = 1, label=lbl
ax.plot(t_plot[0:t_idx[2],2,2], T_plot[0:t_idx[2],2,2], 'o--', markersize=6, lw = 1, label=lbl

ax.set_xlabel('time (t)', fontsize = 12)
ax.set_ylabel('Primary Variable (T)', fontsize = 12)
ax.set_title(''r'Time Integration Convergence Analysis, $\alpha = 1 \Rightarrow$ Implicit' , f
ax.grid(b = True, which = 'major')
ax.grid(b = True, which = 'major')
# ax.set_ylim(0, 2)
ax.legend(loc=0, framealpha = 1)

fig_name = '3b_3.pdf'
path = '/Users/Lampe/Documents/UNM_Courses/ME-500/HW05/'
fig_1DOF_6.savefig(path + fig_name)
```




Time Integration Convergence Analysis, $\alpha = 0.75 \Rightarrow$ Implicit
Analytical Solution
s =0.25
s =0.20
s =0.16
Primary Variable (T)
time (t)


Time Integration Convergence Analysis, $\alpha = 1 \Rightarrow$ Implicit
Analytical Solution
s =0.25
s =0.20
s =0.16
Primary Variable (T)
time (t)

In [ ]: