Summary of Routines Written in Python

In [7]:
```
import sys
from scipy import linalg as LA
import numpy as np
from matplotlib import pyplot as plt

sys.path.append('/Users/Lampe/PyScripts')
import constit_mod_02 as cs

# np.set_printoptions(precision = 2, suppress = True)
```

In [8]:
```
from pylab import *
import matplotlib.pyplot as plt
import matplotlib.pylab as pylab

%matplotlib inline
```

Printing Functions ¶

In [9]:
```
##################################
# printing functions
##################################
def valprint(string, value):
    """ Inforces uniform formatting of scalar value outputs."""
    print("{0:>15}: {1: .2e}".format(string, value))

def matprint(string, value):
    """ inforces uniform formatting of matrix value outputs."""
    print("{0}:".format(string))
    print(value)
```

In [10]:
```
# Driver Program for Constitutive Modeling

# path: integer (0 to 9) - indentifies the type of loading or strain path e.g. the test type
#     0 = strain prescribed - the default
#     1 = uniaxial stress
#     2 = plane stress
#     3 = hydrostatic stress
#     4 = Triaxial Extension
#     5 = Triaxial Compression
#     6 = Relaxation (Constain Strain) 1D viscoelasticity
#     7 = Creep (Constant Stress) 1D viscoelasticity
#     8: undefined, may be defined later
# leg: defines which leg, increment size can vary with leg
# nleg: number of legs for the load path that is being modeled
# term_type: defines how each of the nleg will be terminated
#     1 = n_max, maximium number of steps
#     2 = strs_max, maximium stress value
#     3 = strs_min, minimium stress value
#     4 = strn_max, maximium strain value
#     5 = strn_min, minimium strain value
#     6 = p_max, maximium pressure (mean stress)
#     7 = p_min, minimium pressure (mean stress)
#     8 = t_max, maximum allowable time (irow * t_inc)
# mat_type: integer - defines how the material will be modeled
#     0 = Elastic Material (default)
#     1 = Viscoelastic Material (1D)
#     3 = Elastoplastic Material
#     All Other Values = Error - function will be terminated

# SM: storage matrix
```

Constant values for this assignment

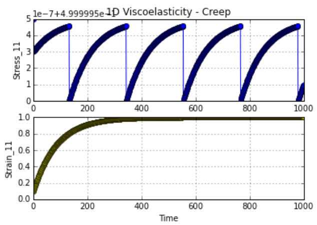
In [11]:
```
#user defined strain increments
#define an increment value for each leg if needed
strn_inc_11 = [0.1]
strn_inc_22 = [0.0]
strn_inc_33 = [0.0]
strn_inc_12 = [0.0]

##########################
### Material Parameters ###
##########################
# Elastic Parameters
Y1, Y2, Y3 = 28.0/6, 28.0/6, 28.0/6 # youngs
nu12, nu23, nu31 = 1.0/6, 1.0/6, 1.0/6 # poissons
G44, G55, G66 = 0.5, 0.5, 0.5 # shear

out_dir = "/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04"
```

Template for calling Driver program

In [27]:
```
run_title = "Test_module"

#### Model Parameters ###
n_leg = 1 # number of legs / changes in strain increment
path_type = [7] # Loading path default is zero

# termination type
term_type = np.ones(n_leg) * [8]

#define termination values
n_max = 150000 #global termination on number of loops

# local terminations dependent on term_type
inc_max = 50000 # per leg
strs_max = 1
strs_min = -1.0
strn_max = 1
strn_min = -1
p_max = 0.5
p_min = -0.5

# mat_type: integer - defines how the material will be modeled
#     0 = Elastic Material (default)
#     1 = Viscoelastic Material
#     3 = Elastoplastic Material
mat_type = 1

# Viscous Parameters
tau_1, tau_b_1 = 10.0, 9.0
tau_2, tau_b_2 = 0, 0
e_max, e_dot_max = 0.002, 0.0001 # cyclic loading parameters
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

strn_func = 0 # zero if constant strain increment, 1 if cyclic loading

t_inc = tau_b_1 / 10.0 # (np.pi * e_max / e_dot_max) / 20.0 #tau_b_1 / 10.0 # time increment for viscoelastic analysis
t_max = 1000 #np.pi * e_max / e_dot_max *2 #10 #

# Plastic Parameters

# call driver
SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
           Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
           e_max, e_dot_max, sigma_0, c_star,
           strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
           mat_type = mat_type, t_inc = t_inc, n_max = n_max,
           inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
           p_max = p_max, p_min = p_min, t_max = t_max, strn_func = strn_func)

out_fig_name = run_title
out_fig_fmt = "pdf"

x1 = 48
y1 = 1

x2 = 48
y2 = 23
#36 backstrin
#48 time

# call plotting device
cs.Plot_Setup(SM, col_namev, out_dir, out_name = out_fig_name, irow = irow,
                       sub_plot = 1,path = path_type[len(path_type) -1],
                       x1 = x1, x2 = x2, y1 = y1, y2 = y2, fmt = out_fig_fmt)
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 10.    9.    0.    0.    0.    0.    0.5   0.    0.    0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.11e+03
```

Out[27]: 'Plotting Complete'



1.iii) Demonstrate progrma works by copmaring numerical solutions to analytical solutions

- calculate analytical solution to relaxation problem

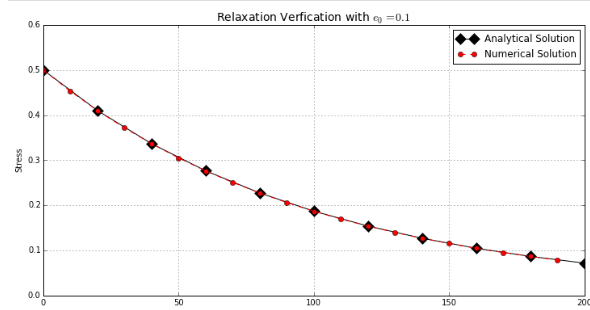
In [28]:
```
tau_1 = 101.0
tau_b_1 = 100.0

e_inc = 0.1
E_11 = 5
sigma_0 = E_11 * e_inc
t = np.arange(0,200.5,20.0)
sigma_relax = np.zeros(len(t))

tau_star = tau_1 * tau_b_1 / (tau_1 - tau_b_1)
strn_0 = sigma_0 / E_11

# relaxation
sigma_relax = sigma_0 / tau_1 * (tau_1 - tau_b_1 + tau_b_1 * exp(-t / tau_b_1))
```

- calculate numerical solution to relaxation problem

In [29]:
```
path_type = [6] # relaxation

tau_1 = 101.0
tau_b_1 = 100.0
tau_2 = 0
ttau_b_2 = 0

t_inc = tau_b_1 / 10.0 #
t_max = 200.0
inc = t_max / t_inc

SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

stress_num = SM[0:inc,1]
t_num = SM[0:inc,48]
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 101.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  2.10e+01
```

In [30]:
```
# plot relaxation verification
fig_relax, ax = plt.subplots(figsize = (12,6))

ax.plot(t, sigma_relax, 'kD-',markersize = 10, label="Analytical Solution")
ax.plot(t_num, stress_num, 'ro--', label="Numerical Solution")

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Stress')
ax.set_title(''r'Relaxation Verfication with $e_0= 0.1$ ', fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_relax.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/1_Rela
_Ver.pdf")

show()
```



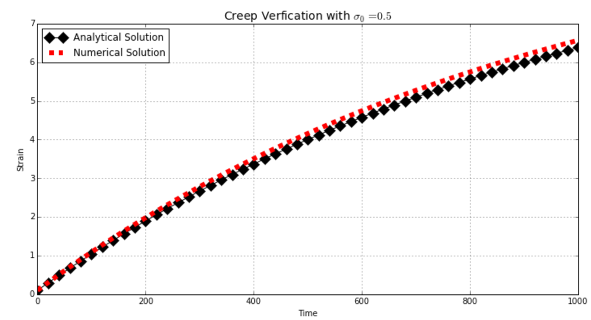
-calculate analytical solution to creep problem

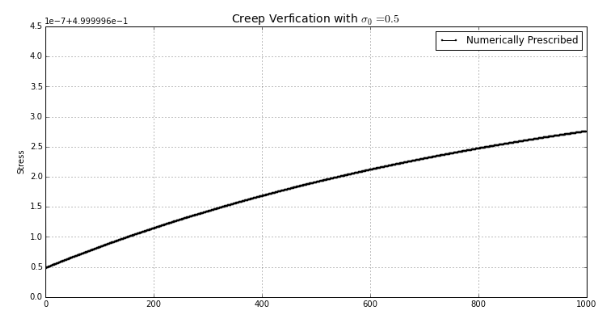
In [16]:
```
tau_1 = 10.1
tau_b_1 = 10.0

e_inc = 0.1
E_11 = 5
sigma_0 = E_11 * e_inc
t = np.arange(0,1020,20.0)
sigma_relax = np.zeros(len(t))

tau_star = tau_1 * tau_b_1 / (tau_1 - tau_b_1)
strn_0 = sigma_0 / E_11

# creep
strn_creep = strn_0 / (tau_1 - tau_b_1) * (tau_1 - tau_b_1 * exp(-t / tau_star))
```

In [17]:
```
path_type = [7] # creep

tau_1 = 10.10
tau_b_1 = 10.00
tau_2 = 0
ttau_b_2 = 0
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

t_inc = tau_b_1 / 10 #
t_max = 1000.0
inc = t_max / t_inc

SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

strn_num = SM[0:inc,23]
t_num = SM[0:inc,48]
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 10.1  10.    0.    0.    0.    0.    0.5   0.    0.    0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.00e+03
```

In [22]:
```
# plot creep verification
fig_creep_strn, ax = plt.subplots(figsize = (12,6))

ax.plot(t, strn_creep, 'kD-',markersize = 10, label="Analytical Solution")
ax.plot(t_num, strn_num, 'r--', lw = 6, label="Numerical Solution")

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Strain')
ax.set_title('Creep Verfication with 'r'$\sigma_0 = 0.5$', fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')

fig_creep_strn.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/1
creep_strn_Ver.pdf")

show()

fig_creep_strs, ax = plt.subplots(figsize = (12,6))

ax.plot(t_num, SM[0:inc, 1], 'ko-',markersize = 2, label="Numerically Prescribed")

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Stress')
ax.set_title('Creep Verfication with 'r'$\sigma_0 = 0.5$', fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_creep_strs.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/1
Creep_strs_Ver.pdf")

show()
```



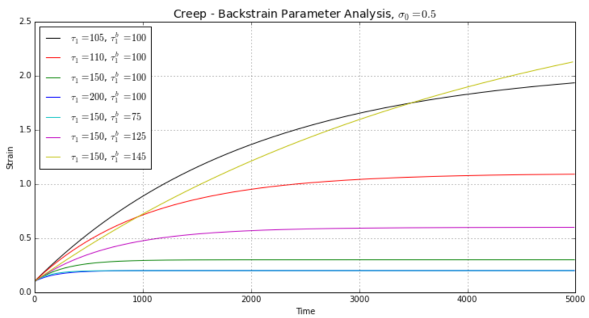


2. Show the effects of choosing various values for parameters

In [13]:
```
path_type = [7] # creep
t1 = np.array([105, 110, 150, 200, 150, 150, 150])
t1b =np.array([100., 100, 100, 100, 75, 125, 145])
t2 = 0
t2b =0
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

time_inc = t1b / 10.0#
t_max = 5000.0
inc = t_max / time_inc

loop_cnt = len(t1b)

parm_effects_creep = np.zeros((np.amax(inc), loop_cnt))
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1[i]
    tau_b_1 = t1b[i]
    t_inc = time_inc[i]
#     tau_2 = t2[i]
#     tau_b_2 = t2b[i]

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    parm_effects_creep[0:inc[i],i] = SM[0:inc[i],23]
    time[0:inc[i],i] = SM[0:inc[i],48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 105.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  5.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 110.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  5.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  5.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 200.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  5.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.    75.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.68e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   125.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  4.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   145.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  3.46e+02
[ 500.    500.    500.    500.    666.67  400.    344.83]
```

In [14]:
```python
# plot creep verification
fig_parm_effects_creep, ax = plt.subplots(figsize = (12,6))

ax.plot(time[0:inc[0],0], parm_effects_creep[0:inc[0],0], 'k-',markersize = 4, label=''r'$\tau_1=105$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[1],1], parm_effects_creep[0:inc[1],1], 'r-',markersize = 4, label=''r'$\tau_1=110$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[2],2], parm_effects_creep[0:inc[2],2], 'g-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[3],3], parm_effects_creep[0:inc[3],3], 'b-',markersize = 4, label=''r'$\tau_1=200$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[4],4], parm_effects_creep[0:inc[4],4], 'c-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 75$')
ax.plot(time[0:inc[5],5], parm_effects_creep[0:inc[5],5], 'm-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 125$')
ax.plot(time[0:inc[6],6], parm_effects_creep[0:inc[6],6], 'y-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 145$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Strain')
ax.set_title(''r'Creep - Backstrain Parameter Analysis, $\sigma_0 = 0.5$', fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_parm_effects_creep.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYE
/HW04/2_creep.pdf")

show()
```



Relaxation

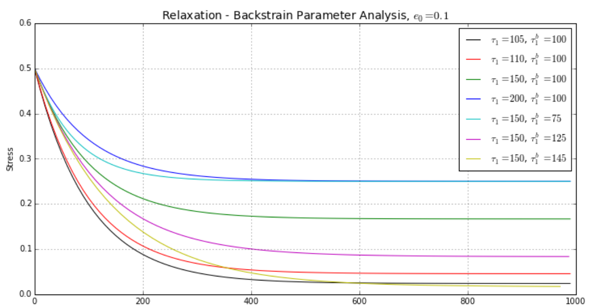
In [31]:
```python
path_type = [6] # relaxation
t1 = np.array([105, 110, 150, 200, 150, 150, 150])
t1b =np.array([100., 100, 100, 100, 75, 125, 145])
t2 = 0
t2b =0
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

time_inc = t1b / 10.0#
t_max = 1000.0
inc = t_max / time_inc

loop_cnt = len(t1b)

parm_effects_relax = np.zeros((np.amax(inc), loop_cnt))
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1[i]
    tau_b_1 = t1b[i]
    t_inc = time_inc[i]
#     tau_2 = t2[i]
#     tau_b_2 = t2b[i]

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    parm_effects_relax[0:inc[i],i] = SM[0:inc[i],1]
    time[0:inc[i],i] = SM[0:inc[i],48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 105.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 110.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 200.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.    75.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.34e+02
Viscoelastic Material
user defined elastic parm
```

```
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   125.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  8.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   145.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  7.00e+01
[ 100.    100.    100.    100.    133.33   80.     68.97]
```

In [32]:

```
# plot relaxatio verification
fig_parm_effects_relax, ax = plt.subplots(figsize = (12,6))

ax.plot(time[0:inc[0],0], parm_effects_relax[0:inc[0],0], 'k-',markersize = 4, label=''r'$\tau_1=105$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[1],1], parm_effects_relax[0:inc[1],1], 'r-',markersize = 4, label=''r'$\tau_1=110$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[2],2], parm_effects_relax[0:inc[2],2], 'g-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[3],3], parm_effects_relax[0:inc[3],3], 'b-',markersize = 4, label=''r'$\tau_1=200$, $\tau^b_1 = 100$')
ax.plot(time[0:inc[4],4], parm_effects_relax[0:inc[4],4], 'c-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 75$')
ax.plot(time[0:inc[5],5], parm_effects_relax[0:inc[5],5], 'm-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 125$')
ax.plot(time[0:inc[6],6], parm_effects_relax[0:inc[6],6], 'y-',markersize = 4, label=''r'$\tau_1=150$, $\tau^b_1 = 145$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Stress')
ax.set_title(''r'Relaxation - Backstrain Parameter Analysis, $e_{0} = 0.1$', fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_parm_effects_relax.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYE
/HW04/2_relax_Ver.pdf")

show()
```



In [41]:

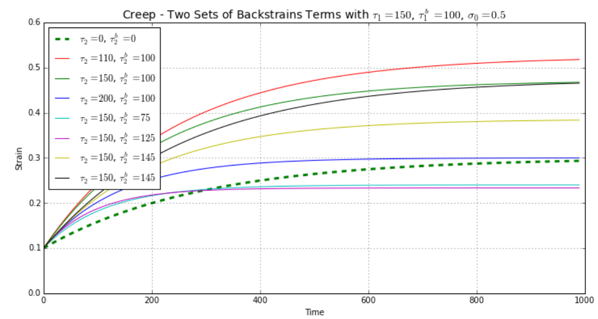
```
##########################
path_type = [7] # creep
t1 = 150
t1b = 100
t2 = np.array([0, 105, 110, 150, 200, 150, 150, 150])
t2b =np.array([0, 100., 100, 100, 100, 75, 125, 145])
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

time_inc = t1b / 10.0#
t_max = 1000.0
inc = t_max / time_inc

loop_cnt = len(t2b)

two_term_creep = np.zeros((inc, loop_cnt))#np.zeros((np.amax(inc), loop_cnt))
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1
    tau_b_1 = t1b
    t_inc = time_inc#[i]
    tau_2 = t2[i]
    tau_b_2 = t2b[i]

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    two_term_creep[0:inc,i] = SM[0:inc,23]#[0:inc[i],i] = SM[0:inc[i],1]
    time[0:inc,i] = SM[0:inc,48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   105.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   110.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
```

```
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   200.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.    75.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   125.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   145.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
100.0
```

In [42]:

```
# plot creep verification
fig_two_term_creep, ax = plt.subplots(figsize = (12,6))

ax.plot(time[0:inc,0], two_term_creep[0:inc,0], 'g--',lw = 3, markersize = 4, label=''r'$\tau_2=0$, $\tau^b_2 = 0$')
ax.plot(time[0:inc,1], two_term_creep[0:inc,1], 'r-',markersize = 4, label=''r'$\tau_2=110$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,2], two_term_creep[0:inc,2], 'g-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,3], two_term_creep[0:inc,3], 'b-',markersize = 4, label=''r'$\tau_2=200$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,4], two_term_creep[0:inc,4], 'c-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 75$')
ax.plot(time[0:inc,5], two_term_creep[0:inc,5], 'm-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 125$')
ax.plot(time[0:inc,6], two_term_creep[0:inc,6], 'y-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 145$')
ax.plot(time[0:inc,7], two_term_creep[0:inc,7], 'k-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 145$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Strain')
ax.set_title(''r'Creep - Two Sets of Backstrains Terms with $\tau_1 = 150$, $\tau^b_1 = 100$, $\sigma_0 = 0.5$', fontsize = 14
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_two_term_creep.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW
4/3_two_term_creep.pdf")

show()
```



Two term relaxation analysis

In [43]:

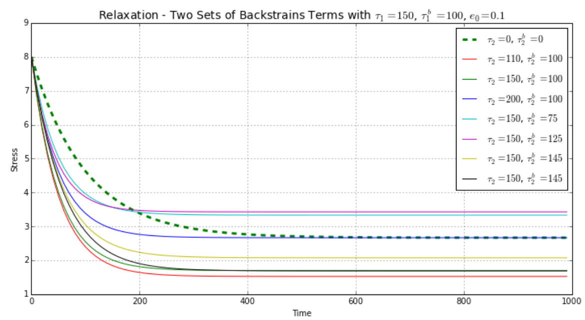
```python
##########################
path_type = [6] # relaxation
t1 = 150
t1b = 100
t2 = np.array([0, 105, 110, 150, 200, 150, 150, 150])
t2b =np.array([0, 100., 100, 100, 100, 75, 125, 145])
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

time_inc = t1b / 10.0#
t_max = 1000.0
inc = t_max / time_inc

loop_cnt = len(t2b)

two_term_relax = np.zeros((inc, loop_cnt))#np.zeros((np.amax(inc), loop_cnt))
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1
    tau_b_1 = t1b
    t_inc = time_inc#[i]
    tau_2 = t2[i]
    tau_b_2 = t2b[i]

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    two_term_relax[0:inc,i] = SM[0:inc,1]#[0:inc[i],i] = SM[0:inc[i],1]
    time[0:inc,i] = SM[0:inc,48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   105.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   110.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   200.   100.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.    75.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   125.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  1.01e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.   150.   145.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
```

```
Number of Calculations:  1.01e+02
100.0
```

In [44]:

```
# plot relax verification
fig_two_term_relax, ax = plt.subplots(figsize = (12,6))


ax.plot(time[0:inc,0], two_term_relax[0:inc,0], 'g--',lw = 3, markersize = 4, label=''r'$\tau_2=0$, $\tau^b_2 = 0$')
ax.plot(time[0:inc,1], two_term_relax[0:inc,1], 'r-',markersize = 4, label=''r'$\tau_2=110$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,2], two_term_relax[0:inc,2], 'g-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,3], two_term_relax[0:inc,3], 'b-',markersize = 4, label=''r'$\tau_2=200$, $\tau^b_2 = 100$')
ax.plot(time[0:inc,4], two_term_relax[0:inc,4], 'c-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 75$')
ax.plot(time[0:inc,5], two_term_relax[0:inc,5], 'm-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 125$')
ax.plot(time[0:inc,6], two_term_relax[0:inc,6], 'y-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 145$')
ax.plot(time[0:inc,7], two_term_relax[0:inc,7], 'k-',markersize = 4, label=''r'$\tau_2=150$, $\tau^b_2 = 145$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time')
ax.set_ylabel('Stress')
ax.set_title(''r'Relaxation - Two Sets of Backstrains Terms with $\tau_1 = 150$, $\tau^b_1 = 100$, $e_{0} = 0.1$', fontsize =
4)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_two_term_relax.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW
4/3_two_term_relax.pdf")

show()
```



4) Analysis of nonlinear term

Analysis of c star

In [35]:

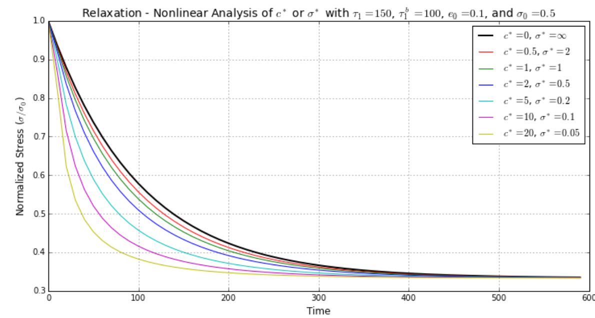
```
#########################
path_type = [6] # relaxation only
t1 = 150
t1b = 100
t2 = 0
t2b = 0
s_0 = 0.5  # sigma_0 is creep stress
c_s = [0.0, 0.5, 1, 2, 5, 10, 20]# c_star is for nonlinear viscoelasticity

time_inc = t1b / 10.0#
t_max = 600.0
inc = t_max / time_inc

loop_cnt = len(c_s)

nl_cstar_relax = np.zeros((inc, loop_cnt))# normalized vector
nl_cstar_relax_un = np.zeros((inc, loop_cnt))# un normalized vector
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1
    tau_b_1 = t1b
    t_inc = time_inc
    tau_2 = t2
    tau_b_2 = t2b
    c_star = c_s[i]
    sigma_0 = s_0

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    nl_cstar_relax_un[0:inc,i] = SM[0:inc,1]#not normalized
    nl_cstar_relax[0:inc,i] = nl_cstar_relax_un[0:inc,i] / SM[0, 1]# normalized
    time[0:inc,i] = SM[0:inc,48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    0.5    0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    1.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
```

```
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    2.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    5.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5   10.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5   20.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
60.0
```

In [37]:

```
# plot  verification
fig_nl_cstar_relax, ax = plt.subplots(figsize = (12,6))

ax.plot(time[0:inc,0], nl_cstar_relax[0:inc,0], 'k-',lw = 2, markersize = 4, label=''r'$c^*=0$, $\sigma^*=\infty$')
ax.plot(time[0:inc,1], nl_cstar_relax[0:inc,1], 'r-',markersize = 4, label=''r'$c^*=0.5$, $\sigma^*=2$')
ax.plot(time[0:inc,2], nl_cstar_relax[0:inc,2], 'g-',markersize = 4, label=''r'$c^*=1$, $\sigma^*=1$')
ax.plot(time[0:inc,3], nl_cstar_relax[0:inc,3], 'b-',markersize = 4, label=''r'$c^*=2$, $\sigma^*=0.5$')
ax.plot(time[0:inc,4], nl_cstar_relax[0:inc,4], 'c-',markersize = 4, label=''r'$c^*=5$, $\sigma^*=0.2$')
ax.plot(time[0:inc,5], nl_cstar_relax[0:inc,5], 'm-',markersize = 4, label=''r'$c^*=10$, $\sigma^*=0.1$')
ax.plot(time[0:inc,6], nl_cstar_relax[0:inc,6], 'y-',markersize = 4, label=''r'$c^*=20$, $\sigma^*=0.05$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time', fontsize = 12)
ax.set_ylabel('Normalized Stress 'r'($\sigma / \sigma_0$)', fontsize = 12)
ax.set_title('Relaxation - Nonlinear Analysis of 'r'$c^*$ or $\sigma^*$ with $\tau_1 = 150$, $\tau^b_1 = 100$, $e_0 = 0.1$, ar
$\sigma_0=0.5$',
             fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_nl_cstar_relax.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW
4/4_cstar_relax.pdf")

show()
```



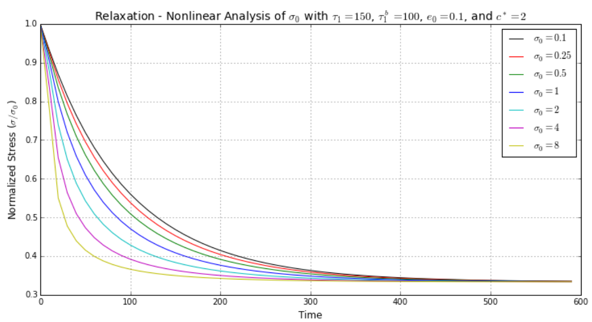
analysis of sigma 0

In [39]:
```
##########################
path_type = [6] # relaxation only
t1 = 150
t1b = 100
t2 = 0
t2b = 0
s_0 = [0.1, 0.25, 0.5, 1, 2, 4, 8]  # sigma_0 is creep stress
c_s = 2.0# c_star is for nonlinear viscoelasticity
E11 = 5.0

time_inc = t1b / 10.0#
t_max = 600.0
inc = t_max / time_inc

loop_cnt = len(s_0)

nl_s0_relax = np.zeros((inc, loop_cnt))# normalized vector
nl_s0_relax_un = np.zeros((inc, loop_cnt))# un normalized vector
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    tau_1 = t1
    tau_b_1 = t1b
    t_inc = time_inc
    tau_2 = t2
    tau_b_2 = t2b
    c_star = c_s
    sigma_0 = s_0[i]
    strn_inc_11 = sigma_0 / E11


    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max, sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max)

    nl_s0_relax_un[0:inc,i] = SM[0:inc,1]#not normalized
    nl_s0_relax[0:inc,i] = nl_s0_relax_un[0:inc,i] / SM[0, 1]# normalized
    time[0:inc,i] = SM[0:inc,48]
print inc
```

```
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.1    2.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.    100.      0.      0.      0.      0.      0.25    2.      0.
    0.  ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.   100.     0.     0.     0.     0.     0.5    2.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.  100.    0.    0.    0.    0.    1.    2.    0.    0.]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.  100.    0.    0.    0.    0.    2.    2.    0.    0.]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.  100.    0.    0.    0.    0.    4.    2.    0.    0.]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 150.  100.    0.    0.    0.    0.    8.    2.    0.    0.]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.10e+01
60.0
```

In [40]:

```
# plot  verification
fig_nl_s0_relax, ax = plt.subplots(figsize = (12,6))

ax.plot(time[0:inc,0], nl_s0_relax[0:inc,0], 'k-',markersize = 4, label=''r'$\sigma_0=0.1$')
ax.plot(time[0:inc,1], nl_s0_relax[0:inc,1], 'r-',markersize = 4, label=''r'$\sigma_0=0.25$')
ax.plot(time[0:inc,2], nl_s0_relax[0:inc,2], 'g-',markersize = 4, label=''r'$\sigma_0=0.5$')
ax.plot(time[0:inc,3], nl_s0_relax[0:inc,3], 'b-',markersize = 4, label=''r'$\sigma_0=1$')
ax.plot(time[0:inc,4], nl_s0_relax[0:inc,4], 'c-',markersize = 4, label=''r'$\sigma_0=2$')
ax.plot(time[0:inc,5], nl_s0_relax[0:inc,5], 'm-',markersize = 4, label=''r'$\sigma_0=4$')
ax.plot(time[0:inc,6], nl_s0_relax[0:inc,6], 'y-',markersize = 4, label=''r'$\sigma_0=8$')

ax.legend(loc=0); # upper left corner
ax.set_xlabel('Time', fontsize = 12)
ax.set_ylabel('Normalized Stress 'r'($\sigma / \sigma_0$)', fontsize = 12)
ax.set_title('Relaxation - Nonlinear Analysis of 'r'$\sigma_0$ with $\tau_1 = 150$, $\tau^b_1 = 100$, $e_0 = 0.1$, and $c^* =
$',
             fontsize = 14)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_nl_s0_relax.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/
_s0_relax.pdf")

show()
```

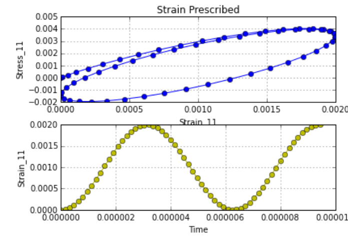


5) Cyclic Analysis

In [25]:

```
out_fig_fmt = "pdf"

#### Model Parameters ###
n_leg = 1 # number of legs / changes in strain increment
path_type = [0] # Loading path default is zero

# termination type
term_type = np.ones(n_leg) * [8]

#define termination values
n_max = 150000 #global termination on number of loops

# local terminations dependent on term_type
inc_max = 50000 # per leg
strs_max = 1
strs_min = -1.0
strn_max = 1
strn_min = -1
p_max = 0.5
p_min = -0.5

mat_type = 1

# Viscous Parameters
tau_1, tau_b_1 = 1e-6, 8e-7
tau_2, tau_b_2 = 0, 0
e_max = 0.002
e_dot_max = [1e-8,  250.0, 500.0, 1000.0] # cyclic loading parameters
sigma_0, c_star = 0.5, 0.0 # sigma_0 is creep stress, c_star is for nonlinear viscoelasticity

strn_func = 1 # zero if constant strain increment, 1 if cyclic loading

loop_cnt = len(e_dot_max)
cycle_strs = np.zeros((inc, loop_cnt))# normalized vector
time = np.zeros((np.amax(inc), loop_cnt))

for i in xrange(loop_cnt):
    t_max = np.pi * e_max / e_dot_max[i] *1.5 #10 #
    valprint("t_max", t_max)
    valprint("e_dot_max", e_dot_max[i])
    t_inc = tau_b_1 / (e_dot_max[i] / 200.0) # time increment for viscoelastic analysis

    SM, col_namev, irow = cs.Driver(run_title, n_leg, path_type, term_type,
       Y1, Y2, Y3, nu12, nu23, nu31, G44, G55, G66, tau_1, tau_b_1, tau_2, tau_b_2,
       e_max, e_dot_max[i], sigma_0, c_star,
       strn_11 = strn_inc_11, strn_22 = strn_inc_22, strn_33 = strn_inc_33, strn_12 = strn_inc_12,
       mat_type = mat_type, t_inc = t_inc, n_max = n_max,
       inc_max = inc_max, strs_max = strs_max, strs_min = strs_min, strn_max = strn_max, strn_min = strn_min,
       p_max = p_max, p_min = p_min, t_max = t_max, strn_func = strn_func)

    x1 = 23
    y1 = 1

    x2 = 48
    y2 = 23

    if i == 0:
        out_fig_name = '5_edot_1e-10'
    elif i == 1:
        out_fig_name = '5_edot_250'
    elif i == 2:
        out_fig_name = '5_edot_500'
    elif i == 3:
        out_fig_name = '5_edot_1000'

    # call plotting device
    cs.Plot_Setup(SM, col_namev, out_dir, out_name = out_fig_name, irow = irow,
                         sub_plot = 1,path = path_type[len(path_type) -1],
                         x1 = x1, x2 = x2, y1 = y1, y2 = y2, fmt = out_fig_fmt)
```

```
        t_max:  9.42e+05
      e_dot_max:  1.00e-08
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[ 0.   0.   0.   0.   0.   0.   0.5  0.   0.   0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.00e+01
        t_max:  3.77e-05
      e_dot_max:  2.50e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[   0.     0.     0.     0.     0.   250.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.00e+01
        t_max:  1.88e-05
      e_dot_max:  5.00e+02
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[   0.     0.     0.     0.     0.   500.     0.5    0.     0.     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.00e+01
        t_max:  9.42e-06
      e_dot_max:  1.00e+03
Viscoelastic Material
user defined elastic parm
[ 4.67  4.67  4.67  0.17  0.17  0.17  0.5   0.5   0.5 ]
user defined viscoelastic parm:
[    0.      0.      0.      0.      0.   1000.      0.5     0.      0.
     0. ]
default elastoplastic parm - current none
mat_type = 1, viscoelastic
Elasticity Matrix:
[[ 5.  1.  1.  0.  0. -0.]
 [ 1.  5.  1.  0.  0. -0.]
 [ 1.  1.  5.  0.  0. -0.]
 [ 0.  0.  0.  1.  0. -0.]
 [ 0.  0.  0.  0.  1. -0.]
 [ 0.  0.  0.  0.  0.  1.]]
Run Completed
 Number of Legs:  1.00e+00
Number of Calculations:  6.00e+01
```



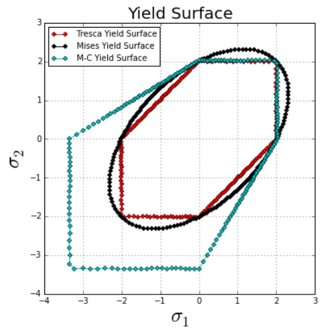
6) Yield Surfaces

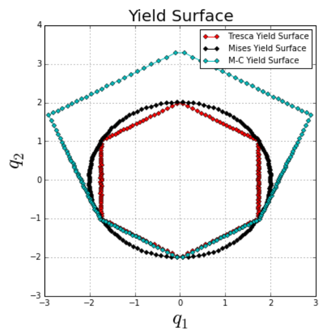
In [3]:
```
yield_mises = cs.Yield_Surface(0, 2, 0, 0)
```

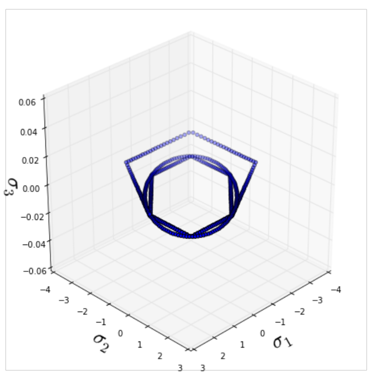
In [4]:
```
yield_tresca = cs.Yield_Surface(1, 2, 0, 0)
```

In [5]:
```
yield_mc = cs.Yield_Surface(2, 0, -15, 0.8)
```

In [6]:
```
fig_yield_1, ax = plt.subplots(figsize = (6,6))

ax.plot(yield_tresca[:,0], yield_tresca[:,1], 'rD-',markersize = 4, label=''r' Tresca Yield Surface')
ax.plot(yield_mises[:,0], yield_mises[:,1], 'kD-',markersize = 4, label=''r' Mises Yield Surface')
ax.plot(yield_mc[:,0], yield_mc[:,1], 'cD-',markersize = 4, label=''r' M-C Yield Surface')

ax.legend(loc=0, fontsize = 10); # upper left corner
ax.set_xlabel(''r'$\sigma_{1}$', fontsize = 25)
ax.set_ylabel(''r'$\sigma_{2}$', fontsize = 25)
ax.set_title('Yield Surface', fontsize = 20)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_yield_1.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/6_yi
ld_surf_sig.pdf")


fig_yield_2, ax = plt.subplots(figsize = (6,6))

ax.plot(yield_tresca[:,3], yield_tresca[:,4], 'rD-',markersize = 4, label=''r' Tresca Yield Surface')
ax.plot(yield_mises[:,3], yield_mises[:,4], 'kD-',markersize = 4, label=''r' Mises Yield Surface')
ax.plot(yield_mc[:,3], yield_mc[:,4], 'cD-',markersize = 4, label=''r' M-C Yield Surface')

ax.legend(loc=0, fontsize = 10); # upper left corner
ax.set_xlabel(''r'$q_1$', fontsize = 25)
ax.set_ylabel(''r'$q_2$', fontsize = 25)
ax.set_title('Yield Surface', fontsize = 20)
ax.grid(b = True, which = 'minor')
ax.grid(b = True, which = 'major')
fig_yield_2.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/6_yi
ld_surf_q.pdf")

from mpl_toolkits.mplot3d import Axes3D
fig_3d = plt.figure(figsize = (8,8))
ax = fig_3d.add_subplot(111, projection = '3d')

ax.scatter(yield_tresca[:,0],yield_tresca[:,1], yield_tresca[:,2])
ax.scatter(yield_mises[:,0],yield_mises[:,1], yield_mises[:,2])
ax.scatter(yield_mc[:,0],yield_mc[:,1], yield_mc[:,2])

ax.set_xlabel(''r'$\sigma_1$', fontsize = 25)
ax.set_ylabel(''r'$\sigma_2$', fontsize = 25)
ax.set_zlabel(''r'$\sigma_3$', fontsize = 25)
ax.view_init(elev = 30, azim = 45)
fig_3d.savefig("/Users/Lampe/Documents/UNM_Courses/ME-562_CONSTITUTIVEMODELINGANDASSOCIATEDALOGORITHMS_SCHREYER/HW04/6_yield_s
rf_3d.pdf")
```







Yield Function

In [45]:

```
def Yield_Func(P1, P2, P3, q, func, sigma_test, theta = 0, c = 0):
    """Function used to calculate the user defined yield criterion
    func = integer used to define which yield function to use

    P1, P2, P3 = principal values
    q = mises stress
    sigma_test = yield strength of material in uniaxial strain
    theta = internal angle of friction (degrees)
    c = cohesion
    """

    if func == 0: # mises yield criterion
        sigma_cr = sigma_test
        F = q / sigma_cr - 1
    if func == 1: # rankine yield criterion
        sigma_cr = sigma_test/2.0
        F_12 = ((P1 - P2)/2) / sigma_cr - 1
        F_13 = ((P1 - P3)/2) / sigma_cr - 1
        F_21 = ((P2 - P1)/2) / sigma_cr - 1
        F_23 = ((P2 - P3)/2) / sigma_cr - 1
        F_31 = ((P3 - P1)/2) / sigma_cr - 1
        F_32 = ((P3 - P2)/2) / sigma_cr - 1
        F_list = [F_12, F_13, F_21, F_23, F_31, F_32]
        F = max(F_list)
    if func == 2: # Mohr-Coulomb failure criteria
        theta_cr = -theta * (np.pi / 180.0) # convert to radians
        mu = 1 / np.tan(2 * theta)

        F_12 = ((P1 - P2) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P1) #/ sigma_cr - 1
        F_13 = ((P1 - P3) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P1) #/ sigma_cr - 1
        F_21 = ((P2 - P1) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P2) #/ sigma_cr - 1
        F_23 = ((P2 - P3) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P2) #/ sigma_cr - 1
        F_31 = ((P3 - P1) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P3) #/ sigma_cr - 1
        F_32 = ((P3 - P2) * (np.sin(theta_cr)*np.cos(theta_cr) - mu * np.sin(theta_cr)**2) - c + mu * P3) #/ sigma_cr - 1
        F_list = [F_12, F_13, F_21, F_23, F_31, F_32]
        F = max(F_list)

#         valprint("theta cr", theta_cr)
#         valprint("mu", mu)
#         matprint("F func", F_list)

    return F
```

Yield Surface Function

In [ ]:

```
def Yield_Surface(model, yield_stress, theta_cr = 0, cohesion = 0):
    """ definition of yield functions
        model: defines what model with be used for the yield function
            0 = mises
            1 = tresca
            2 = mohr-coulomb
    """

    theta_degv = np.arange(0, 360, 1) # angle in degrees
    r_inc = 0.02 # increment for radius of failure surface
    # yield stress from uniaxial tension test

    theta_radv = np.zeros(len(theta_degv))
    yield_surfacev = np.zeros((len(theta_degv),18))

    sig_1v, sig_2v = np.zeros(len(theta_degv)), np.zeros(len(theta_degv))
    measv = np.zeros((1,50))
    inc = 0
    F= -1.0
    loop_max = 100000

    for i in range(len(theta_degv)):
        loop_count = 0
        theta_radv[i] = theta_degv[i] * np.pi / 180.0
        r = 0.075# initial radius
        while F < 0.0:
            P1 = r * np.cos(theta_radv[i])#maximum principal stress
            P2 = r * np.sin(theta_radv[i])#Int principal stress
            P3 = 0 #min principal stress

            P_vect = [P1, P2, P3]
            P_vect.sort()
            P_vect.reverse()
            sigma = np.diag(P_vect)

            sigma_sp = 1.0/3.0 * np.trace(sigma) * np.eye(3) #spherical (isotropic) stress matrix
            sigma_dv = sigma - sigma_sp #deviatoric stress matrix
```

```
        #calculate stress invariants
        I1 = np.trace(sigma)
        J2 = 1.0 / 2.0 * np.trace(np.dot(sigma_dv, sigma_dv))
        J3 = 1.0 / 3.0 * np.trace(np.dot(sigma_dv, np.dot(sigma_dv, sigma_dv)))

        #calculate other stress measures
        p = 1.0 / 3.0 * I1 #tensile pressure or mean stress

        sigma_dev_1 = P1 - p #maximum deviatoric stress
        sigma_dev_2 = P2 - p #intermediate stress stress
        sigma_dev_3 = P3 - p #minimum deviatoric stress

        q1 = np.sqrt(3.0) / 2 * (P2 - P1)
        q2 = -3.0 / 2.0 * (sigma_dev_1 + sigma_dev_2)

        # check that terms are equivalent
        mises_strs = np.sqrt(3.0 * J2)
        q = np.sqrt(q1**2 + q2**2)

        #Lode angle calcs
        lode_r = np.sqrt(2.0 * J2)
        lode_z = I1 / np.sqrt(3.0) #coordinate is parallel to hydrostatic axis

        c_2 = 3.0*np.sqrt(6.0)*LA.det(sigma_dv/lode_r)
        if c_2 > 1.0:
#             print "c_2 greater than 1.0"
            c_2 = 1.0
        if c_2 < -1.0:
#             print "c_2 less than -1.0"
            c_2 = -1.0
        lode_theta = 1.0/3.0*np.arcsin(c_2)

        #stress triaxiality
        triax = p / mises_strs
        ################################

        F = Yield_Func(P1 = P1, P2 = P2, P3 = P3, q = q, func = model, sigma_test = yield_stress,
                       theta =  theta_cr, c = cohesion)

        r = r + r_inc
        loop_count = loop_count + 1
#         valprint("radius", r)

        if loop_count > loop_max:
            valprint("F",F)
            sys.exit("too man loops")

    yield_surfacev[inc, 0] = P1 # sigma_11
    yield_surfacev[inc, 1] = P2 # sigma_22
    yield_surfacev[inc, 2] = P3 # sigma_33
    yield_surfacev[inc, 3] = q1 # in deviatoric plane
    yield_surfacev[inc, 4] = q2 # in deviatoric plane
    yield_surfacev[inc, 5] = p # mean pressure
    yield_surfacev[inc, 6] = q # mises stress
    yield_surfacev[inc, 7] = lode_r # radius from origin in deviatoric plane
    yield_surfacev[inc, 8] = lode_z
    yield_surfacev[inc, 9] = lode_theta
    yield_surfacev[inc, 10] = triax
    yield_surfacev[inc, 11] = F

    inc = inc + 1
    F = -1.0
    r = 0

return yield_surfacev
```