stability_over_parameter_space

November 13, 2025

[1]:
```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np
```

Suppose we have the second order ordinary differential equation (ODE)

$$\frac{d^2x}{dt^2} - (\alpha - \beta)x = 0$$

where $\alpha, \beta \in \{x \in \mathbb{R} | x \geq 0\}$ are the parameters controlling the ODE. Note that the characteristic equation is

$$r^2 = \alpha - \beta$$

where the growth rates are given by

$$\sigma(\alpha, \beta) = \sqrt{\alpha - \beta}.$$

The solutions grow exponentially if $\alpha > \beta$, are stationary if $\alpha = \beta$, and oscillate if $\alpha < \beta$. While this is nice mathematically, if α and β have relevance to a problem of scientific interest studying the growth rate in this format is not very helpful. Introducing the parameter space allows us to better understand how the growth rates relate to the parameters. The parameter space (in this context) plots the growth rates or frequency over a space defined by the parameters.

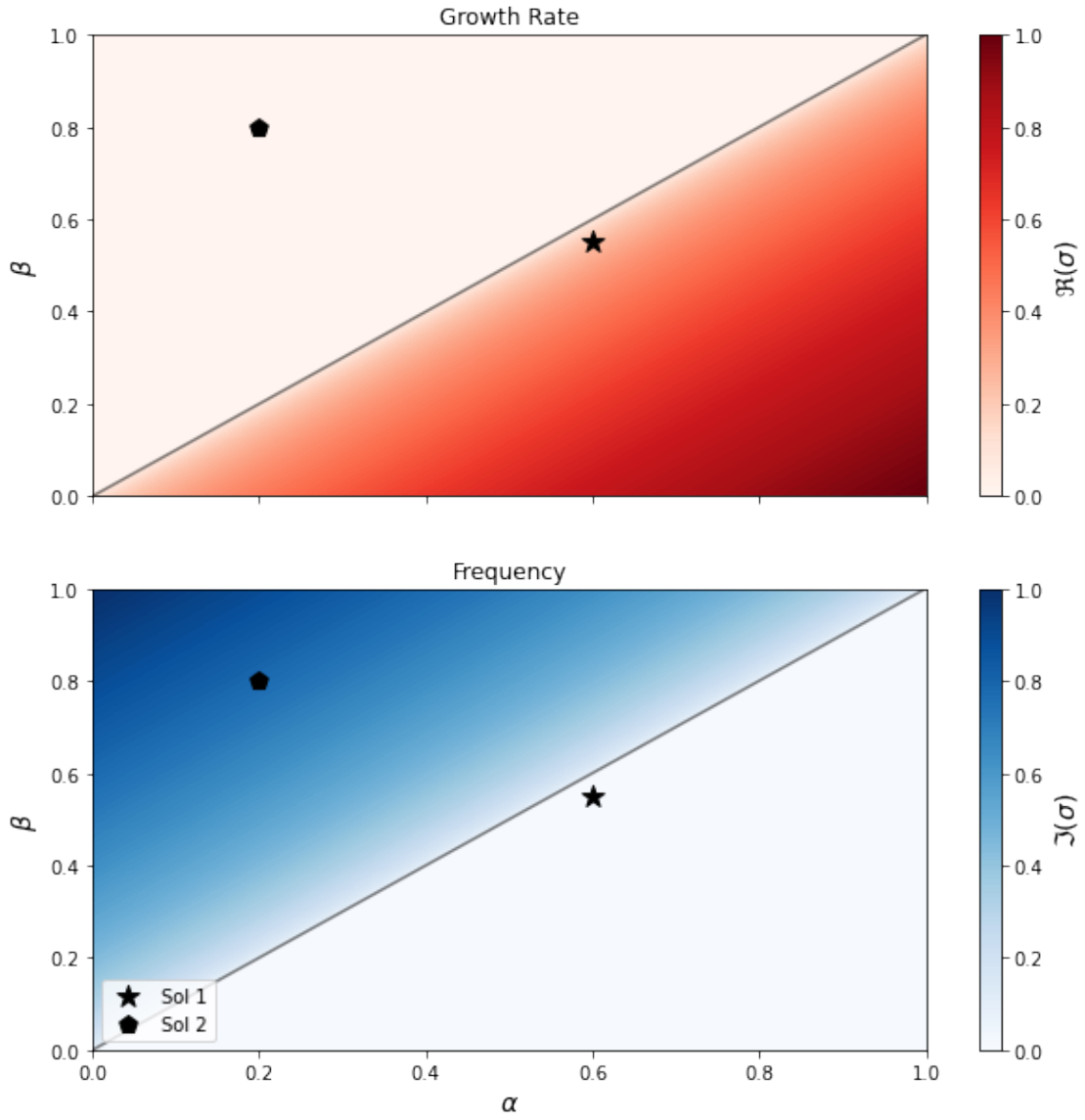
We will plot the parameter space after writing a function using python to plot it.

[2]:
```
def stability_gr(a,b):
    return np.sqrt(a-b+0*1j) # growth rate function
```

[3]:
```
dx = 500 # define resolution of parameter space

upper_limit_stbly = 1 # upper limit of parameter space

# define alpha and beta space
alpha_vls = np.array([np.arange(0,upper_limit_stbly+1/dx,upper_limit_stbly/
 ↪dx)]).T@np.ones((1,dx+1))
beta_vls = np.array([np.arange(0,upper_limit_stbly+1/dx,upper_limit_stbly/dx)]).
 ↪T@np.ones((1,dx+1))
```

```
# calculate the growth rates
growth_rates = stability_gr(alpha_vls, beta_vls.T)
```

[4]:
```
fig, axs = plt.subplots(2,figsize=(10,10),sharex=True)

# plot the growth rates over parameter space
cf_real = axs[0].contourf(alpha_vls,beta_vls.T,growth_rates.
 ↪real,cmap="Reds",levels=100)
axs[0].plot(np.arange(0,upper_limit_stbly+0.1,0.1),np.
 ↪arange(0,upper_limit_stbly+0.1,0.1),c="k",alpha=0.5)
axs[0].set_ylabel(r"$\beta$",fontsize=14)
axs[0].set_title("Growth Rate")

axs[0].scatter(0.6,0.55,label="Sol 1",c="k",marker="*",s=150)
axs[0].scatter(0.2,0.8,label="Sol 2",c="k",marker="p",s=100)

cbar_real = fig.colorbar(cf_real,ax=axs[0],ticks=np.arange(0,1.1,0.2))
cbar_real.set_label(r"$\Re(\sigma)$",fontsize=14)

# plot the frequencies over the parameter space
cf_imag = axs[1].contourf(alpha_vls,beta_vls.T,growth_rates.
 ↪imag,cmap="Blues",levels=100)
axs[1].plot(np.arange(0,upper_limit_stbly+0.1,0.1),np.
 ↪arange(0,upper_limit_stbly+0.1,0.1),c="k",alpha=0.5)
axs[1].set_xlabel(r"$\alpha$",fontsize=14)
axs[1].set_ylabel(r"$\beta$",fontsize=14)
axs[1].set_title("Frequency")

axs[1].scatter(0.6,0.55,label="Sol 1",c="k",marker="*",s=150)
axs[1].scatter(0.2,0.8,label="Sol 2",c="k",marker="p",s=100)

cbar_imag = fig.colorbar(cf_imag,ax=axs[1],ticks=np.arange(0,1.1,0.2))
cbar_imag.set_label(r"$\Im(\sigma)$",fontsize=14)

axs[1].legend(loc="lower left")

plt.show()
```



Above is the parameter space for the ODE we are considering in this problem. We see above the line $\alpha = \beta$ the solutions are purely oscillatory, and below the line the solutions grow exponentially. Two solutions for initial conditions $x(t = 0) = 1$ and $dx(t = 0)/dt = 0$ are chosen so that one solution oscillates, and the other grows exponentially (albiet the parameters chosen are for relatively slow growth to faciltate comparison).

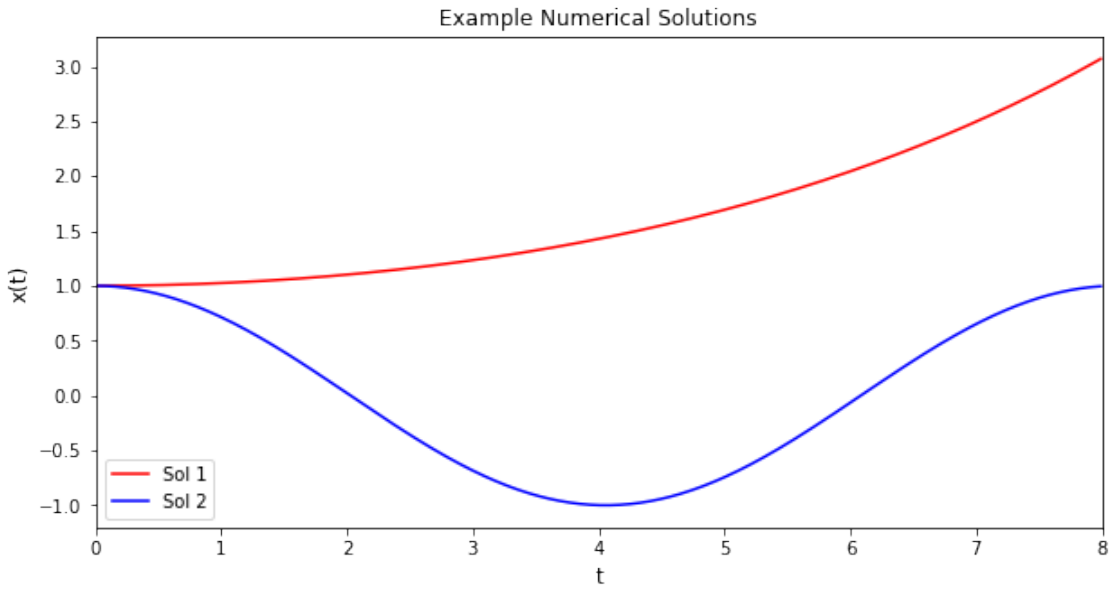
The solutions for Sol 1 which has parameters $(\alpha, \beta) = (0.6, 0.55)$ and Sol 2 which has parameters $(\alpha, \beta) = (0.2, 0.8)$ are below.

[5]:
```
def ode_one(v, t, a, b):  # differential equation
    coeff = a-b  # coefficient
    x, y = v
```

```
    return [y, coeff*x]  # note this is defined as a system of two ODEs for␣
↪numerical purposes
```

[6]:
```
t = np.arange(0,8,0.01)  # time domain
v0 = [0,1]  # initial condition
sol1 = odeint(ode_one, v0, t, args=(0.6,0.55))  # numerical solver for sol 1
sol2 = odeint(ode_one, v0, t, args=(0.2,0.8))  # numerical solver for sol 2
```

[7]:
```
fig = plt.figure(figsize=(10,5))

# plot solutions over time
plt.plot(t,sol1[:,1],c="r",label="Sol 1")
plt.plot(t,sol2[:,1],c="b",label="Sol 2")

# label the axes, title, domain limits, etc.
plt.xlabel("t",fontsize=12)
plt.ylabel("x(t)",fontsize=12)

plt.title("Example Numerical Solutions")

plt.xlim([0,8])

plt.legend(loc="lower left")

plt.show()
```



To check that the exact and numerical solution are consistent we will plot a numerical solution

with the exact solution for Sol1. Note that in this case the exact solution is given by:

$$x(t) = 0.5\exp(\sqrt{\alpha - \beta}t) + 0.5\exp(-\sqrt{\alpha - \beta}t)$$

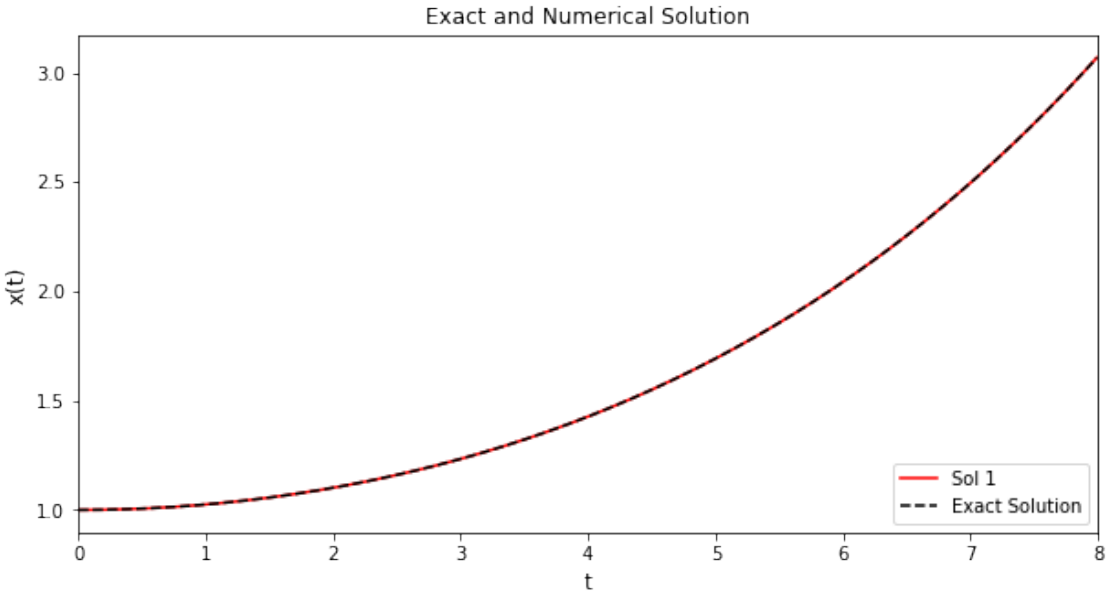
and we can calculate the growth rate to be

[8]:
```
grwth_rt = stability_gr(0.6,0.55).real
print("The growth rate for Sol 1 is "+str(round(grwth_rt,3)))
```

```
The growth rate for Sol 1 is 0.224
```

The plots of exact vs numerical solution below show that the numerical and exact solutions are consistent:

[9]:
```
exact_sol = 0.5*np.exp(grwth_rt*t)+0.5*np.exp(-grwth_rt*t) # calculate exact␣
↪solution

fig = plt.figure(figsize=(10,5))

# plot exact and numerical solution
plt.plot(t,sol1[:,1],c="r",label="Sol 1")
plt.plot(t,exact_sol,c="k",linestyle="--",label="Exact Solution")

# labels, titled, domain restriction, etc
plt.xlabel("t",fontsize=12)
plt.ylabel("x(t)",fontsize=12)

plt.xlim([0,8])

plt.title("Exact and Numerical Solution")

plt.legend(loc="lower right")

plt.show()
```



For a more interesting parameter space, we will look at the ODEn for the dampled harmonic oscillator

$$\frac{d^2x}{dt^2} + \gamma\frac{dx}{dt} + \delta x = 0$$

where $\gamma, \delta \in \mathbb{R}$, and the solution of the characteristic equation is given by

$$\sigma = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\delta}}{2}.$$

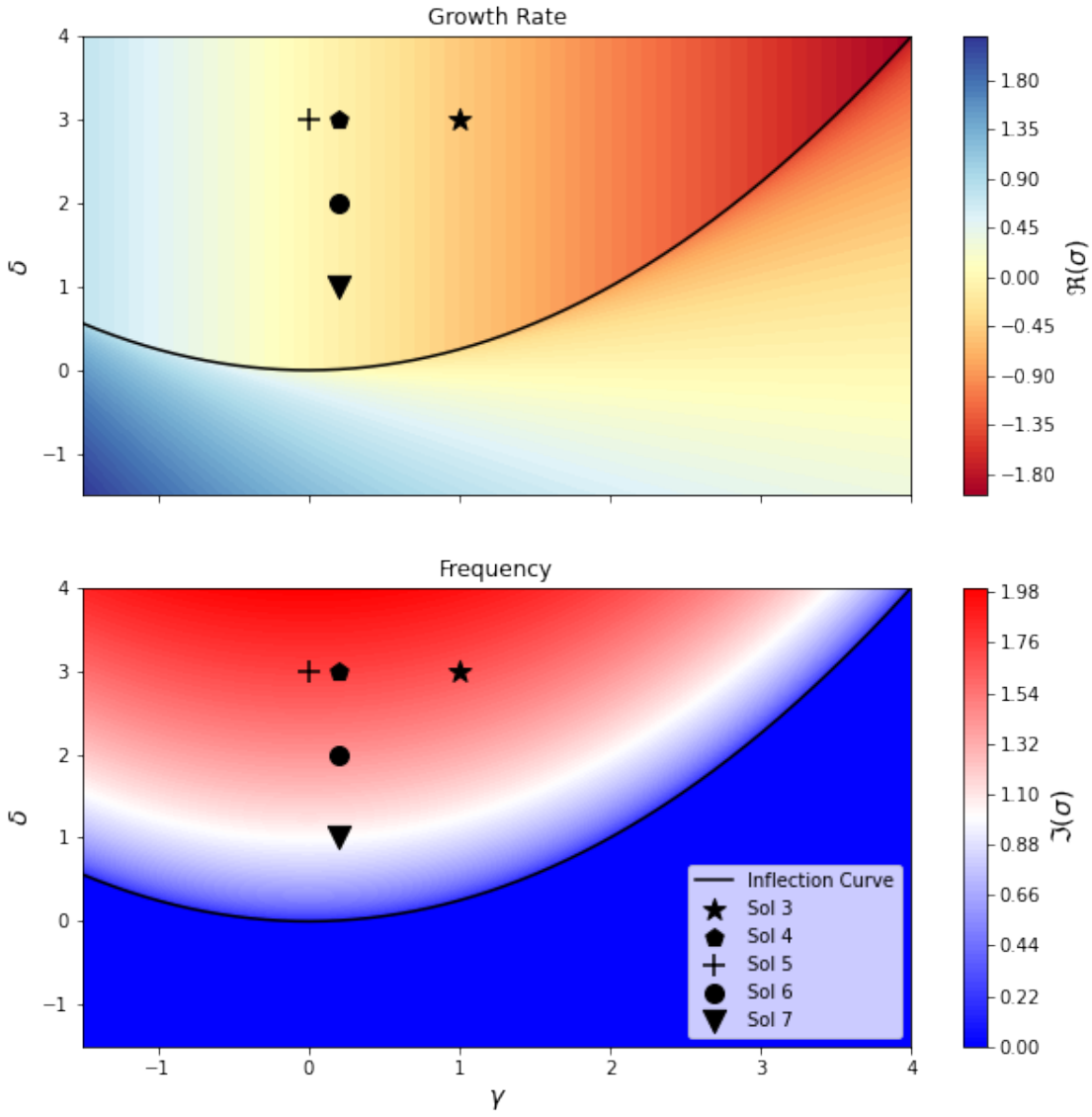
We calculate the growth rate by taking the real component of it and the frequency of solutions by taking the abosulte value of the imaginary component.

[10]:
```
def dmpd_hrmc_osc_gr_rts(g,d): # characteristic soltuions i.e. growth rate␣
↪values
    frst = -g
    scnd = np.sqrt(g**2-4*d+0*1j)

    return 0.5*(frst+scnd)
```

[11]:
```
dx = 1000 # define parameter space
upper_limit = 4
lower_limit = -1.5
domain = np.arange(lower_limit, upper_limit+1/dx, (upper_limit-lower_limit)/dx)
gamma_vls = np.array([domain]).T@np.ones((1, dx+1))
delta_vls = np.array([domain]).T@np.ones((1, dx+1))
```

```
hrmc_osc_gr_rts = dmpd_hrmc_osc_gr_rts(gamma_vls, delta_vls.T) # calculate␣
↪growth rates
```

[12]:
```
fig, axs = plt.subplots(2,figsize=(10,10),sharex=True)

# plot the growth rate over parameter space
cf_real = axs[0].contourf(gamma_vls,delta_vls.T,hrmc_osc_gr_rts.
↪real,cmap="RdYlBu",levels=100)
axs[0].set_ylabel(r"$\delta$",fontsize=14)
axs[0].set_title("Growth Rate")

# plot inflection curve
axs[0].plot(domain,domain**2/4,c="k",label="Inflection Curve")

# plot solution points in parameter space
axs[0].scatter(1,3,label="Sol 3",c="k",marker="*",s=150)
axs[0].scatter(0.2,3,label="Sol 4",c="k",marker="p",s=100)
axs[0].scatter(0,3,label="Sol 5",c="k",marker="+",s=150)
axs[0].scatter(0.2,2,label="Sol 6",c="k",marker="o",s=100)
axs[0].scatter(0.2,1,label="Sol 7",c="k",marker="v",s=150)

cbar_real = fig.colorbar(cf_real,ax=axs[0])
cbar_real.set_label(r"$\Re(\sigma)$",fontsize=14)

# plot the frequency over parameter space
cf_imag = axs[1].contourf(gamma_vls,delta_vls.T,np.abs(hrmc_osc_gr_rts.
↪imag),cmap="bwr",levels=100)
axs[1].set_xlabel(r"$\gamma$",fontsize=14)
axs[1].set_ylabel(r"$\delta$",fontsize=14)
axs[1].set_title("Frequency")

# plot solution points in parameter space
axs[1].scatter(1,3,label="Sol 3",c="k",marker="*",s=150)
axs[1].scatter(0.2,3,label="Sol 4",c="k",marker="p",s=100)
axs[1].scatter(0,3,label="Sol 5",c="k",marker="+",s=150)
axs[1].scatter(0.2,2,label="Sol 6",c="k",marker="o",s=100)
axs[1].scatter(0.2,1,label="Sol 7",c="k",marker="v",s=150)

# plot inflection curve
axs[1].plot(domain,domain**2/4,c="k",label="Inflection Curve")

axs[1].legend(loc="lower right")

cbar_imag = fig.colorbar(cf_imag,ax=axs[1])
cbar_imag.set_label(r"$\Im(\sigma)$",fontsize=14)

plt.show()
```



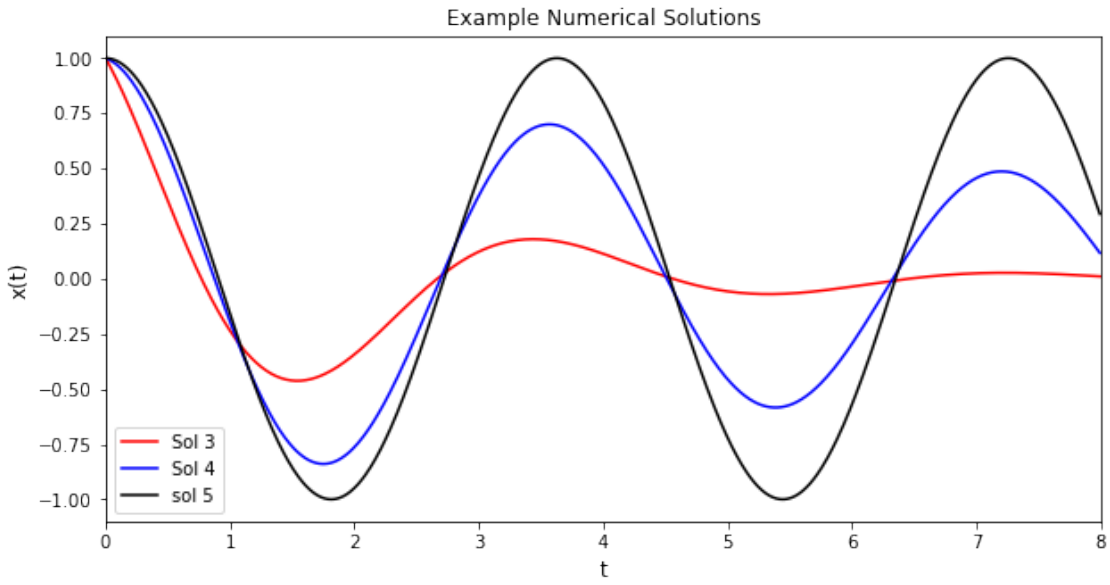
The parameter space is plotted above, note that the inflection curve, the point where the growth rates have a singularity and the solutions goes from oscillating to not oscilating, is a parabola. Further, the solutions for the damped harmonic oscillator can be oscillating and exponentially decaying. Below we have solution plotted for different growth rates and similar frequencies and same growth rate but different frequencies.

[13]:
```
def dmpd_hrmc_osc_eqn(v, t, g, d):
    u1, u2 = v
    return [u2, -d*u1-g*u2]
```

[14]:
```
sol3 = odeint(dmpd_hrmc_osc_eqn, v0, t, args=(1,3))
sol4 = odeint(dmpd_hrmc_osc_eqn, v0, t, args=(0.2,3))
```

```
sol5 = odeint(dmpd_hrmc_osc_eqn, v0, t, args=(0,3))
```

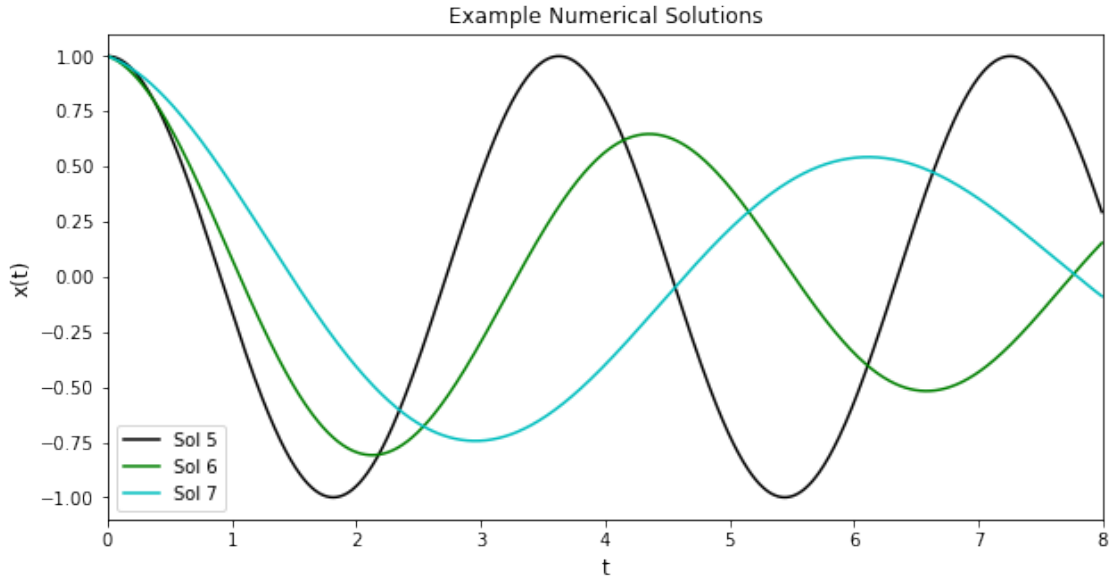
[15]:
```
fig = plt.figure(figsize=(10,5))

plt.plot(t,sol3[:,1],c="r",label="Sol 3")
plt.plot(t,sol4[:,1],c="b",label="Sol 4")
plt.plot(t,sol5[:,1],c="k",label="sol 5")

plt.xlabel("t",fontsize=12)
plt.ylabel("x(t)",fontsize=12)

plt.title("Example Numerical Solutions")

plt.xlim([0,8])

plt.legend(loc="lower left")

plt.show()
```



[16]:
```
sol6 = odeint(dmpd_hrmc_osc_eqn, v0, t, args=(0.2,2))
sol7 = odeint(dmpd_hrmc_osc_eqn, v0, t, args=(0.2,1))
```

[17]:
```
fig = plt.figure(figsize=(10,5))

plt.plot(t,sol5[:,1],c="k",label="Sol 5")
plt.plot(t,sol6[:,1],c="g",label="Sol 6")
plt.plot(t,sol7[:,1],c="c",label="Sol 7")
```

```
plt.xlabel("t",fontsize=12)
plt.ylabel("x(t)",fontsize=12)

plt.title("Example Numerical Solutions")

plt.xlim([0,8])

plt.legend(loc="lower left")

plt.show()
```



Source:

"15.6: Damped Oscillations." Physics LibreTexts, Libretexts, 1 Oct. 2024, phys.libretexts.org/Bookshelves/University_Physics/University_Physics_(OpenStax)/Book%3A_University_Ph *Mechanics_Sound_Oscillations_and_Waves*(OpenStax)/15%3A_Oscillations/15.06%3A_Damped_Oscillations.

[ ]: