روش اویلر برای حل عددی معادلات دیفرانسیل جفت شده

مثال:

معادله دیفرانسیل توصیف کننده حرکت سیستم جرم و فنر $mx" + cx' + kx = 0$ است که می‌توان آن را به صورت $x"(t) + 2\xi\omega x'(t) + \omega^2 x(t) = 0$ بازنویسی کرد. به کمک روش اویلر این معادله را در حالت های مختلف حل میکنیم.

$$x"(t) = -2\xi\omega x'(t) - \omega^2 x(t)$$

ابتدا این معادله دیفرانسیل مرتبه دو را به صورت دو معادله دیفرانسیل جفت شده مرتبه یک مینویسیم:

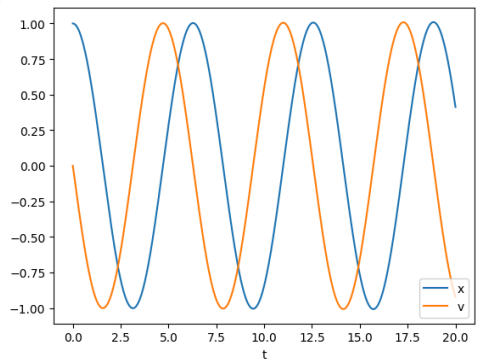
$$x'(t) = v(t)$$

$$v'(t) = -2\xi\omega v(t) - \omega^2 x(t)$$

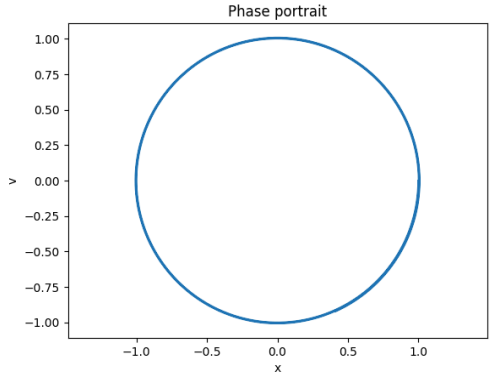
حالت اول: هیچ اصطکاکی وجود ندارد پس $\xi = 0$:

```
def xdot(v):
  return v

def vdot(x,v):
  omega=1.0
  xi = 0.0
  return (- (2.0 * xi * omega * v ) - (omega*omega*x))

start_time = 0.0
dt = .001
stop_time =20.0

t=np.arange(start_time,stop_time,dt)
x=np.zeros_like(t)
v=np.zeros_like(t)

x[0] = 1.0
v[0] = 0.0

for i in range(0,len(t)-1):
  x[i+1] = x[i] + dt * xdot(v[i])
  v[i+1] = v[i] + dt * vdot(x[i],v[i])
```

```
plt.plot(t,x,label="x")
plt.plot(t,v,label="v")
plt.xlabel("t")
# plt.ylabel("x")
plt.legend()
```

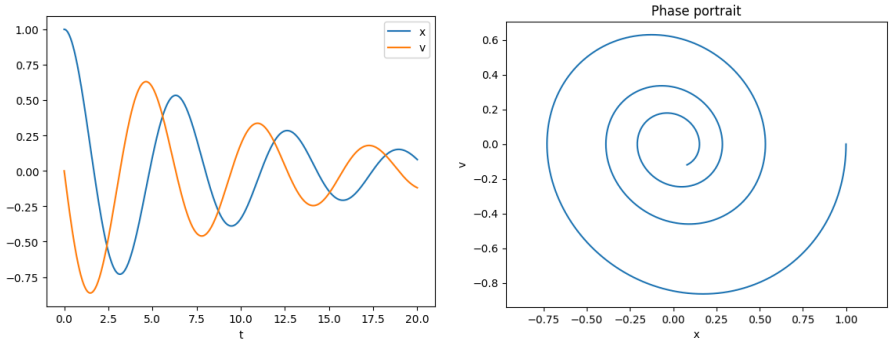



```
plt.plot(x,v)
plt.xlabel("x")
plt.ylabel("v")
plt.axis('equal')
plt.title("Phase portrait")
```



حالت اول: هیچ اصطکاکی وجود دارد پس $\xi = 0.1$:

```
def xdot(v):
  return v

def vdot(x,v):
  Omega = 1.0
  xi = 0.10
  return (- (2.0 * xi * omega * v ) - (omega*omega*x))

start_time = 0.0
dt = .001
stop_time =20.0

t=np.arange(start_time,stop_time,dt)
x=np.zeros_like(t)
v=np.zeros_like(t)

x[0] = 1.0
v[0] = 0.0

for i in range(0,len(t)-1):
  x[i+1] = x[i] + dt * xdot(v[i])
  v[i+1] = v[i] + dt * vdot(x[i],v[i])
```



اگر بخواهیم اثر نویز را هم داشته باشیم

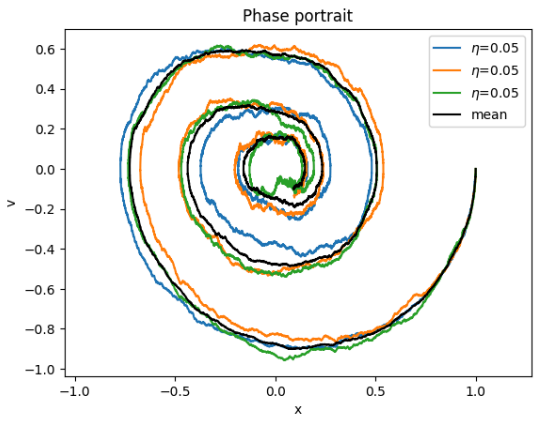
$$x''(t) = \; -2\xi\omega x'(t) - \omega^2 x(t) + \eta\, Noise(t)$$

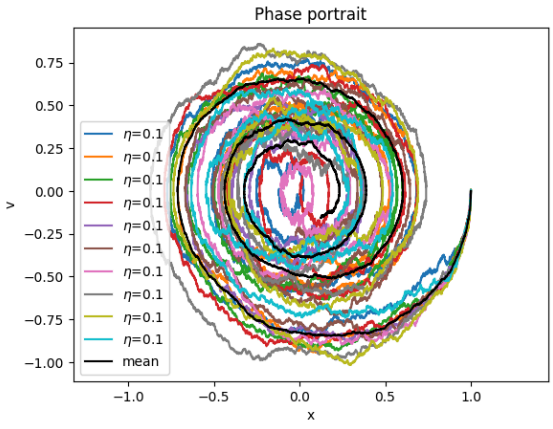
$$x'(t) = v(t)$$

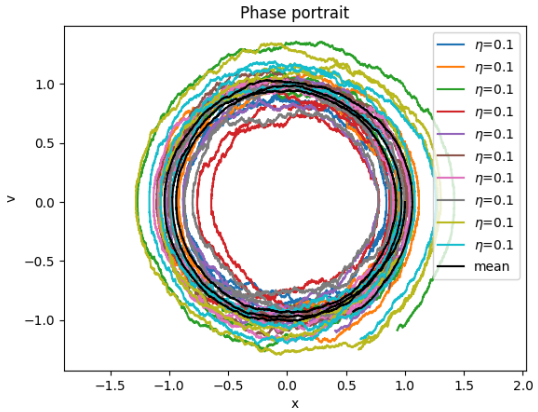
$$v'(t) = \; -2\xi\omega v(t) - \omega^2 x(t) + \eta\, Noise(t)$$

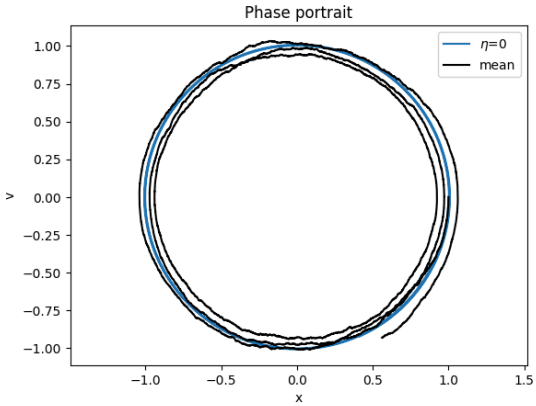
```
def HO(noise_power,xi):
  def xdot(v):
    return v

  def vdot(x,v,xi=0.1):
    omega=1.0
    # print(xi)
    return (- (2.0 * xi * omega * v ) - (omega*omega*x) )
  def noise(eta=0.0):
    eta = noise_power
    return eta*np.random.normal(loc=0.0,scale=1.0)

  start_time = 0.0
  dt = .001
  stop_time =20.0

  t=np.arange(start_time,stop_time,dt)
  x=np.zeros_like(t)
  v=np.zeros_like(t)

  x[0] = 1.0
  v[0] = 0.0

  for i in range(0,len(t)-1):
    x[i+1] = x[i] + dt * xdot(v[i])
    v[i+1] = v[i] + dt * vdot(x[i],v[i],xi) + np.sqrt(dt)*noise()
  return x,v

vels=[]
pos=[]
noise_pow=[0.1 for i in range(10)]
for iNoise in noise_pow:
  x,v=HO(iNoise,xi=0.0)
  vels.append(v)
```

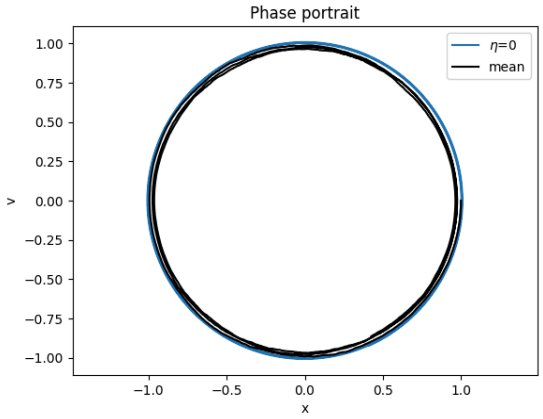
```
  pos.append(x)
  plt.plot(x,v,label=f"$\eta$={iNoise}")
mean_x=sum(pos)/float(len(noise_pow))
mean_v=sum(vels)/float(len(noise_pow))
plt.plot(mean_x,mean_v,'-k',label=f"mean")
plt.xlabel("x")
plt.ylabel("v")
plt.axis('equal')
plt.legend()
plt.title("Phase portrait")

plt.figure()
x,v=HO(0,0)
plt.plot(x,v,label=f"$\eta$={0}")
plt.plot(mean_x,mean_v,'-k',label=f"mean")
plt.xlabel("x")
plt.ylabel("v")
plt.axis('equal')
plt.legend()
plt.title("Phase portrait")
```

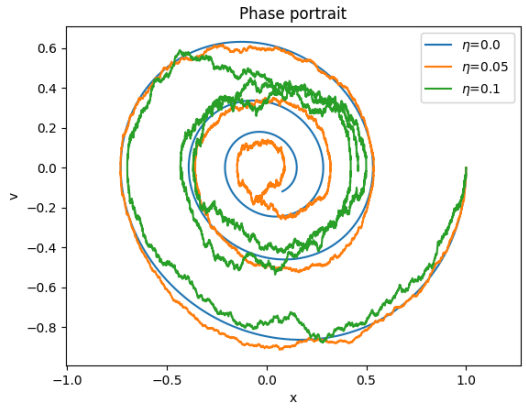





Phase portrait
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
η=0.1
mean
x
v


Phase portrait
η=0
mean
x
v


Phase portrait
η=0
mean
x
v


Phase portrait
η=0.0
η=0.05
η=0.1
x
v