Predictor-Corrector Method

When step size, h=0.2

In [1]:

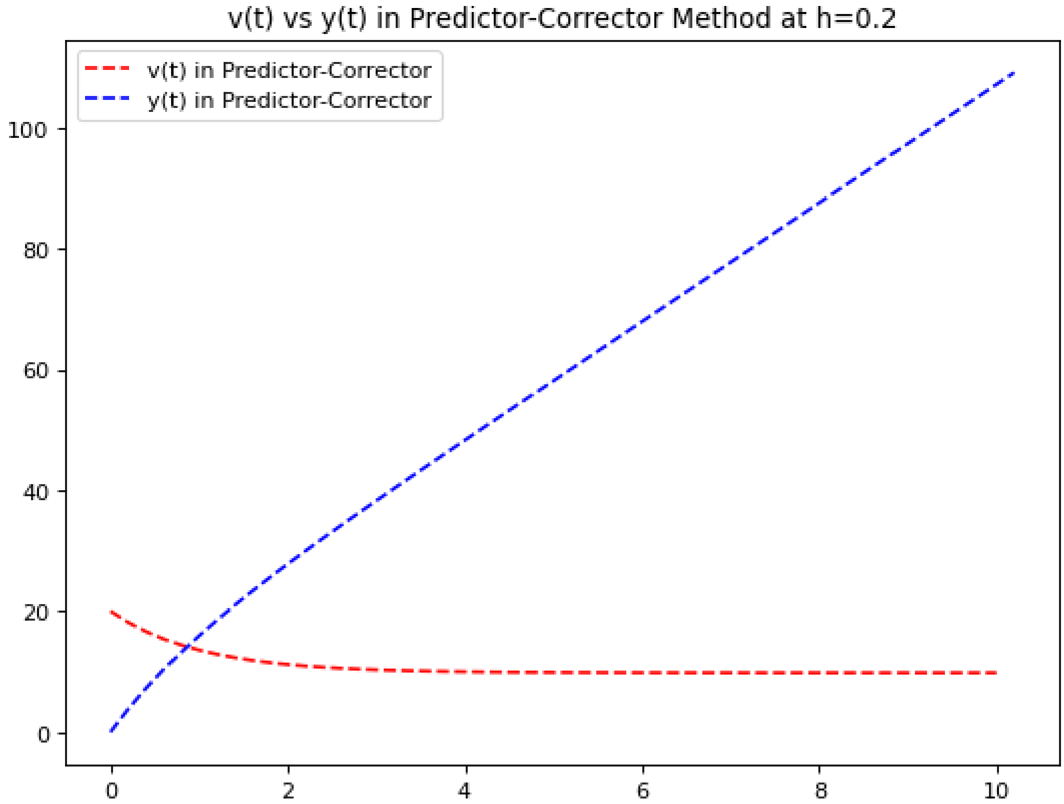
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
```

In [2]:

```
figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')
# Solving for the Predictor-Corrector method at step-size h=0.2

g = 9.81
m = 1
c = 1
def f(t,v):
     return g-(c/m)*v
a = 0
b = 10
va = 20
N = 50
h = (b-a)/N
t = np.arange(a, b+h, h)
v = np.zeros((N+1,))
v[0] = va

 #using fourth oreder Runge-Kutta to obtain the first 3 points
for i in range(0,N):
    if i in range(0,3):
        k1 = h * f(t[i],v[i])
        k2 = h * f(t[i] + (h/2.0), v[i] +(k1/2.0))
        k3 = h * f(t[i] + (h/2.0), v[i] + (k2/2.0))
        k4 = h * f(t[i] + h, v[i] + k3)

        v[i + 1] = v[i] + (k1 + 2.0*k2 + 2.0*k3 + k4)/6.0
    else:
        v[i + 1] = v[i] + h*(55.0 * f(t[i],v[i]) - 59.0 * f(t[i-1],v[i-1]) +
        v[i + 1] = v[i] + h*(9.0 * f(t[i+1], v[i + 1]) + 19.0 * f(t[i],v[i])

h1 = 0.2
def g1(v):
    g = 9.8
    return g-v
v0 = [20]
t0 = [0]
y0 = [0]

# solving the v(t) used for y(t)
for i in range(N+1):
    v1 = v0[i] + h1*g1(v0[i])
    v0.append(v1)
    t0.append(t0[i]+h1)

# solving for y(t)
for j in range(N+1):
    y1 = y0[j] + (h1/2)*(v0[j] + v0[j+1])
    y0.append(y1)
y0 = list(np.around(np.array(y0),3))
v = list(np.around(np.array(v),3))
print("y(t) =", y0)
print("v(t) =", v)
```

```
print("t =", t)
legends = []
plt.plot(t,v,"--",label="v(t) in Predictor-Corrector", color="r")
plt.plot(t0,y0,"--", label="y(t) in Predictor-Corrector", color="b")
plt.legend(loc="best")
plt.title("v(t) vs y(t) in Predictor-Corrector Method at h=0.2")
plt.show()
```

```
y(t) = [0.0, 3.796, 7.225, 10.36, 13.26, 15.972, 18.534, 20.975, 23.32, 25.58
8, 27.794, 29.951, 32.069, 34.155, 36.216, 38.257, 40.282, 42.293, 44.295, 4
6.288, 48.274, 50.255, 52.232, 54.206, 56.177, 58.145, 60.112, 62.078, 64.04
2, 66.006, 67.969, 69.931, 71.893, 73.854, 75.815, 77.776, 79.737, 81.698, 8
3.658, 85.618, 87.579, 89.539, 91.499, 93.459, 95.42, 97.38, 99.34, 101.3, 10
3.26, 105.22, 107.18, 109.14]
v(t) = [20.0, 18.153, 16.641, 15.402, 14.389, 13.559, 12.879, 12.323, 11.867,
11.494, 11.189, 10.939, 10.734, 10.567, 10.43, 10.317, 10.225, 10.15, 10.088,
10.038, 9.997, 9.963, 9.935, 9.912, 9.894, 9.879, 9.866, 9.856, 9.848, 9.841,
9.835, 9.831, 9.827, 9.824, 9.821, 9.819, 9.818, 9.816, 9.815, 9.814, 9.813,
9.813, 9.812, 9.812, 9.812, 9.811, 9.811, 9.811, 9.811, 9.811, 9.81]
t = [ 0.   0.2  0.4  0.6  0.8  1.   1.2  1.4  1.6  1.8  2.   2.2  2.4  2.6
  2.8  3.   3.2  3.4  3.6  3.8  4.   4.2  4.4  4.6  4.8  5.   5.2  5.4
  5.6  5.8  6.   6.2  6.4  6.6  6.8  7.   7.2  7.4  7.6  7.8  8.   8.2
  8.4  8.6  8.8  9.   9.2  9.4  9.6  9.8 10. ]
```



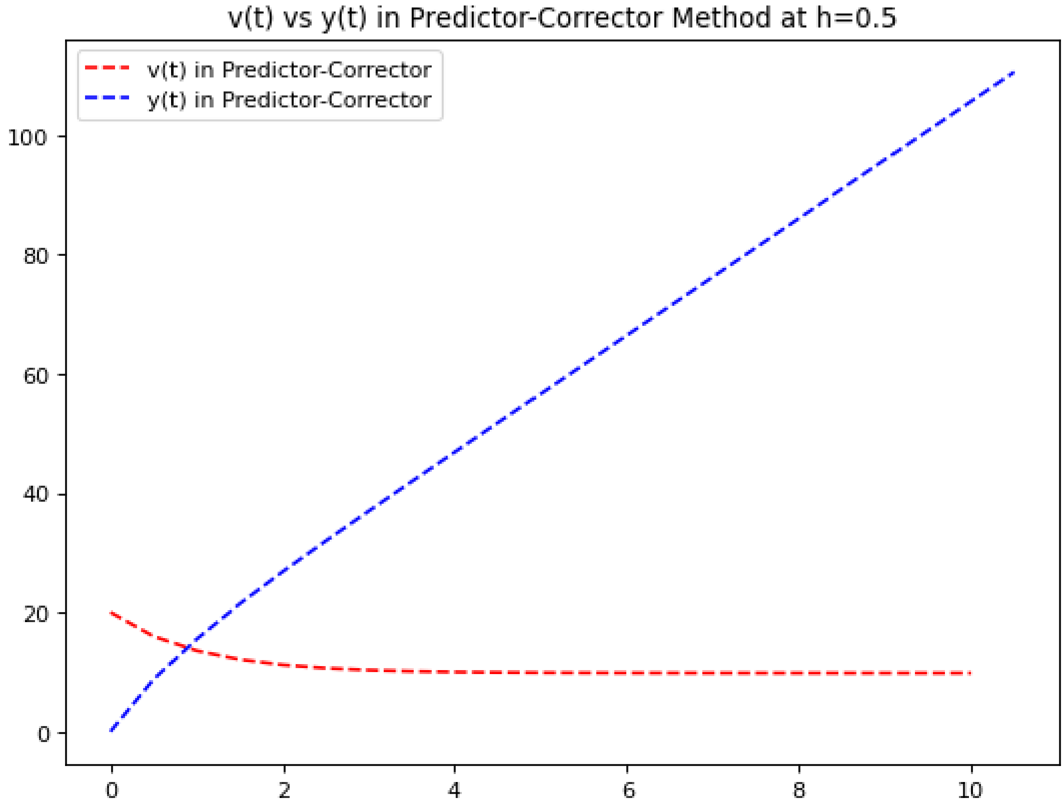
When step size, h=0.5

In [3]:

```
figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

# Solving for the Predictor-Corrector method at step-size h=0.5


N = 20
h = (b-a)/N
t = np.arange(a, b+h, h)
v = np.zeros((N+1,))
v[0] = va

 #using fourth oreder Runge-Kutta to obtain the first 3 points
for i in range(0,N):
```

```
    if i in range(0,3):
        k1 = h * f(t[i],v[i])
        k2 = h * f(t[i] + (h/2.0), v[i] +(k1/2.0))
        k3 = h * f(t[i] + (h/2.0), v[i] + (k2/2.0))
        k4 = h * f(t[i] + h, v[i] + k3)

        v[i + 1] = v[i] + (k1 + 2.0*k2 + 2.0*k3 + k4)/6.0
    else:
        v[i + 1] = v[i] + h*(55.0 * f(t[i],v[i]) - 59.0 * f(t[i-1],v[i-1]) +
        v[i + 1] = v[i] + h*(9.0 * f(t[i+1], v[i + 1]) + 19.0 * f(t[i],v[i])

h1 = 0.5
def g1(v):
    g = 9.8
    return g-v
v0 = [20]
t0 = [0]
y0 = [0]

# solving the v(t) used for y(t)
for i in range(N+1):
    v1 = v0[i] + h1*g1(v0[i])
    v0.append(v1)
    t0.append(t0[i]+h1)

# solving for y(t)
for j in range(N+1):
    y1 = y0[j] + (h1/2)*(v0[j] + v0[j+1])
    y0.append(y1)
y0 = list(np.around(np.array(y0),3))
v = list(np.around(np.array(v),3))
print("y(t) =", y0)
print("v(t) =", v)
print("t =", t)
legends = []
plt.plot(t,v,"--",label="v(t) in Predictor-Corrector", color="r")
plt.plot(t0,y0,"--", label="y(t) in Predictor-Corrector", color="b")
plt.legend(loc="best")
plt.title("v(t) vs y(t) in Predictor-Corrector Method at h=0.5")
plt.show()
```

```
y(t) = [0.0, 8.725, 15.538, 21.394, 26.772, 31.911, 36.93, 41.89, 46.82, 51.7
35, 56.643, 61.546, 66.448, 71.349, 76.25, 81.15, 86.05, 90.95, 95.85, 100.7
5, 105.65, 110.55]
v(t) = [20.0, 15.993, 13.562, 12.086, 11.18, 10.635, 10.307, 10.109, 9.99, 9.
918, 9.875, 9.849, 9.834, 9.824, 9.819, 9.815, 9.813, 9.812, 9.811, 9.811, 9.
81]
t = [ 0.   0.5  1.   1.5  2.   2.5  3.   3.5  4.   4.5  5.   5.5  6.   6.5
  7.   7.5  8.   8.5  9.   9.5 10. ]
```



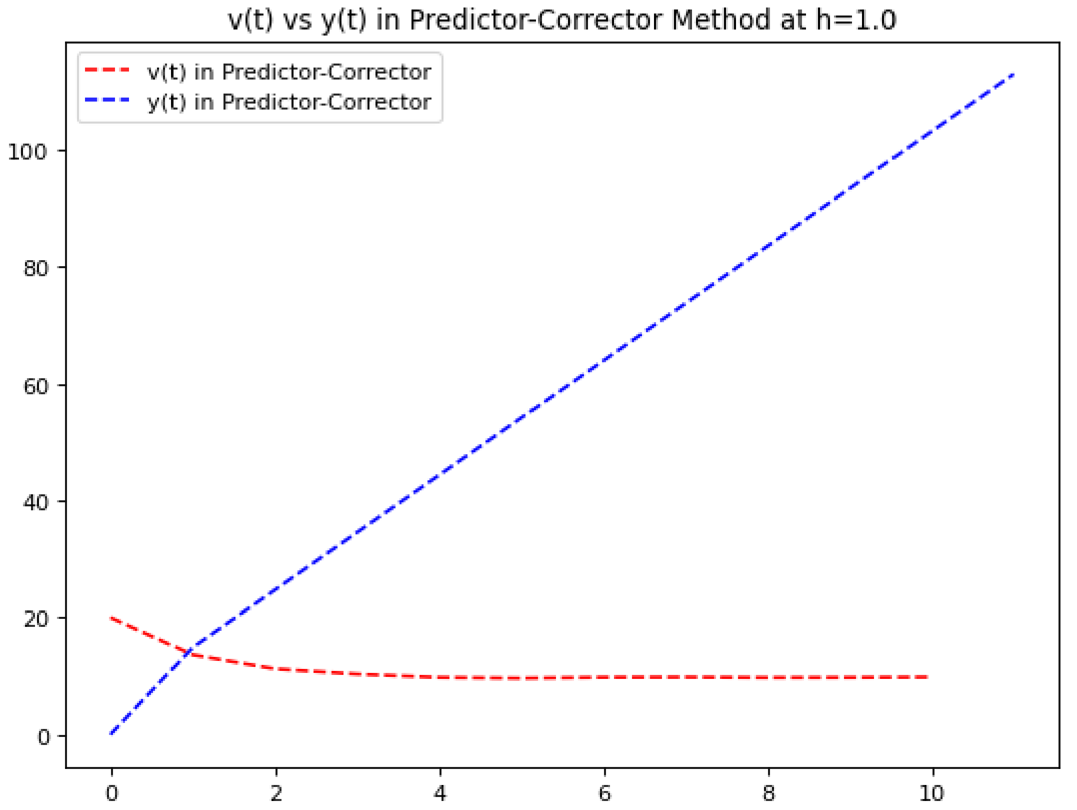
When step size, h=1

In [4]:

```
figure(num=None, figsize=(8, 6), dpi=80, facecolor='w', edgecolor='k')

# Solving for the Predictor-Corrector method at step-size h=1.0

va = 20
N = 10
h = (b-a)/N
t = np.arange(a, b+h, h)
v = np.zeros((N+1,))
v[0] = va

 #using fourth oreder Runge-Kutta to obtain the first 3 points
for i in range(0,N):
    if i in range(0,3):
        k1 = h * f(t[i],v[i])
        k2 = h * f(t[i] + (h/2.0), v[i] +(k1/2.0))
        k3 = h * f(t[i] + (h/2.0), v[i] + (k2/2.0))
        k4 = h * f(t[i] + h, v[i] + k3)

        v[i + 1] = v[i] + (k1 + 2.0*k2 + 2.0*k3 + k4)/6.0
    else:
        v[i + 1] = v[i] + h*(55.0 * f(t[i],v[i]) - 59.0 * f(t[i-1],v[i-1]) +
        v[i + 1] = v[i] + h*(9.0 * f(t[i+1], v[i + 1]) + 19.0 * f(t[i],v[i])

h1 = 1.0
def g1(v):
    g = 9.8
    return g-v
v0 = [20]
t0 = [0]
y0 = [0]

# solving the v(t) used for y(t)
for i in range(N+1):
    v1 = v0[i] + h1*g1(v0[i])
```

```
    v0.append(v1)
    t0.append(t0[i]+h1)

# solving for y(t)
for j in range(N+1):
    y1 = y0[j] + (h1/2)*(v0[j] + v0[j+1])
    y0.append(y1)
y0 = list(np.around(np.array(y0),3))
v = list(np.around(np.array(v),3))
print("y(t) =", y0)
print("v(t) =", v)
print("t =", t)
legends = []
plt.plot(t,v,"--",label="v(t) in Predictor-Corrector", color="r")
plt.plot(t0,y0,"--", label="y(t) in Predictor-Corrector", color="b")
plt.legend(loc="best")
plt.title("v(t) vs y(t) in Predictor-Corrector Method at h=1.0")
plt.show()
```

```
y(t) = [0.0, 14.9, 24.7, 34.5, 44.3, 54.1, 63.9, 73.7, 83.5, 93.3, 103.1, 11
2.9]
v(t) = [20.0, 13.631, 11.243, 10.347, 9.777, 9.635, 9.799, 9.834, 9.745, 9.76
7, 9.841]
t = [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```




```
In [ ]:
```