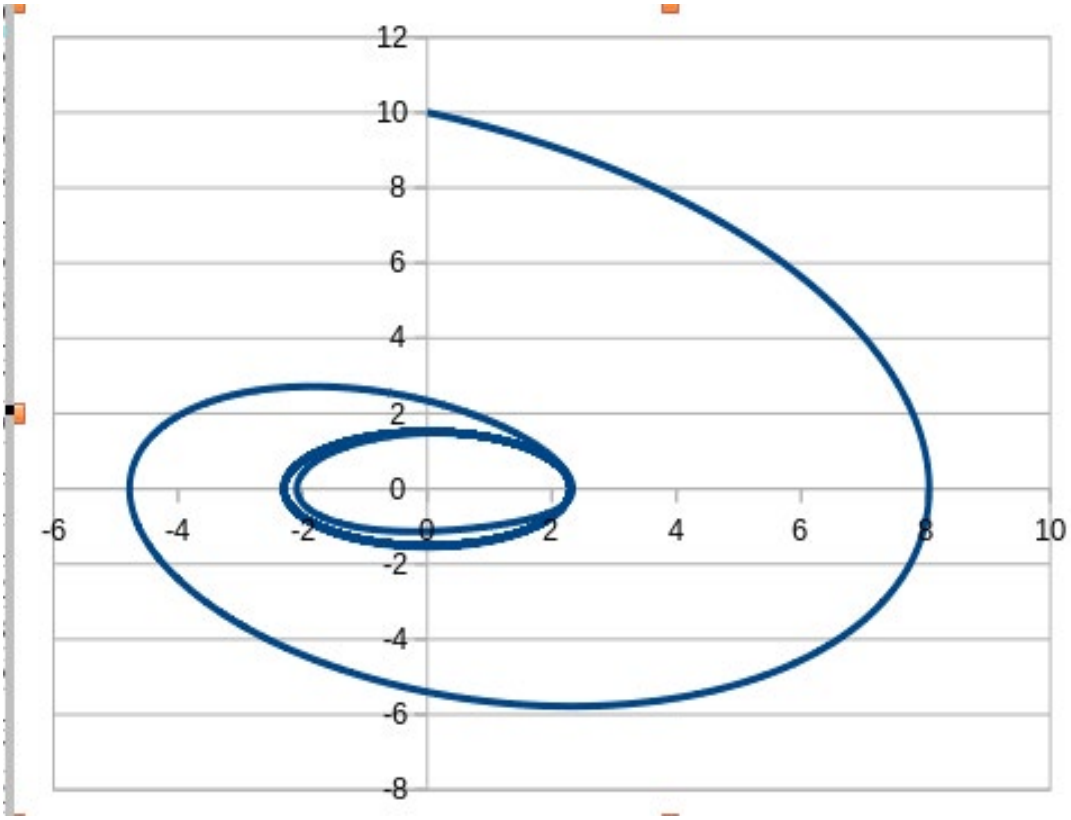
Forced damped Oscillator with Runge-kutta




```
#include <iostream>
#include <fstream>
#include <cmath>
const float gama=0.5;
const float omega0=1.0;
const float A=1.5;
const float omegaD=0.66;
using namespace std;



float f1(float x1, float x2, float x3, float t) {
return x2;
}
```

```
float f2(float x1, float x2, float x3, float t) {
return -gama * x2 - pow(omega0,2) *x1 + A * cos(x3);
}

float f3(float x1, float x2, float x3, float t) {
return omegaD;
}

int main (){
   float T=100.0;
   float tau=0.001,t;
   int nmax=T/tau;
   int n;
   float* x1=new float[nmax+1];
   float* x2=new float[nmax+1];
   float* x3=new float[nmax+1];
   float k1[3],k2[3],k3[3],k4[3];
   ofstream output ("rkg4_results.ods");
   cout << "x1[0] (initial position) =";       cin >> x1[0];
   cout << "x2[0] (initial velocity) =";       cin >> x2[0];
   cout << "x3[0] (initial phase) =";                    cin >> x3[0];


   for (int n = 0;n < nmax; n++) {
      t=n * tau;

      k1[0] = f1(x1[n], x2[n], x3[n],t);
      k1[1] = f2(x1[n], x2[n], x3[n],t);
```

```
    k1[2] = f3(x1[n], x2[n], x3[n],t);

    k2[0] = f1(x1[n]+0.5*tau*k1[0],x2[n]+0.5*tau*k1[1],x3[n]+0.5*tau*k1[2],t+0.5*tau);
    k2[1] = f2(x1[n]+0.5*tau*k1[0],x2[n]+0.5*tau*k1[1],x3[n]+0.5*tau*k1[2],t+0.5*tau);
    k2[2] = f3(x1[n]+0.5*tau*k1[0],x2[n]+0.5*tau*k1[1],x3[n]+0.5*tau*k1[2],t+0.5*tau);

    k3[0] = f1(x1[n]+0.5*tau*k2[0],x2[n]+0.5*tau*k2[1],x3[n]+0.5*tau*k2[2],t+0.5*tau);
    k3[1] = f2(x1[n]+0.5*tau*k2[0],x2[n]+0.5*tau*k2[1],x3[n]+0.5*tau*k2[2],t+0.5*tau);
    k3[2] = f3(x1[n]+0.5*tau*k2[0],x2[n]+0.5*tau*k2[1],x3[n]+0.5*tau*k2[2],t+0.5*tau);

    k4[0] = f1(x1[n]+tau*k3[0],x2[n]+tau*k3[1],x3[n]+tau*k3[2],t+tau);
    k4[1] = f2(x1[n]+tau*k3[0],x2[n]+tau*k3[1],x3[n]+tau*k3[2],t+tau);
    k4[2] = f3(x1[n]+tau*k3[0],x2[n]+tau*k3[1],x3[n]+tau*k3[2],t+tau);

    x1[n+1] = x1[n] + (tau/6.0)*(k1[0] + 2 * k2[0] + 2 * k3[0] + k4[0]);
    x2[n+1] = x2[n] + (tau/6.0)*(k1[1] + 2 * k2[1] + 2 * k3[1] + k4[1]);
    x3[n+1] = x3[n] + (tau/6.0)*(k1[2] + 2 * k2[2] + 2 * k3[2] + k4[2]);
}

for (n=0;n<nmax;n++) {
    t = n * tau;
    output << t << "\t" << x1[n] << "\t" << x2[n] << "\t" << x3[n] << endl;
}
delete[] x1;
delete[] x2;
delete[] x3;

output.close();
return 0;
```

}