Computational methods and their application using Python

Finding common interests and potential for cooperation

Joan Jani[1]

[1]Polytechnic University of Tirana
Faculty of Mathematics and Physics

Erasmus+ Program, December 2021

Table of Contents

Table of Contents

Polytechnic University of Tirana (upt.al)

FACULTIES of PUoT

- Faculty of Civil Engineering (FCE)
- Faculty of Architecture and Urban Planning (FAUP)
- Faculty of Information Technology (FIT)
- Faculty of Mechanical Engineering (FME)
- Faculty of Geology and Mining (FGM)
- Faculty of Electrical Engineering (FEE)
- Faculty of Mathematical Engineering and Physical Engineering (FME& PhE)
- Institute Geosciences and Energy, Water and Environment (IGEWE)

Faculty of Mathematical Engineering and Physical Engineering

Dean : Prof. Dr. Shkelqim Kuka

Department of Eng. Mathematics
Prof. Dr. Luela Prifti

Department of Eng. Physics
Prof Altin Gjevori

Study program

- Bachelor Eng. Mathematics
- Master in Eng. Mathematics
- Master in System Analysis (Signals ans Systems)
- Master in Actuary (Risk Analysis)

- Bachelor in Eng. Physics
- Master in Eng. Physics (Materials, Photovoltaics,)
- Master in Medical physics (Radiotherapy, Imaging, PET-CT, MRI)

Life in FIMIF

Our Labs

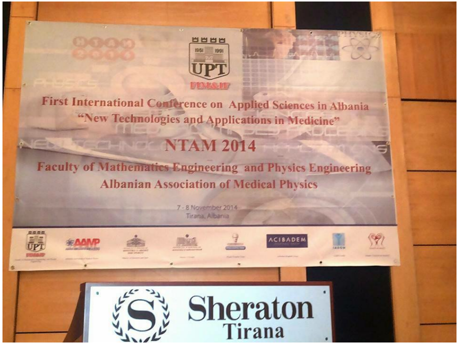
Computer Lab

Library

Conferences Organized

NTAM 2014

International Conference "New Technologies and Applications in Medicine"

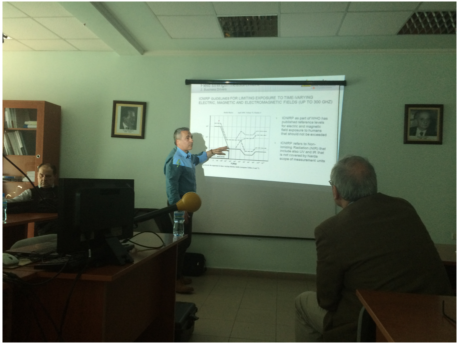


Conferences Organized

ICEAS 2017

International Conference on Applied Sciences and Engineering

International Collaboration



International Collaboration

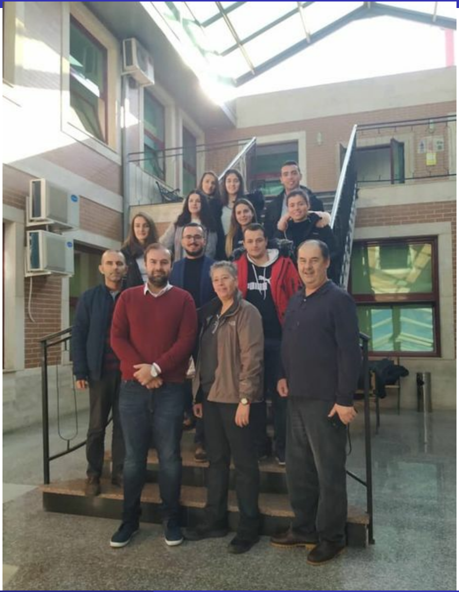


Table of Contents

Tools that we use

Linux

Octave

Python

A demonstration

From order to Chaos

Newton's second law of motion

$$\frac{d^2\vec{r}}{dt^2} = \vec{a}(t, \vec{r}, \vec{v}) \tag{1}$$

where

$$\vec{a} = (t, \vec{r}, \vec{v}) = \frac{\vec{F}}{m} \quad \text{and} \quad \vec{v} = \frac{d\vec{r}}{dt} \tag{2}$$

with initial conditions

$$\vec{r}(t_0) = \vec{r}_0 \quad \text{and} \quad \vec{v}_(t_0) = \vec{v}_0 \tag{3}$$

only one solution $\vec{r}(t)$.

Differential equation

We define a system of two first order equation

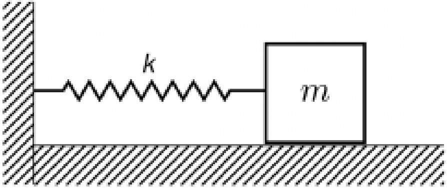
$$\frac{d\vec{r}}{dt} = \vec{v} \quad \text{dhe} \quad \frac{d\vec{v}}{dt} = \vec{a}(t, \vec{r}, \vec{v}) \tag{4}$$

For one dimensional motion we will have

$$\frac{dx}{dt} = v \quad \text{and} \quad \frac{dv}{dt} = a \tag{5}$$

Oscillator

System of spring with constant k and object with mass m



By applying Newton's law

$$\sum F = ma \Rightarrow -kx = ma \Rightarrow \boxed{-kx = m\frac{d^2x}{dt^2}}$$

System of equations

Differential equations of motion:

$$m\frac{d^2x}{dt^2} = -kx \quad \text{and} \quad \frac{dx}{dt} = v \tag{6}$$

and we write the first order equations

$$\frac{dv}{dt} = -\frac{k}{m}x \quad \text{and} \quad \frac{dx}{dt} = v \tag{7}$$

From analog to digital world

Euler's method

We define:

$t_i = 0$: Initial time

t_f : Final time

$t_f - t_i$: Time deference, divided in $N-1$ segments with duration $\Delta t = h$, where

$$h = (t_f - t_i)/(N-1)$$

So the speed could be written: $(x_{n+1} - x_n)/\Delta t \approx x'_n$
and the system of equations:

$$\begin{aligned} v_{n+1} &= v_n + a_n \Delta t \\ x_{n+1} &= x_n + v_n \Delta t \end{aligned} \tag{8}$$

acceleration a:

$$F = ma \Rightarrow -kx = ma \Rightarrow a = -\frac{kx}{m}$$

.

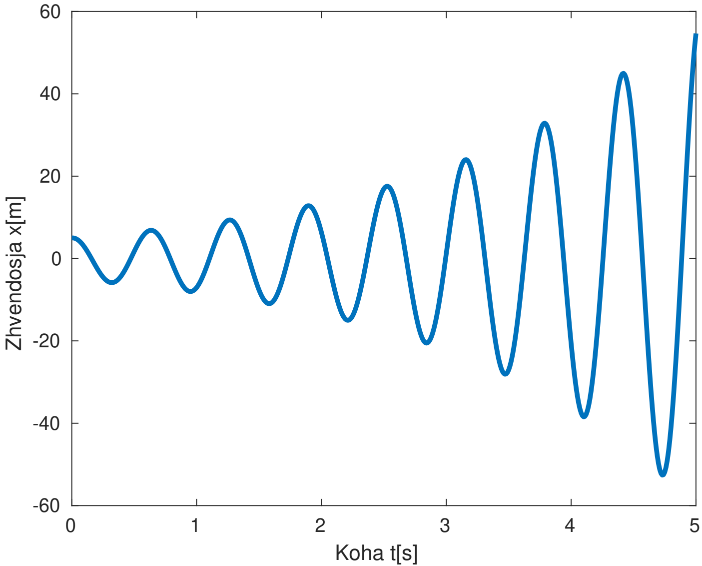
Code 1/2

```
clear;clc;close all;
% Mass
m=1;
% Elasticity constant k
k=100;
% Initial conditions
x(1)=5;
v(1)=0;
dt=0.01;
% Number of points
N=500;
% Time
t=0:dt:dt*N;
```

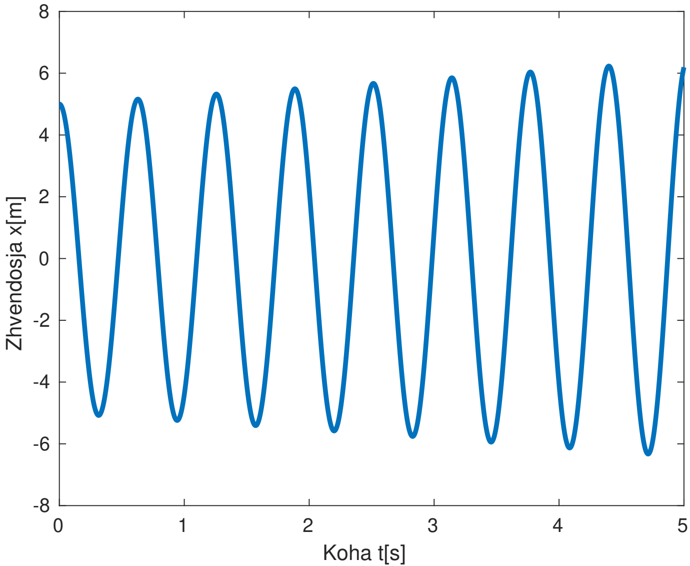
Euler 1/2

```
 % Euler
for i=1:N
  % speed
  v(i+1)=v(i)-(k/m)*x(i)*dt;
  % position
  x(i+1)=x(i)+v(i)*dt;

end
% Plots
plot(t,x)
xlabel('Koha t[s]')
ylabel('Zhvendosja x[m]')
```

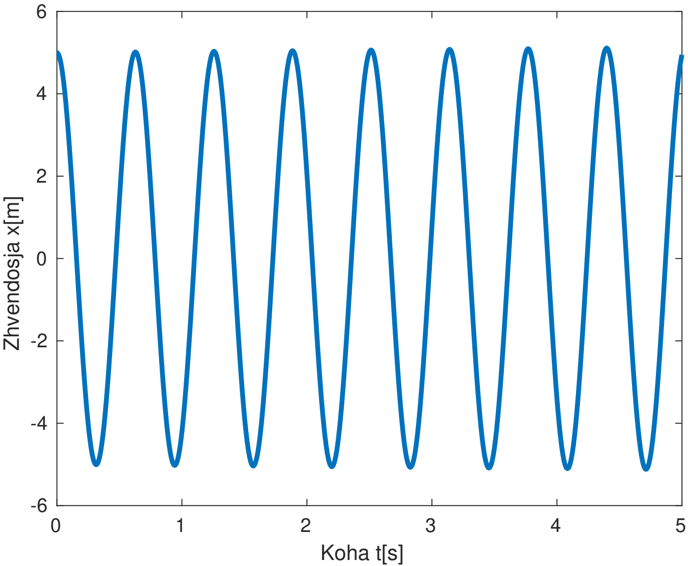
Results for $N = 500$ and $dt = 0.01s$



Results $N = 5000$ and $dt = 0.001s$



Results for $N = 50000$ and $dt = 0.0001s$



Conclusion

We need a new method

Metoda Euler-Cromer

A very small modification

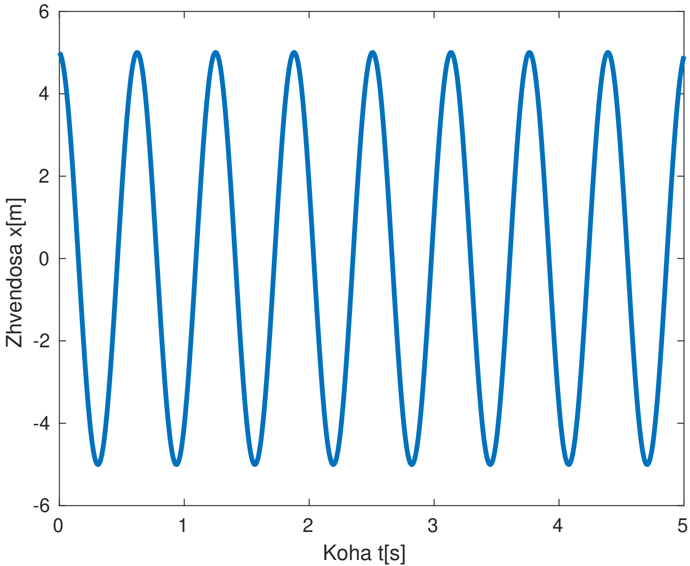
$$
\begin{aligned}
v_{n+1} &= v_n + a_n \Delta t \\
x_{n+1} &= x_n + v_{n+1} \Delta t
\end{aligned} \tag{9}
$$

We run again the script with this minor change

Euler-Cromer 1/2

```
 % Implementimi i metodes Euler−Cromer

for i=1:N
  % Llogaritja e shpejtesise
  v(i+1)=v(i)−(k/m)*x(i)*dt;
  % Llogaritja e pozicionit
  x(i+1)=x(i)+v(i+1)*dt;

end
% Krijimi i grafikut te zhvendosjes ne lidhje me kohes
plot(t,x)
xlabel('Koha t[s]')
ylabel('Zhvendosja x[m]')
```

Euler-Cromer për $N = 500$ dhe $dt = 0.01s$



Runge - Kutta

Let the system be:

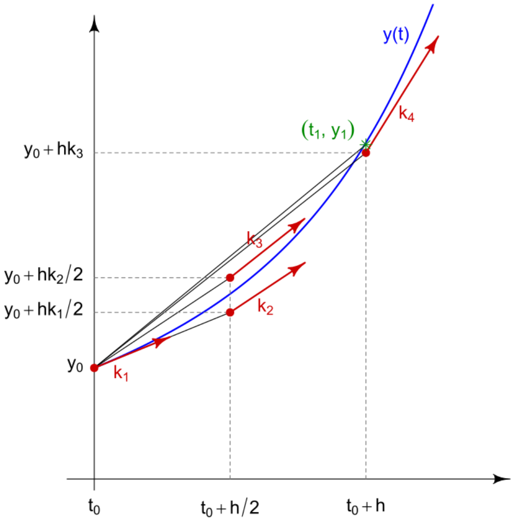
$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

The solution will be

$$y_{n+1} = y_n + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right),$$
$$t_{n+1} = t_n + h$$

Slopes used by the classical Runge-Kutta method

$$k_1 = f(t_n, y_n),$$
$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right),$$
$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right),$$
$$k_4 = f\left(t_n + h, y_n + hk_3\right).$$



Optimizations

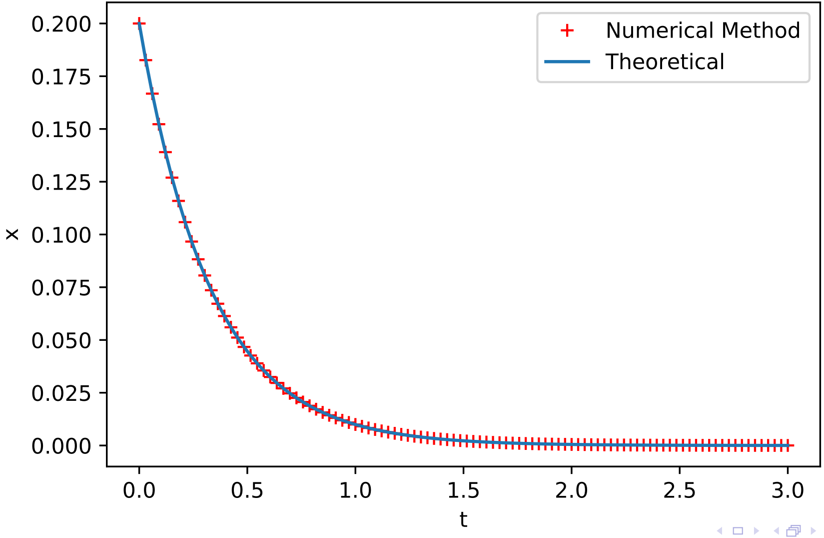
Why to programm when there are ready tools

Example

$$\frac{dx}{dt} = -kx \Rightarrow \frac{dx}{x} = -kdt \Rightarrow = \int_{0.2}^{x} \frac{dx}{x} = -\int_{0}^{3} kdt \Rightarrow \boxed{x = 0.2e^{-kt}}$$

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
k = 3
N = 100
t = np.linspace(0,3,N)
# Differential equation
def f(x,t):
    dxdt = -k*x
    return dxdt
x = 0.2
# Solver
solution = odeint(f,x,t)
plt.plot(solution)
```

Solution of example

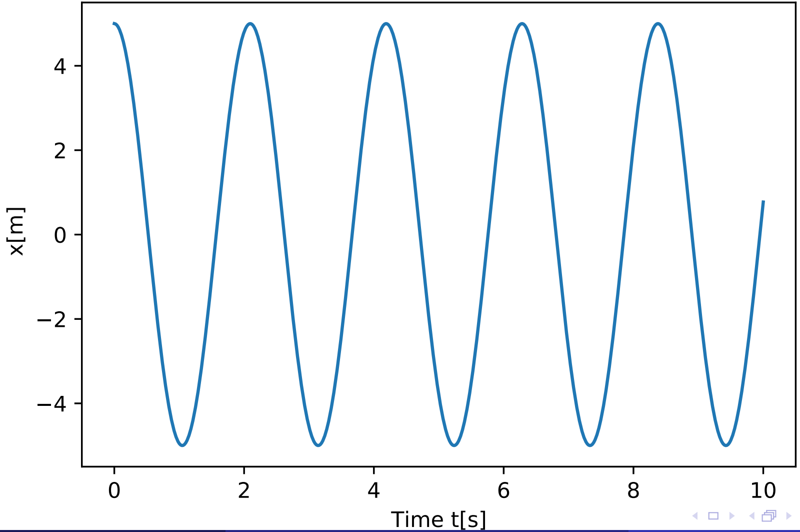


Oscillator

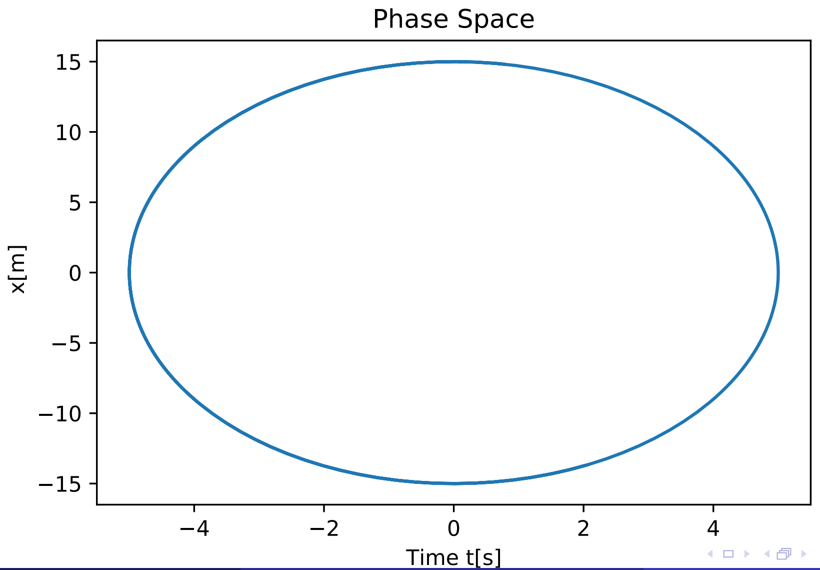
$$\frac{dv}{dt} = -\frac{k}{m}x \quad \text{and} \quad \frac{dx}{dt} = v \tag{10}$$

```
# Oscilator
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# System Parameters
[k, m] = [9, 1]
# Initial Conditions
x = [5,0]
# Simulation parameters
N = 1000
t = np.linspace(0,10,N)
def f(x,t):
    dvdt = -(k/m)*x[0]
    dxdt = x[1]
    return [dxdt,dvdt]
solution = odeint(f,x,t)
```

Solution



Phase space

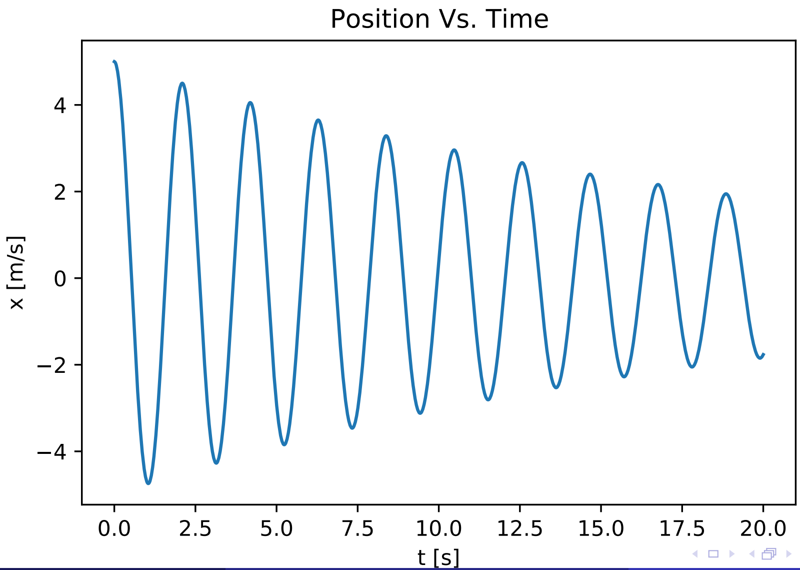


Dumped Oscilator - Position Vs. Time

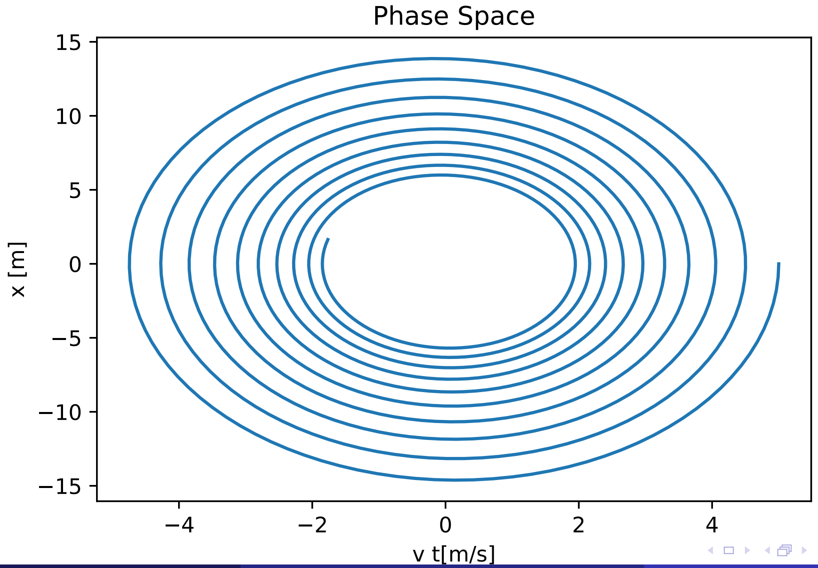
$$m\ddot{x} = -kx - \beta\dot{x} \tag{11}$$

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# System Parameters
[k, m, b] = [9, 1, 0.1]
# Initial Conditions
x = [5,0]
# Simulation parameters
N = 1000
t = np.linspace(0,20,N)
def f(x,t):
    dvdt = -(k/m)*x[0]-(b/m)*x[1]
    dxdt = x[1]
    return [dxdt,dvdt]
solution = odeint(f,x,t)
```

Dumped Oscilator - Position Vs. Time



Dumped Oscilator - Phase space



Damped Driven Oscillator

The system with equation:

$$m\ddot{x} = -kx - \beta\dot{x} + A\sin(\omega t)$$

with solution

$$A_t = \frac{A}{\sqrt{m^2(\omega_o^2 - \omega^2)^2 + \beta^2\omega^2}}$$

$$\delta = \tan^{-1}\left(\frac{\beta\omega}{m(\omega_o^2 - \omega^2)}\right)$$

$$x_t(t) = A_t \sin(\omega t - \delta)$$

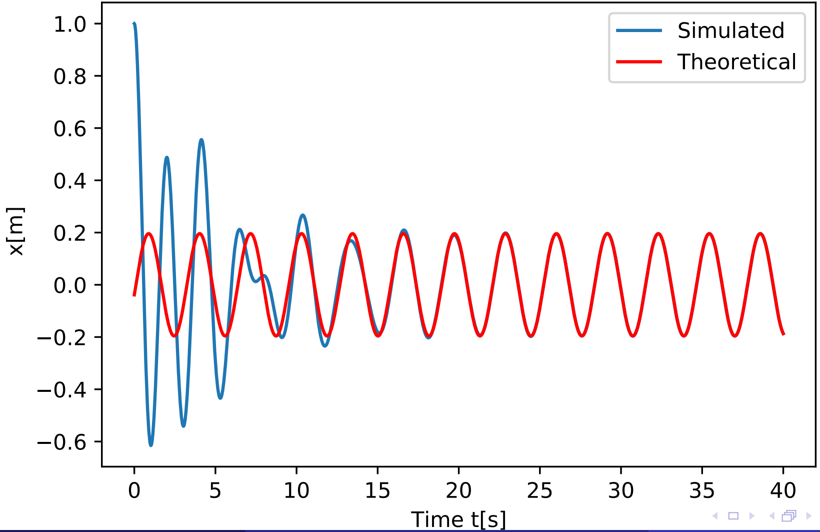
Damped Driven Oscillator

```python
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# System Parameters
[k, m, b, A] = [9, 1 ,0.5 ,1]
# Natural frequency
w_o = np.sqrt(k/m)
w = 2
# Initial Conditions
x=[1,0]
# Simulation parameters
N = 5000
t = np.linspace(0,40,N)
def f(x,t):
    dvdt = (-(k)*x[0]-b*x[1]+A*np.sin(w*t))/m
    dxdt = x[1]
    return [dxdt,dvdt]
```

Damped Driven Oscillator

```
solution = odeint(f,x,t)

At = (A)/np.sqrt((m**2)*(w_o**2-w**2)**2+(b*w)**2)
delta = np.arctan((b*w)/(m*(w_o**2-w**2)))
xt  = At*np.sin(w*t - delta)

plt.plot(t,solution[:,0],label='Simulated')
plt.plot(t,xt,'r',label='Theoretical')
plt.legend()
plt.xlabel('Time t[s]')
plt.ylabel(' x[m]')
plt.savefig('Stable.pdf')
plt.show()
```

Damped Driven Oscillator - Position Vs. Time



Damped Driven Oscillator - Resonance

$$A_t = \frac{A}{\sqrt{(\omega_o^2 - \omega^2)^2 + b^2\omega^2}}$$

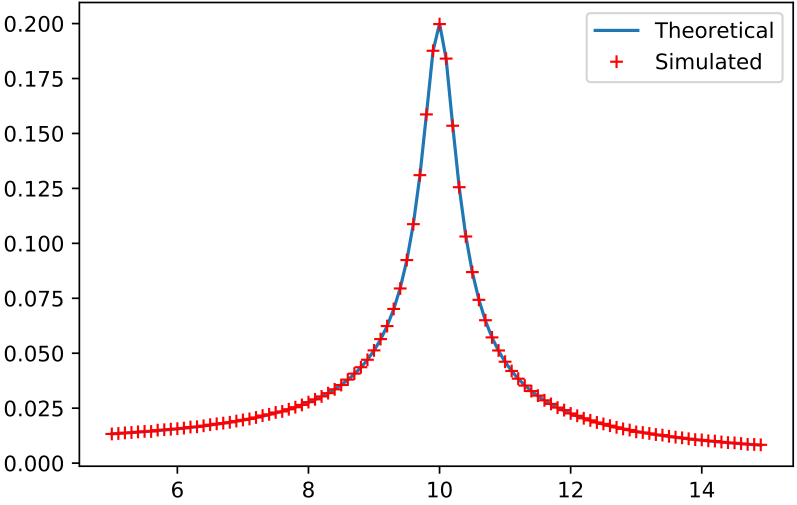
Damped Driven Oscillator

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# System Parameters
[k, m, b, A] = [100, 1 ,0.5 ,1]
# Natural frequency
w_o = np.sqrt(k/m)
# Range of frequencies
wt = np.arange(w_o-w_o/2,w_o+w_o/2,0.1)
# Initial Conditions
x=[1,0]
# Simulation parameters
N = 500
t = np.linspace(0,40,N)
Amax = np.zeros(len(wt))

def f(x,t):
    dydt = (-(k)*x[0]-b*x[1]+A*np.sin(w*t))/m
```

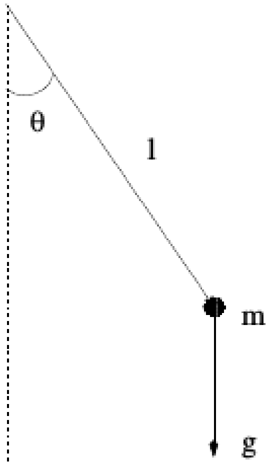
Damped Driven Oscillator

```
for i in range(len(wt)):
    w = wt[i]
    solution = odeint(f,x,t)
    Amax[i] = np.max(solution[-100:,0])

# For theoritical
At = A/np.sqrt((w_o**2-wt**2)**2+b**2*wt**2)

plt.plot(wt,At,label='Theoretical')
plt.plot(wt,Amax,'r+',label='Simulated')
plt.legend()
plt.savefig('resonance.pdf')
plt.show()
```

Damped Driven Oscillator- Resonance



Pendulum



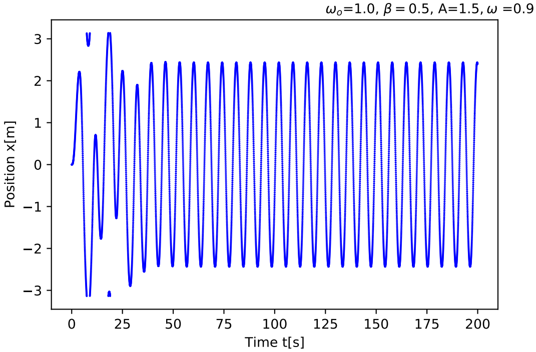
$$-ml\ddot{\theta} = mg\sin(\theta)$$

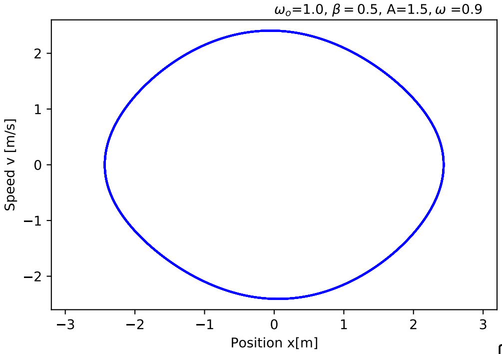
$$\ddot{\theta} = -\frac{g}{L}\sin(\theta)$$

Driven dumped pendulum

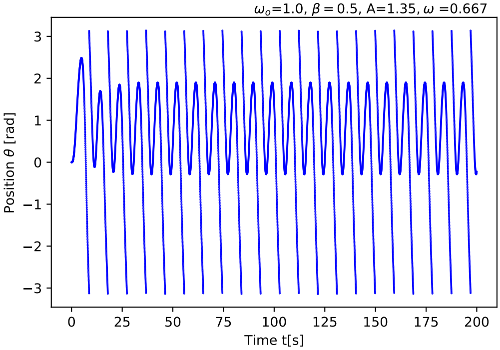
```
# Driven dumped pendulum
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt

# System Parameters
[g, L, b, A, w] = [10, 10 ,0.5 ,1.5, 0.9]
# Initial Conditions
x=[0.5,0]
# Simulation parameters
N = 1000
t = np.linspace(0,70,N)
def f(x,t):
    dvdt = -(g/L)*np.sin(x[0])-b*x[1]+A*np.sin(w*t)
    dxdt = x[1]
    return [dxdt,dvdt]
```

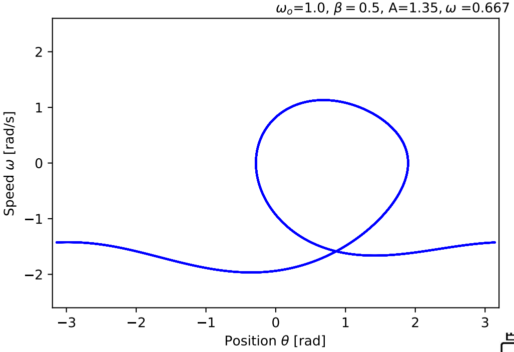
Driven dumped pendulum $A = 1.5$ $\omega = 0.9$



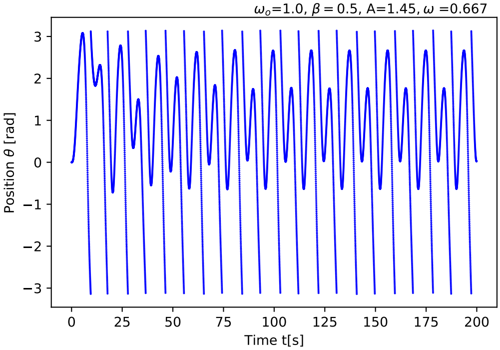


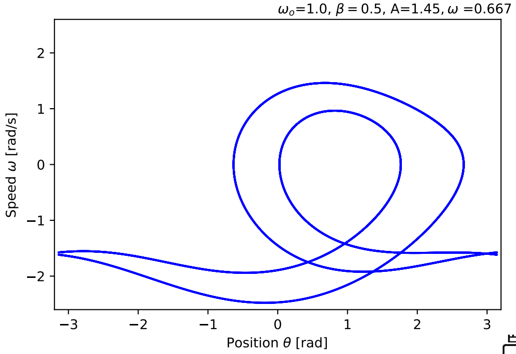
Driven dumped pendulum $\omega = 0.667$ A $= 1.35$



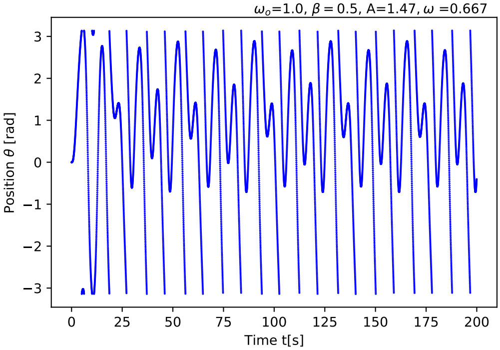


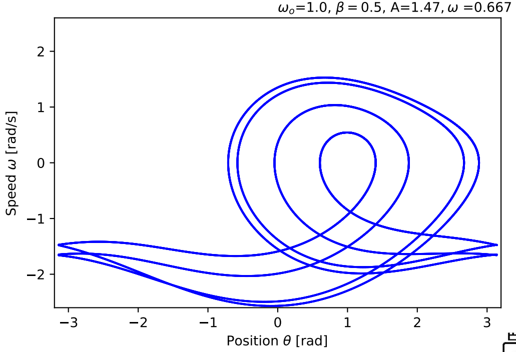
Driven dumped pendulum $\omega = 0.667$ A $= 1.45$



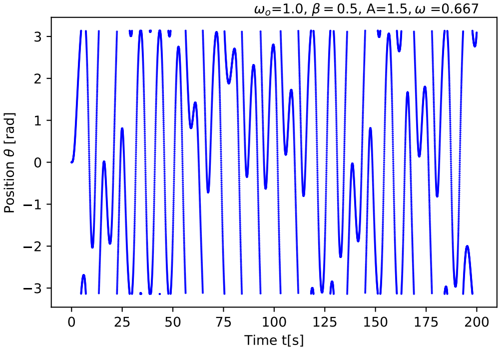


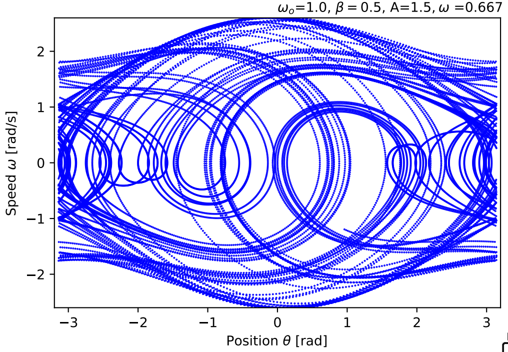
Driven dumped pendulum $\omega = 0.667$ A $= 1.47$





Driven dumped pendulum $\omega = 0.667$ A $= 1.50$





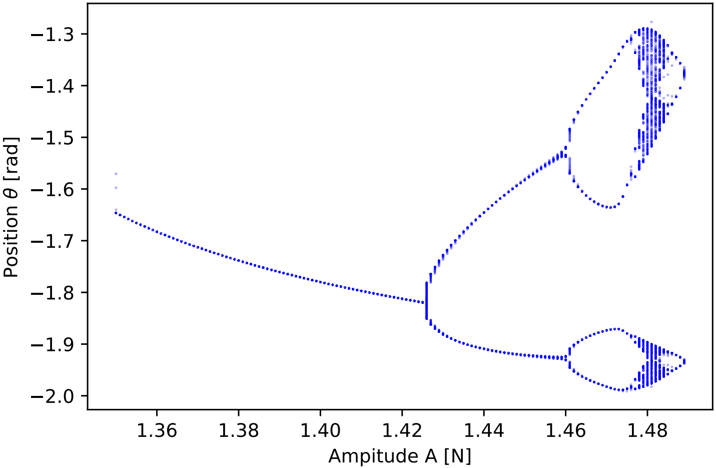
Driven dumped pendulum - Bifurcation

```
from scipy.integrate import odeint
import numpy as np
import matplotlib.pyplot as plt
# System Parameters
[g, L, b, A, w] = [10, 10 ,0.5 ,1.15, 0.6667]
# Initial Conditions
x=[0,0]
# Simulation Parameters
T = 2*np.pi/w
N  = 501
ti = 0
tf = T
# Poincare points and coordinate matrices
Np = 25000
Xp =[]
Yp =[]

def f(x,t):
    dvdt = -(g/L)*np.sin(x[0])-b*x[1]+A*np.sin(w*t)
    dxdt = x[1]
```

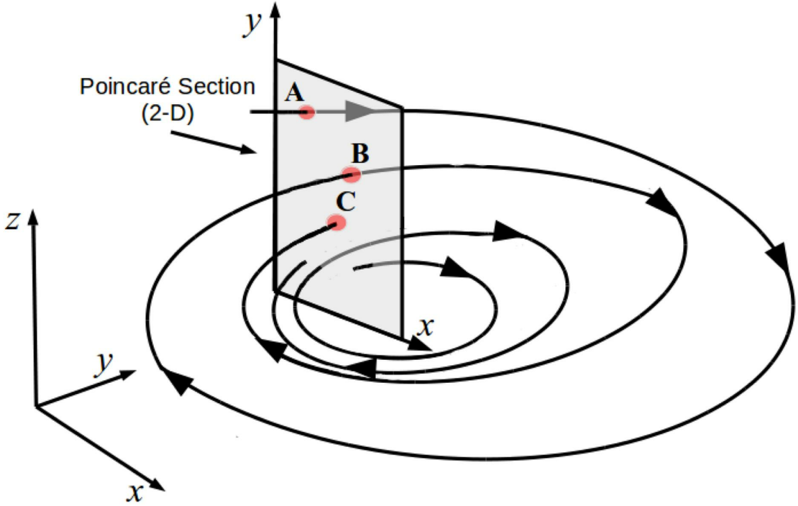
Driven dumped pendulum - Bifurcation

```
for j in range(len(A_bif)):
    A = A_bif[j]
    for i in range(Np):
        t =  np.linspace(ti,tf,N) + i*T
        solution = odeint(f,x,t)
        # Pozition only at -pi to pi
        for i, position in enumerate(solution[:,0]):
            while position > np.pi:
                position = position-(2.0*np.pi)
            while position <- np.pi:
                position = position+(2.0*np.pi)
        solution[i,0] = position
        x = solution[-1,:]
        Xp.append(x[0])
        Yp.append(x[1])
    Bif[j,:] = Yp
    Xp=[]
    Yp=[]
```

Driven dumped pendulum - Bifurcation



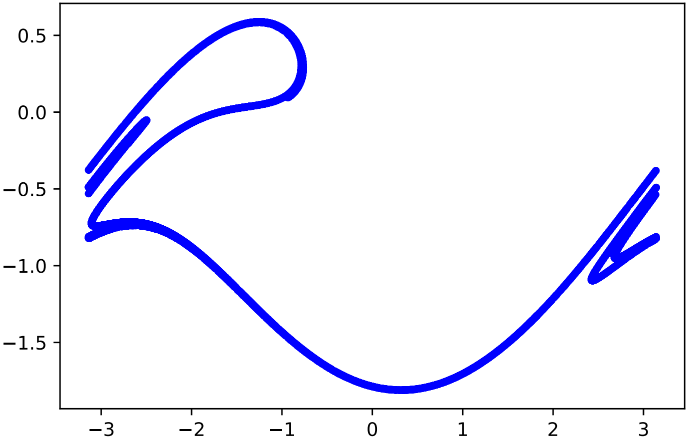
Driven dumped pendulum - Poincare section



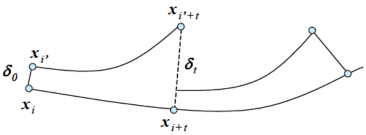
Driven dumped pendulum - Poincare section

```
for i in range(Np):
    t =  np.linspace(ti,tf,N) + i*T
    solution = odeint(f,x,t)
    # Pozition only at -pi to pi
    for i, position in enumerate(solution[:,0]):
        while position > np.pi:
            position = position-(2.0*np.pi)
        while position <- np.pi:
            position = position+(2.0*np.pi)
    solution[i,0] = position
    x = solution[-1,:]
    Xp.append(x[0])
    Yp.append(x[1])
```

Driven dumped pendulum - Poincare section



Calculation of Lyapunov



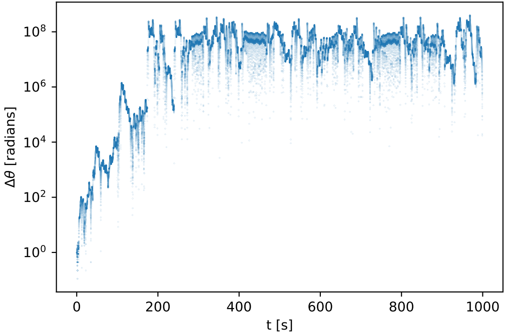
$$\delta_t \cong \delta_0 e^{\lambda t} \qquad \lambda = \frac{1}{N}\sum_{j=1}^{N} \log \frac{\delta_{t,j}}{\delta_{0,j}}$$

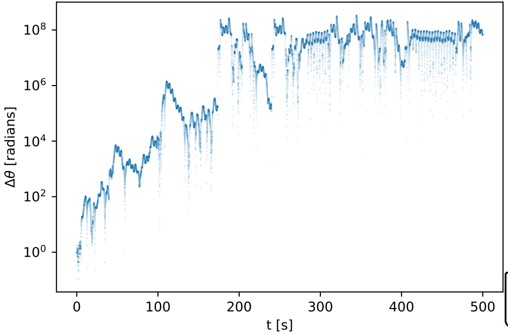
Eksponenti Lyapunov:

- $\lambda > 0$ chaos
- $\lambda = 1$ limit cycle
- $\lambda < 0$ loss of energy

Initial conditions θ_1 and θx_2 where $\theta x_2 = x_1 + \epsilon$:

$$\Delta\theta = |\theta_2 - \theta 2|$$





Calculation of Lyapunov

$$\lambda = 0.06629152230465289$$

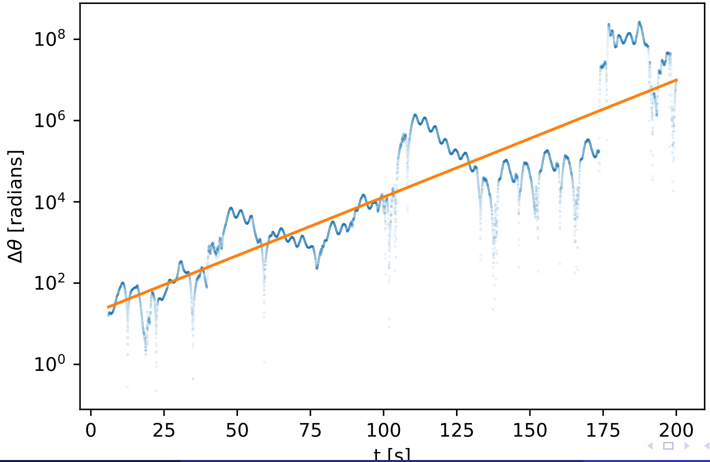
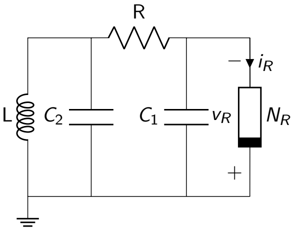
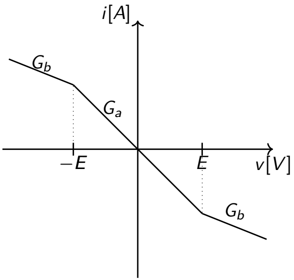


Table of Contents

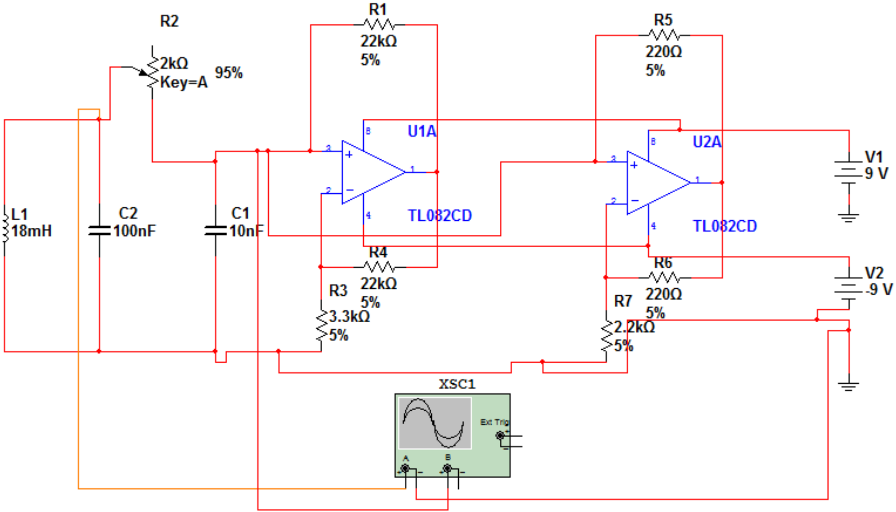




$$C_1 \frac{dv_1}{dt} = \frac{v_2 - v_1}{R} - h(v_1)$$
$$C_2 \frac{dv_2}{dt} = \frac{v_1 - v_2}{R} + i_L$$
$$L \frac{di_L}{dt} = -v_2$$

Non linear resistance the key factor for chaos.

Chua chaotic circuit



Simulation using MULTISIM

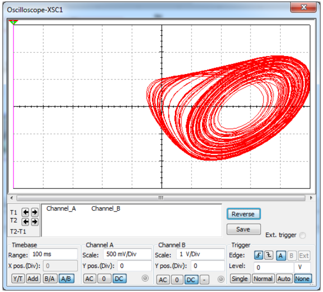


Figure: Phase space $R_2 = 1.72$

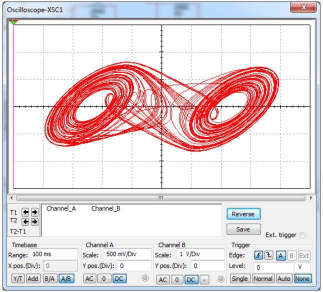


Figure: Realization $R_2 = 1.9\ k\Omega$

Simulation with MULTISIM

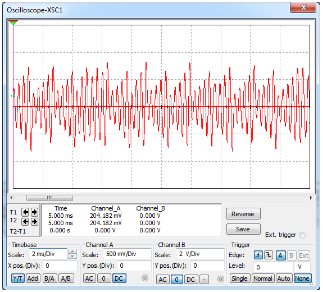


Figure: Voltage across C_1 and $R_2 = 1.9\ k\Omega$

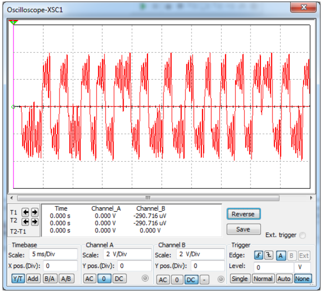
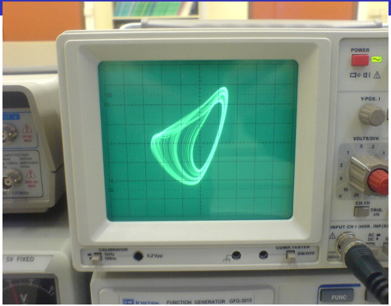
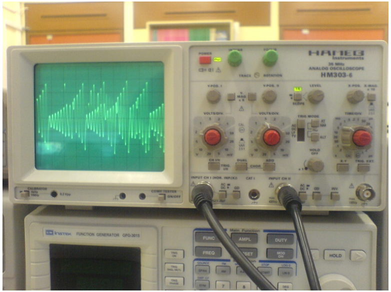
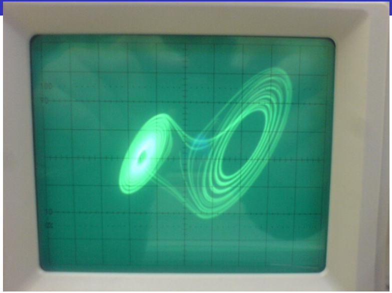
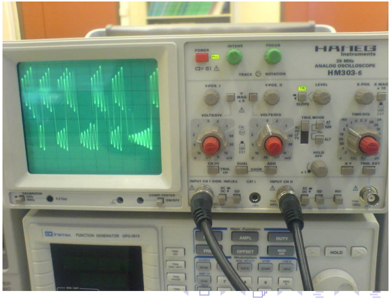


Figure: Voltage across C_2 and $R_2 = 1.9\ k\Omega$

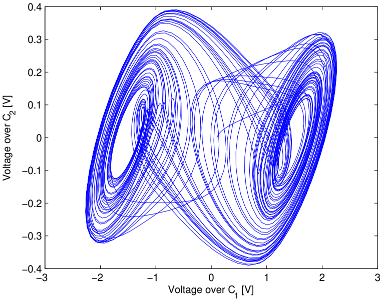

POWER
5V FIXED
FUNCTION GENERATOR GFG-3015
FUNC

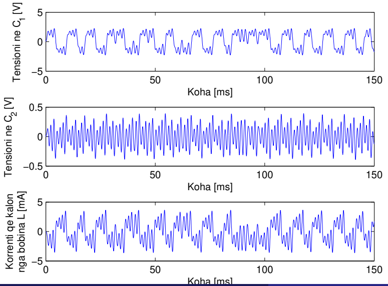

HAMEG
HM303-6

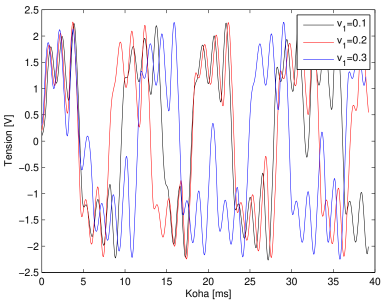


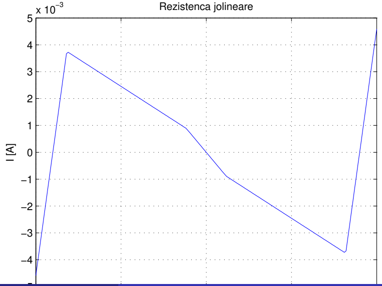

HAMEG
ANALOG OSCILLOSCOPE
HM303-6

Simulimi me MATLAB

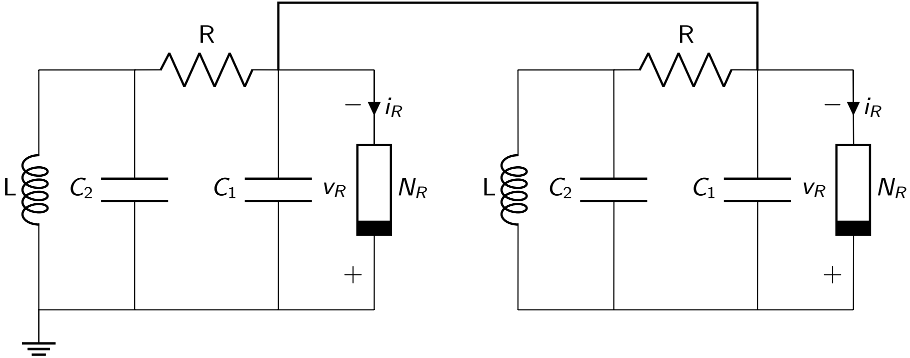








Synchronization



$$C_1 \frac{dv_1}{dt} = \frac{v_2 - v_1}{R} - h(v_1)$$
$$C_2 \frac{dv_2}{dt} = \frac{v_1 - v_2}{R} + i_L$$
$$L \frac{di_L}{dt} \ldots$$

- Same equations for both system
- $v_1 = v_1'$
- Runge – Kutta for solution

Table of Contents

Devices

We have at the Departament the Narda measuring devices

Narda EHP-50F

Spectrum Analyzer
1Hz-400kHz

Narda SRM 3006

Spectrum Analyzer
9 kHz to 6 GHz

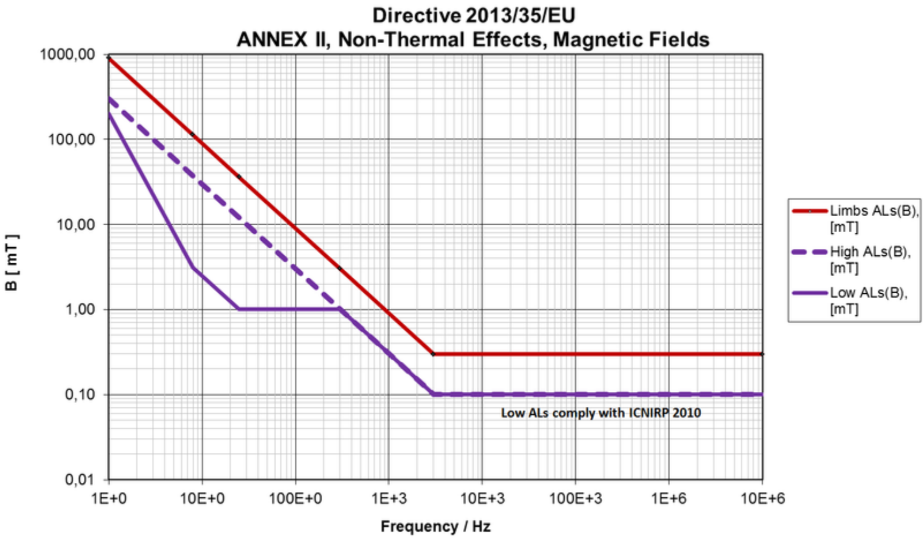
Limits

1. ICNIRP. Guidelines for limiting exposure to time-varying electric, magnetic,and electromagnetic fields (up to 300 GHz). Health Phys 1998;74(4)
2. ICNIRP. Factsheet on the guidelines for limiting exposure to time-varying electric and magnetic fields (1Hz-100kHz). Health Phys 2010;99(6)
3. Directive 2013/35/EU of the European Parliament and of the Council of 26 June 2013 on the minimum health and safety requirements regarding the exposure of workers to the risks arising from physical agents (electromagnetic fields). Off J Eur Union L 179/1.
4. VENDIM Nr. 843, datë 3.12.2014 "PËR MBROJTJEN E PUNËMARRËSVE NGA RISQET LIDHUR ME RREZATIMET JOJONIZUESE NË VENDIN E PUNËS"

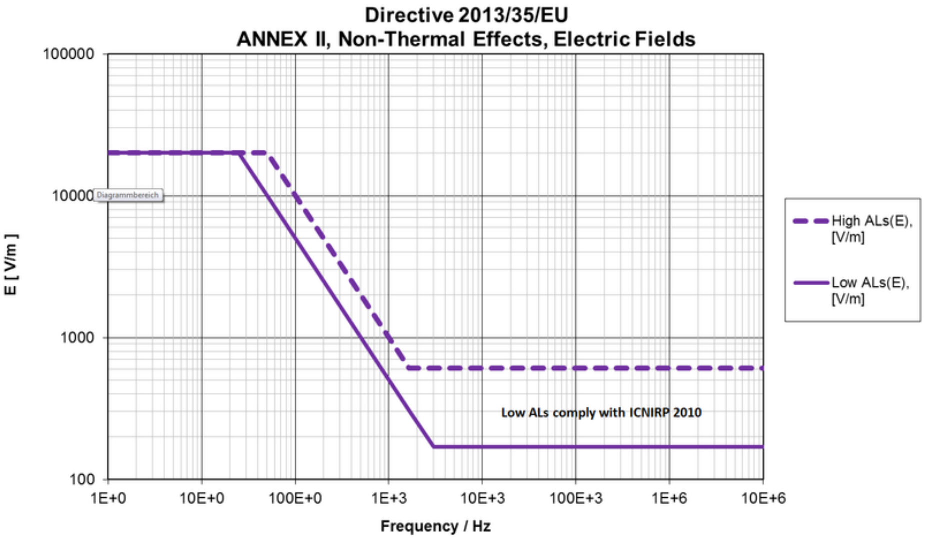
Limits

- Exposure limit values (ELVs)' means values established on the basis of biophysical and biological considerations, in particular on the basis of scientifically well-established short-term and acute direct effects
  - 'health effects ELVs' means those ELVs above which workers might be subject to adverse health effects, such as thermal heating or stimulation of nerve and muscle tissue
  - sensory effects ELVs' means those ELVs above which workers might be subject to transient disturbed sensory perceptions and minor changes in brain functions;
- 'action levels (ALs)' means operational levels established for the purpose of simplifying the process of demonstrating the compliance with relevant ELVs or, where appropriate, to take relevant protection or prevention measures specified in this Directive.

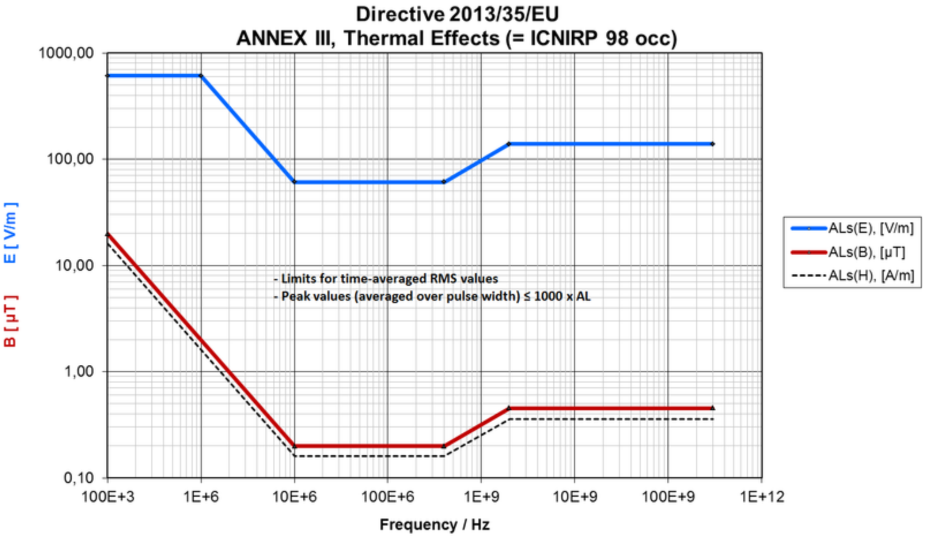
2013/35/EU - Magnetic Field



2013/35/EU - Electric Field



2013/35/EU - Thermal Effects

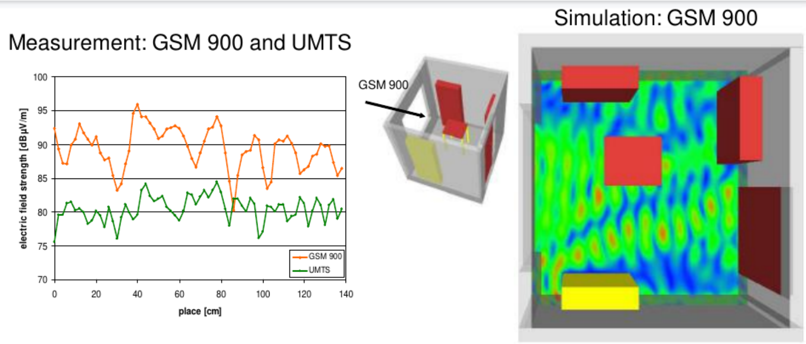


Limits for Greece

FEK 1105/B/6-9-2000 kai 3431 FEK 13/A/3-2-2006

Radio service	E_{max} (ICNIRP 1998)	E_{max} (Greece, general) 84 % of ICNIRP	E_{max} (Greece, close to schools, hospitals etc.) 77 % of ICNIRP
FM radio (100 MHz)	28.0 V/m	23.4 V/m	21.7 V/m
DVB-T (600 MHz)	33.7 V/m	28.2 V/m	26.1 V/m
GSM-900 (940 MHz)	42.2 V/m	35.3 V/m	32.7 V/m
GSM- / LTE-1800 (1850 MHz)	59.1 V/m	49.5 V/m	45.8 V/m
UMTS, LTE-2600 (>2000 MHz)	61.0 V/m	51.0 V/m	47.2 V/m

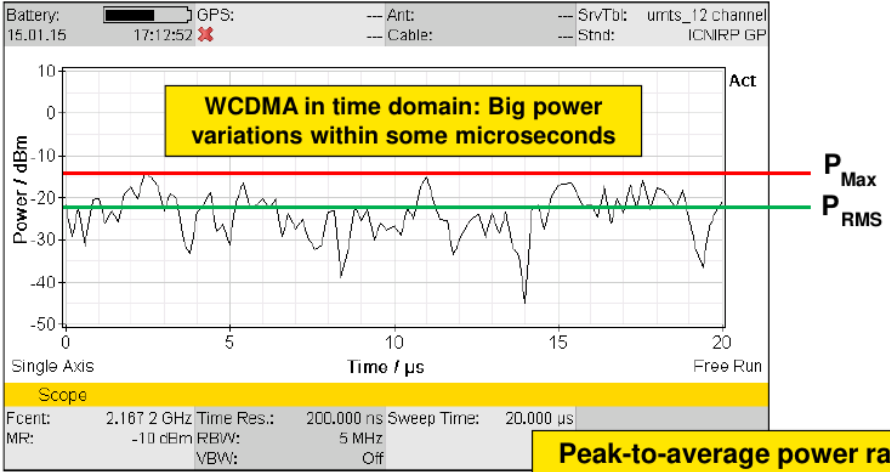
Space Variations



- Small scale variations (fast fading) especially at indoor scenarios due to multipath propagation
- Variations of 10 dB and more (green ≈6 dB, blue ≈ 20 dB lower than red)
- Also present in outdoor scenarios (FM radio at traffic light)

Measurements at only one fixed point not representative!

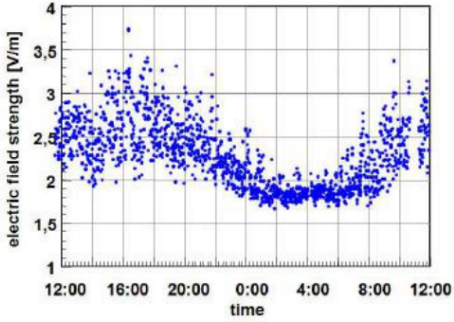
Small Variations in time



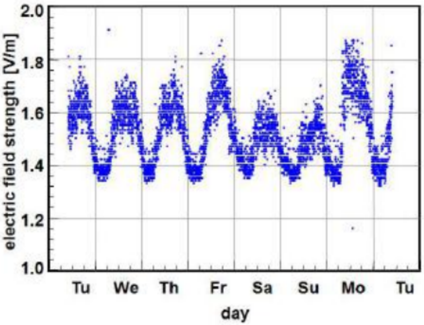
Peak-to-average power ratio:
PAPR = P_{Max}/P_{RMS}

Large scale variations in time

Measurement: GSM 24 hours

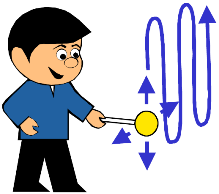


Measurement: GSM 7 days



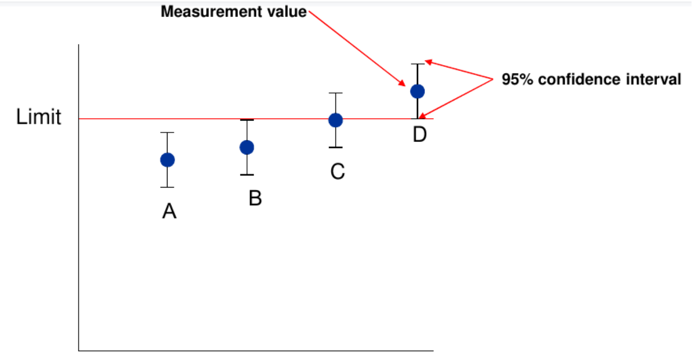
- Large scale variations due to traffic load (power control).
- Measurements at only one fixed time not representative!
- Measurements at arbitrarily chosen 6 min interval not representative!

Measuring procedures



- Volume scan with hand guided antenna
- Up to approx. 1.75...2.0 m height (standing person)
- Sampling on a (fine) grid geometries possible
- Take measurents during heavy traffic

Uncertainty calculation



- A: With a certainty of more than 97.5% the real exposure is below the limit.
- B: With a certainty of 97.5% the real exposure is below the limit.
- C: With a certainty of 50% the real exposure is below the limit.
- D: With a certainty of 97.5% the real exposure is above the limit.