loge Documentation

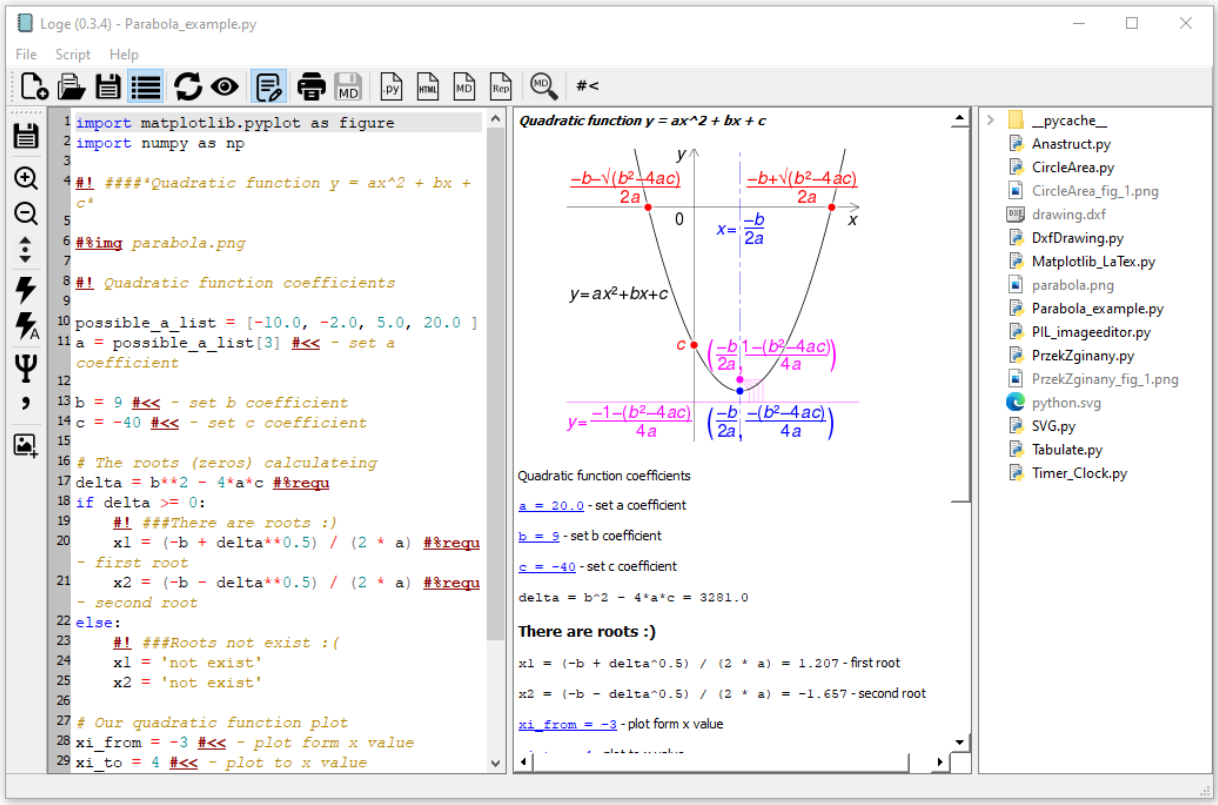
Release 0.1

Albert Defler, Łukasz Laba

Jan 10, 2023

Contents:

Easy and fast dynamic report generation with Python 3



CHAPTER 1

About Loge

1.1 What Loge is

Loge is a tool to create interactive report for scientific calculation you made with python script. You can use Loge to create reports ready for publication. Loge use .py file format. All additional Loge's syntax is hidden inside python comments so the python engine does not see it, this is still python standard code file you can execute. Loge comments is based on easy to use Markdown language syntax. Loge has its own code editor and file browser included but you can also use your favorite editor to edit script file and get live report preview when file changed.

Loge is based on Python3 and pyqt5.

Loge is continuation of SeePy project. SeePy is based on Python2 and pyqt4. SeePy project is is no longer developed.

1.2 Mission

The mission of the project is to provide a simple and practical tool that will interest engineers to use Python language in their daily work.

1.3 Stage of development

At the moment Loge is available on PyPI as beta stage software.

1.4 Source code

Source code git repository is available on Bitbucket.

1.5 License

1.6 Development team

The current co-lead Loge developers:

- Copyright (C) 2017-2022 Lukasz Laba <lukaszlaba@gmail.com>
- Copyright (C) 2017 Albert Defler <alnw@interia.eu>

CHAPTER 2

Installing

2.1 Loge requirements

To run Loge the following Python environment must be installed

1 - Python 3

2 - Minimal non-standard python dependencies

- pyqt5
- mistune
- pillow

3 - Optional non-standard python dependencies

- unum - SI unit calculation
- matplotlib - matplotlib and LaTex display
- svgwrite - SVG graphic display
- tabulate - working with nice looking text table
- dxf2svg - dxf file drawing display
- anastruct - 2D structure analysis

2.2 How to run Loge?

Loge is available through PyPI and can be install with pip command. To install Loge with minimal requirements use pip by typing:

```
pip install loge
```

To make all features available install optional dependencies by taping:

```
pip install unum matplotlib svgwrite tabulate dxf2svg anastruct
```

To run Loge use command `loge` from your system command line. Command `python -m loge` works as well.

On Windows system you can also find and run `loge.exe` file in `..\Python3\Scripts` folder. For easy run make shortcut for this file.

If new version of Loge package available upgrade it by typing

```
pip install --upgrade loge
```

2.3 OS compatibility

Windows (10) and Linux (xubuntu) tested.

CHAPTER 3

Features

Here is extra syntax you can use in your python script to get Loge report

CHAPTER 4

Comments

4.1 One line comment

(indentation acceptable)

```
#! Your Markdown comment
```

4.2 Multi line comment

(indentation not acceptable)

```
#!
'''
Your miltiline Markdown comment

you can write long text
'''
```

4.3 Variable with line comment

(indentation acceptable)

```
a = 30 #! Comment
```

or if a value defined use

```
a #! Comment
```

4.4 Calling variable value in Loge comment

You can call variable value in comments using `%(name)s` as it show below

```
a = 1
b = 2
#! Values are %(a)s and %(b)s
```

or you can use `val_name` and `var_name`

```
a = 1
b = 2
#! Values are val_a and val_b
#! Variables are var_a and var_b
```

CHAPTER 5

Python code

(indentation not acceptable)

5.1 Showing python code used in your script

You can show multi-line python code from your *.py script as it show below

```
#%code
text = 'Python is cool'
for i in text:
    print i
#%
```

or short syntax for one line code

```
text = 'Python is cool' #%code
```

CHAPTER 6

Images from file

(indentation acceptable)

6.1 Showing image in report

You can show any image file from directory where your *.py script is stored. Most image file format allowed (including SVG).

```
#%img image.jpg
```

6.2 Showing dxf file

You can also show graphic from dxf file. Python dxf2svg package converter is used to display dxf.

More about dxf2svg at https://bitbucket.org/lukaszlaba/dxf2svg

```
#%img drawing.dxf [framename] [500]
```

The integer number is image size you will get in report.

6.3 Embedding image from clipboard

The code editor toolbar include feature that allow embed image directly from clipboard. It transform clipboard data into image file and is save in the same location where the loge script is located. After image file is saved #%img commend is added to script.

CHAPTER 7

Image from PIL / PILLOW

(indentation acceptable)

7.1 Displaying PIL.Image instance in report

You can display image from PIL.Image instance using `#%pil` syntax.

```
from PIL import Image
imagefilepath = '/home/.../someimage.jpg' #<<<< Image path -
im = Image.open(imagefilepath) #%pil
im2 = im.resize((200,200)) #%pil
im3 = im.rotate(10) #%pil
im #%pil
```

CHAPTER 8

Matplotlib

(indentation acceptable)

8.1 Showing Matplotlib figure

You can add to Loge report Matplotlib figure - matplotlib.pyplot instance is needed

```
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(-1.0, 2.0, 0.01)
s1 = np.cos(9*np.pi*t) + 3 * t ** 2
plt.plot(t, s1)#%plt
plt.clf()
```

or you can use:

```
plt #%plt
```

CHAPTER 9

Anastruct

(indentation acceptable)

9.1 Showing Anastruct figure

Anastruct is a python package that allow analyse 2D frames and trusses. It determine the bending moments, shear forces, axial forces and displacements. You can add Anastruct figures to Loge report. Anastruct figures base on Matplotlib.

More about anastruct https://anastruct.readthedocs.io/ and https://github.com/ritchie46/anaStruct

```
from anastruct import SystemElements
ss = SystemElements()
import matplotlib.pyplot as plt
ss.add_element(location=[[0, 0], [3, 4]])
ss.add_element(location=[[3, 4], [8, 4]])
ss.add_support_hinged(node_id=1)
ss.add_support_fixed(node_id=3)
ss.q_load(element_id=2, q=-10)
ss.solve()

fig = ss.show_shear_force(show=False)
plt #%plt
plt.clf()
```

CHAPTER 10

Tabulate table

(indentation acceptable)

10.1 Showing Tabulate table

More about tabulate at https://github.com/astanin/python-tabulate

```
from tabulate import tabulate
table = [["spam",42],["eggs",451],["bacon",0]]
headers = ["item", "qty"]
tabulate_tab = tabulate(table, headers, tablefmt="fancy_grid") #%tab
```

or you can use:

```
tabulate_tab #%tab
```

CHAPTER 11

LaTex

(indentation acceptable)

11.1 Rendering LaTex syntax from comment

```
#%tex s(t) = \mathcal{A}\mathrm{sin}(2 \omega t)
```

you can call variables

```
a = 23
#%tex f(x) = %(a)s * y
```

11.2 Rendering python code as LaTex syntax

```
pi = 3.14 #! - pi value
r = 40 #! - circle radius
# from formula
Area = pi * r ** 2 #%tex
Area #! - what we get
```

11.3 Rendering LaTex syntax from python string

```
LaTexString = '\lim_{x \to \infty} \exp(-x) = 0'
LaTexString #%stringtex
```

CHAPTER 12

SVG graphic

(indentation acceptable)

12.1 Rendering SVG syntax from python string

```
svgsyntaxstring='''
<svg>
    <circle cx="30" cy="30" r="20" fill="tan" />
</svg>
'''
svgsyntaxstring #%svg
```

12.2 Rendering SVG `svgwrite.drawing` instance from `svgwrite` package

More about svgwrite at https://svgwrite.readthedocs.io

```
import svgwrite
svg_document = svgwrite.Drawing()
svg_document.add(svg_document.rect(size = (40, 40), fill = "tan"))
svg_document #%svg
```

CHAPTER 13

Raport interaction

(indentation acceptable)

13.1 Interactive python variable changing

```
a = 120 #! - this is not interactive variable in your report
b = 30 #<< - this is interactive variable in your report click it to change it
#! the values are %(a)s and %(b)s
```

You can get other display effect using #<<, #<<< or #<<<<

```
b = 30 #<< your comment
b = 30 #<<< your comment
b = 30 #<<<< your comment
```

Special cases

- If your variable value is True or False then interactive CheckBox will be displayed on report when #<<< or #<<<< used. When you click CheckBox the value will change to opposite bool value.

```
dosomething = True #<<<< Do something
if dosomething:
    a = 1
    b = 3
    c = a + b #%requ
    #! Done ...
```

- If you variable name has 'filepath' or 'dirpath' in it name then file or directory browse dialog will be open after clicked.

```
image_filepath = '/home/image.ipg' #<<<< Image file path to open -
image_filepath #! - this is your path
```

(continues on next page)

(continued from previous page)

```
search_dirpath = '/home' #<<<< Directory to scan for data -
search_dirpath #! - your is your path
```

13.2 Interactive python variable selecting from list

If your variable is equal some list element

```
list = [1, 2, 3, 4]
variable = list[1]
```

You can make this choice interactive

```
list = [1, 2, 3, 4]
variable = list[1] #<< - select variable value
```

You can use #<<, #<<< or #<<<< to get different display effect

```
list = [1, 2, 3, 4]
variable = list[1] #<< - select variable value
variable = list[1] #<<< - select variable value
variable = list[1] #<<<< - select variable value
```

Examples of use

Example 1

```
car_list = ['volvo', 'toyota', 'saab', 'fiat']
your_car = car_list[1] #<<< - select your car
#! Your car is %(your_car)s .
```

Example 2

```
material_list = ['steel', 'concrete', 'plastic', 'wood']
material = material_list[1] #<<<< Material is -
#! The %(material)s will be used to make something.
```

Example 3

```
temperature_range = range(10,30,1)
room_temperature = temperature_range[2] #<<<< Select the room temperature -
#! Temperature %(room_temperature)s Celsius degree selected.
```

CHAPTER 14

Mathematical expressions and equations

(indentation acceptable)

You can add to Loge report expressions and equations with optional comment using `#%requ` or `#%equ` (first give result value, second not)

```
a = 1
b = 3
c = 3*a + 4*b #%requ - your comment
c = 3*a + 4*b #%requ
c = 3*a + 4*b #%equ - your comment
c = 3*a + 4*b #%equ
3*a + 4*b #%requ - your comment
3*a + 4*b #%requ
3*a + 4*b #%equ - your comment
3*a + 4*b #%equ
```

Python code is format to look more natural

- when you working with math package the `math.` prefix be deleted
- python power sign `**` is change to `^`

```
import math
math.pi #%equ
math.sin(1) #%equ
2**2 #%equ
```

When you working with SI units using Unum package

- `u.m` change to `[m]` e.t.c
- `(mathexpression).asUnit()` change to `mathexpression`

```
from unum import units as u
5 * u.m + 10 * u.mm #%requ
(5 * u.m + 10 * u.mm).asUnit(u.km) #%requ
```

CHAPTER 15

Loge timer

You can run timer option to run your script regularly every specifed time space. This option is available from Timer toolbar (time unit in those toolbar is second). Here is a script that show current time on report - try use timer for it.

```
import time
timestring = time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime())
#! ##The time is val_timestring
```

If you want to make timer on when script is open specify timer parameter in script:

```
import time
timestring = time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime())
#! ##The time is val_timestring
#%timer 300 ON
```

Specified time is 300 millisecond and timer will be ON.

If you want set timer but not make it ON after opening the script:

```
import time
timestring = time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime())
#! ##The time is val_timestring
#%timer 300 OFF
```

CHAPTER 16

Saveas with dependences

If you use saveas option Loge will save copy of your python script and other files if those names are linked inside script code(for example images you has linked with `#%img`).

CHAPTER 17

Code editor Greek letters usage

You can simply paste the needed unicode characters into your code. For Greek letters there is a build in feature included. To write Greek letter, write its roman equivalent and when the cursor is after the letter use Ctrl+G shortcut. Here is the list of available Greek letters.

Name	Uppercase	Lowercase	Roman equivalent
alpha	Α	α	A
beta	Β	β	B
chi	Χ	χ	C
delta	Δ	δ	D
epsilon	Ε	ε	E
eta	Η	η	H
gamma	Γ	γ	G
iota	Ι	ι	I
kappa	Κ	κ	K
lambda	Λ	λ	L
mu	Μ	μ	M
nu	Ν	ν	N
omega	Ω	ω	W
omicron	Ο	ο	O
phi	Φ	φ	F
phi(alternate)		φ	J
pi	Π	π	P
psi	Ψ	ψ	Y
rho	Ρ	ρ	R
sigma	Σ	σ	S
tau	Τ	τ	T
theta	Θ	θ	Q
theta(alternate)	ϑ		J
upsilon	Υ	υ	U
xi	Ξ	ξ	X
zeta	Ζ	ζ	Z

alt text

CHAPTER 18

Loge in action

Here you can find a few Tebe use case examples - it does not contain all available features. There are other example scripts and tutorial included in Tebe.

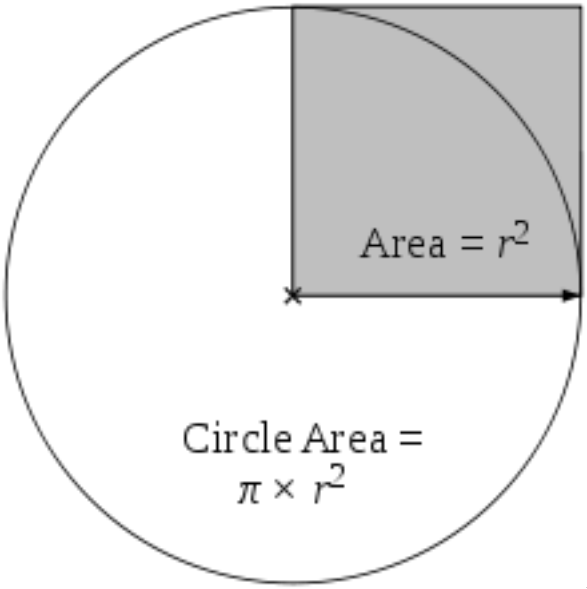
18.1 Circle area

Python code:

```
import math

#! ### Circle Area calculation

#%img action_circle_fig_1.png

#! For input data

r = 89 #<< - circle radius

Area = math.pi * r**2 #%requ - formula that everyone know

#! So, the area of circle with var_r is val_Area.
```

Loge output report:

18.1.1 Circle Area calculation



Alt text

For input data

`r = 89` - circle radius

`Area = pi * r^2 = 24884.5554` - formula that everyone know

So, the area of circle with `r = 89` is `24884.5554` .

18.2 Matplotlib figure

Python code:

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

t1 = -1.0 #! - start argument value for figures plots
t2 =  2.0 #! - end argument value for figures plots

t = np.arange(t1, t2, 0.01)

#! Figure 1
s1 = np.cos(9*np.pi*t) + 3 * t ** 2 #%equ
plt.figure(1)
plt.plot(t, s1) #%plt
plt.clf()

#! Figure 2
s2 = np.sin(18*np.pi*t**2)
plt.figure(2)
plt.plot(t, s2)
```

(continues on next page)

(continued from previous page)

```
plt #%plt
plt.clf()

#! All figures in one
plt.figure(4)
plt.plot(t, s1)
plt.plot(t, s2)
plt #%plt
plt.clf()
```

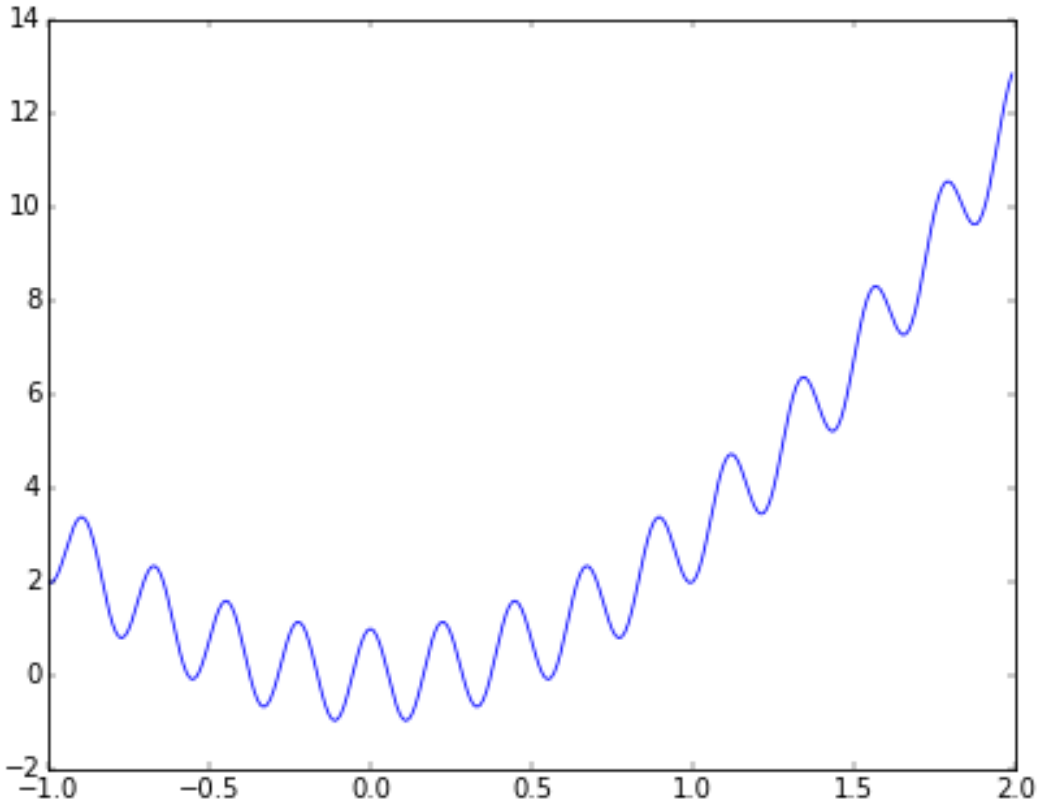
Loge output report:

`t1 = -1.0` - start argument value for figures plots

`t2 = 2.0` - end argument value for figures plots

Figure 1

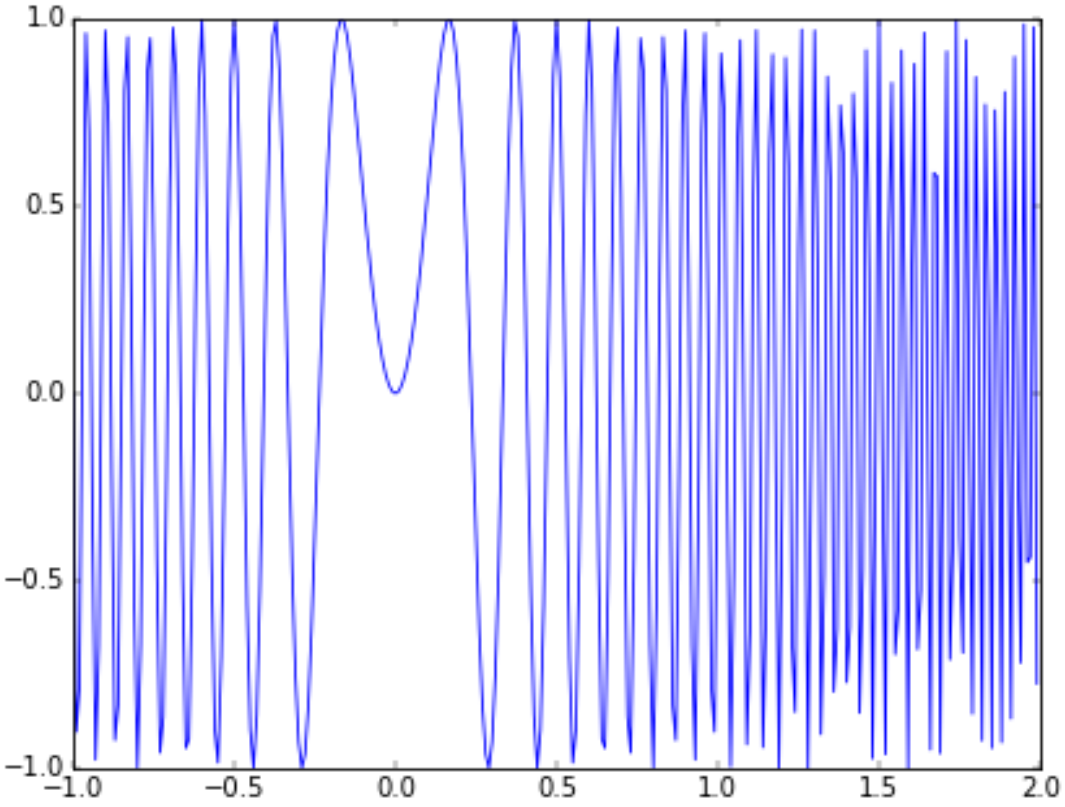
`s1 = np.cos(9*np.pi*t) + 3 * t ^ 2` - figure formula



Alt text

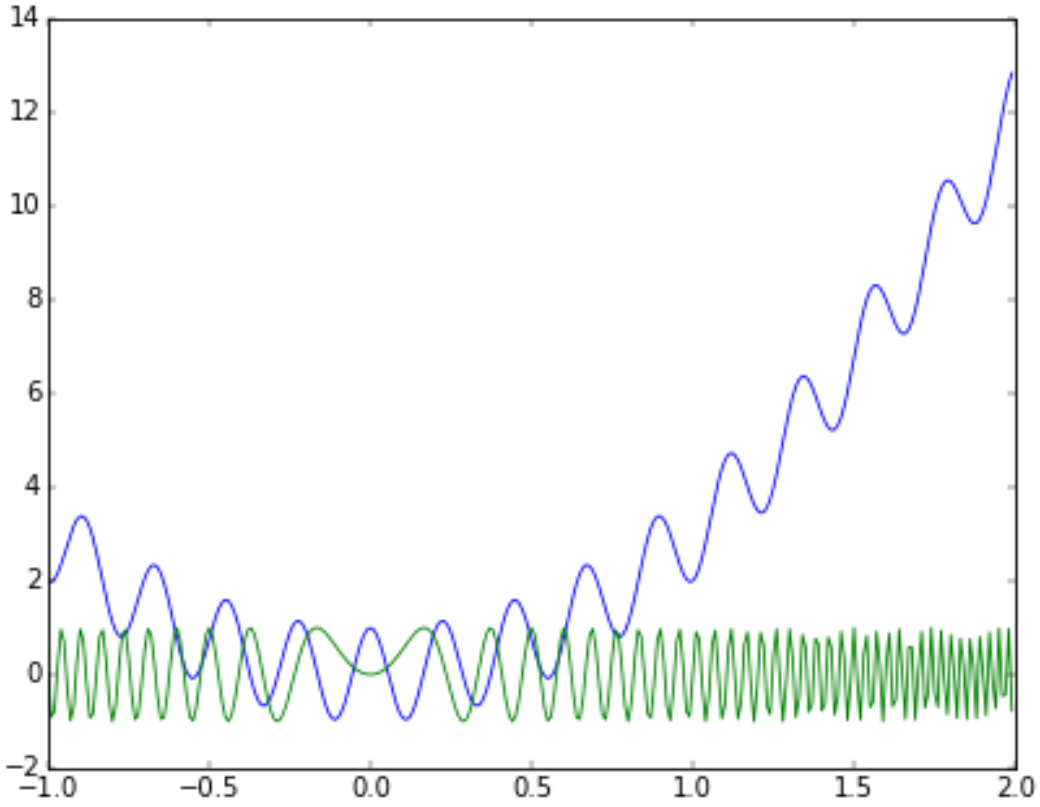
Figure 2

`s2 = np.sin(18*np.pi*t^2)` - figure formula



Alt text

All figures in one



Alt text

18.3 SI unit calculation

SI unit calculation can by made with Unum pagkage. Here you can find Unum documentation. Please note that Loge has some special features that help diplay Unum - you can find it in Features page.

Python code:

```
import unum.units as u

#! Stress calculation

#! Input data:
F = 1000.0*u.N #<< - force value
b = 20.0 * u.cm #<< - section width
h = 10.0 * u.cm #<< - section height

#! Result calculation:
A = b * h #%requ - section area
Sigma = (F / A).asUnit(u.Pa) #%requ - stress value
```

(continues on next page)

(continued from previous page)

```
#! So for section dimensions var_b , var_h and force var_F we get stress value val_
↪Sigma.
```

Loge output report:

Stress calculation

Input data:

`F = 1000.0 [N]` - force value

`b = 20.0 [cm]` - section width

`h = 10.0 [cm]` - section height

Result calculation:

`A = b * h = 200.0 [cm2]` - section area

`Sigma = F / A = 50000.0 [Pa]` - stress value

So for section dimensions `b = 20.0 [cm]`, `h = 10.0 [cm]` and force `F = 1000.0 [N]` we get stress value `50000.0 [Pa]`.

18.4 Foundation dynamic analysis

Python code:

```
# -*- coding: utf-8 -*-

import matplotlib.pyplot as plt
import numpy as np

import strupy.units as u
from unum.units import um
u.um = um

import math

#! ### Fundament blokowy po wentylator

show = True #<<< (zobacz podstawy teoretyczne)
if show:
    #%img action_foundation_fig_1.png
    None

#! ---

#! ### Dane wejściowe
#! Geometria fundamentu
a_f = 3.8 *  u.m #<< - szerokość fundamentu
```

(continues on next page)

(continued from previous page)

```
b_f = 5.20* u.m #<< - długość fundamentu
h_f = 1.4 * u.m #<< - wysokość fundamentu
m_f = a_f * b_f * h_f * 2100 * u.kg / u.m3 #! - masa fundamentu betonowego

#! Dane urzadzenia
m_u = 4640.0 * u.kg #<< - masa urzadzenia
n_m = 991.0 #<< - predkość obrotowa [rpm]
n_m / 60.0 #%requ - częstość w [Hz]
omega_m = n_m / (60.0 * u.s) * 2 * math.pi #! - predkość obrotowa [rad/s]
h_u = 3.2 * u.m #<< - wysokość srodka cęźkośći maszyny wzgl. spodu fund
P_d = 1.57 * u.kN #<< - siła dynamiczna
h_s = 3.2 * u.m #<< - wysokość przyłozenia siły wzgl. spodu fundamentu

#! Podłoże gruntowe
C_o = 18.0 * u.MPa / u.m #<< - dynamiczny wsp. podloża (wg. tab.1 PN)

#! ---
#! ### Obliczenia
M_c = m_u + m_f #! - masa calkowita
F = a_f * b_f #! - powierzchnia podstawy fundamentu
p = M_c / F * 10.0 * u.N / u.kg
p = p.asUnit(u.kPa) #! - nacisk statyczny na grunt
#! Współczynniki sztywności podłoża
C_z = C_o * (1 + 2*(a_f + b_f) / (u.m**-1 * F)) * (p/(20*u.kPa))**0.5 #! - dynamiczny␣
↪współczynnik podłoza
C_x = 0.7 * C_z #! - ddynamiczny współczynnik podłoza
C_fi = C_o * (1 + 2*(a_f + 3 * b_f) / (u.m**-1 * F)) * (p/(20*u.kPa))**0.5 #! -␣
↪dynamiczny współczynnik podłoża

#! Sztywność podłoża
K_z = C_z * F
K_z = K_z.asUnit(u.kN / u.m) #! - sztywność kier. z

K_x = C_x * F
K_x = K_x.asUnit(u.kN / u.m) #! - sztywność kier. x

I_y = a_f**3 * b_f / 12 #! - moment bezwladności podstawy

K_fi = I_y * C_fi
K_fi = K_fi.asUnit(u.kNm) #! - sztywność na obrót

z_k = (m_f * h_f/2 + m_u * h_u) / M_c #! - wysokość środka ciężkośći układu

#! Położenie punktu kontroli drgań
zp = 3.2 * u.m #<< wysokość od podstawy fundamentu
xp = 0.0 * u.m #<< odleglość w poziomie od osi fundamentu

delta = 0.1 #<< - błąd szacowania czestotliwości degań własnych
#! ### Kątowa częstość drgań pionowych

Lambda_z = (K_z / M_c)**0.5 #%requ - w [rad/s] (czestość maszyny var_omega_m)
Lambda_zmin = Lambda_z * (1 - delta) #%requ - dolna granica szacowanej czestotliwości
Lambda_zmax = Lambda_z * (1 + delta) #%requ - górna granica szacowanej czestotliwości
#! Szacowana czestość rezonansowa w przedziale val_Lambda_zmin - val_Lambda_zmax

A_oz = P_d / K_z
A_oz =A_oz.asUnit(u.um)#! - przemieszczenie pionowe pod dzialaniem statycznym sily Pd
```

(continues on next page)

(continued from previous page)

```
#! ### Katowa czestość drgan wahadlowych
Lambda_fi = (K_x * K_fi /( M_c * (K_x * z_k**2 + K_fi)) )**0.5 #! - w [rad/s]␣
↪(czestość maszyny var_omega_m)
Lambda_fimin = Lambda_fi * (1 - delta) #%requ - dolna granica szacowanej␣
↪czestotliwości
Lambda_fimax = Lambda_fi * (1 + delta) #%requ - górna granica szacowanej␣
↪czestotliwości
#! Szacowana czestość rezonansowa w przedziale val_Lambda_fimin - val_Lambda_fimax

A_o1 = P_d / K_x * (1 + K_x * h_s * zp / K_fi)
A_o1 = A_o1.asUnit(u.um)#! - przemieszczenie poziome pod dzialaniem statycznym sily Pd

A_o2 = P_d / K_fi * h_s * xp
A_o2 = A_o2.asUnit(u.um)#! - przemieszczenie pionowe pod dzialaniem statycznym sily Pd

#! ### Rodzaj strojenia
#! Ze wzgledu na czestotliwość var_Lambda_z
if  Lambda_zmin < omega_m < Lambda_zmax:
    #! ####!!!Praca w strefie rezonansu var_Lambda_zmin < var_omega_m < var_Lambda_
↪zmax !!!
    None
if  omega_m < Lambda_zmin:
    None
    #! OK - Strojenie wysokie var_omega_m < var_Lambda_zmin
if  Lambda_zmax < omega_m:
    None
    #! OK - Strojenie niskie var_Lambda_zmax < var_omega_m

#! Ze wzgledu na czestotliwość var_Lambda_fi
if  Lambda_fimin < omega_m < Lambda_fimax:
    #! ####!!!Praca w strefie rezonansu var_Lambda_fimin < var_omega_m < var_Lambda_
↪fimax !!!
    None
if  omega_m < Lambda_fimin:
    None
    #! OK - Strojenie wysokie var_omega_m < var_Lambda_fimin
if  Lambda_fimax < omega_m:
    None
    #! OK - Strojenie niskie var_Lambda_fimax < var_omega_m

show = True #<<< (zobacz wykres strojenia)
if show:
    x_max = 1.2*max(omega_m, Lambda_zmax, Lambda_fimax).asNumber()

    y1 = np.array([0, 0, 0.3, 0.3, 0, 0])
    x1 = np.array([0, Lambda_zmin.asNumber(), Lambda_zmin.asNumber(), Lambda_zmax.
↪asNumber(), Lambda_zmax.asNumber(), x_max])
    x1a = np.array([0, Lambda_z.asNumber(), Lambda_z.asNumber(), Lambda_z.asNumber(),␣
↪Lambda_z.asNumber(), x_max])

    y2 = y1
    x2 = np.array([0, Lambda_fimin.asNumber(), Lambda_fimin.asNumber(), Lambda_fimax.
↪asNumber(), Lambda_fimax.asNumber(), x_max])
    x2a = np.array([0, Lambda_fi.asNumber(), Lambda_fi.asNumber(), Lambda_fi.
↪asNumber(), Lambda_fi.asNumber(), x_max])
```

(continues on next page)

(continued from previous page)

```
    xm1 = np.arange(0.0, omega_m.asNumber(), 10.0)
    ym1 = 1.0 * (xm1 / omega_m.asNumber())**2
    ym2 = np.array([0, 0, 1.0, 0, 0])
    xm2 = np.array([0, omega_m.asNumber(), omega_m.asNumber(), omega_m.asNumber(), x_
↪max])

    plt.figure(1)
    plt.fill(x1, y1, alpha=0.3, label='szacowany obszar Lambda_z', color='blue')
    plt.plot(x1a, y1, label='Lambda_z', color='blue')


    plt.fill(x2, y2, alpha=0.3, label='szacowany obszar Lambda_fi', color='green')
    plt.plot(x2a, y2, label='Lambda_fi', color='green')


    plt.plot(xm1, ym1, label='sila wymuszajaca', color='yellow')
    plt.plot(xm2, ym2, label='omega_m', color='red')
    plt.legend()
    plt #%plt
    #plt.show()
    plt.clf()

#! ### Naprezenia pod fundamentem
p #! - nacisk statyczny na grunt od fundamentu i maszyny
sigma_pov = P_d / F
sigma_pov = sigma_pov.asUnit(u.kPa)#! - statyczny nacisk od sily wymuszajacej pionowo

sigma_poh = (P_d * h_s)/(a_f**2 * b_f / 6)
sigma_poh = sigma_poh.asUnit(u.kPa)#! - statyczny nacisk od sily wymuszajacej poziomo


#! ### Amplitudy drgań

gamma = 0.13 #! - tłumienie gruntu

#!-----------------------------------------------------------------------------
#! ###Praca na pełnych obrotach val_omega_m
#! Dopuśćzalne amplitudy drgań dla val_omega_m
omega_m / (2*math.pi) * 60.0 #%requ x [obr]
show = True #<<< (zobacz wykres PN)
if show:
    #%img action_foundation_fig_3.png
    None
A_xdop = 110.0 * u.um    #<< - poziomych
A_zdop = 70.0 * u.um   #<< - pionowych
#! #### Drgania pionowe
if omega_m < Lambda_zmin:
    Lambda_zmod = Lambda_zmin #%requ strojenie wysokie
if omega_m > Lambda_zmax:
    Lambda_zmod = Lambda_zmax #%requ strojenie niskie
if Lambda_zmin <= omega_m <= Lambda_zmax:
    Lambda_zmod = omega_m #%requ praca w strefie rezonansu!!!
    Lambda_zmod = Lambda_zmod * 0.999

eta_z = omega_m / Lambda_zmod #%requ

tlumienie = False
```

(continues on next page)

(continued from previous page)

```
if 0.75<eta_z<1.25:
    #! fundament pracuje w strefie rezonanasu
    tlumienie = True #<<< - dodatkwowo uzglednic tłumienie var_gamma (nie zalecane)

if tlumienie:
    ni = 1 / ((1 - eta_z**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
else:
    ni = 1 / ((1 - eta_z**2)**2)**0.5 #%requ - wzmocnienie

A_dz = ni * A_oz   #%requ - Amplituda drgań pionowych

#! #### Drgania wahadłowe
if omega_m < Lambda_fimin:
    Lambda_fimod = Lambda_fimin #%requ strojenie wysokie
if omega_m > Lambda_fimax:
    Lambda_fimod = Lambda_fimax #%requ strojenie niskie
if Lambda_fimin <= omega_m <= Lambda_fimax:
    Lambda_fimod = omega_m #%requ praca w strefie rezonansu!!!
    Lambda_fimod = Lambda_fimod * 0.999

eta_fi = omega_m / Lambda_fimod #%requ

tlumienie = False
if 0.75<eta_fi<1.25:
    #! fundament pracuje w strefie rezonanasu
    tlumienie = False #<<< - dodatkwowo uzglednic tłumienie var_gamma (nie zalecane)

if tlumienie:
    ni = 1 / ((1 - eta_fi**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
else:
    ni = 1 / ((1 - eta_fi**2)**2)**0.5 #%requ - wzmocnienie
A_do1 = ni * A_o1   #%requ - amplituda drgań poziomych
A_do2 = ni * A_o2   #%requ - amplituda drgań pionowych
#! #### Drgania całkowite
A_x = A_do1 #%requ - całkowita amplituda drgań poziomych var_A_xdop
A_z = A_dz + A_do2 #%requ - całkowita amplituda drgań pionowych var_A_zdop

if (A_x > A_xdop) or (A_z > A_zdop):
    None
    #! ### !! Przekroczone amplitudy drgań !!

if Lambda_zmax < omega_m:
    #!-------------------------------------------------------------------------------
    #! ### Rezonans przejśćiowy dla var_Lambda_zmax (drgania pionowe)
    omega_r = Lambda_zmax #!
    omega_r / (2*math.pi) * 60.0 #%requ x [obr]
    #! Dopuśćzalne amplitudy drgań dla val_omega_r
    show = True #<<< (zobacz wykres PN)
    if show:
        #%img action_foundation_fig_3.png
        None
    A_xdop = 120.0 * u.um #<< - poziomych
    A_zdop = 80.0 * u.um #<< - pionowych

    P_dred = P_d * (omega_r / omega_m)**2 #%requ -siła dynamiczna dla obrotów val_
↪omega_r
    #! Drgania pionowe
```

(continues on next page)

(continued from previous page)

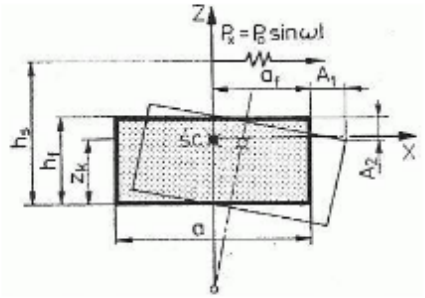
```
    eta_z = omega_r / (Lambda_z) #%requ
    ni = 1 / ((1 - eta_z**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
    A_dz = P_dred / P_d * ni * A_oz   #%requ - Amplituda drgań pionowych

    #! Drgania wahadłowe
    eta_fi = omega_r / (1.25* Lambda_fi) #%requ
    ni = 1 / ((1 - eta_fi**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
    A_do1 = P_dred / P_d * ni * A_o1   #%requ - amplituda drgań poziomych
    A_do2 = P_dred / P_d * ni * A_o2   #%requ - amplituda drgań pionowych
    #! Drgania całkowite
    A_x = A_do1 #%requ - całkowita amplituda drgań poziomych var_A_xdop
    A_z = A_dz + A_do2 #%requ - całkowita amplituda drgań pionowych var_A_zdop

    if (A_x > A_xdop) or (A_z > A_zdop):
        None
        #! ### !! Przekroczone amplitudy drgań !!

if Lambda_fimax < omega_m:
    #!-------------------------------------------------------------------------------
    #! ### Rezonans przejściowy dla Lambda_fimax (drgania wahadłowe)
    omega_r = Lambda_fi #!
    omega_r / (2*math.pi) * 60.0 #%requ x [obr]
    #! Dopuśćzalne amplitudy drgań dla val_omega_r
    show = True #<<< (zobacz wykres PN)
    if show:
        #%img action_foundation_fig_3.png
        None
    A_xdop = 140.0 * u.um  #<< - poziomych
    A_zdop = 100.0 * u.um  #<< - pionowych

    P_dred = P_d * (omega_r / omega_m)**2 #%requ -siła dynamiczna dla obrotów val_
↪omega_r
    #! Drgania pionowe
    eta_z = omega_r / (0.75 * Lambda_z) #%requ
    ni = 1 / ((1 - eta_z**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
    A_dz = P_dred / P_d * ni * A_oz   #%requ - Amplituda drgań pionowych

    #! Drgania wahadłowe
    eta_fi = omega_r / (Lambda_fi) #%requ
    ni = 1 / ((1 - eta_fi**2)**2 +  gamma**2)**0.5 #%requ - wzmocnienie
    A_do1 = P_dred / P_d * ni * A_o1   #%requ - amplituda drgań poziomych
    A_do2 = P_dred / P_d * ni * A_o2   #%requ - amplituda drgań pionowych
    #! Drgania całkowite
    A_x = A_do1 #%requ - całkowita amplituda drgań poziomych var_A_xdop
    A_z = A_dz + A_do2 #%requ - całkowita amplituda drgań pionowych var_A_zdop

    if (A_x > A_xdop) or (A_z > A_zdop):
        None
        #! ### !! Przekroczone amplitudy drgań !!

#! ---
show = True #<<< (pomocnicze - zobacz wykres wsp. dynamicznego)
if show:
    #%img action_foundation_fig_2.png
    None
```

Loge output report:

18.4.1 Fundament blokowy po wentylator

(zobacz podstawy teoretyczne)

Drgania wahadłowe Lipiński tab. 2-11



$$A_1 = A_1^0 v_1 = \frac{P_0}{K_x}\left(1 + \frac{K_x h_s h_f}{K_\varphi}\right) v_1 \qquad [2\text{-}92]$$

$$A_2 = A_2^0 v_1 = \frac{P_0}{K_\varphi} h_s a_f v_1 \qquad [2\text{-}93]$$

$$\lambda_1 = \sqrt{\frac{K_x K_\varphi}{m(K_x z_k^2 + K_\varphi)}} = \varkappa \lambda_x,$$

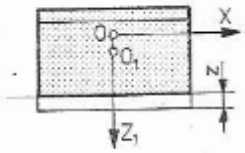
v_z — wg wzoru [2-90];

gdzie $\eta_1 = \dfrac{\omega}{\lambda_1}$

$$v_1 = \frac{1}{\sqrt{(1-\eta_1^2)^2 + \gamma^2 \eta_1^2}}$$

Drgania pionowe Lipiński tab. 2-8

1. Drgania własne pionowe



Równanie drgań własnych pionowych

$$m\ddot{z} + K_z z = 0$$

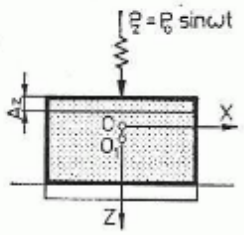
lub

$$\ddot{z} + \lambda_z^2 z = 0$$

Prędkość kątowa drgań własnych pionowych

$$\lambda_z = \sqrt{\frac{K_z}{m}}$$

1. Drgania pionowe (przesuwne)



Równanie drgań wymuszonych układu

$$m\ddot{z} + K_z z = P_{0z} \sin\omega t \qquad [2\text{-}76]$$

Amplituda drgań wymuszonych

$$A_z = \frac{P_{0z}}{K_z - m\omega^2} = \frac{P_{0z}}{m(\lambda_z^2 - \omega^2)} = A_{0z} v \qquad [2\text{-}77]$$

gdzie:

$A_{0z} = \dfrac{P_{0z}}{K_z}$ — przemieszczenie układu pod statycznym działaniem siły P_{0z}

$v = \left|\dfrac{1}{1-\eta^2}\right|$ — współczynnik dynamiczny

$\eta = \dfrac{\omega}{\lambda_z}$ — stosunek częstości wzbudzającej do własnej pionowej

Alt text

18.4.2 Dane wejściowe

Geometria fundamentu

`a_f = 3.80 [m]` - szerokość fundamentu

`b_f = 5.20 [m]` - dlugość fundamentu

`h_f = 1.40 [m]` - wysokość fundamentu

`m_f = 58094.40 [kg]` - masa fundamentu betonowego

Dane urzadzenia

`m_u = 4640.00 [kg]` - masa urzadzenia

`n_m = 991.0` - predkość obrotowa [rpm]

`n_m / 60.0 = 16.5167` - częstość w [Hz]

`omega_m = 103.78 [1/s]` - predkość obrotowa [rad/s]

`h_u = 3.20 [m]` - wysokość srodka cęźkośći maszyny wzgl. spodu fund

`P_d = 1.57 [kN]` - siła dynamiczna

`h_s = 3.20 [m]` - wysokość przyłozenia siły wzgl. spodu fundamentu

Podloże gruntowe

`C_o = 18.00 [MPa/m]` - dynamiczny wsp. podloża (wg. tab.1 PN)

18.4.3 Obliczenia

`M_c = 62734.40 [kg]` - masa calkowita

`F = 19.76 [m2]` - powierzchnia podstawy fundamentu

`p = 31.75 [kPa]` - nacisk statyczny na grunt

Współczynniki sztywności podłoża

`C_z = 43.34 [MPa/m]` - dynamiczny współczynnik podłoza

`C_x = 30.34 [MPa/m]` - ddynamiczny współczynnik podłoza

`C_fi = 67.21 [MPa/m]` - dynamiczny współczynnik podłoża

Sztywność podłoża

`K_z = 856345.26 [kN/m]` - sztywność kier. z

`K_x = 599441.68 [kN/m]` - sztywność kier. x

`I_y = 23.78 [m4]` - moment bezwladności podstawy

`K_fi = 1598099.91 [kNm]` - sztywność na obrót

`z_k = 0.88 [m]` - wysokość środka cięźkośći układu

Połozenie punktu kontroli drgań

`zp = 3.20 [m]` wysokość od podstawy fundamentu

`xp = 0.00 [m]` odleglość w poziomie od osi fundamentu

`delta = 0.1` - błąd szacowania czestotliwości degań własnych

18.4.4 Kątowa częstość drgań pionowych

`Lambda_z = (K_z / M_c)^0.5 = 116.83 [1/s]` - w [rad/s] (czestość maszyny `omega_m = 103.78 [1/s]`)

`Lambda_zmin = Lambda_z * (1 - delta) = 105.15 [1/s]` - dolna granica szacowanej czestotliwości

`Lambda_zmax = Lambda_z * (1 + delta) = 128.52 [1/s]` - górna granica szacowanej czestotliwości

Szacowana czestość rezonansowa w przedziale `105.15 [1/s]` - `128.52 [1/s]`

`A_oz = 1.83 [um]` - przemieszczenie pionowe pod dzialaniem statycznym sily Pd

18.4.5 Kątowa czestość drgan wahadlowych

`Lambda_fi = 85.94 [1/s]` - w [rad/s] (czestość maszyny `omega_m = 103.78 [1/s]`)

`Lambda_fimin = Lambda_fi * (1 - delta) = 77.35 [1/s]` - dolna granica szacowanej czestotliwości

`Lambda_fimax = Lambda_fi * (1 + delta) = 94.54 [1/s]` - górna granica szacowanej czestotliwości

Szacowana czestość rezonansowa w przedziale `77.35 [1/s]` - `94.54 [1/s]`

`A_o1 = 12.68 [um]` - przemieszczenie poziome pod dzialaniem statycznym sily Pd

`A_o2 = 0.00 [um]` - przemieszczenie pionowe pod dzialaniem statycznym sily Pd

18.4.6 Rodzaj strojenia

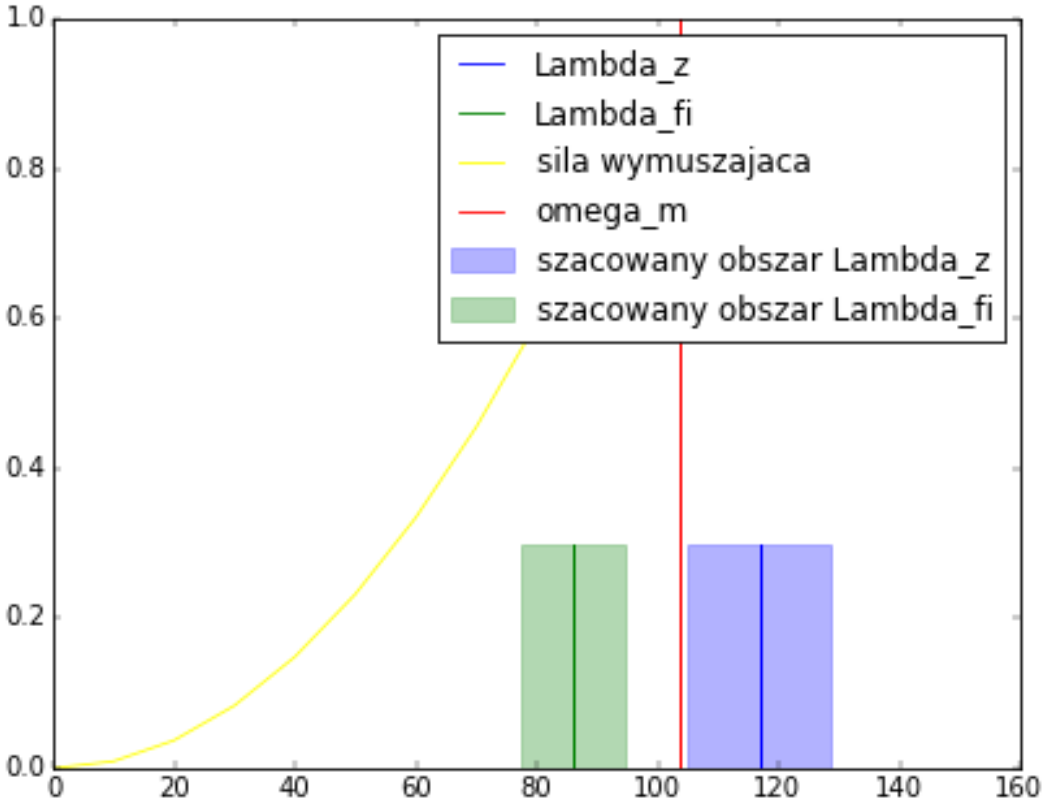
Ze wzgledu na czestotliwość `Lambda_z = 116.83 [1/s]`

OK - Strojenie wysokie `omega_m = 103.78 [1/s]` < `Lambda_zmin = 105.15 [1/s]`

Ze wzgledu na czestotliwość `Lambda_fi = 85.94 [1/s]`

OK - Strojenie niskie `Lambda_fimax = 94.54 [1/s]` < `omega_m = 103.78 [1/s]`

(zobacz wykres strojenia)



Alt text

18.4.7 Naprezenia pod fundamentem

`p = 31.75 [kPa]` - nacisk statyczny na grunt od fundamentu i maszyny

`sigma_pov = 0.08 [kPa]` - statyczny nacisk od sily wymuszajacej pionowo

`sigma_poh = 0.40 [kPa]` - statyczny nacisk od sily wymuszajacej poziomo

18.4.8 Amplitudy drgań

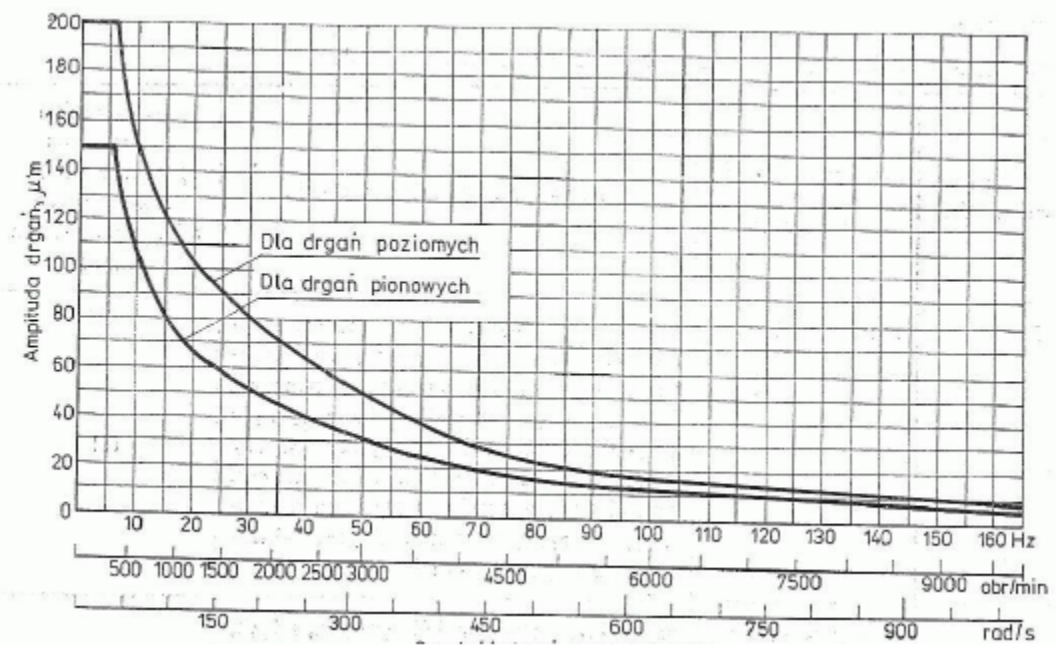
`gamma = 0.13` - tłumienie gruntu

18.4.9 Praca na pełnych obrotach `103.78 [1/s]`

Dopuśćzalne amplitudy drgań dla `103.78 [1/s]`

`omega_m / (2*pi) * 60.0 = 991.00 [1/s]` x [obr]

(zobacz wykres PN)



Alt text

`A_xdop = 110.00 [um]` - poziomych

`A_zdop = 70.00 [um]` - pionowych

Drgania pionowe

`Lambda_zmod = Lambda_zmin = 105.15 [1/s]` strojenie wysokie

`eta_z = omega_m / Lambda_zmod = 0.99 []`

fundament pracuje w strefie rezonanasu

- dodatkwowo uzglednic tłumienie `gamma = 0.13` (nie zalecane)

`ni = 1 / ((1 - eta_z^2)^2 + gamma^2)^0.5 = 7.54 []` - wzmocnienie

`A_dz = ni * A_oz = 13.83 [um]` - Amplituda drgań pionowych

Drgania wahadłowe

`Lambda_fimod = Lambda_fimax = 94.54 [1/s]` strojenie niskie

`eta_fi = omega_m / Lambda_fimod = 1.10 []`

fundament pracuje w strefie rezonanasu

- dodatkwowo uzglednic tłumienie `gamma = 0.13` (nie zalecane)

`ni = 1 / ((1 - eta_fi^2)^2)^0.5 = 4.88 []` - wzmocnienie

`A_do1 = ni * A_o1 = 61.82 [um]` - amplituda drgań poziomych

`A_do2 = ni * A_o2 = 0.00 [um]` - amplituda drgań pionowych

Drgania całkowite

`A_x = A_do1 = 61.82 [um]` - całkowita amplituda drgań poziomych `A_xdop = 110.00 [um]`

`A_z = A_dz + A_do2 = 13.83 [um]` - całkowita amplituda drgań pionowych `A_zdop = 70.00 [um]`

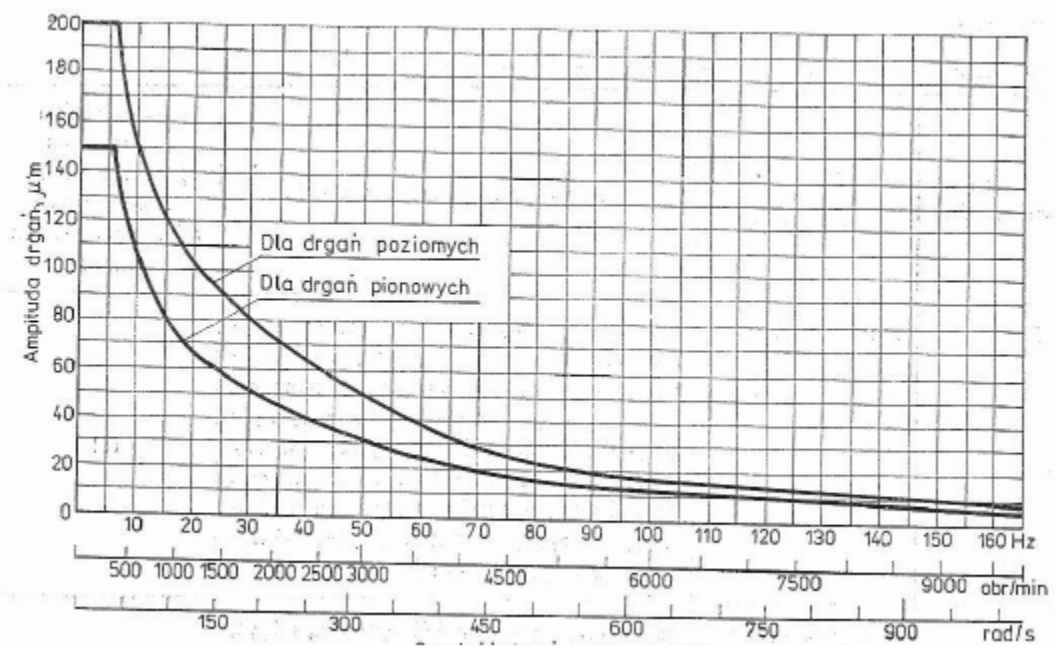
18.4.10 Rezonans przejśćiowy dla Lambda_fimax (drgania wahadłowe)

`omega_r = 85.94 [1/s]`

`omega_r / (2*pi) * 60.0 = 820.68 [1/s]` x [obr]

Dopuśćzalne amplitudy drgań dla `85.94 [1/s]`

(zobacz wykres PN)



Alt text

`A_xdop = 140.00 [um]` - poziomych

`A_zdop = 100.00 [um]` - pionowych

`P_dred = P_d * (omega_r / omega_m)^2 = 1.08 [kN]` -siła dynamiczna dla obrotów `85.94 [1/s]`

Drgania pionowe

`eta_z = omega_r / (0.75 * Lambda_z) = 0.98 []`

`ni = 1 / ((1 - eta_z^2)^2 + gamma^2)^0.5 = 7.38 []` - wzmocnienie

`A_dz = P_dred / P_d * ni * A_oz = 9.28 [um]` - Amplituda drgań pionowych

Drgania wahadłowe

`eta_fi = omega_r / (Lambda_fi) = 1.00 []`

`ni = 1 / ((1 - eta_fi^2)^2 + gamma^2)^0.5 = 7.69 []` - wzmocnienie

`A_do1 = P_dred / P_d * ni * A_o1 = 66.89 [um]` - amplituda drgań poziomych

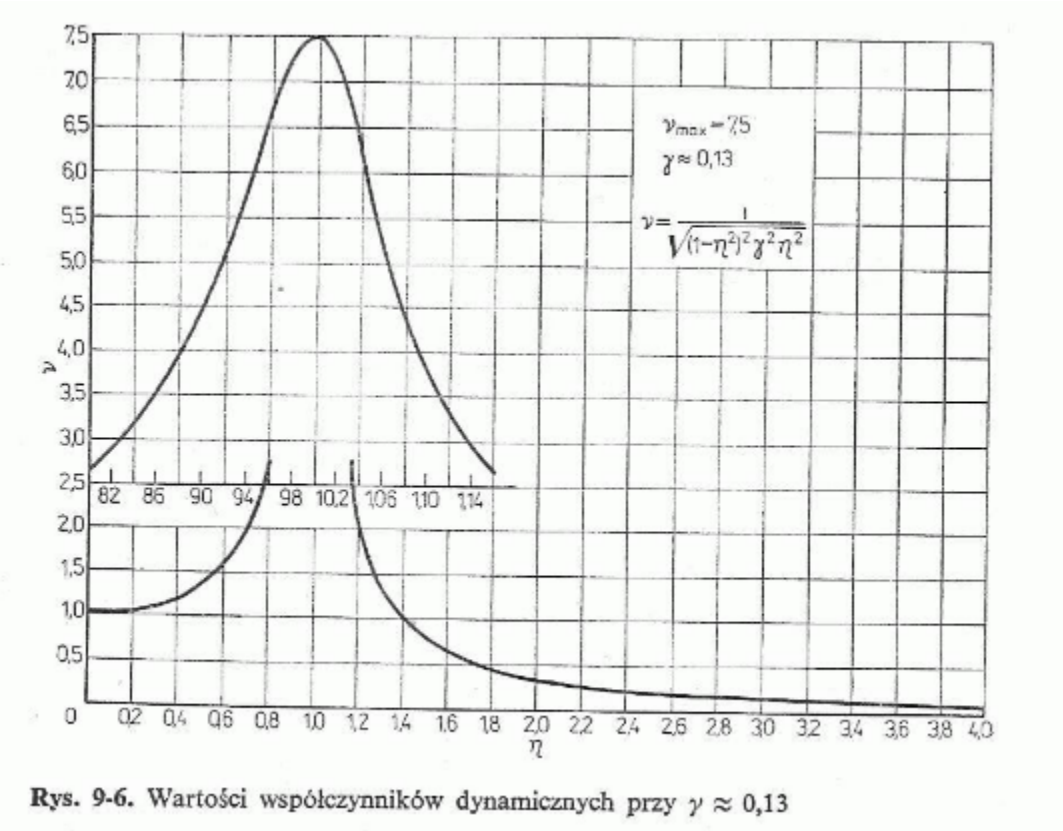
`A_do2 = P_dred / P_d * ni * A_o2 = 0.00 [um]` - amplituda drgań pionowych

Drgania całkowite

`A_x = A_do1 = 66.89 [um]` - całkowita amplituda drgań poziomych `A_xdop = 140.00 [um]`

`A_z = A_dz + A_do2 = 9.28 [um]` - całkowita amplituda drgań pionowych `A_zdop = 100.00 [um]`

(pomocnicze - zobacz wykres wsp. dynamicznego)



Rys. 9-6. Wartości współczynników dynamicznych przy $\gamma \approx 0{,}13$

Alt text

18.5 Video examples

Here you can find some youtube videos for Loge.

CHAPTER 19

Indices and tables

- genindex
- modindex
- search