Универзитет у Београду
Електротехнички факултет

Рад са језицима Micro-Cap и SymPy

Пројектни рад

Ментор:

Кандидат:

Драгана Нинковић
2019/0052 ОС

Ана Ћирковић

2019/0119 ОС

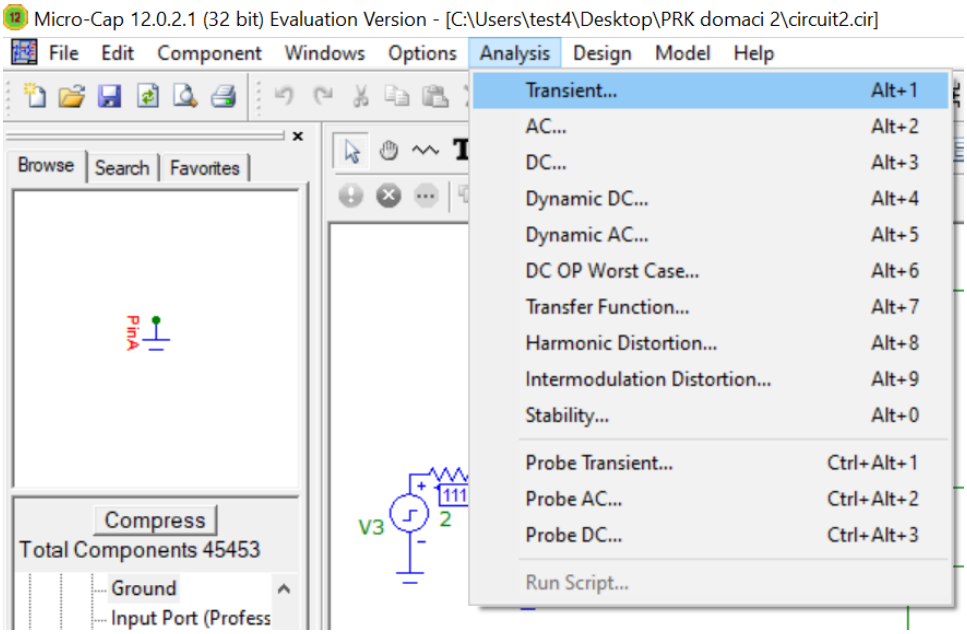
Београд, Новембар 2020.

Садржај

Анализа кола у Micro-Cap-у

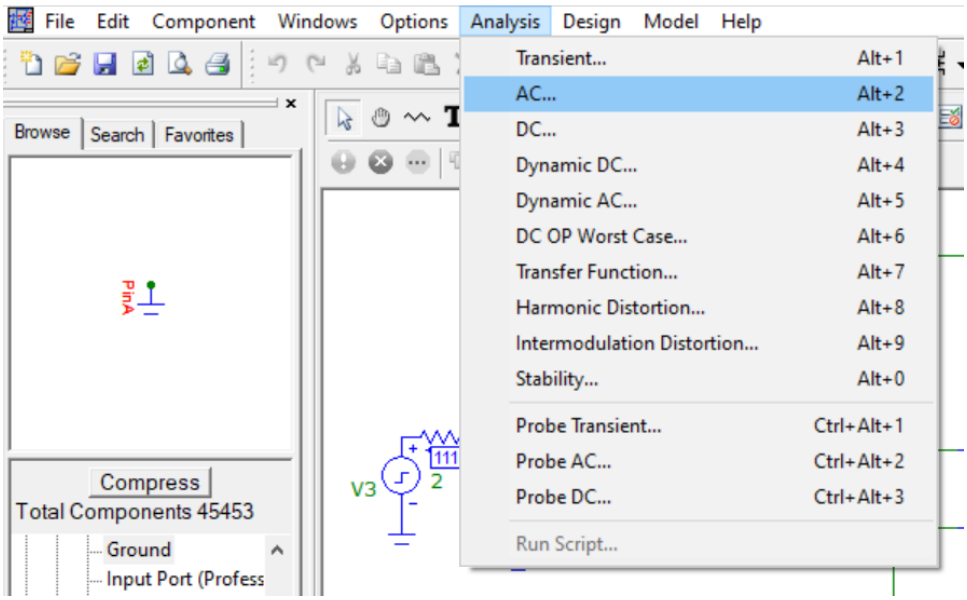
У задатку 1 (налази се на страни 6) су за анализу кола коришћене транзијентна анализа и АС анализа. Транзијентна анализа је коришћена за импулсни и одскочни одзив док је АС анализа коришћена за амплитудску и фазну карактеристику.

За одкочни одзив се усваја да је подуба Хевисајдова. Хевисајдова функција је реализована у Micro-Cap-у помоћу pulse-а са одговарајућим карактеристикама. За импулсни одзиве се усваја Диракова побуда. Диракова функција је у Micro-Cap-у реализована као правоугаони импулс.
За оба одзива је коришћена транзијентна анализа која као излазни резултат даје промену напона у времену, а приликом анализе рачуна напон у гранама односно чворовима.



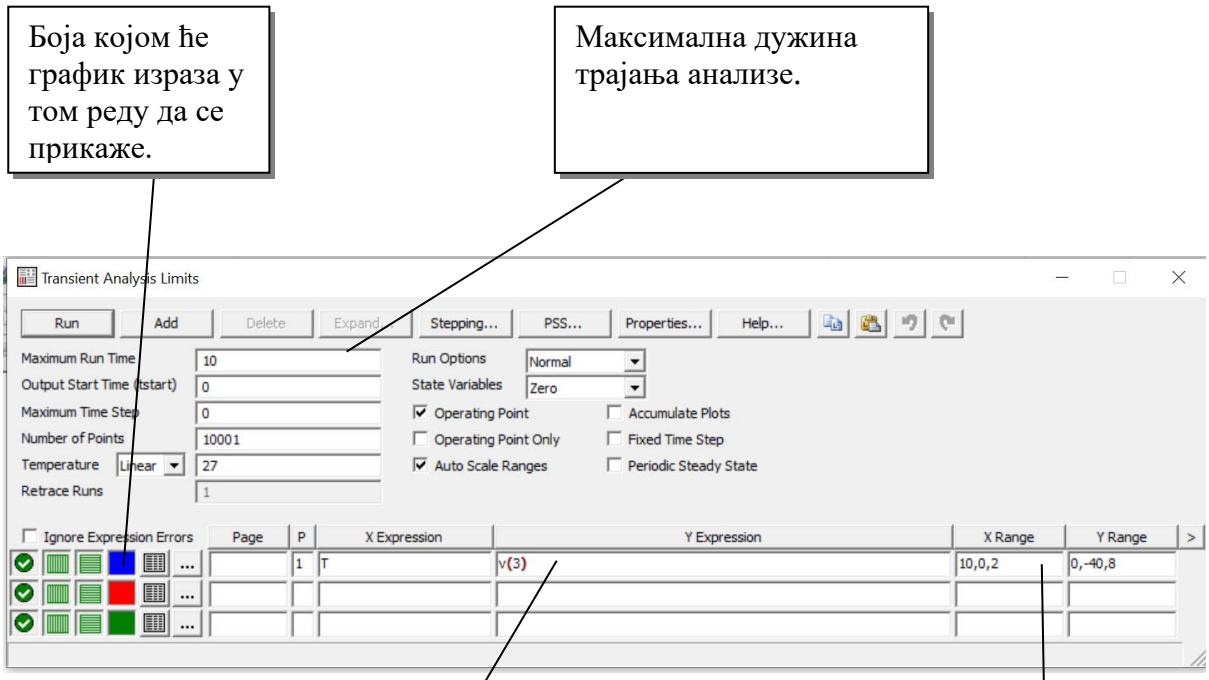
Слика 1: Транзијантна анализа

Амплитудски одзив је модул фреквенцијског одзива,а импулсни одзив је аргумент фреквенцијског одзива. За оба одзива се користи АС анализа која приказује понашање амплитуде и фазе сигнала у зависности од фреквенције. Она омогућава истраживање понашања малих сигнала.



Слика 2: AC анализа

На слици је приказан прозор који се појави када покренемо транзијентну анализу. Преко прозора могу да се подесе x и y осе графика, као и време трајања анализе.



Слика 3: Покретање транзијентне анализе

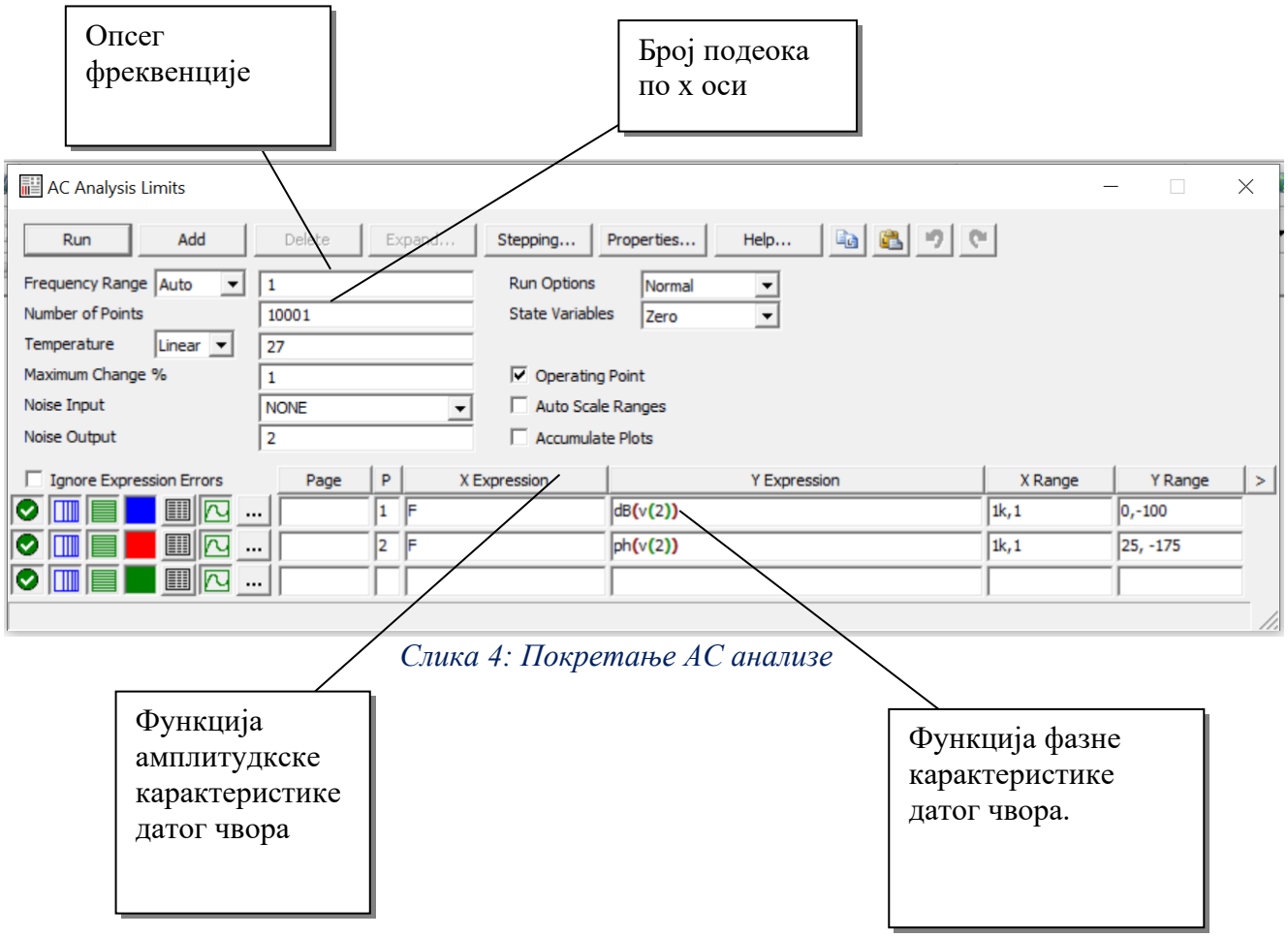
Боја којом ће график израза у том реду да се прикаже.

Максимална дужина трајања анализе.

Место где уписујемо потенцијал односно струју са шеме чији график желимо да прикажемо. Такође се могу наћи и изрази који су функције од потенијала и струја.

Подешавање x и y осе графика. Могуће је ставити три параметра. Први параметар представља максималну вредност на датој оси, други параметар представља минималну вредност, а трећи параметар. Уколико се на стави број подеока програм ће сам поделити.

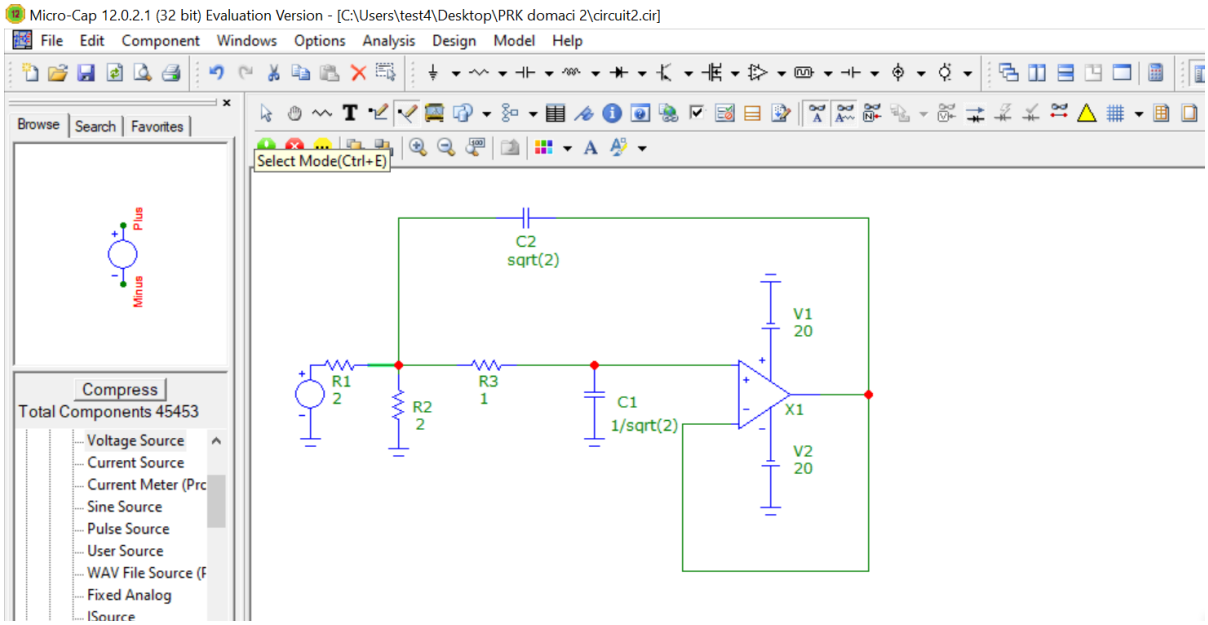
На слици је приказан прозор који се појави када покренемо AC анализу.



Слика 4: Покретање AC анализе

Фаза се исписује у степенима, а не у радијаниима.

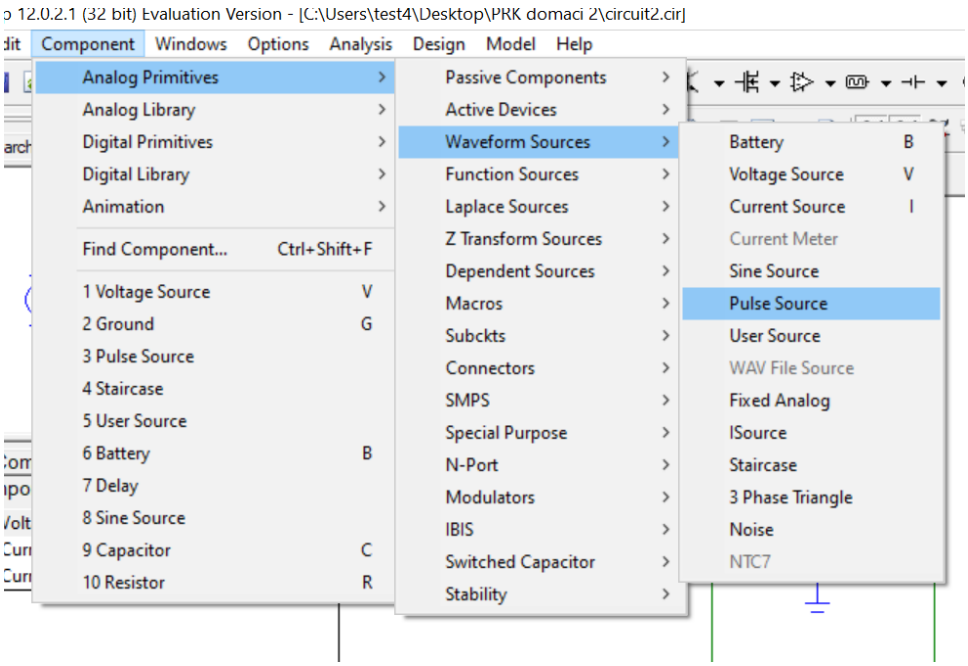
Задатак 1



Слика 5: Приказ задатог кола у Micro-Cap

На слици изнад у Micro Cap-у је приказано коло са Sallen & Key LP-LQ активним филтром

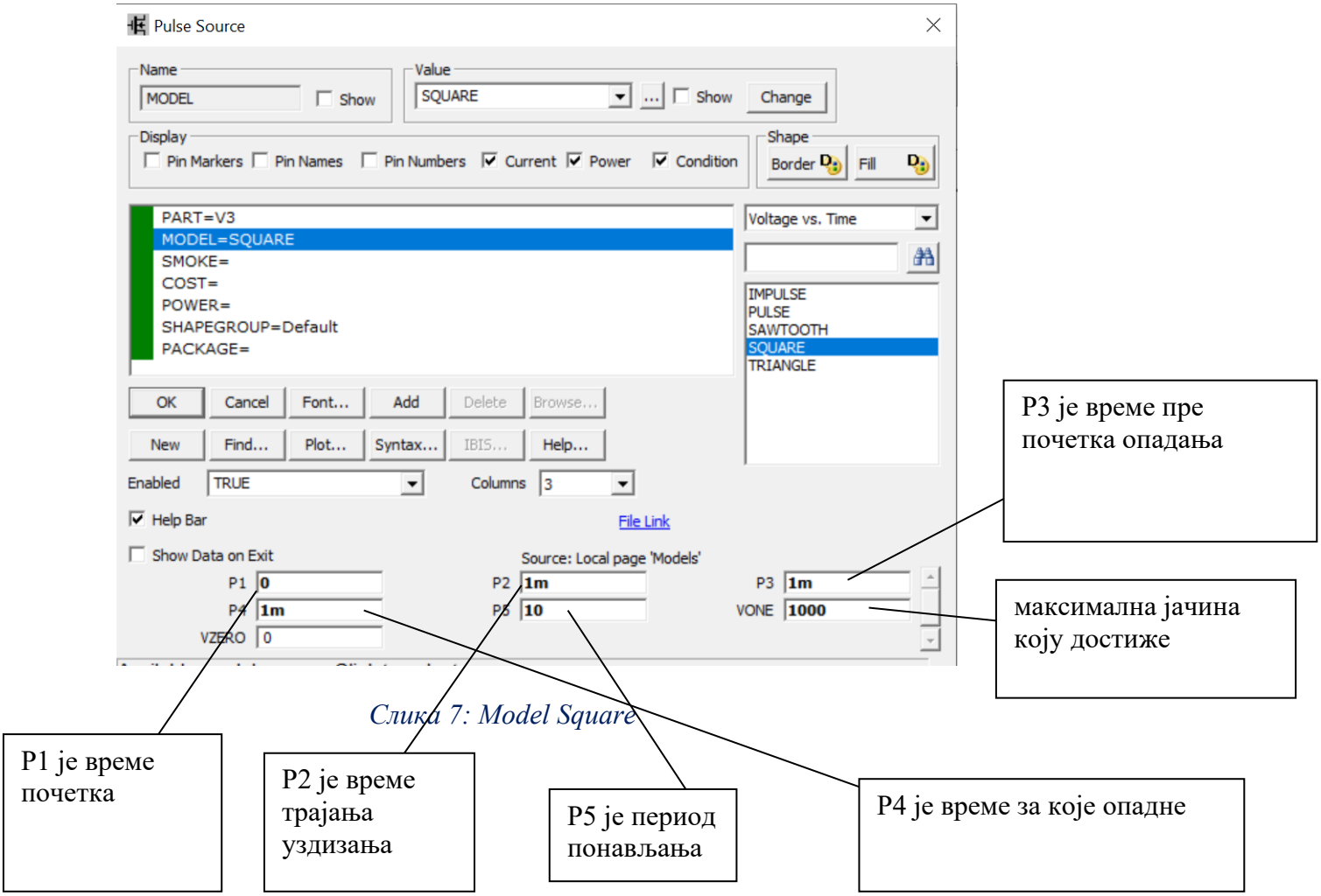
За импулсни и одскочни одзив користити Pulse Source. На наредној слици је приказан начин на који се може доћи до Pulse Sourca



Слика 6: Pulse Source

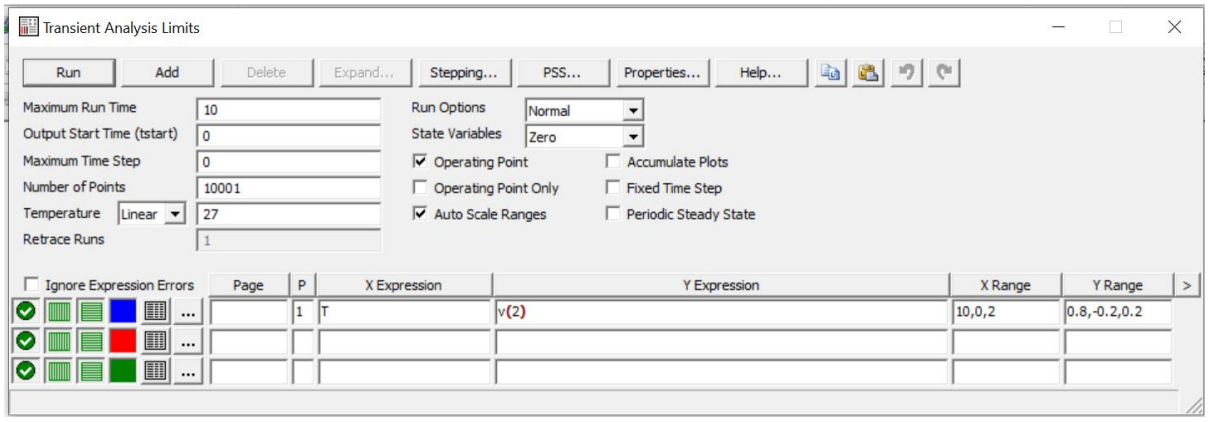
а) импулсни и одскочни одзив

За импулсни одзив користити модел генератора Square. На наредној слици приказано подешавање модела. Приметимо да је 1m*1000 = 1 чиме је испуњено да је површина правоугаоника једнака.



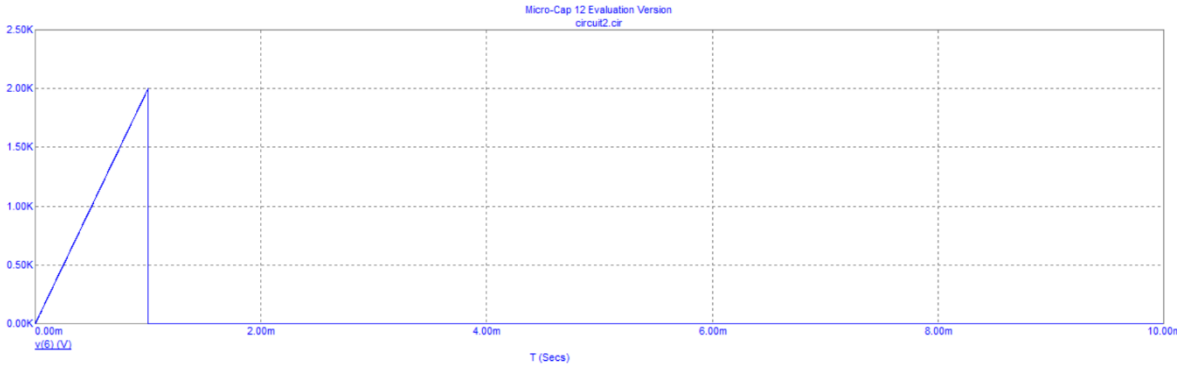
Слика 7: Model Square

На слици је приказан прозор за покретање транзијентне анализе као и унети параметри (све што се налази на слици је објашњено на страни 4)



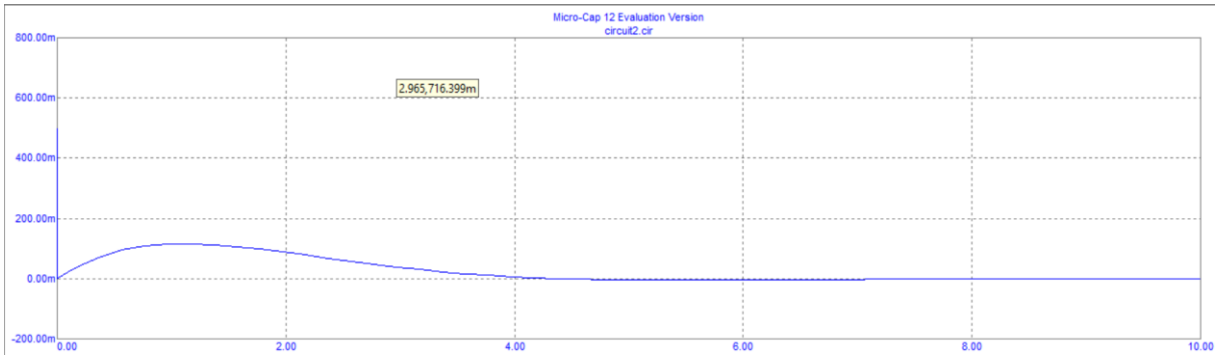
Слика 8: Покретање транзијентне анализе

На слици је приказана Диракова функција:

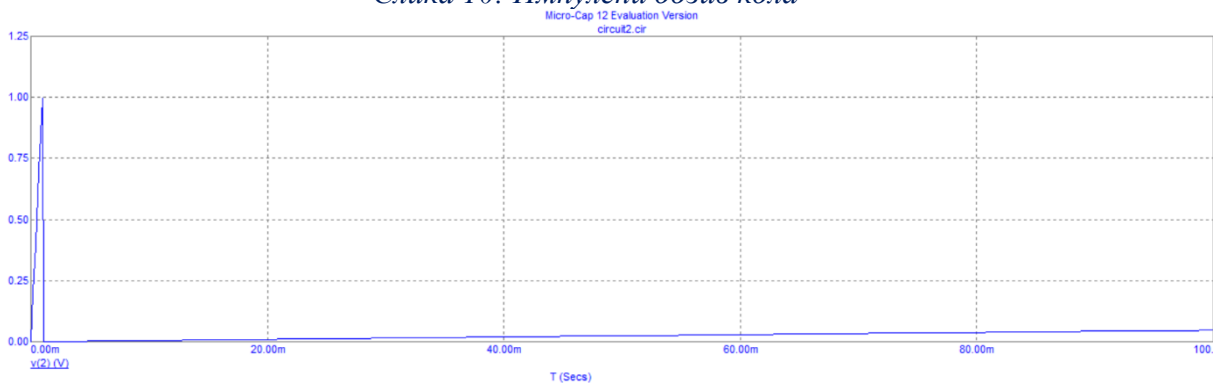


Слика 9: График Диракове функције

На слици је приказан импулсни одзив кола:

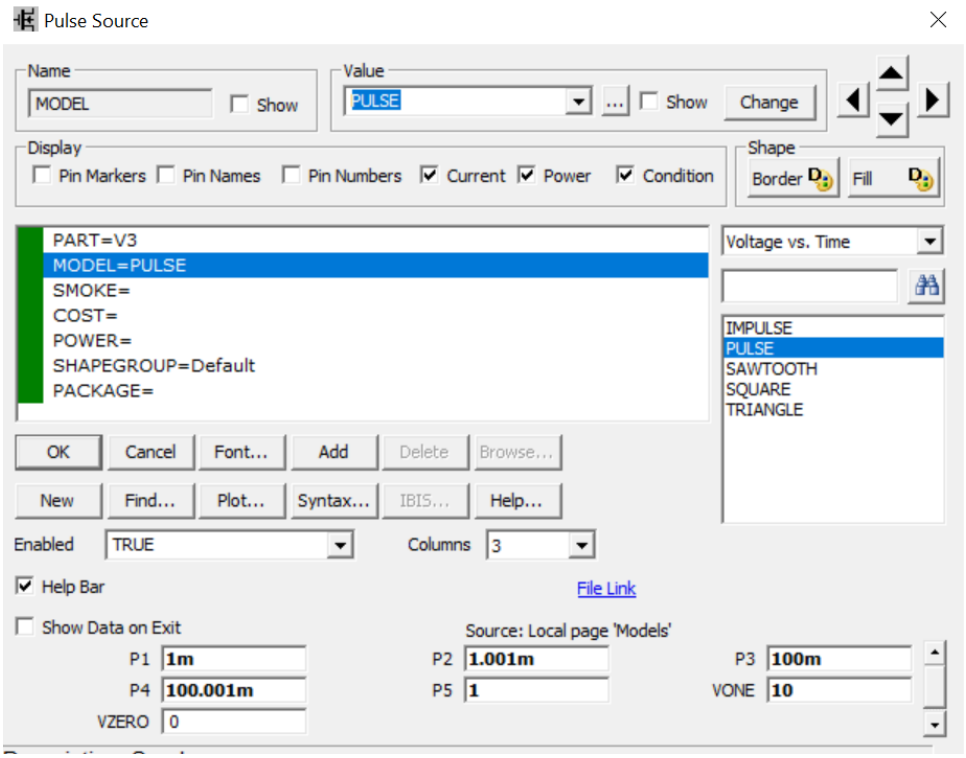


Слика 10: Импулсни одзив кола



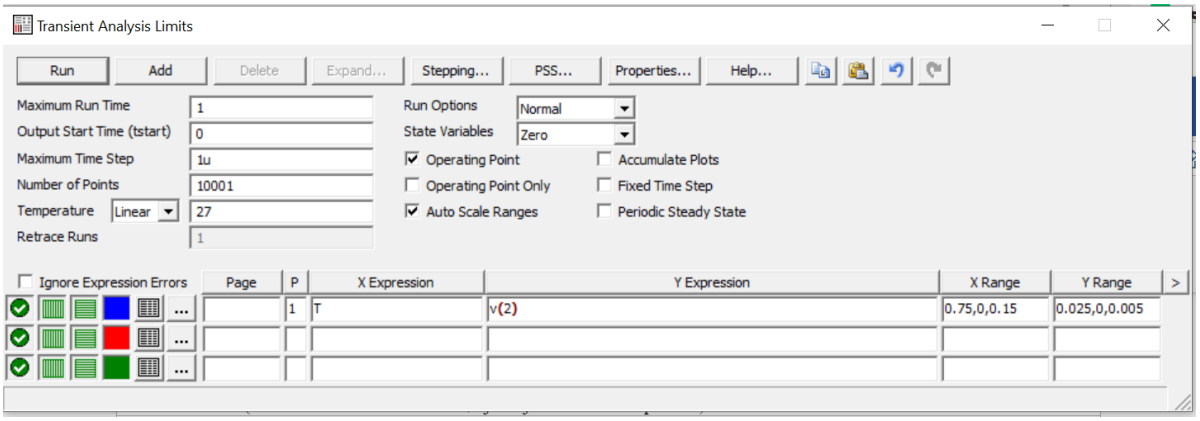
Слика 11: Импулсни одзив кола око нуле

За одскочни одзив користити модел генератора Pulse. На наредној слици приказано подешавање модела, значење параметара за намештање модела је исто као и на слици за импулсни одзив.



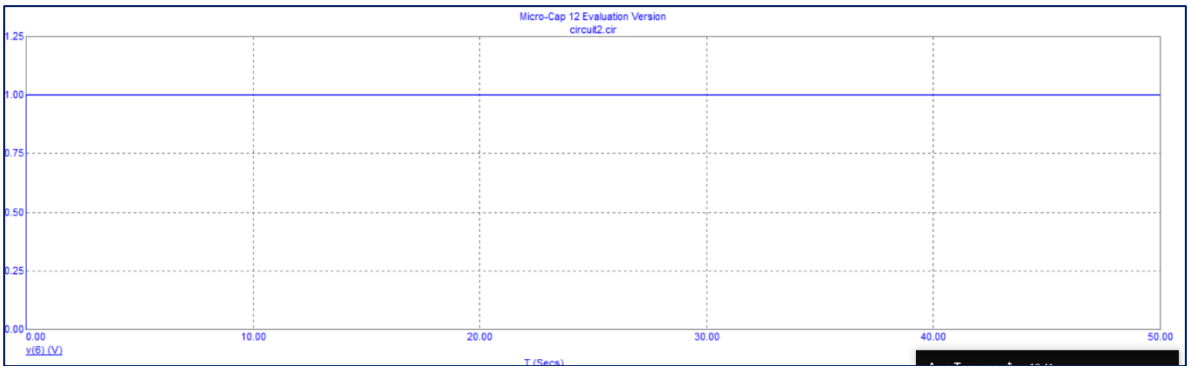
Слика 12: Model Pulse

На слици је приказан прозор за покретање транзијентне анализе као и унети параметри (све што се налази на слици је објашњено на страни 4)



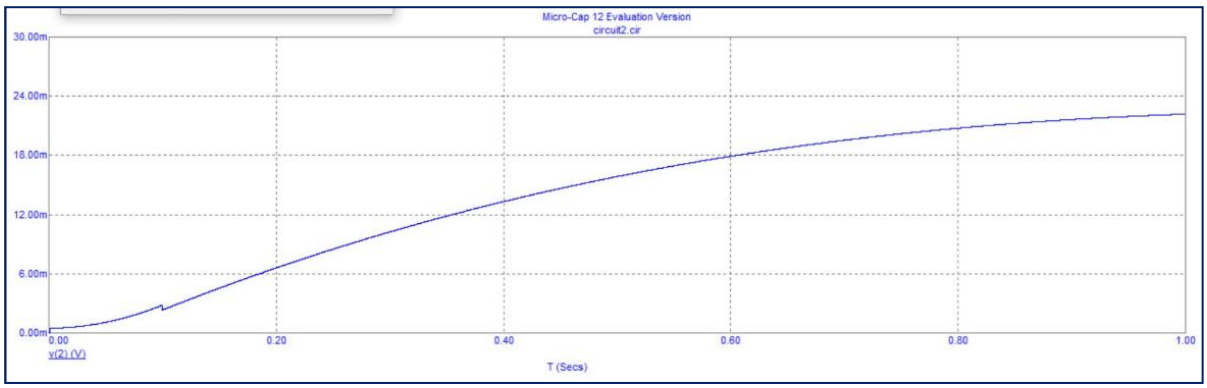
Слика 13: Покретање транзијентне анализе

На слици је приказана Хевисајдова функција (за моделовање ове функције користили смо Pulse):

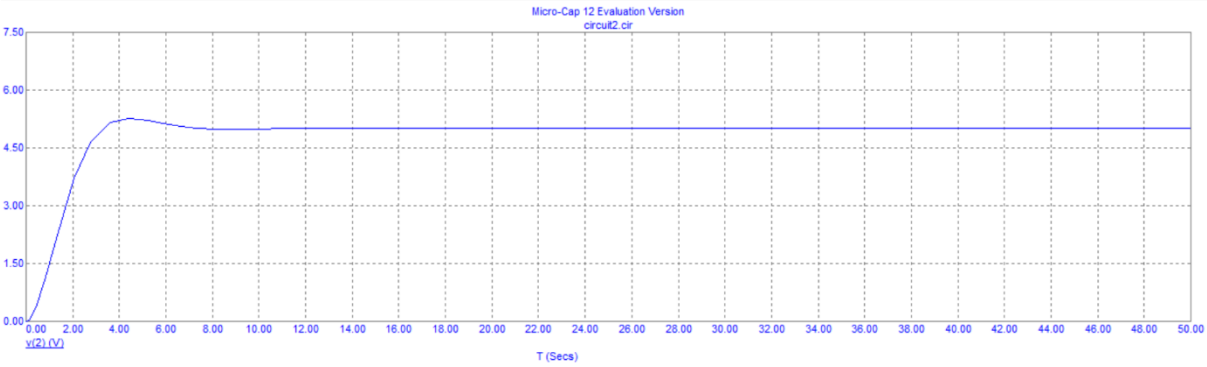


Слика 14: График Хевисајдове функције

На слици је приказан одскочни одзив кола:



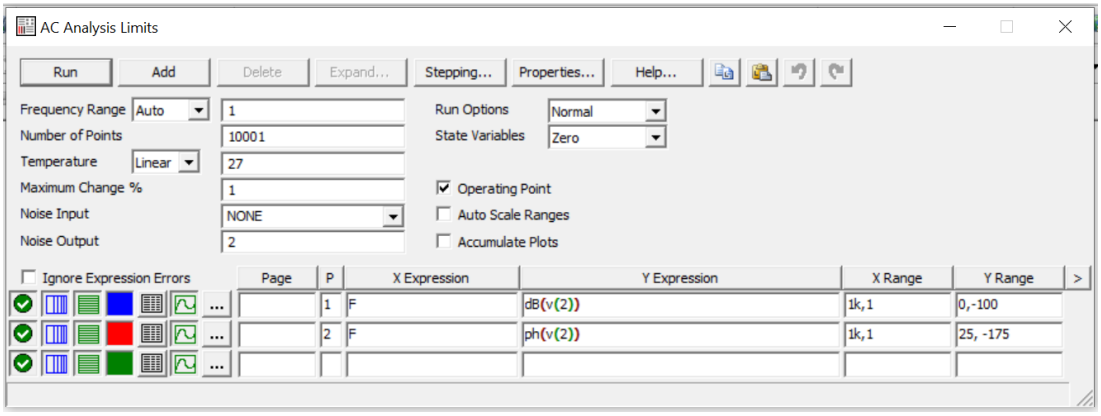
Слика 15: Одскочни одзив кола



Слика 16: Одскочни одзив кола

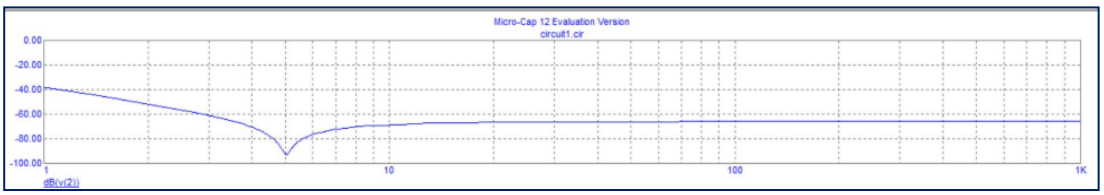
б) амплитудска и фазна карактеристика

На слици је приказ прозор за покретање AC анализе, као и унети параметри (све што се налази на слици је објашњено на страни 5)



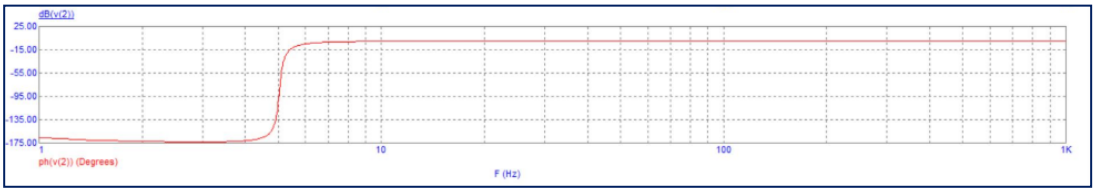
Слика 17: Покретање транзијентне анализе

На слици је приказана амплитудска карактеристика кола:



Слика 18: Амплитудска карактеристика кола

На слици је приказана фазна карактеристика кола:



Слика 19: Фазна карактеристика кола

Увод у SymPy

SymPy је библиотека која се може бесплатно преузети са сајта (https://www.sympy.org). Може да се инсталира на било који компјутер са Python-ом који има mpmath Python библиотеку пре тога интсалирану.Многи пројекти као што су Cadabra,Optalang, SageMath, Lcapy, Quantum Programming in Python и многи слични користе SymPy. SymPy се инсталира преко командне линије где се упише pip install sympy. Да би се SymPy користио морају се укључити следеће библиотеке:

```
>>> import sympy
... from sympy import *
... from sympy import symbols, ones, Eq
... from sympy.solvers.solveset import solveset
... x,y=symbols('x y')
... solveset (Eq(x**2-9,0), x )
```

$$\{-3, 3\}$$

Слика 20: SymPy укључивање библиотека

На слици након наредби је решена једначина $x^2 - 9 = 0$. Да би се једначина решила морају се прво дефинисати коришћени симболи. То се ради на следећи начин:
x,y = symbols ('x y')
Да би се операциј извршила позива се наредба solvest (equation, variable, domain) где се на прво место ставља једначина која треба да се израчуна, а на друго варијабле. Једначина се дефинише као Eq(x, y) где се на место x пише лева страна једнакости, а на месту у десна (као што је наведено горе у примеру).

Поред једначине са једном непознатом могу се решити и:

- квадратне једначине

```
>>> solveset(Eq(x**2-3*x, -2),x)
```

$$\{1, 2\}$$

Слика 21: Решење квадратне једначине у SymPy

- једначине са две непознате

```
linsolve([Eq(x-y,16),Eq( x + y ,1) ], (x, y))
```

$$\left\{\left(\frac{17}{2}, -\frac{15}{2}\right)\right\}$$

Слика 22: Решење једначине са 2 непознате у SymPy


```
linsolve([Eq(x-y,16),Eq( x + y ,1) ], (x, y))
```

$$\left\{\left(\frac{17}{2}, -\frac{15}{2}\right)\right\}$$

- интеграли

```
>>> expr=x**2 + x + 1
... integrate(expr, x)
```

$$\frac{x^3}{3} + \frac{x^2}{2} + x$$

Слика 23: Интеграли у SymPy

- symplify( ):

```
>>> simplify((x**3+1)/(x**2 - x + 1))
```

$$x + 1$$

Слика 24: Наредба symplify у SymPy

- диференцијалне једначине

```
>>> eqn=Eq(f(x).diff(x)+f(x), cos(x))
... eqn
```

$$f(x) + \frac{d}{dx} f(x) = \cos(x)$$

Слика 25: Диференцијалне једначине у SymPy

- комплексни бројеви:

Када се неки симбол дефинише потребно је написати a=Symbol('a' , real=True) иначе SymPy сматра да постоји и имагинарни и реални део.

```
>>> b=Symbol('b')
>>> b=2+3*I
>>> b
```

$$2 + 3i$$

```
>>> Abs(b)
```

$$\sqrt{13}$$

```
>>> conjugate(b)
```

$$2 - 3i$$

Слика 26: Комплексни бројеви у SymPy

- субституција (мења једну променљиву другом, или написаним бројем):

```
>>> expr= a*a*a + 6*a*a - 5*a + sin(pi*a)+7
>>> expr
```

$$a^3 + 6a^2 - 5a + \sin(\pi a) + 7$$

```
>>> expr.subs(a,1/3)
```

$$\frac{\sqrt{3}}{2} + \frac{163}{27}$$

Слика 27: Субституција у SymPy

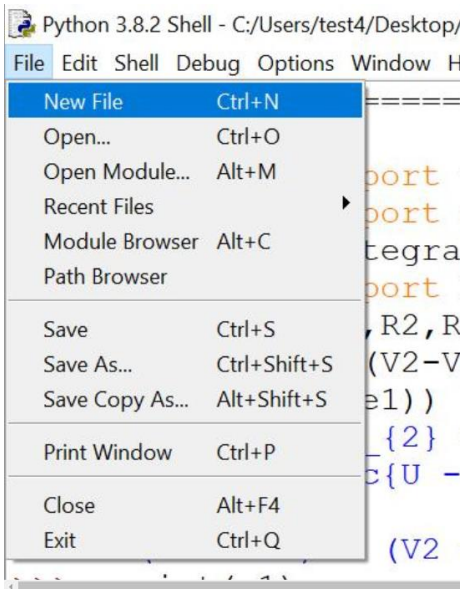
- evalf( ):

```
>>> expr=a/b
>>> expr.evalf(20, subs={a:100, b:7})
```

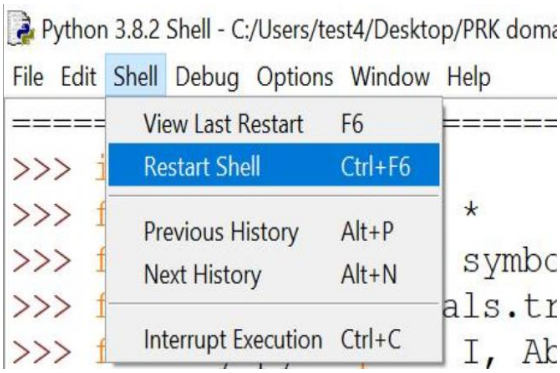
$$14.285714285714285714$$

Слика 28: Наредба evalf у SymPy

На наредним сликама видимо покретање програма(на првој слици) и рестартовање и undo рестарта (на другој слици).

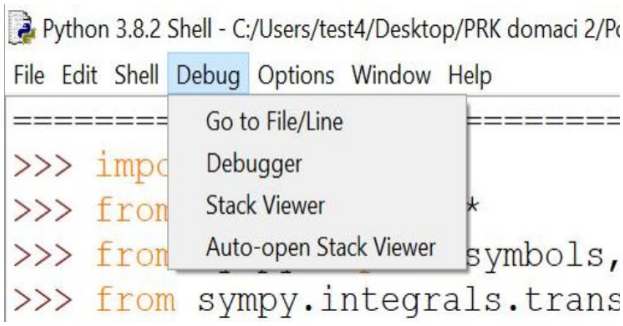


Слика 29: Покретање програма

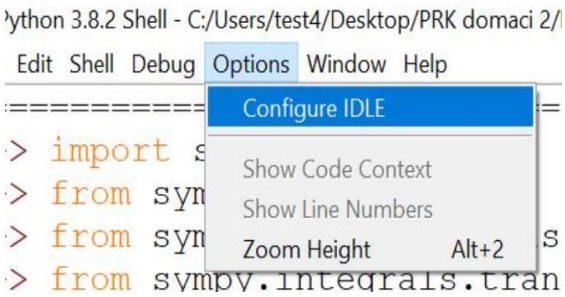


Слика 30: Рестартовање

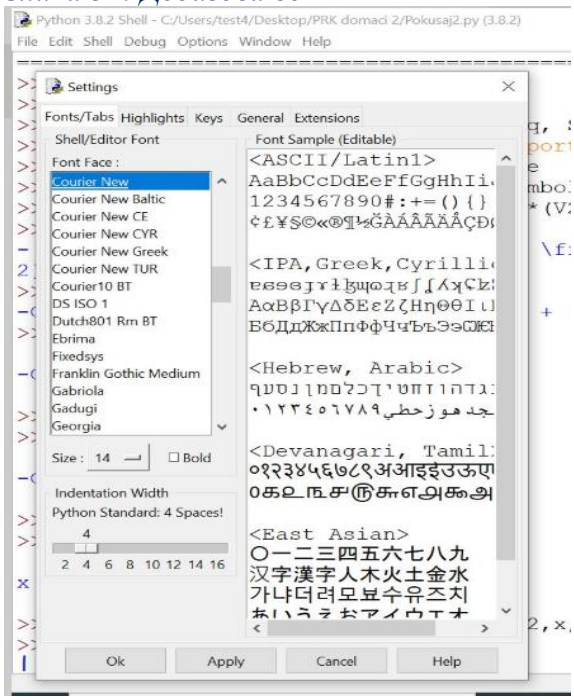
На следећим сликама се види дебаговање (на првој слици), и подешавање екрана (преостале слике).



Слика 31: Дебаговање



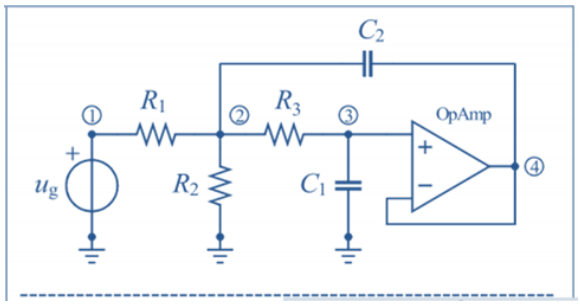
Слика 32: Подешавање екрана



Слика 33: Подешавање екрана

Задатак 2

На слици испод представљено је коло са Sallen & Key LP-LQ активним филтром.



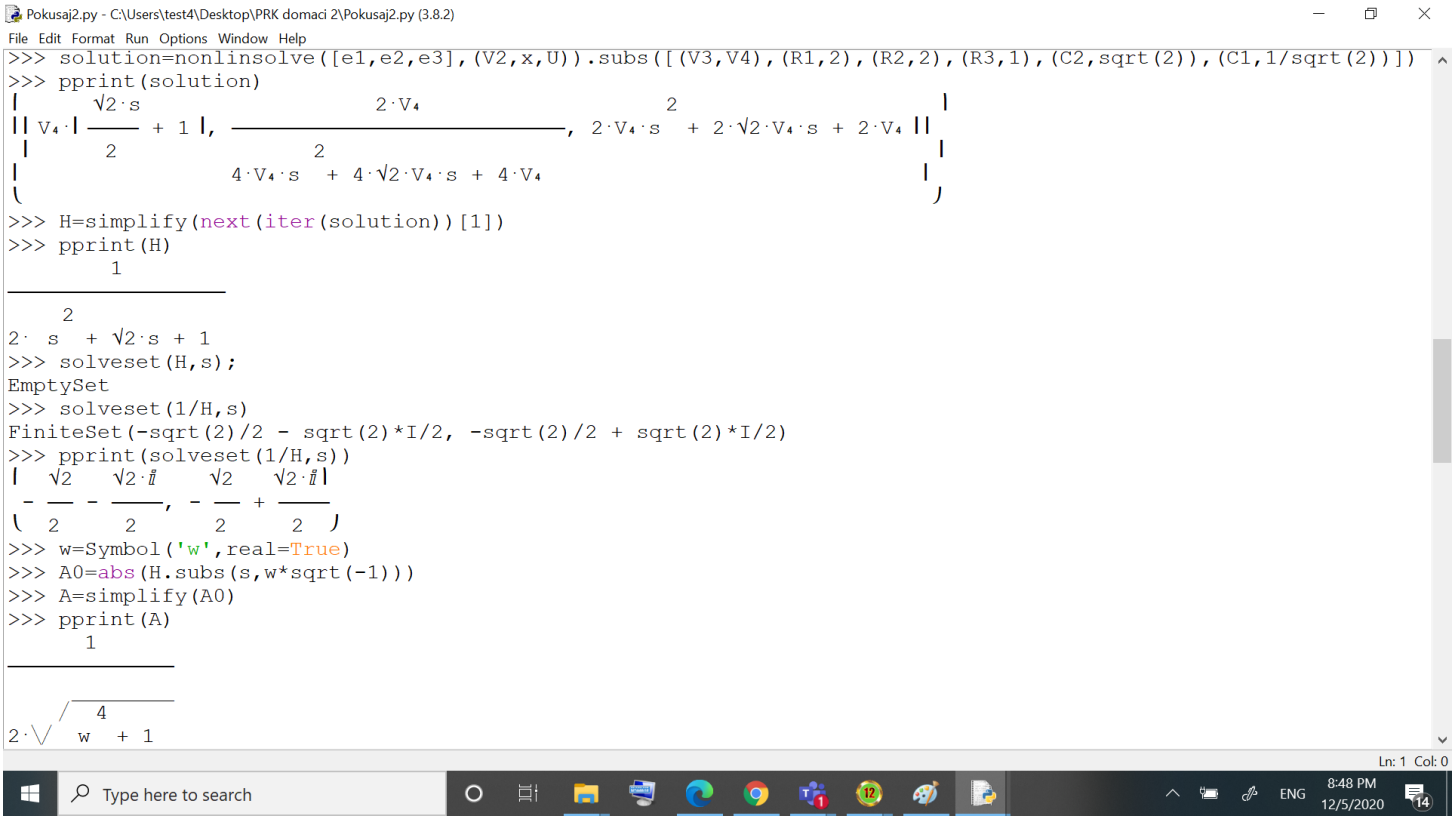
Слика 34: Коло за задатак 2

Одредити трансфер функцију, нуле и полове, импулсни и одскочни одзив, амплитудски одзив, 3 dB пропусни опсег овог филтра, ако је R = 1 Ω, Ω = 1 rad/s.

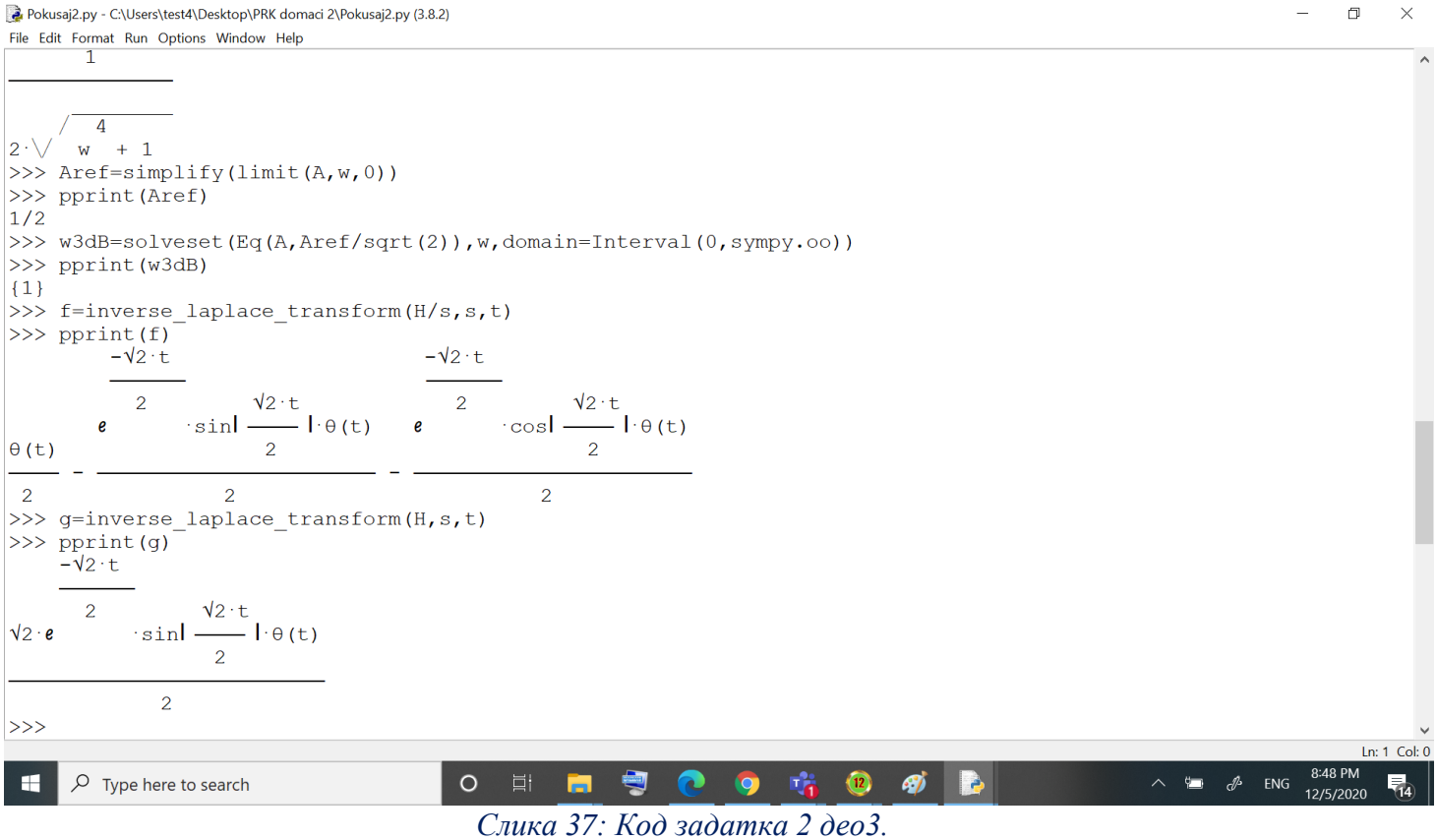
Коло је решено помоћу SymPy софтверског алата. У Python програму су прво дефинисане варијабле, где су V2, V3 и V4 потенцијали одговарајућих чворова, R отпорници, C кондензатори, t време и s комплексна учестаност. e1 и e2 су једначине за КЗС. X је трансфер функција. e1, e2 и e3 су једнаки нули. Уз помоћ nonlinsolve ( ) наредбе израчуна се V2 (из јендчине e2 и убаци у једначину e1) у зависности од V4, израчуна се U (из једначине e1) и x односно трансфер функцију. Помоћу наредбе subs( ) се уводи смена унутар једначина након њиховог израчунавања. Последњом наредбом се извуче израз за трансфер функцију из solution-a. На наредне 2 слике је приказано цело рсшење задатка.

```
Pokusaj2.py - C:\Users\test4\Desktop\PRK domaci 2\Pokusaj2.py (3.8.2)
File  Edit  Format  Run  Options  Window  Help
>>> import sympy
>>> from sympy import *
>>> from sympy import symbols, ones, Eq, S
>>> from sympy.integrals.transforms import laplace_transform
>>> from sympy import I, Abs, conjugate
>>> V2,V3,V4,U,R1,R2,R3,C1,C2,t,s,x=symbols("V2,V3,V4,U,R1,R2,R3,C1,C2,t,s,x")
>>> e1=(U-V2)/R1-(V2-V3)/R3-V2/R2-C2*s*(V2-V4)
>>> pprint(latex(e1))
- C_{2} s \left(V_{2} - V_{4}\right) - \frac{V_{2} - V_{3}}{R_{3}} - \frac{V_{
2}}{R_{2}} + \frac{U - V_{2}}{R_{1}}
>>> print(e1)
-C2*s*(V2 - V4) - (V2 - V3)/R3 - V2/R2 + (U - V2)/R1
>>> pprint(e1)
                  V₂ - V₃   V₂   U - V₂
-C₂⋅s⋅(V₂ - V₄) - ─────── - ── + ─────
                    R₃      R₂     R₁
>>> e2=(V2-V3)/R3-C1*s*V3
>>> pprint(e2)
          V₂ - V₃
-C₁⋅V₃⋅s + ───────
             R₃
>>> e3=x-V4/U
>>> pprint(e3)
    V₄
x - ──
    U
>>> solution=nonlinsolve([e1,e2,e3],(V2,x,U)).subs([(V3,V4),(R1,2),(R2,2),(R3,1),(C2,sqrt(2)),(C1,1/sqrt(2))])
>>> pprint(solution)
⎡      √2⋅s                    2⋅V₄                        2                  ⎤
⎢⎛V₄⋅⎜──── + 1⎞, ──────────────────────────, 2⋅V₄⋅s  + 2⋅√2⋅V₄⋅s + 2⋅V₄⎥
⎣⎝    ⎝ 2  ⎠                                                               ⎦
                                                                  Ln: 1  Col: 0
Type here to search                                  ENG  8:46 PM  12/5/2020
```

Слика 35: Код задатка 2 део1.



Слика 36: Код задатка 2 део2.



Слика 37: Код задатка 2 део3.

Уз помоћ наредбе symplify() се x поједностави односно све што може да се скрати се скрати.

- Трансфер функција односно x једнако је:

```
>>> H=simplify(next(iter(solution))[1])
>>> pprint(H)
          1
──────────────────
     2
2⋅ s  + √2⋅s + 1
```

Слика 38:Трансфер функција

- На наредној слици видимо са су нуле празан скуп:

```
>>> solveset(H,s);
EmptySet
```

Слика 39:Нуле

- Полови су:

```
>>> solveset(1/H,s)
FiniteSet(-sqrt(2)/2 - sqrt(2)*I/2, -sqrt(2)/2 + sqrt(2)*I/2)
>>> pprint(solveset(1/H,s))
⎧  √2    √2⋅ⅈ     √2    √2⋅ⅈ⎫
⎨- ── - ─────, - ── + ─────⎬
⎩  2      2       2      2  ⎭
```

Слика 40:Полови

- Амплитудски одзив је:

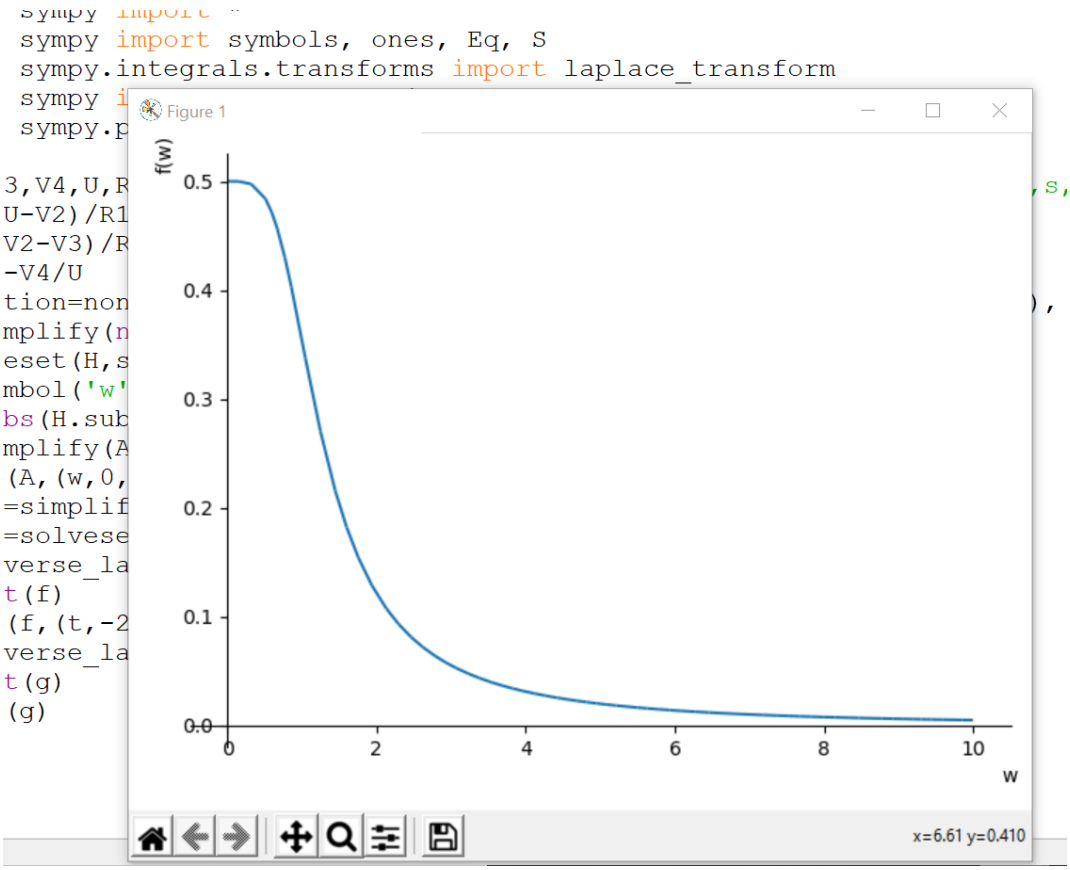
```
>>> w=Symbol('w',real=True)
>>> A0=abs(H.subs(s,w*sqrt(-1)))
>>> A=simplify(A0)
>>> pprint(A)
        1
─────────────────
      ╱  4
2⋅   ╱   w  + 1
   ╲╱
```

```
>>> Aref=simplify(limit(A,w,0))
>>> pprint(Aref)
1/2
```

Слика 41:Амплитудски одзив

```
plot(A)
```

Слика 42:Код за цртање амплитудске карактеристике



Слика 43: График амплитудског одзива

На слици изнад је коришћена наредба abs( ) која рачуна апсолутну вредност израза односно у овом случају модуо комплексног броја. Она имплицитно позива функцију Abs( ) која може да примени операцију над симболичким изразом.

- 3dB пропусни опсег:

```
>>> w3dB=solveset(Eq(A,Aref/sqrt(2)),w,domain=Interval(0,sympy.oo))
>>> pprint(w3dB)
{1}
```

Слика 44: 3dB пропусни опсег

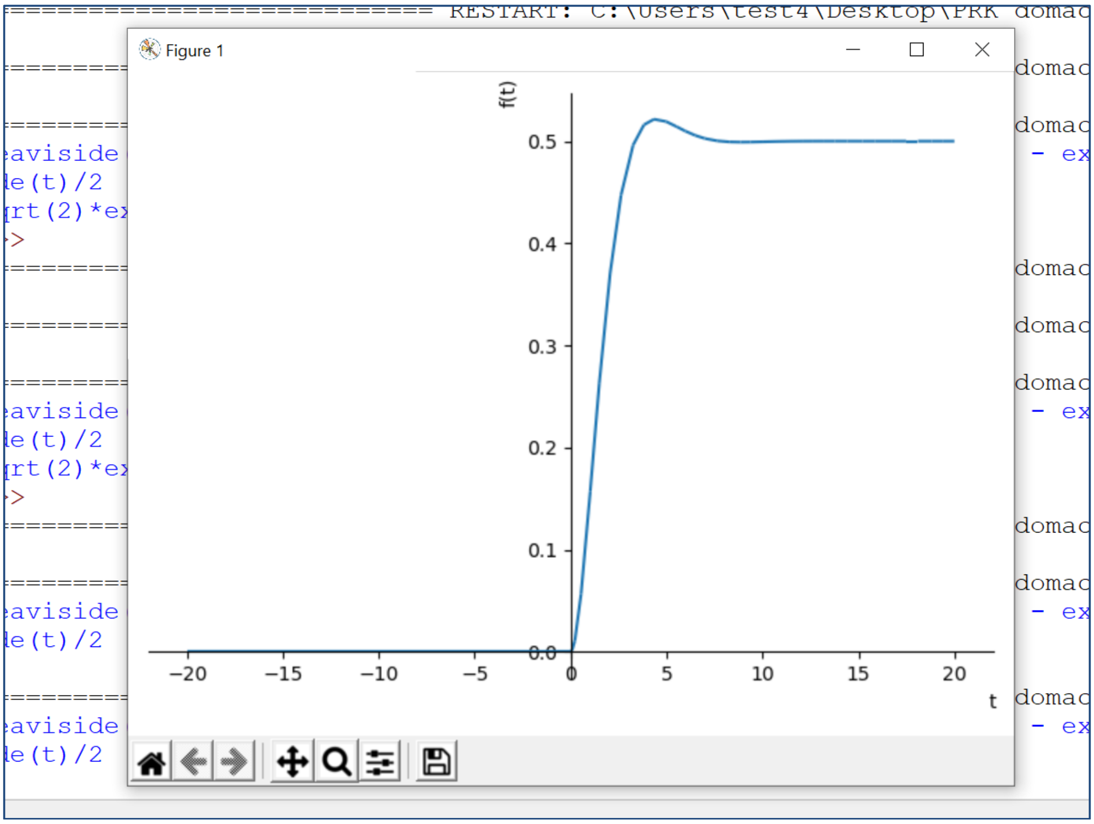
- Одскочни одзив:

```
>>> f=inverse_laplace_transform(H/s,s,t)
>>> pprint(f)
       -√2·t                  -√2·t
       ─────                  ─────
         2      ⎛√2·t⎞          2      ⎛√2·t⎞
      ℯ     ·sin⎜────⎟·θ(t)  ℯ     ·cos⎜────⎟·θ(t)
θ(t)            ⎝ 2  ⎠                 ⎝ 2  ⎠
──── - ─────────────────── - ───────────────────
 2              2                     2
```

Слика 45: Одскочни одзив

```
plot(f,(t,-20,20))
```

Слика 46:Код за цртање одскочног одзива



Слика 47: График одскочног одзива

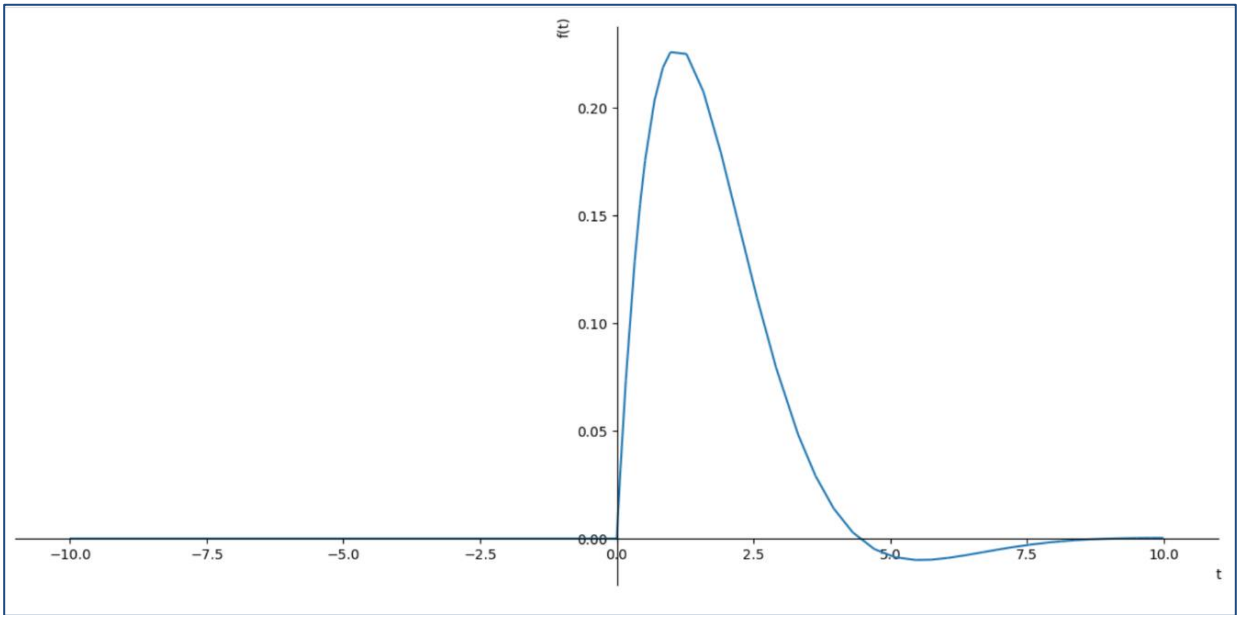
- Импулсни одзив:

```
>>> g=inverse_laplace_transform(H,s,t)
>>> pprint(g)
    -√2·t
    ─────
      2      ⎛√2·t⎞
√2·ℯ     ·sin⎜────⎟·θ(t)
             ⎝ 2  ⎠
────────────────────────
           2
>>>
```

Слика 48: Импулсни одзив

```
plot(g)
```

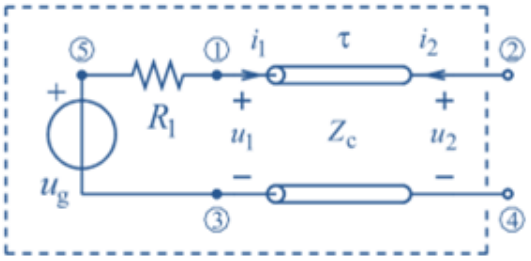
Слика 49:Код за цртање импулсног одзива



Слика 50: График импулсног одзива

Задатак 3

На слици је приказан идеалан вод дужине D.



Слика 51: Приказ кола задатка 3

Једначине за улазни и излазни напон су:

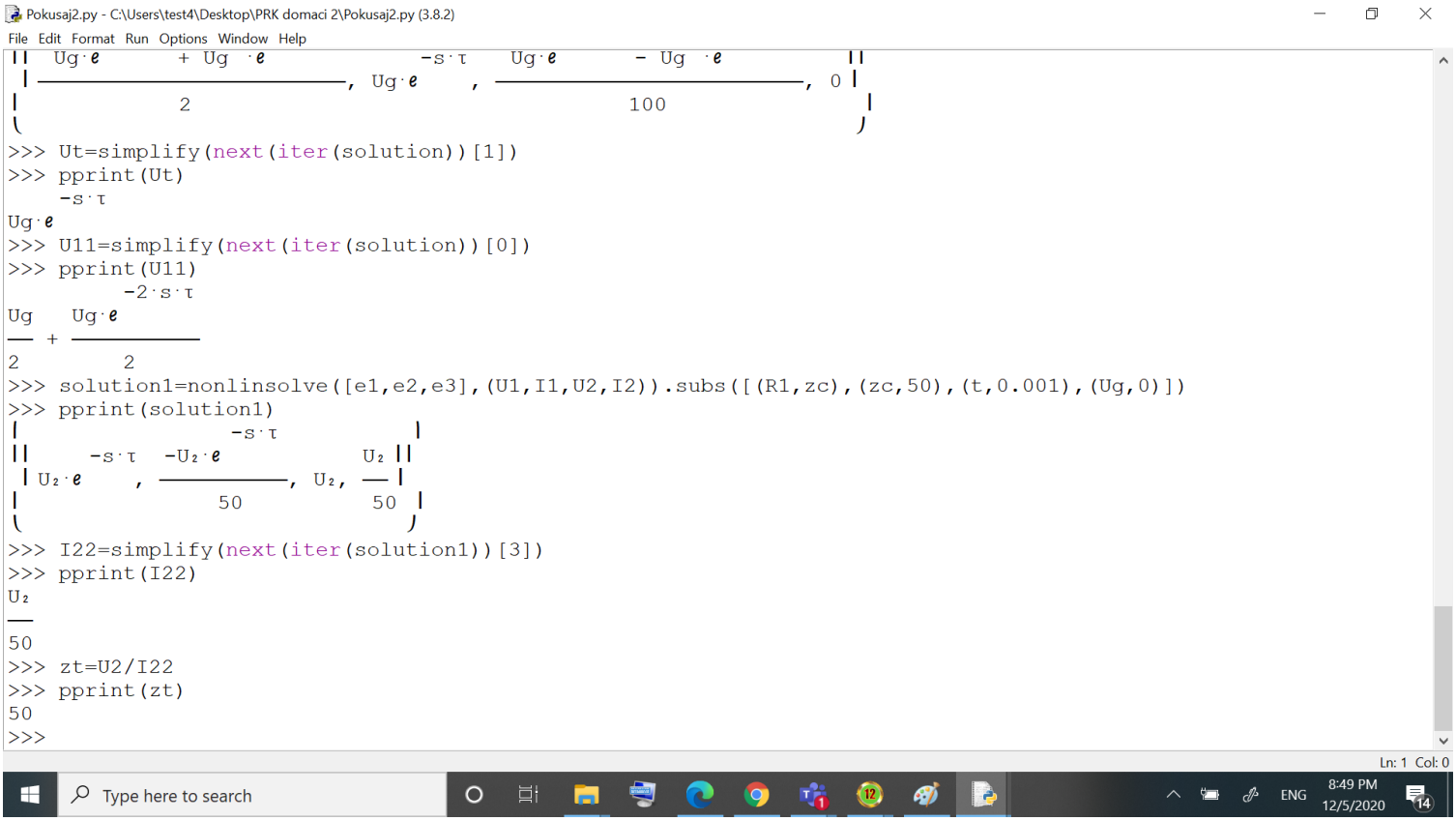
$$\begin{cases} u_1(t) = Z_c i_1(t) + Z_c i_2(t-\tau) + u_2(t-\tau) \\ u_2(t) = Z_c i_2(t) + Z_c i_1(t-\tau) + u_1(t-\tau) \end{cases}$$

Слика 52: Једначине улазног и излазног напона

Прво су дефинисане варијабле , а потом једначине e1, e2 и e3.При чему је коришћено својство помераја аргумената.На наредне 2 слике је приказано цело решење задатка.

```
Pokusaj2.py - C:\Users\test4\Desktop\PRK domaci 2\Pokusaj2.py (3.8.2)
File  Edit  Format  Run  Options  Window  Help
>>> import sympy
>>> from sympy import *
>>> from sympy import symbols, ones, Eq, S
>>> from sympy.integrals.transforms import laplace_transform
>>> from sympy import I, Abs, conjugate
>>> R1,t,s,Ug,Tau, zc,zt,I1,I2,U1,U2=symbols("R1,t,s,Ug,tau, zc,zt,I1,I2,U1,U2");
>>> e1=zc*I1+zc*I2*exp(-s*Tau)+U2*exp(-s*Tau)-U1
>>> pprint(e1)
                -s·τ             -s·τ
I₁·zc + I₂·zc·ℯ      - U₁ + U₂·ℯ
>>> e2=zc*I2+zc*I1*exp(-s*Tau)+U1*exp(-s*Tau)-U2
>>> pprint(e2)
       -s·τ                -s·τ
I₁·zc·ℯ      + I₂·zc + U₁·ℯ      - U₂
>>> e3=U1+R1*I1-Ug
>>> pprint(e3)
I₁·R₁ + U₁ - Ug
>>> solution=linsolve([e1,e2,e3],(U1,U2,I1,I2)).subs([(R1,zc),(zc,50),(t,0.001),(I2,0)])
>>> pprint(solution)
⎡       2·s·τ         -2·s·τ                   2·s·τ         -2·s·τ     ⎤
⎢  Ug·ℯ       + Ug ·ℯ             -s·τ   Ug·ℯ       - Ug ·ℯ           ⎥
⎢ ──────────────────────────, Ug·ℯ    , ──────────────────────────, 0 ⎥
⎣             2                                  100                  ⎦
>>> Ut=simplify(next(iter(solution))[1])
>>> pprint(Ut)
    -s·τ
Ug·ℯ
>>> U11=simplify(next(iter(solution))[0])
>>> pprint(U11)
```

Слика 53: Код задатка 3 део1.



Слика 54: Код задатка 3 део2.

. Пошто је коло отворено струја I2 је нула.

e1 је:

```
>>> e1=zc*I1+zc*I2*exp(-s*Tau)+U2*exp(-s*Tau)-U1
>>> pprint(e1)
-s·τ -s·τ
I₁·zc + I₂·zc·e - U₁ + U₂·e
```

Слика 55: e1

e2 је:

```
>>> e2=zc*I2+zc*I1*exp(-s*Tau)+U1*exp(-s*Tau)-U2
>>> pprint(e2)
-s·τ -s·τ
I₁·zc·e + I₂·zc + U₁·e - U₂
```

Слика 56: e2

e3 је:

```
>>> e3=U1+R1*I1-Ug
>>> pprint(e3)
I₁·R₁ + U₁ - Ug
```

Слика 57: e3

Улазни напон је:

```
>>> U11=simplify(next(iter(solution))[0])
>>> pprint(U11)
          -2·s·τ
Ug    Ug·ℯ
── + ──────────
2          2
```

Слика 58: Улазни напон

```
U11=U11.subs(Ug,laplace_transform(Heaviside(t),t,s)[0])
```

Слика 59:Убацивање вредности

```
          -2·s·τ
 1     ℯ
─── + ──────
2·s      2·s
```

Слика 60:Улазни напон када се убаци вредност

```
θ(t)    θ(t - 2·τ)
──── + ──────────
 2          2
```

Слика 61:Улазни напон у врмеенском домену

```
pprint(inverse_laplace_transform(U11,s,t))
```

Слика 62:Команда за исписивање напона у временском домену

Излазни напон једнак је Тевененовом напону и израчунат је на следећи начин.

Тевененов генератор Ut је:

```
>>> Ut=simplify(next(iter(solution))[1])
>>> pprint(Ut)
    -s·τ
Ug·ℯ
```

Слика 63: Тевененов генератор

```
Ut=Ut.subs(Ug,laplace_transform(Heaviside(t),t,s)[0])
```

Слика 64:Убацивање вредности

```
 -s·τ
ℯ
─────
  s
```

Слика 65:Излазни напон

```
θ(t - τ)
```

Слика 66:Излатни напон у временском домену

```
pprint(inverse_laplace_transform(Ut,s,t))
```

Слика 67:Код за исписивање излазног напона у временском домену

Исто као када су постављене једначине коришћено је да је множење функције $e - s\tau$ у Лапласовом домену аналогно транслирању времена за τ у временском домену. Зато је Ut=Ug(t- **τ**), а $U_1 = \frac{1}{2} U_g(t) + \frac{1}{2} Ug(t- 2\tau)$.
Импеданса Тевененовог генератора је:

```
>>> solution1=nonlinsolve([e1,e2,e3],(U1,I1,U2,I2)).subs([(R1,zc),(zc,50),(t,0.001),(Ug,0)])
>>> pprint(solution1)
⎧                  -s⋅τ           ⎫
⎪     -s⋅τ  -U₂⋅ℯ            U₂ ⎪
⎨U₂⋅ℯ     , ──────────, U₂, ── ⎬
⎪                50          50 ⎪
⎩                              ⎭
>>> I22=simplify(next(iter(solution1))[3])
>>> pprint(I22)
U₂
──
50
>>> zt=U2/I22
>>> pprint(zt)
50
>>>
```

Слика 68: Импенданса Тевененовог генератора

Са слике видимо да је $Z_t = 50\ \Omega$.

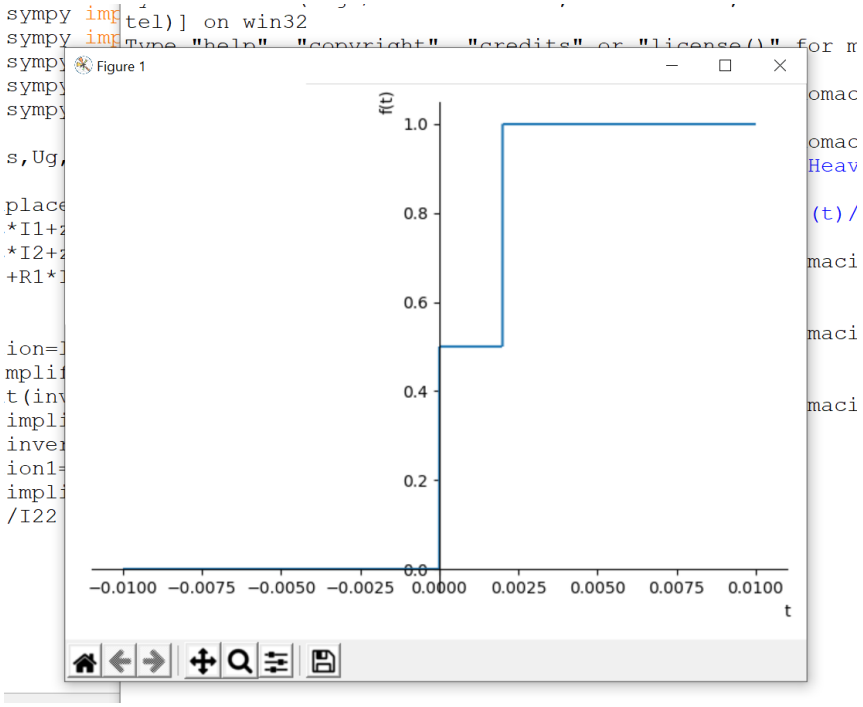
За приказивање графика функције одзива користи се одговарајућа библиотека коју укључујемо наредбом:

```
from sympy.plotting import plot.
```

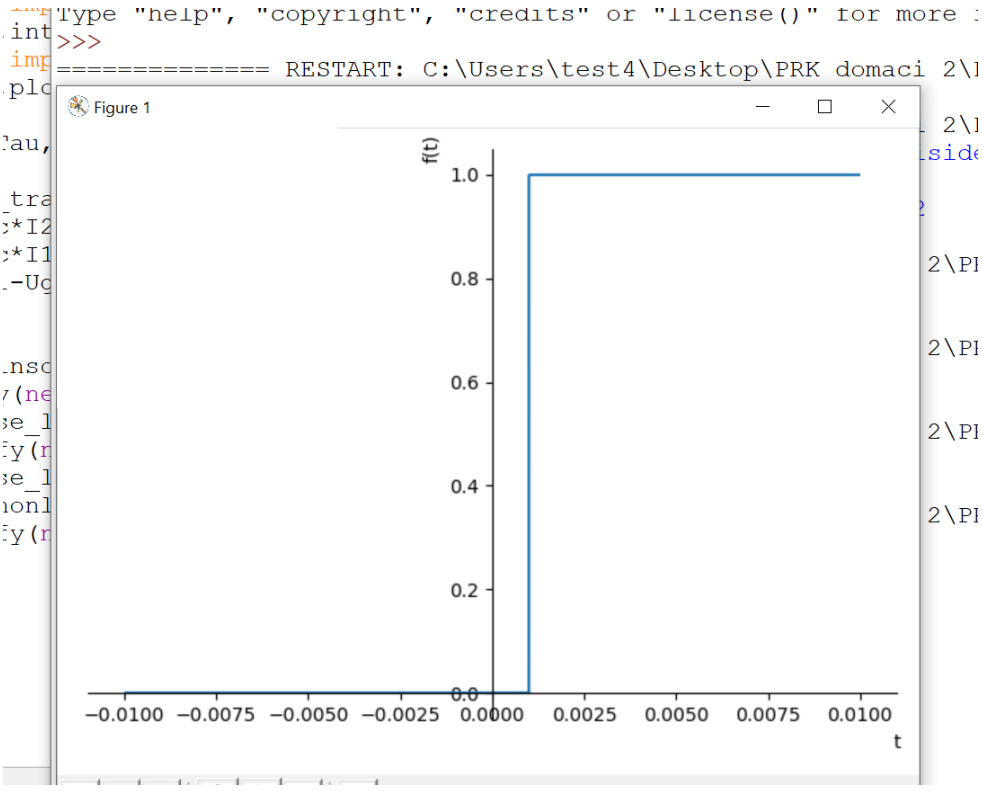
Како желимо да прикажемо график функције одзива нацртаћемо функцију Ut, што се ради наредбом:

```
plot(inverse_laplace_transform(Ut,s,t))
```

Након чега нам изађе следећи прозор



Слика 69: График функције улазног напона вода



Слика 70: График функције излазног напона вода