Simulando Redes Elétricas Inteligentes com OpenDSS e Python

Paulo Radatz - paulo.radatz@gmail.com

Rodolfo Londero - rodolfopl@gmail.com

Ênio Viana - eniocc@gmail.com

10 de Julho de 2022

SBSE2022
IX Simpósio Brasileiro de Sistemas Elétricos

Universidade Federal de Santa Maria 1960

UNISINOS

INCT

Conteúdo

- Instrutores
- Números do OpenDSS no Mundo e Brasil
- OpenDSS Visão Geral
- Controlando o OpenDSS usando Linguagem de Programação
- Demonstração
- Hands-on

Instrutores

Instrutor Paulo Radatz

Graduação Sanduíche:

- Engenharia Elétrica, Escola Politécnica da Universidade de São Paulo (USP-SP) (2010-2015)
- Politécnico de Milano – Milão, Itália (2012-2013)
- Melhor aluno de toda à escola de engenharia da USP-SP formado em 2015

Mestrado:

- Engenharia Elétrica, Escola Politécnica da Universidade de São Paulo (USP-SP) (2016-2019)
- Com tema: "Impacto das funções inteligentes de inversores de sistemas fotovoltaicos na operação de redes de distribuição de energia elétrica"

OpenDSS:

- Engenheiro / Cientista no Electric Power Research Institute (EPRI) – EUA
- 7 anos de experiência com o OpenDSS
- Criador do maior canal do YouTube sobre o OpenDSS do mundo: https://www.youtube.com/PauloRadatz
- Palestrante / Instrutor em diversos encontros, workshops e treinamentos sobre o OpenDSS. Incluindo o treinamento online do EPRI
- Desenvolvedor do OpenDSS
- Desenvolveu a primeira versão do SIGPerdas da Sinapsis
- Professor do curso de difusão "Análise de Sistemas Elétricos de Potência através do OpenDSS" fornecido pelo PECE da POLI-USP
- Professor no MBA em Redes de Distribuição de Energia Elétrica fornecido pelo PECE da POLI-USP
- Desenvolvedor do pacote Python py-dss-interface

Instrutor Rodolfo Londero

Graduação:

- Análise e Desenvolvimento de Sistemas, Instituto Federal Farroupilha- Alegrete/RS (2010-2013)
- Engenharia Elétrica, Universidade Federal do Pampa (Unipampa) Alegrete/RS (2015-2019)
  - Tema: Análise do transformador de estado sólido em substituição ao transformador convencional de redes de distribuição de energia para a conexão de sistemas de geração distribuída

Mestrado:

- Engenharia Elétrica, Universidade Federal de Santa Maria (USFM) (2019-2022)
  - Tema: “Metodologia para simulações de arcos elétricos aplicada ao estudo de energia incidente em sistemas elétricos de potência”

OpenDSS:

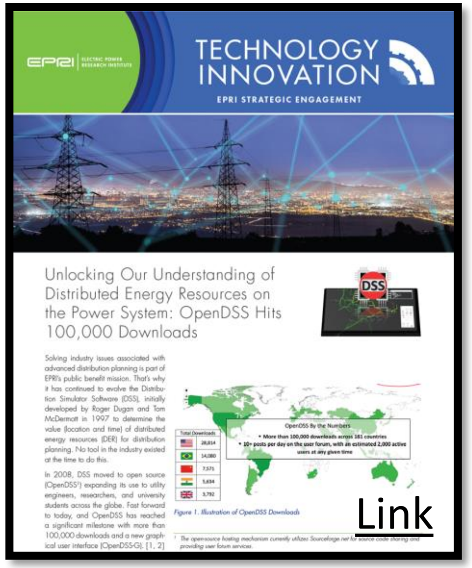
- Utiliza OpenDSS desde 2018
- Desenvolvedor do pacote Python py-dss-interface
- Programador desde 2013

Instrutor Ênio Viana

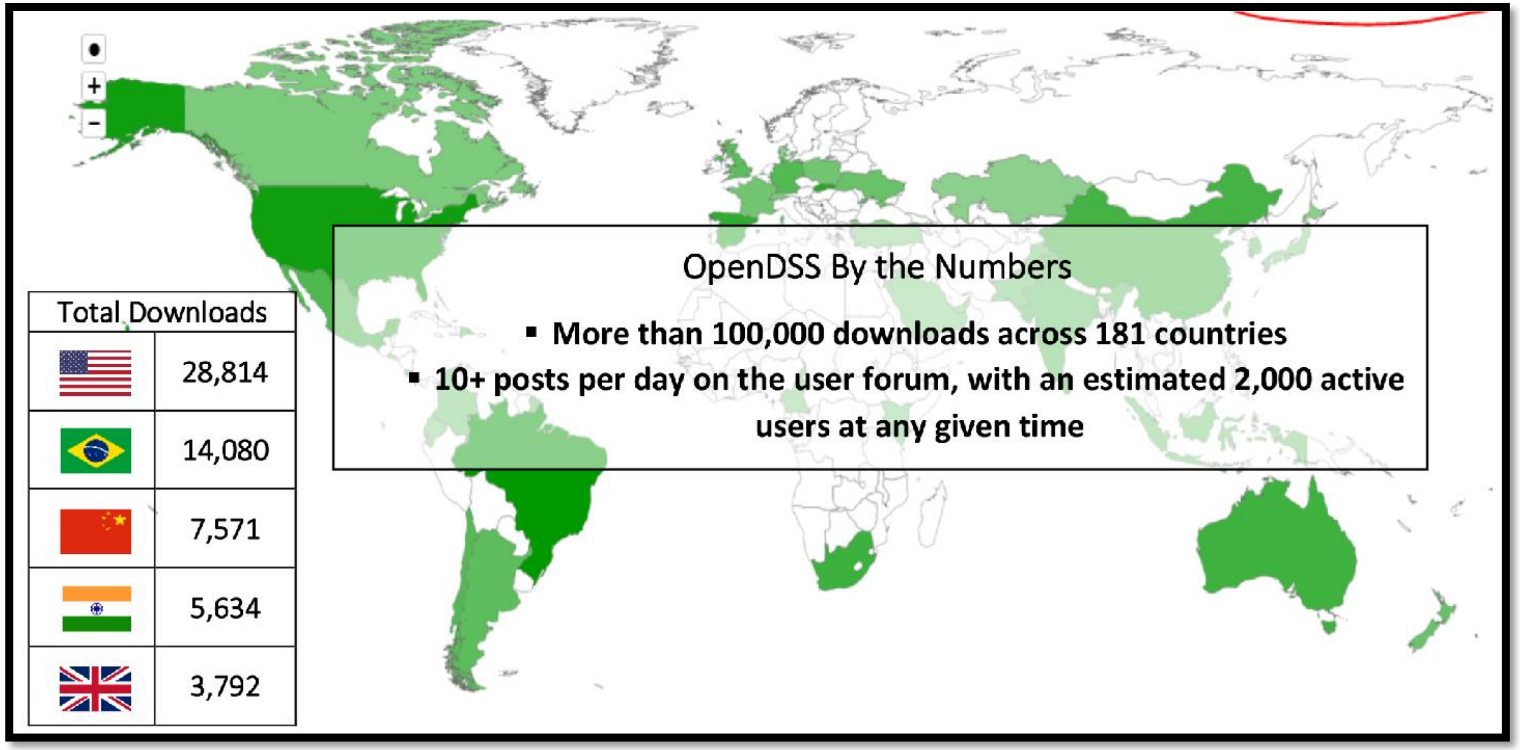
- **Bacharel** em Ciência da Computação
- **Especialização** em Perícia Forense Aplicada à Informática
- **Especialização** em Pós Graduação em Desenvolvimento de Aplicações em Dispositivos Móveis
- **Mestrado** em Engenharia Elétrica, Universidade Federal do Piauí (UFPI) (2017-2019)
  - Tema: “Heurística de Factibilização para Algoritmos Evolutivos na Reconfiguração de Redes em Sistemas de Distribuição de Energia Elétrica”
- **Doutorado** pausado em Ciência da Computação, Universidade Federal do Piauí (UFPI)
- Utiliza **OpenDSS** desde 2019; **Desenvolvedor** do pacote py-dss-interface
- Programador desde 2006
- Perito Criminal Oficial de TI - Perícia Forense do Estado do Ceará

Números do OpenDSS no Mundo e Brasil

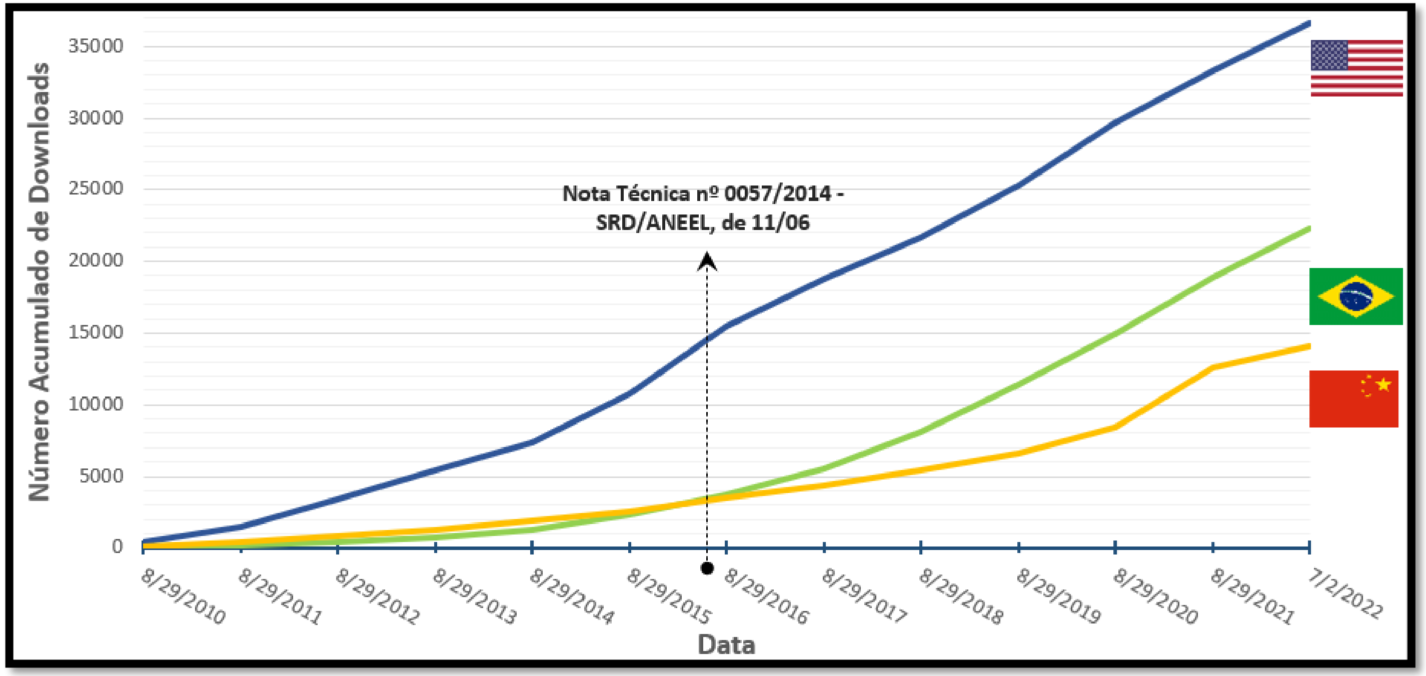
Números do OpenDSS nos 100000 Downloads



Link



Crescimento do uso do OpenDSS no Brasil



Total	
	36655
	22315
	14116

PRODIST Módulo 7

ANEEL – Motivos pela escolha do OpenDSS

OpenDSS no Brasil

OpenDSS Visão Geral

O que é o OpenDSS?

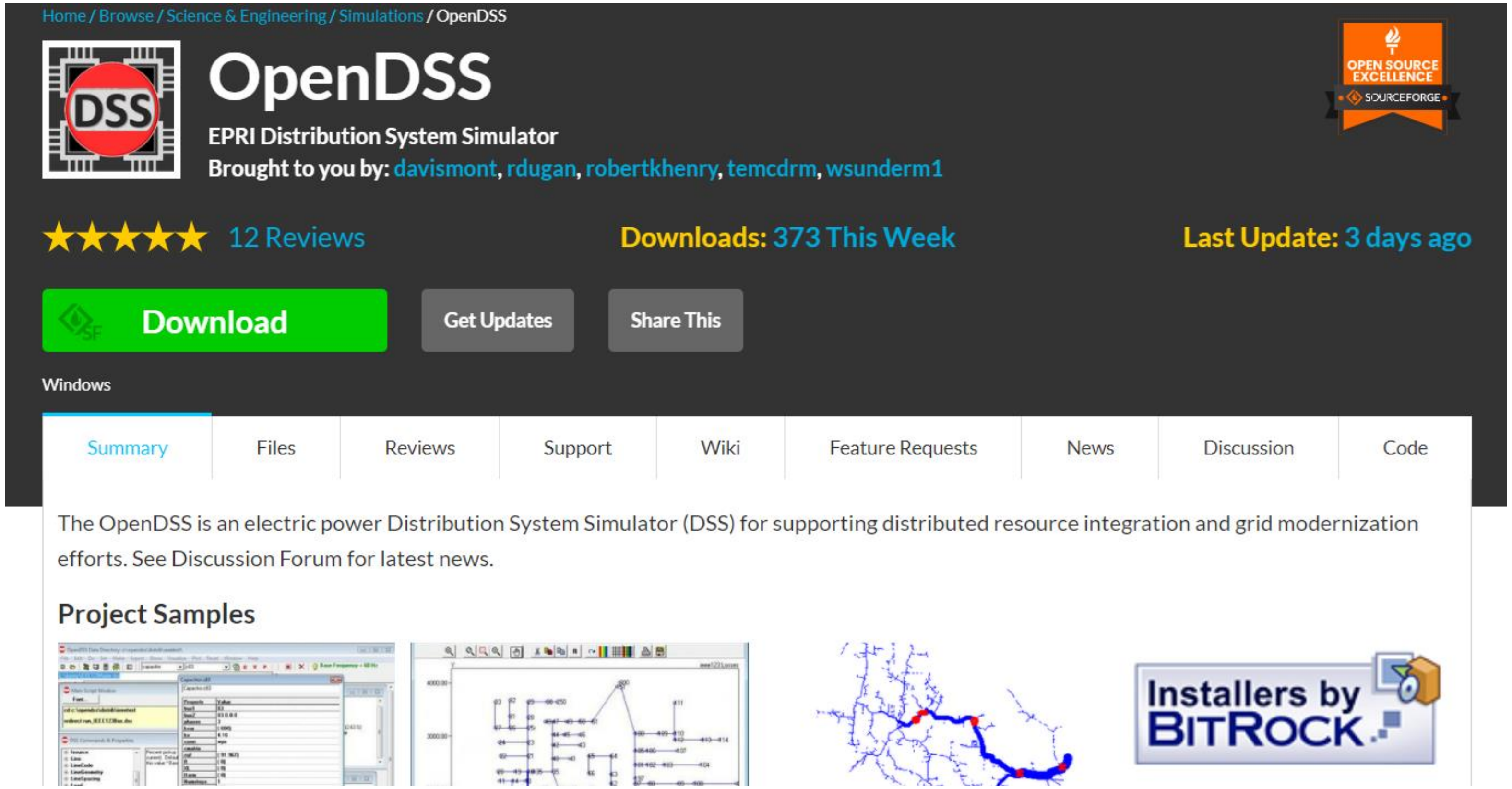
- **O que é**
  - Software de simulação de circuitos elétricos no domínio da frequência (**fasorial**), baseado em **linhas de códigos**

- **O que não é**
  - Não é um software de simulação de transitórios eletromagnéticos (domínio do tempo)

Site oficial do OpenDSS
Características Fundamentais do OpenDSS

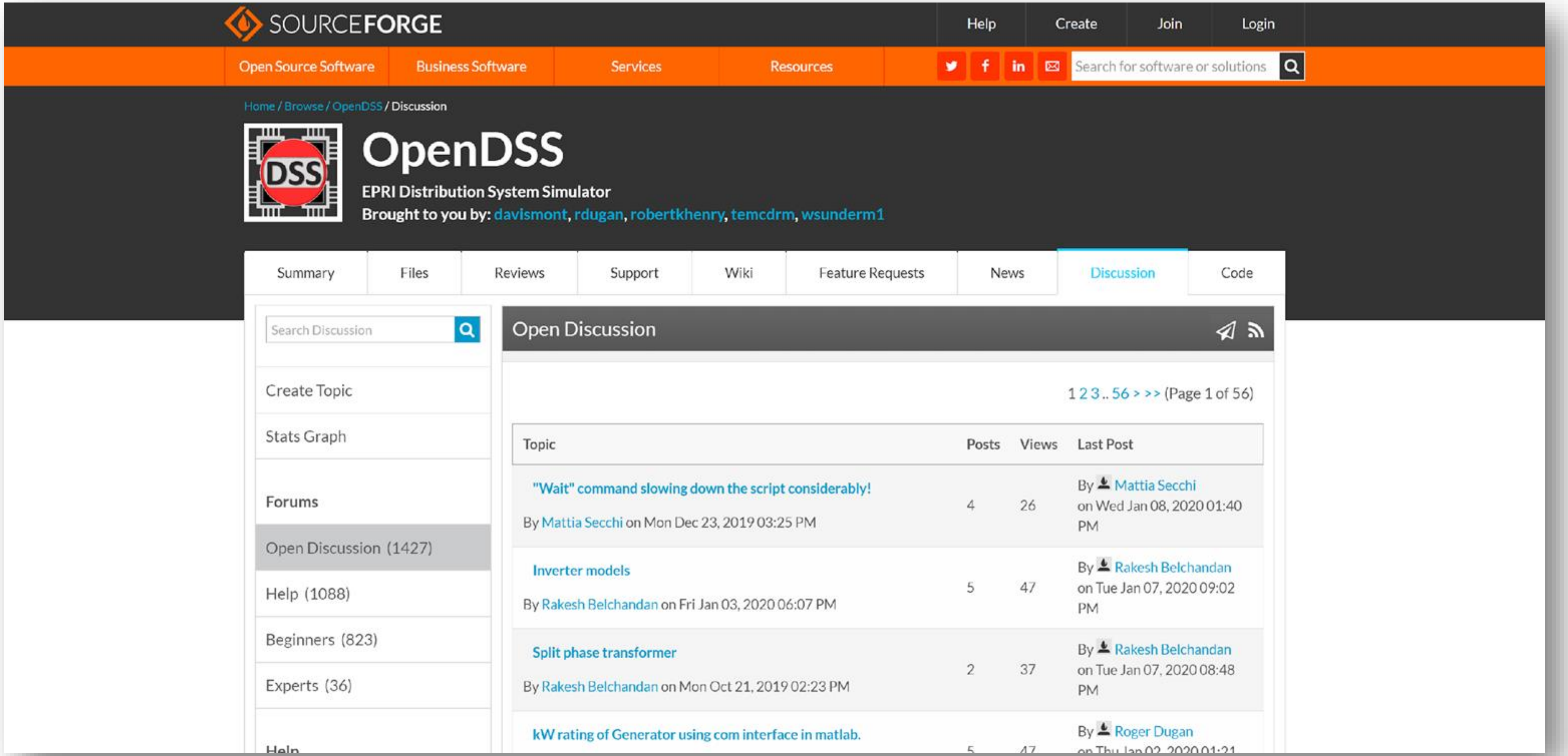
Download e Instalação

- [Version 9.4.0.6]



Fórum e Nosso grupo no Telegram

Fórum
Nosso grupo no Telegram



Recursos

- Pasta do OpenDSS: C:\Program Files\OpenDSS

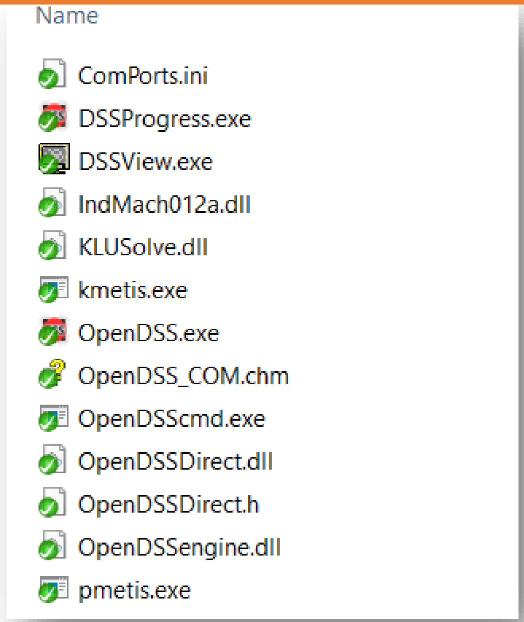
Program Files › OpenDSS

Name	Date modified	Type	Size
Doc	1/8/2020 10:44 AM	File folder	
EPRITestCircuits	1/8/2020 10:44 AM	File folder	
Examples	1/8/2020 10:44 AM	File folder	
IEEETestCases	1/8/2020 10:44 AM	File folder	
Training	1/8/2020 10:44 AM	File folder	
x64	1/8/2020 10:44 AM	File folder	
x86	1/8/2020 10:44 AM	File folder	
License.txt	3/19/2019 11:56 AM	Text Document	2 KB
readme.txt	10/11/2019 1:55 PM	Text Document	8 KB

Controlando o OpenDSS usando Linguagem de Programação

Versões

C:\Program Files\OpenDSS\x64



1. OpenDSS.exe Standalone EXE
2. OpenDSSEngine.dll *In-process* COM server
3. OpenDSSDirect.dll Direct Call DLL
4. OpenDSSCmd.exe Standalone EXE sem interface

- Cada uma está disponível em 32 e 64-bit
- KLUSolve.dll, “sparse matrix solver”
- DSSView.exe, programa separado para processamento de saídas gráficas

Por que controlar o OpenDSS por meio de outras linguagens de programação?

Implementar algoritmos não presentes no OpenDSS

Desenvolver novos elementos

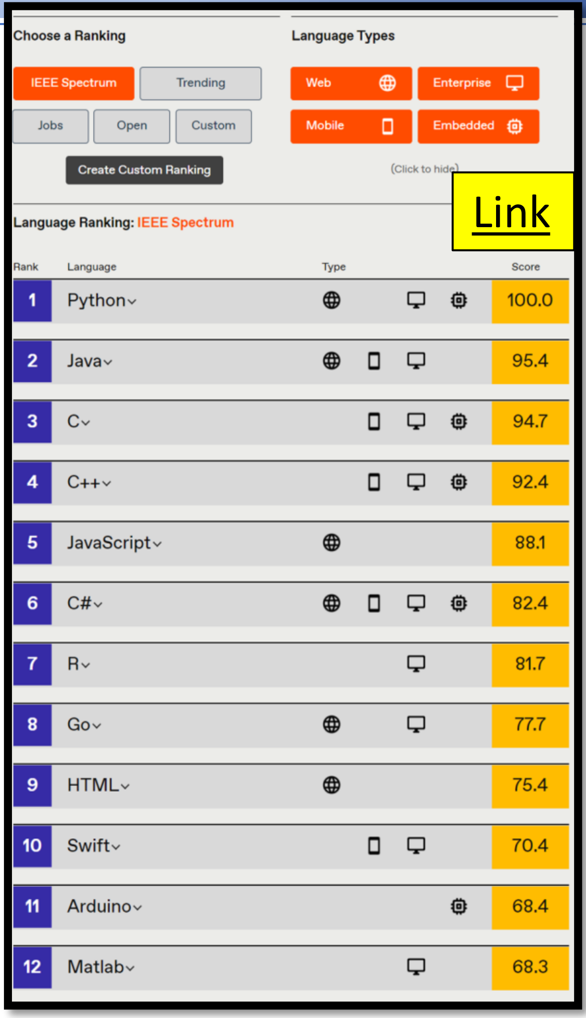
Automatizar tarefas repetitivas e/ou manuais

Obter resultados relevantes

Quais linguagens de programação podem ser usadas?

- Excel VBA
- VB.net
- C#
- C/C++
- Delphi, Free Pascal
- MATLAB
- Python
- Java
- LabView
- R
- Fortran (for DLLs, with DirectDLL)
- Julia (with DirectDLL)

Oportunidade de usar a linguagem de sua preferência



Choose a Ranking: IEEE Spectrum | Trending | Jobs | Open | Custom | Create Custom Ranking

Language Types: Web | Enterprise | Mobile | Embedded (Click to hide)

Link

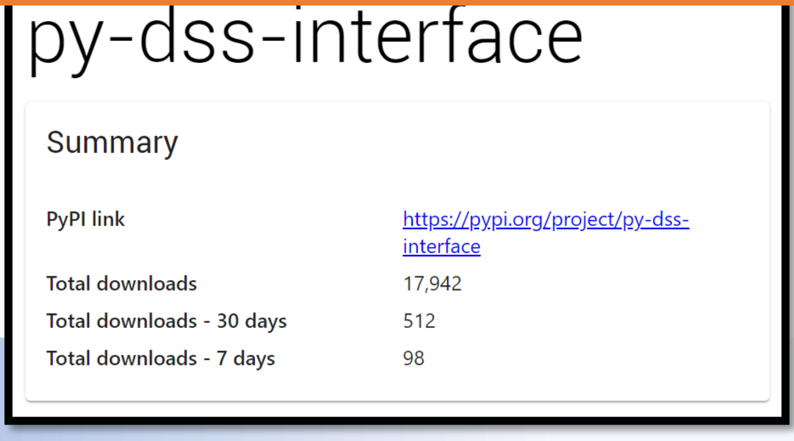
Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	JavaScript		88.1
6	C#		82.4
7	R		81.7
8	Go		77.7
9	HTML		75.4
10	Swift		70.4
11	Arduino		68.4
12	Matlab		68.3

Python integration: py-dss-interface

- Pacote Python
- Usa a versão DirectDLL official do OpenDSS
  - OpenDSS Version 9.3.0.1 vem com o pacote
  - Usuário pode usar sua versão do OpenDSS
- Criado a partir do documento "Direct connection Shared Library (DLL) for OpenDSS" feito pelo Davis Montenegro
- Instalando do PyPI
  - *pip install py-dss-interface*

https://pepy.tech/project/py-dss-interface



Examples

SBSE2022
IX Simpósio Brasileiro de Sistemas Elétricos

Demonstração

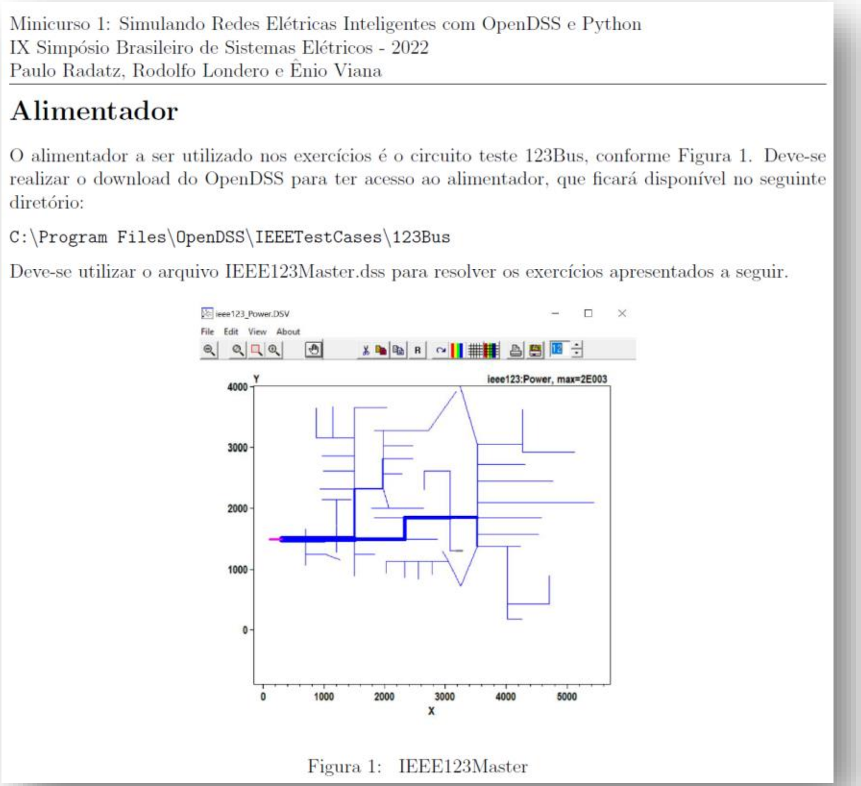
Conteúdo

- OpenDSS standalone
  - Abrir um alimentador
  - Rodar uma simulação estática
    - Obter resultados
  - Rodar uma simulação temporal
    - Obter resultados dos medidores e monitores
    - Demand Interval
- OpenDSS + Python
  - Criar o objeto dss
  - O método **text**
  - Lendo e escrevendo propriedades
  - Lendo resultados

Hands-on

Explicação do Hands-on

- O Hands-on é dividido em duas partes:
  - Simulação Estática
  - Simulação Temporal
- Responder usando OpenDSS e OpenDSS+Python:
  - Simulação Estática
    - Parte 3 (Não precisa F e G do item 4)
  - Simulação Temporal
    - Parte 3
    - Parte 4



Minicurso 1: Simulando Redes Elétricas Inteligentes com OpenDSS e Python
IX Simpósio Brasileiro de Sistemas Elétricos - 2022
Paulo Radatz, Rodolfo Londero e Ênio Viana

Alimentador

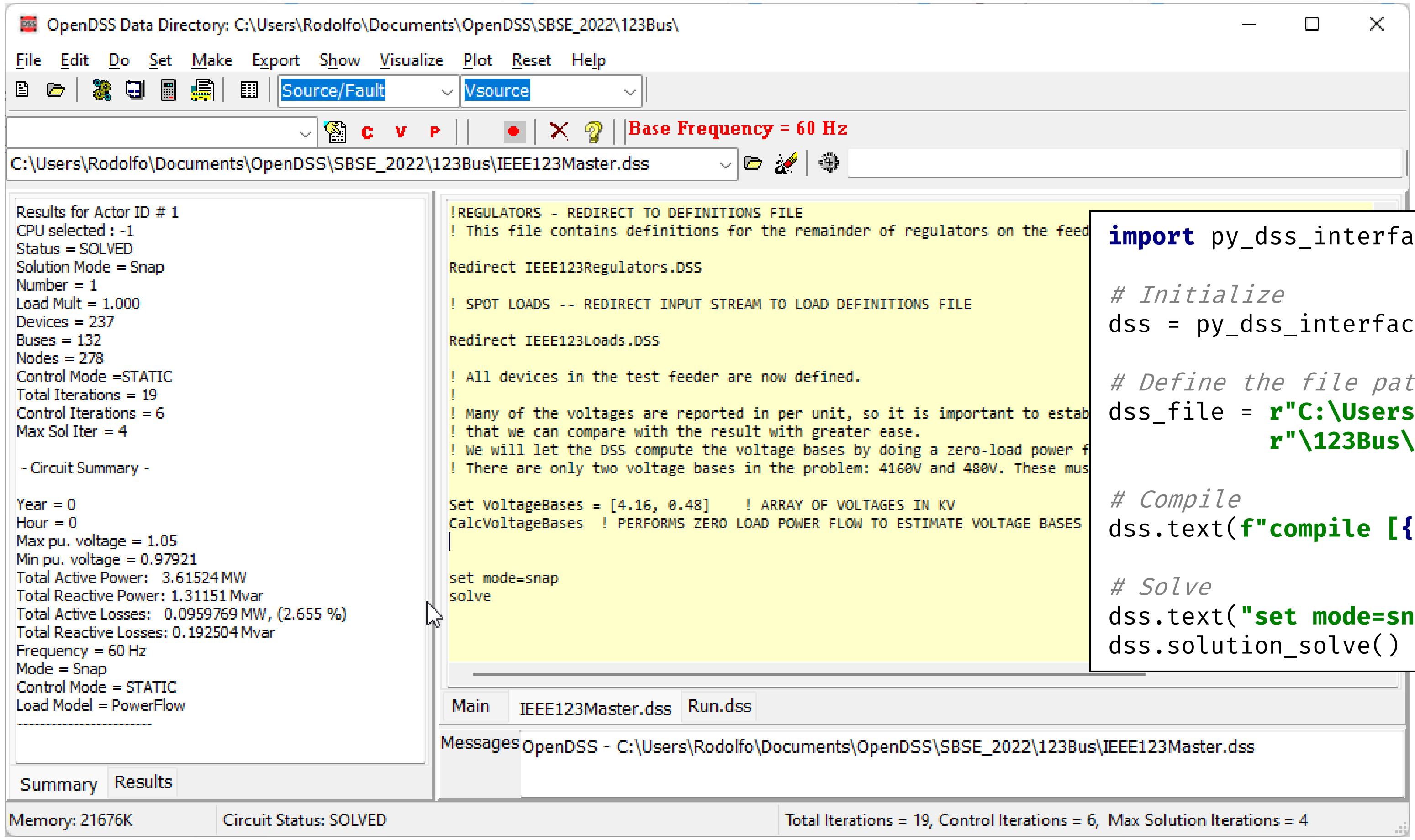
O alimentador a ser utilizado nos exercícios é o circuito teste 123Bus, conforme Figura 1. Deve-se realizar o download do OpenDSS para ter acesso ao alimentador, que ficará disponível no seguinte diretório:

`C:\Program Files\OpenDSS\IEEETestCases\123Bus`

Deve-se utilizar o arquivo IEEE123Master.dss para resolver os exercícios apresentados a seguir.

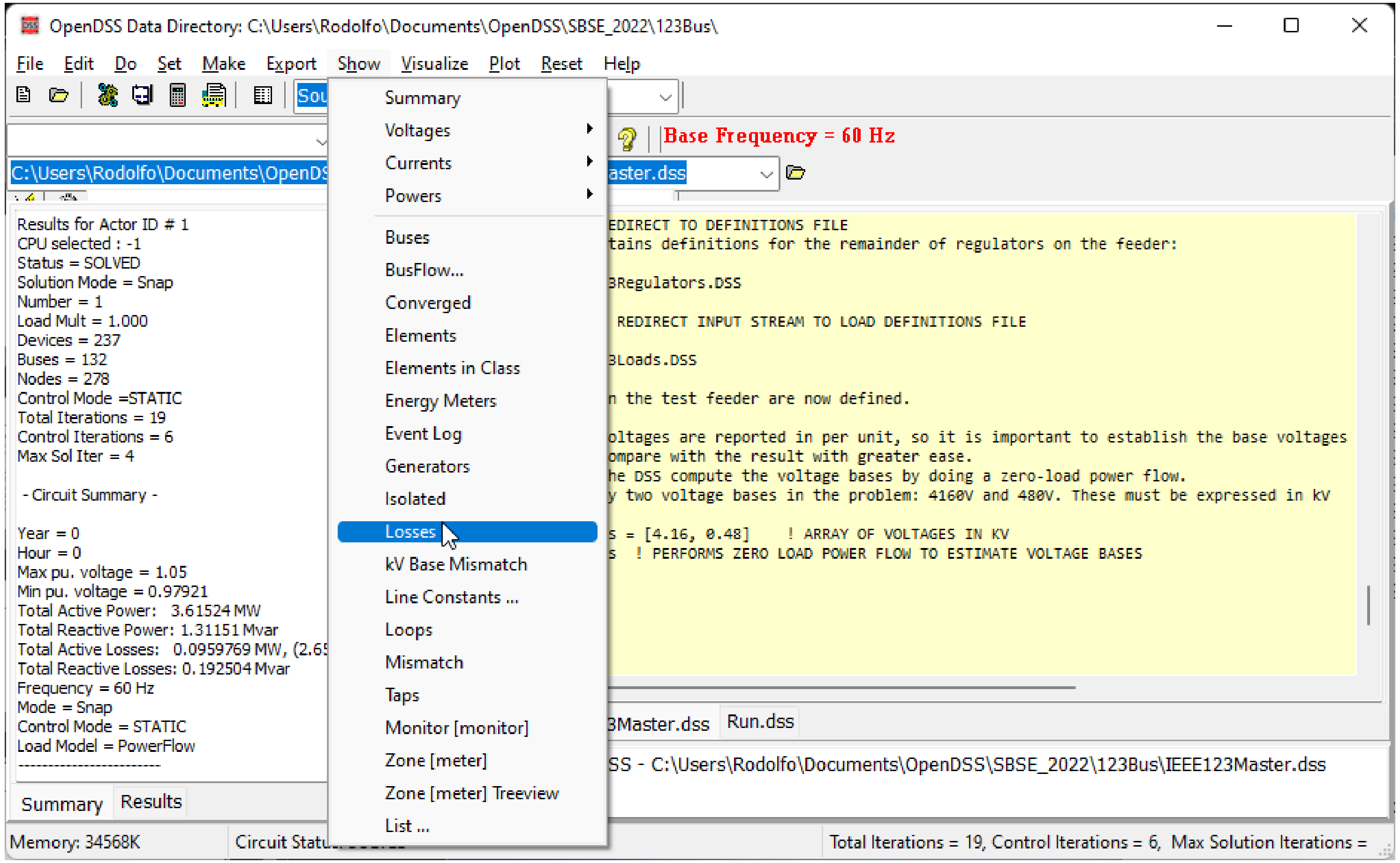
Figura 1: IEEE123Master

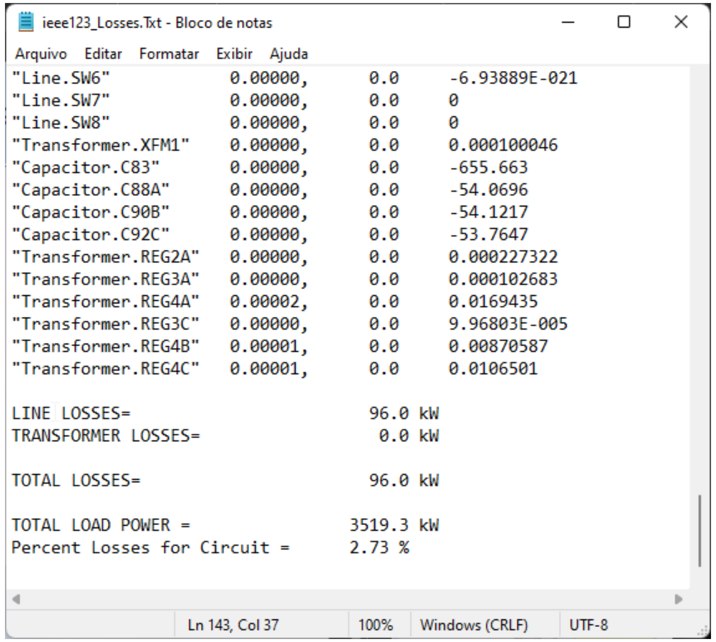
Dicas – Simulação Estática - Summary




```python
import py_dss_interface

# Initialize
dss = py_dss_interface.DSSDLL()

# Define the file path
dss_file = r"C:\Users\Rodolfo\Documents\OpenDSS\SBSE_2022\Exemplo_Python" \
           r"\123Bus\IEEE123Master.dss"

# Compile
dss.text(f"compile [{dss_file}]")

# Solve
dss.text("set mode=snap")
dss.solution_solve()
```

Dicas – Simulação Estática – Perdas



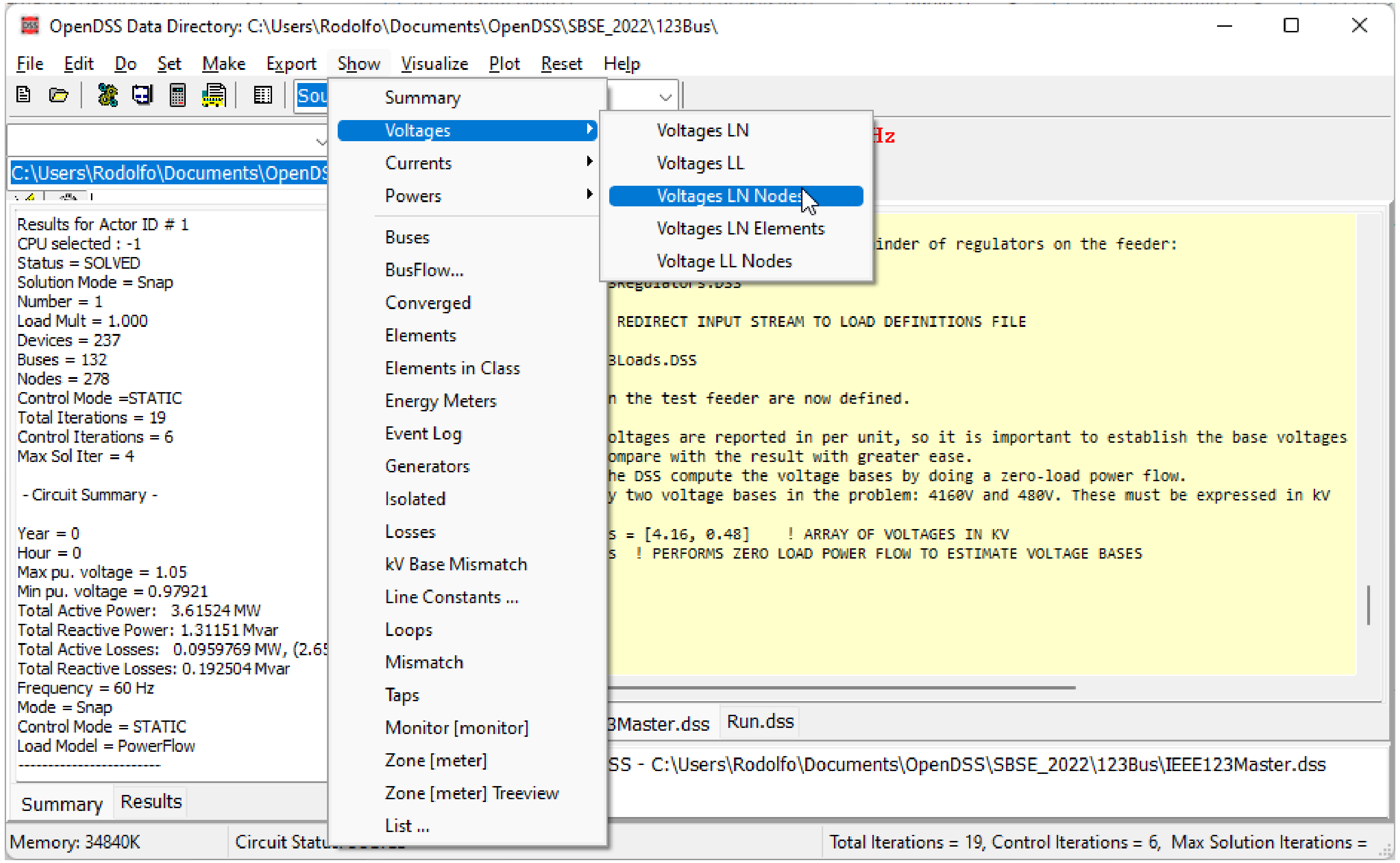



```python
total_losses = dss.circuit_losses()
total_losses_active = total_losses[0]
total_losses_reactive = total_losses[1]

print("\nLosses")
print(f"Active Losses: {total_losses_active / 1000} kW")
print(f"Reactive Losses: {total_losses_reactive / 1000} kvar")
```

Dicas – Simulação Estática – Perdas

```python
# Getting losses
total_losses = dss.circuit_losses()
total_losses_active = total_losses[0]
total_losses_reactive = total_losses[1]

print("\nLosses")
print(f"Active Losses: {total_losses_active / 1000} kW")
print(f"Reactive Losses: {total_losses_reactive / 1000} kvar")

# Getting Lines and Transformers losses
line_losses = dss.circuit_line_losses()
line_losses_active = line_losses[0]
line_losses_reactive = line_losses[1]

print("\nLine losses")
print(f"Line Active Losses: {line_losses_active} kW")
print(f"Line Reactive Losses: {line_losses_reactive} kvar")

transformer_losses = total_losses_active/1000 - line_losses_active
print("\nTransformer losses")
print(f"Transformer Active Losses: {transformer_losses} kW")
```

Dicas – Simulação Estática – Tensões



ieee123_VLN_Node.Txt - Bloco de notas

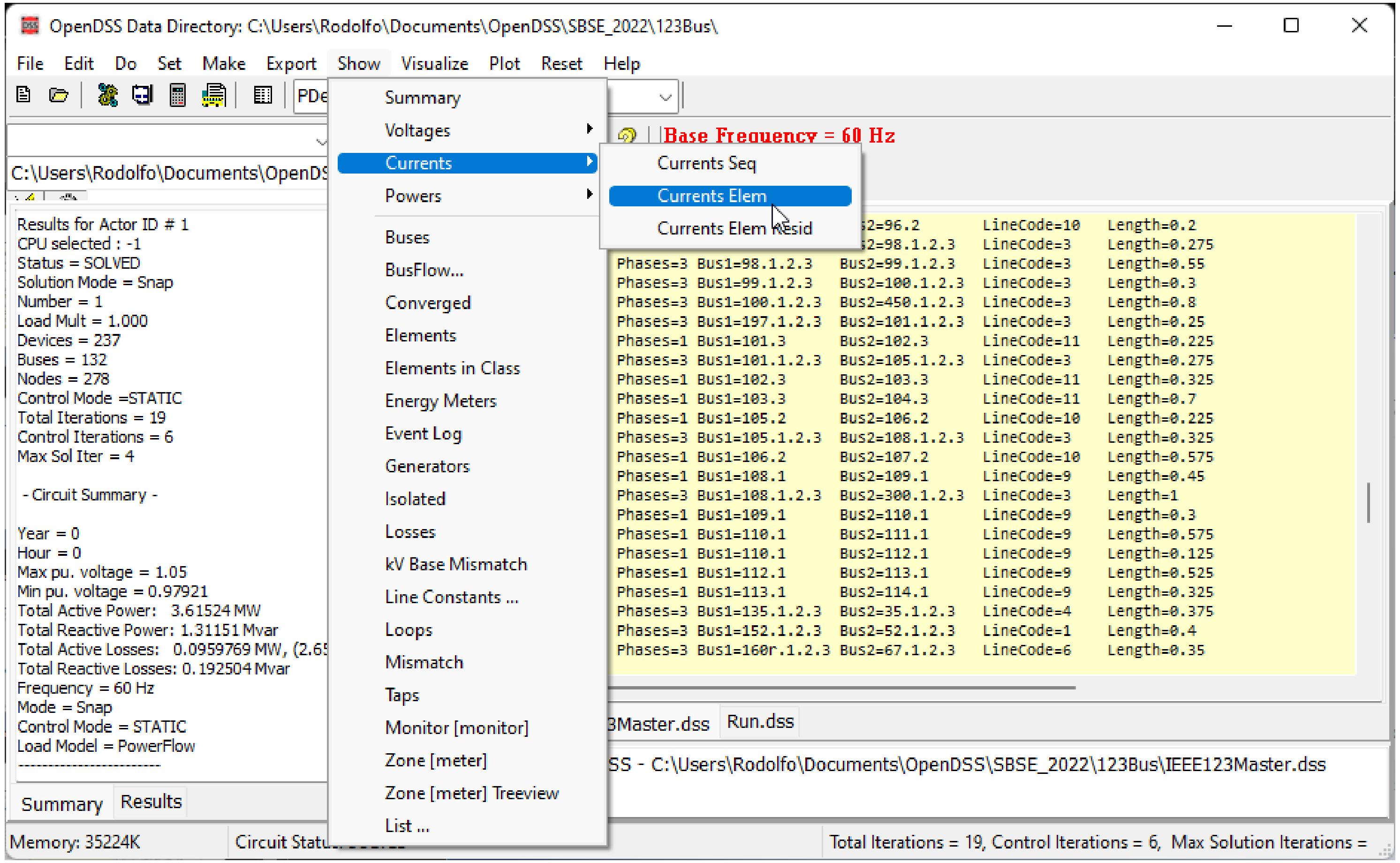
Arquivo Editar Formatar Exibir Ajuda

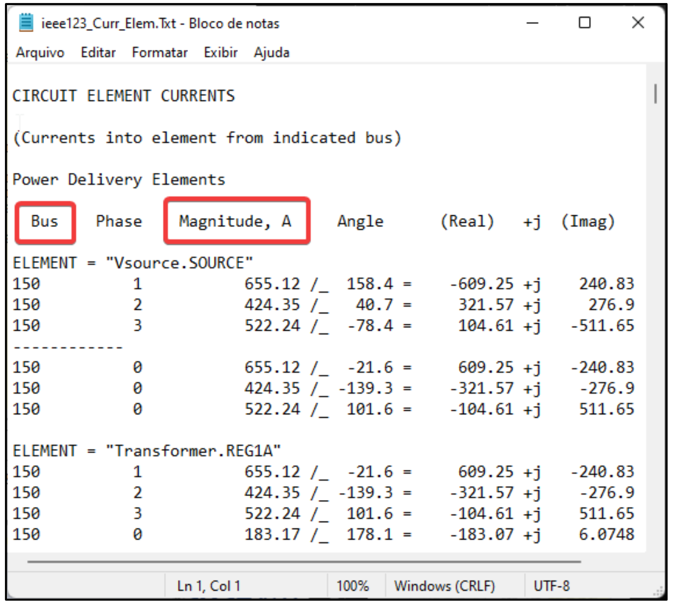
LINE-GROUND and LINE-LINE VOLTAGES BY BUS & NODE

Bus	Node	VLN (kV)	Angle	pu	Base kV	Node-Node
150	1	2.4018 /_	0.0	0.99999	4.160	1-2
-	2	2.4018 /_	-120.0	0.99999	4.160	2-3
-	3	2.4018 /_	120.0	0.99999	4.160	3-1
150R ...	1	2.4918 /_	0.0	1.0375	4.160	1-2
-	2	2.4918 /_	-120.0	1.0375	4.160	2-3
-	3	2.4918 /_	120.0	1.0375	4.160	3-1
149	1	2.4918 /_	0.0	1.0375	4.160	1-2
-	2	2.4918 /_	-120.0	1.0375	4.160	2-3
-	3	2.4918 /_	120.0	1.0375	4.160	3-1
1	1	2.4617 /_	-0.6	1.025	4.160	1-2
-	2	2.4861 /_	-120.3	1.0351	4.160	2-3
-	3	2.4704 /_	119.6	1.0286	4.160	3-1
2	2	2.4856 /_	-120.3	1.0349	4.160	
3	3	2.4665 /_	119.6	1.0269	4.160	
7	1	2.4393 /_	-1.1	1.0156	4.160	1-2
-	2	2.4821 /_	-120.6	1.0334	4.160	2-3

```python
# Bus 152
dss.circuit_set_active_bus("152")

voltages_nodes = dss.bus_vmag_angle()
vmag1 = voltages_nodes[0]
vmag2 = voltages_nodes[2]
vmag3 = voltages_nodes[4]

voltages_nodes_pu = dss.bus_pu_vmag_angle()
v1pu = voltages_nodes_pu[0]
v2pu = voltages_nodes_pu[2]
v3pu = voltages_nodes_pu[4]
```

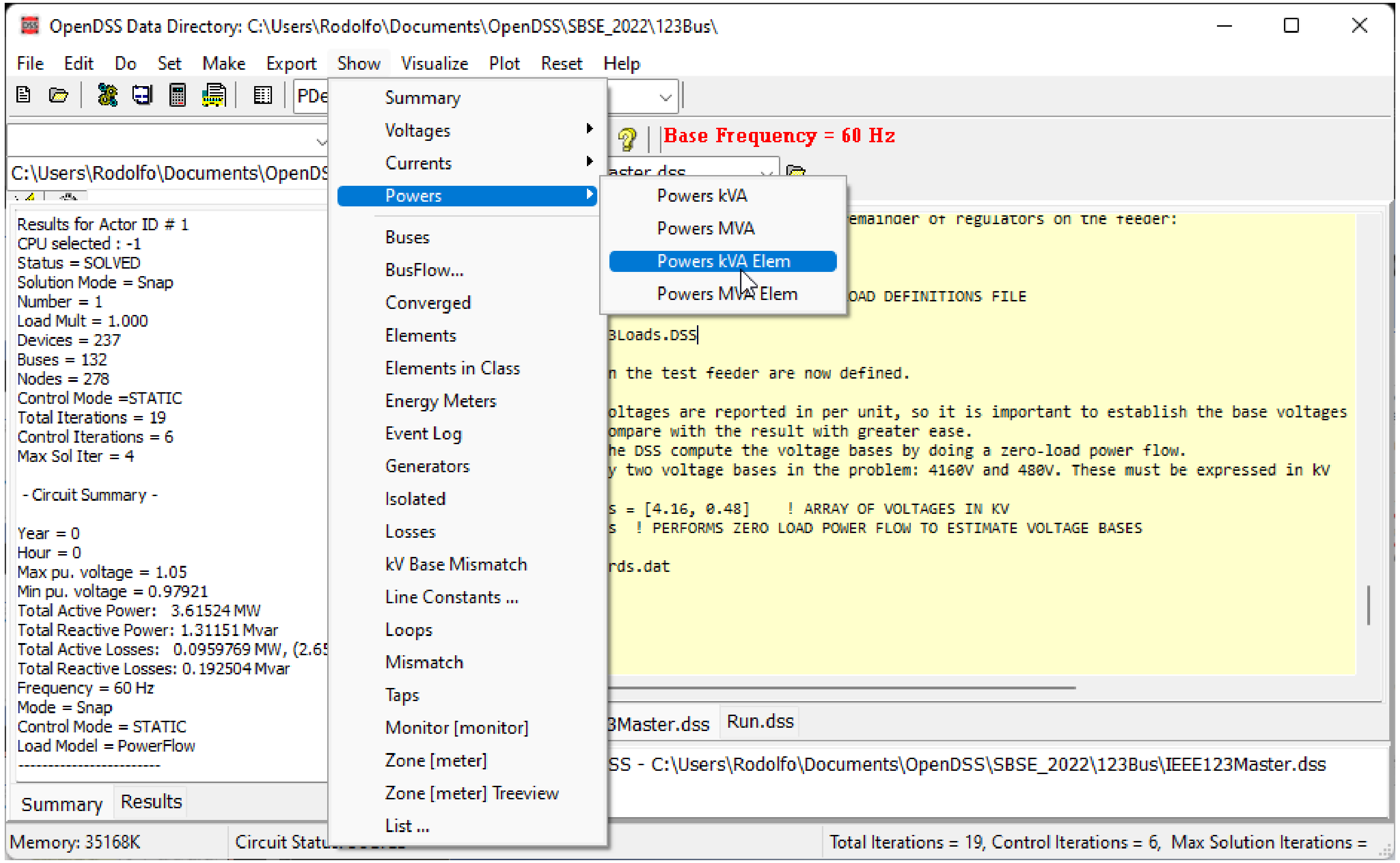
Dicas – Simulação Estática – Correntes

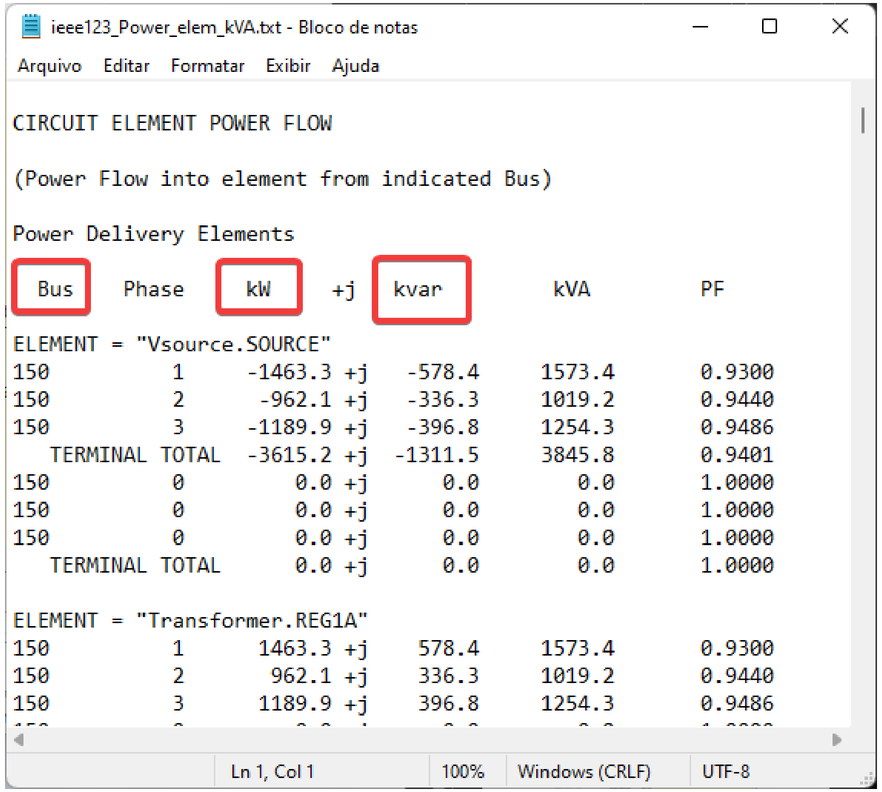




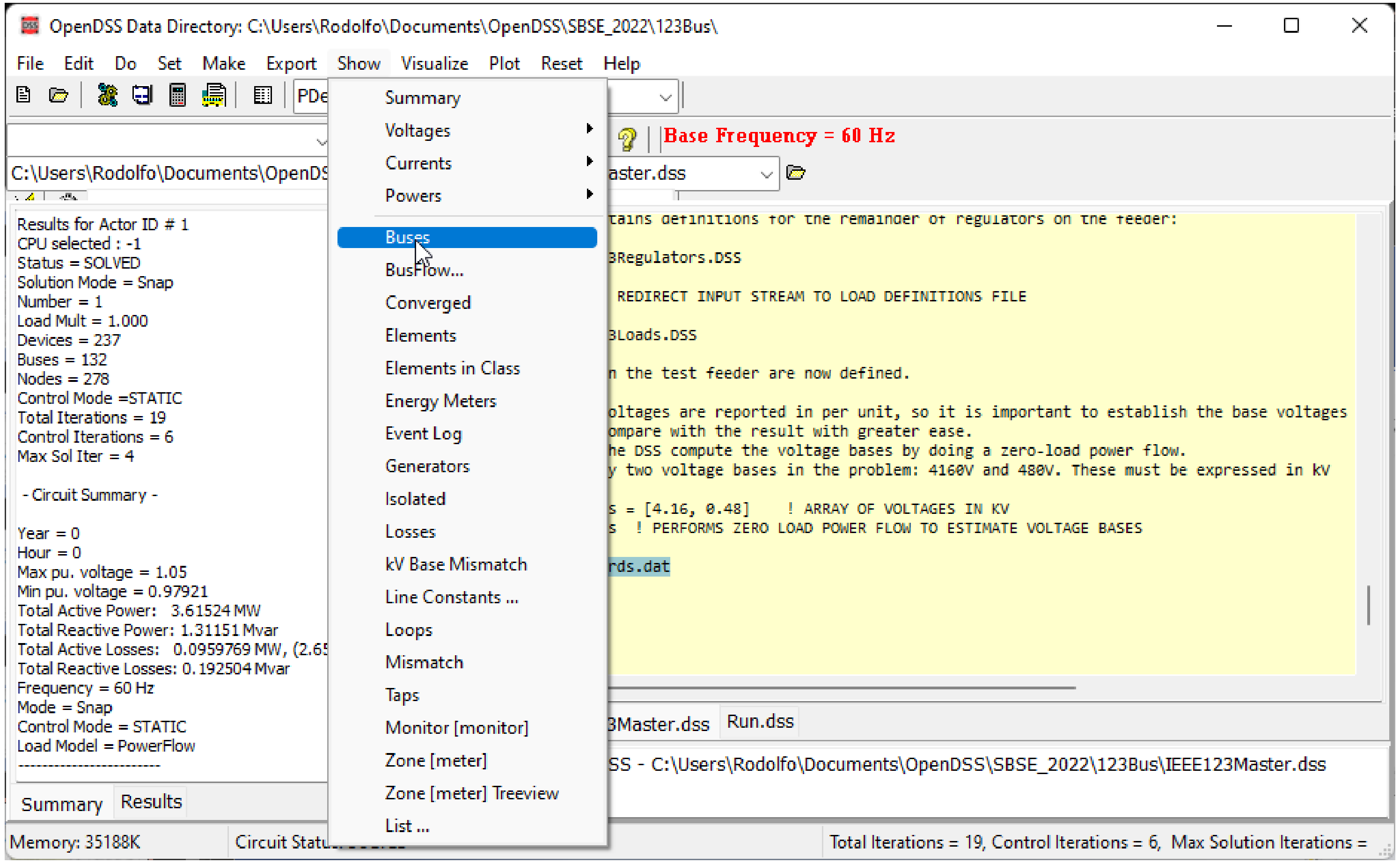

```
dss.circuit_set_active_element(dss.bus_line_list()[0])
bus_currents = dss.cktelement_currents_mag_ang()[:6][::2]
bus_active_power = dss.cktelement_powers()[:6][0::2]
bus_reactive_power = dss.cktelement_powers()[:6][1::2]
```

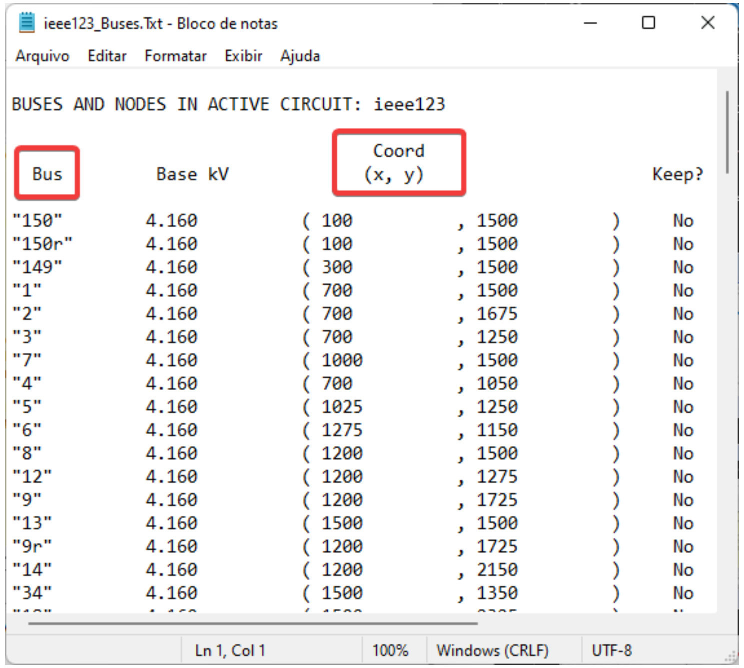
Dicas – Simulação Estática – Potências





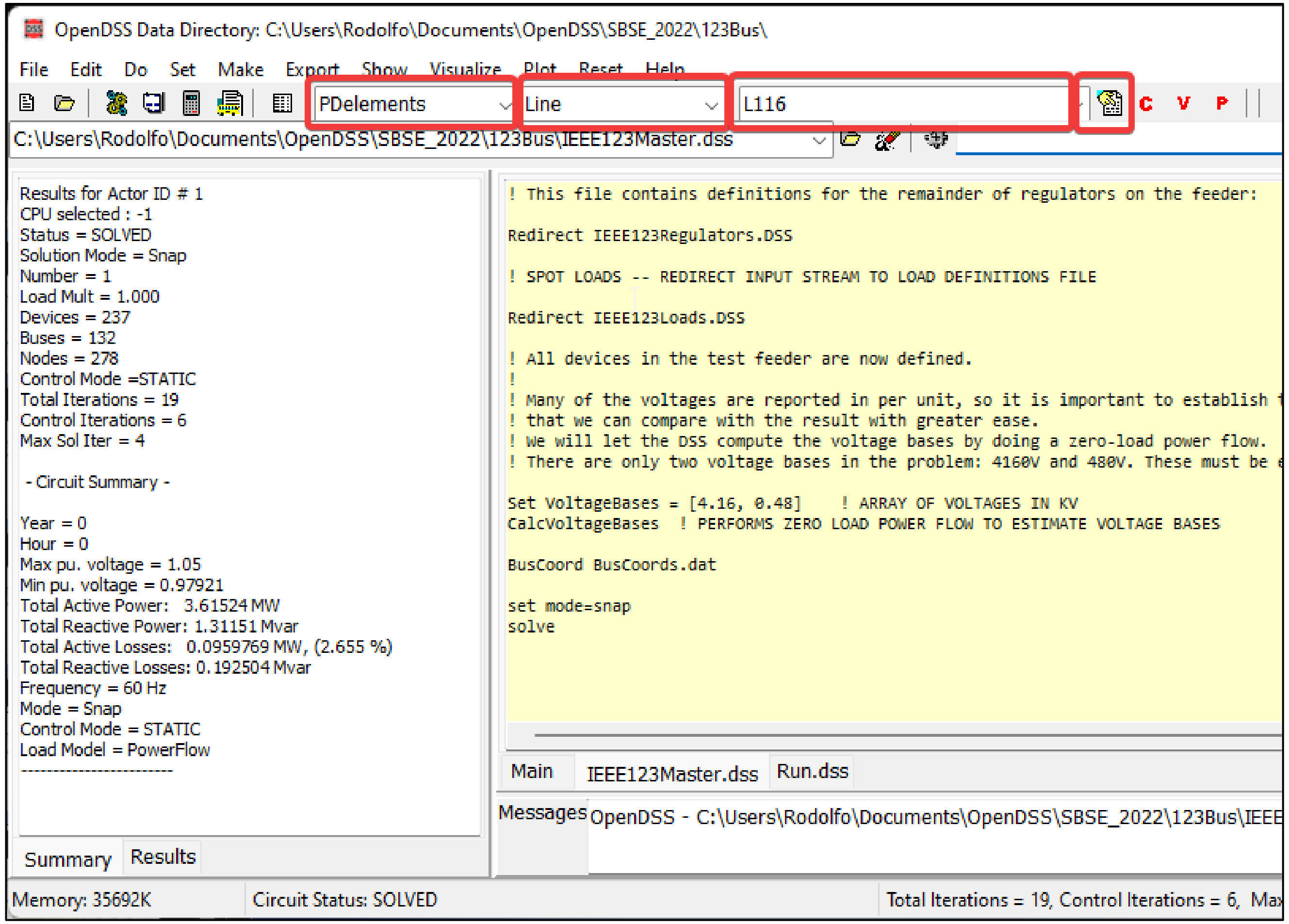
Dicas – Simulação Estática – Potências





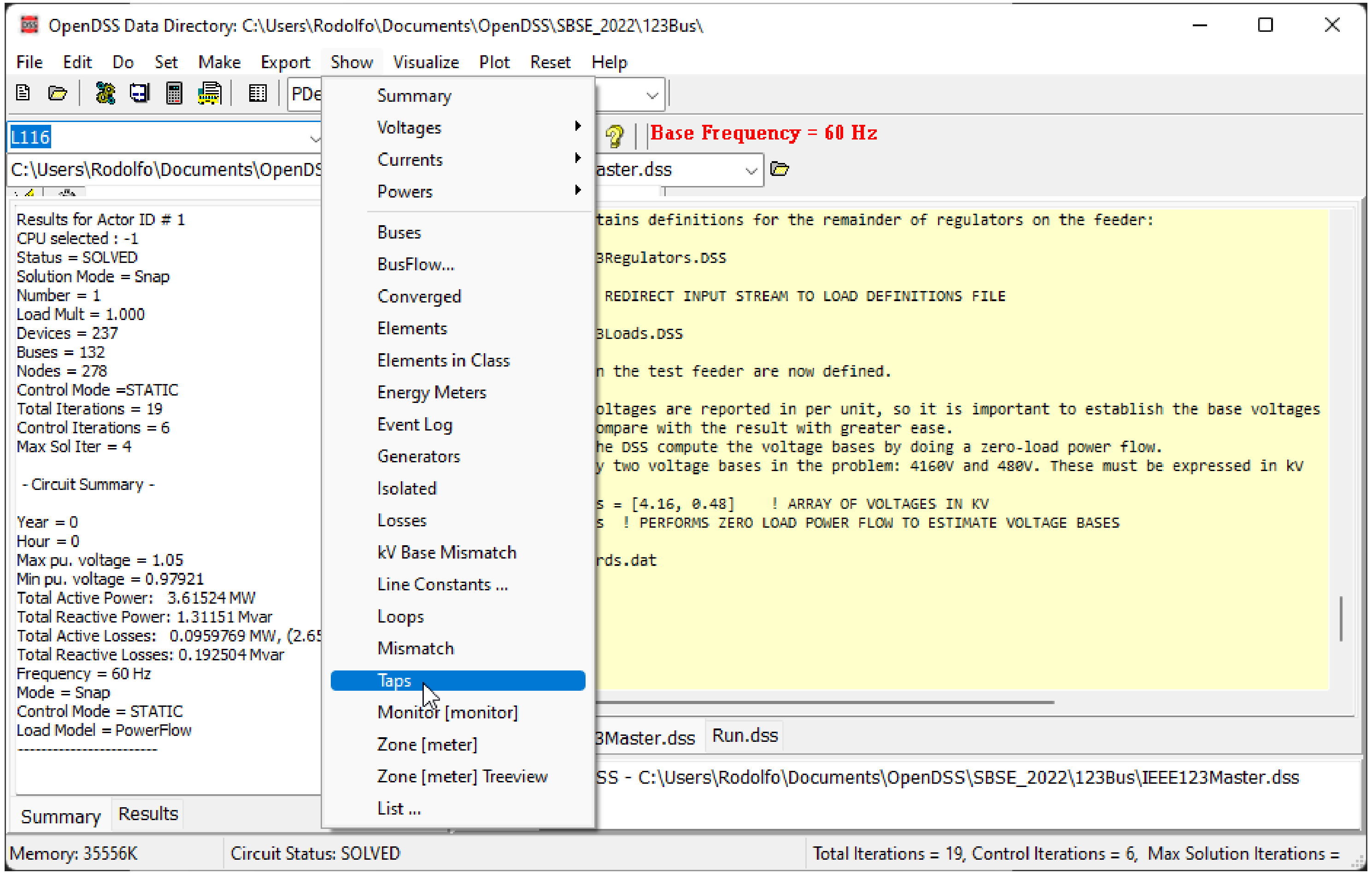

```
bus_x = dss.bus_read_x()
bus_y = dss.bus_read_y()
```

Dicas – Simulação Estática – Linha



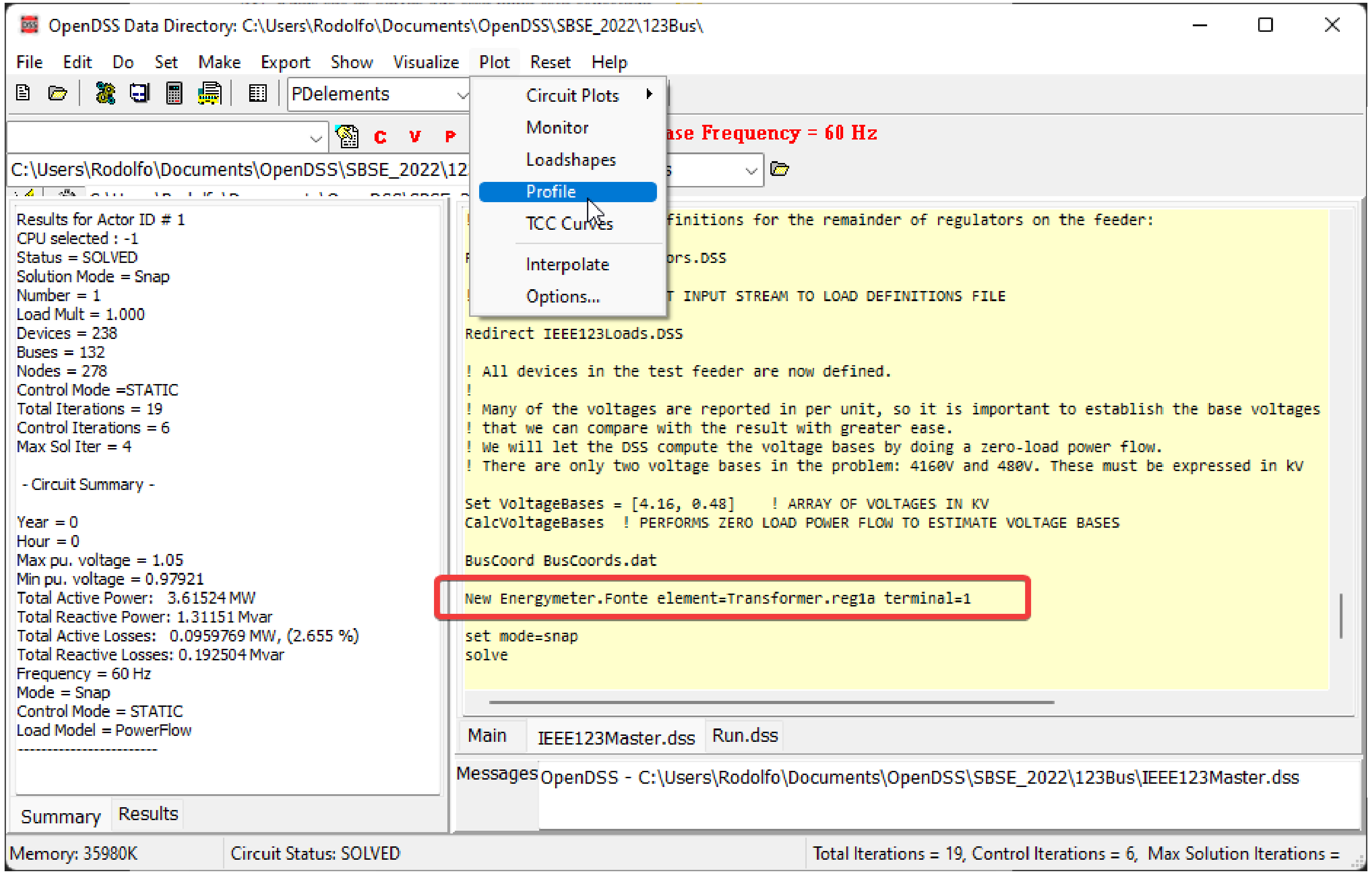

```python
# Line L116
dss.lines_write_name("L116")
bus1 = dss.lines_read_bus1()
bus2 = dss.lines_read_bus2()
linecode = dss.lines_read_linecode()

rmatrix = dss.lines_read_rmatrix()
xmatrix = dss.lines_read_xmatrix()
cmatrix = dss.lines_read_cmatrix()

dss.circuit_set_active_bus(bus1.split(".")[0])
bus1_voltage_nodes = dss.bus_vmag_angle()[::2]

dss.circuit_set_active_bus(bus2.split(".")[0])
bus2_voltage_nodes = dss.bus_vmag_angle()[::2]

dss.circuit_set_active_element("Line.l116")
line_currents = dss.cktelement_currents_mag_ang()[:6][::2]
line_active_power = dss.cktelement_powers()[:6][::2]
line_reactive_power = dss.cktelement_powers()[:6][1::2]

voltages_terminal_1 = dss.cktelement_voltages_mag_ang()[::2][:3]
voltages_terminal_2 = dss.cktelement_voltages_mag_ang()[::2][3:]
```

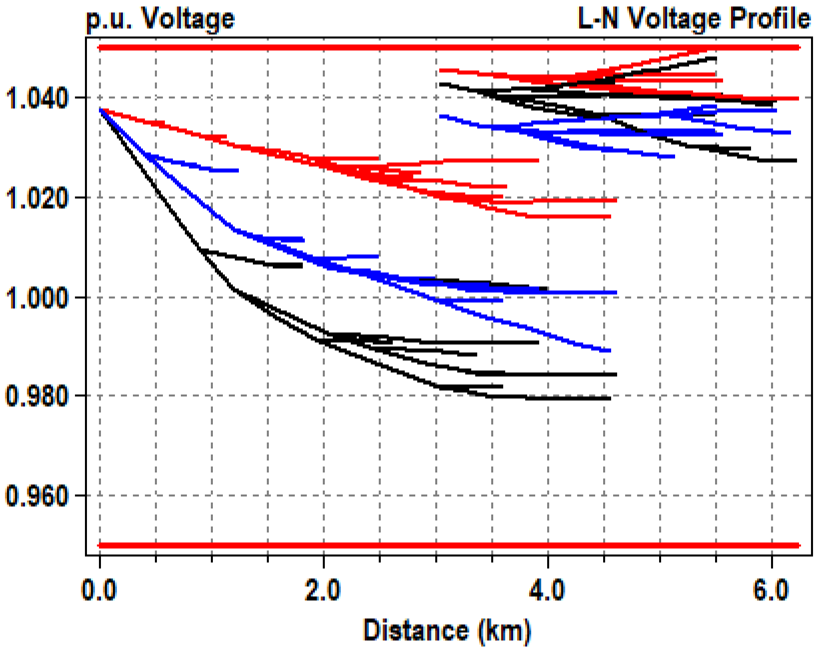
Dicas – Simulação Estática – Taps



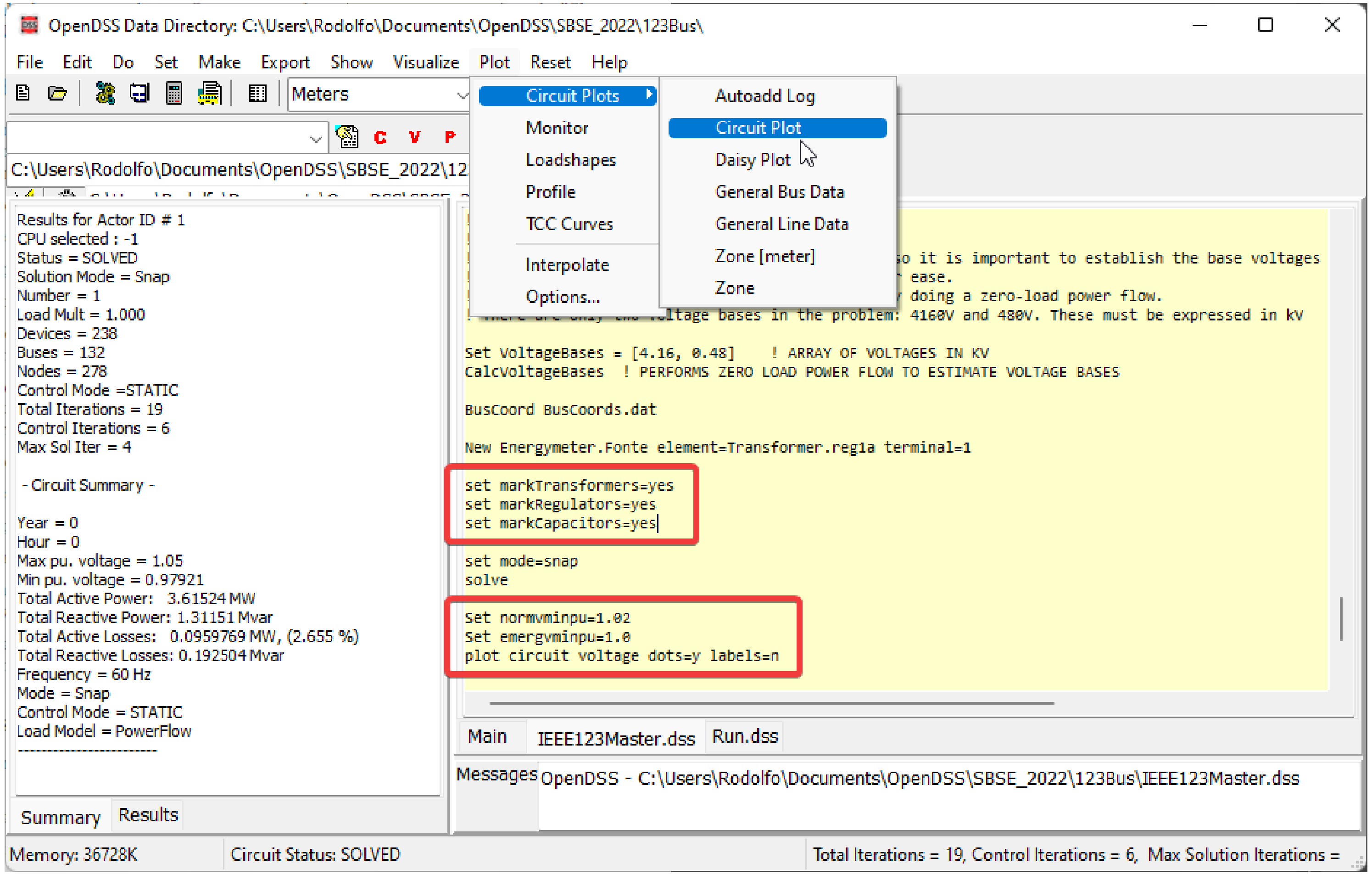

```python
dss.regcontrols_first()
for _ in range(dss.regcontrols_count()):
    print(f"{dss.regcontrols_read_name()} with tap number = {dss.regcontrols_read_tap_number()}")
    dss.regcontrols_next()
```

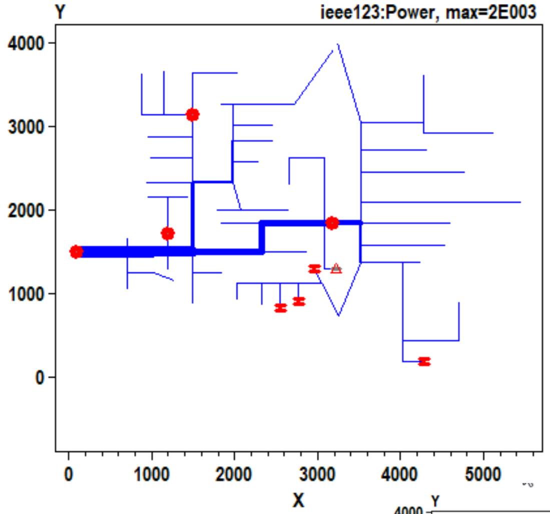
Dicas – Simulação Estática – Perfil de Tensão

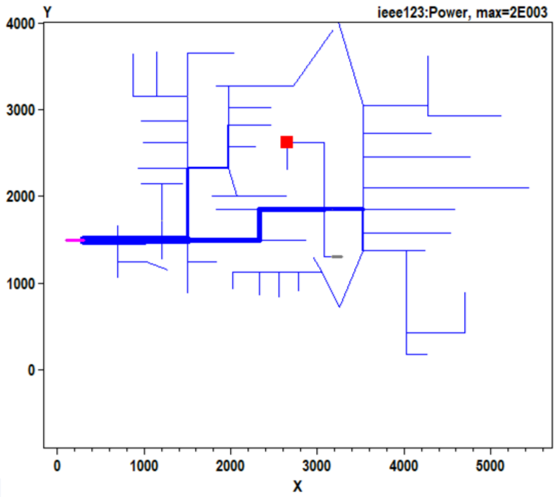




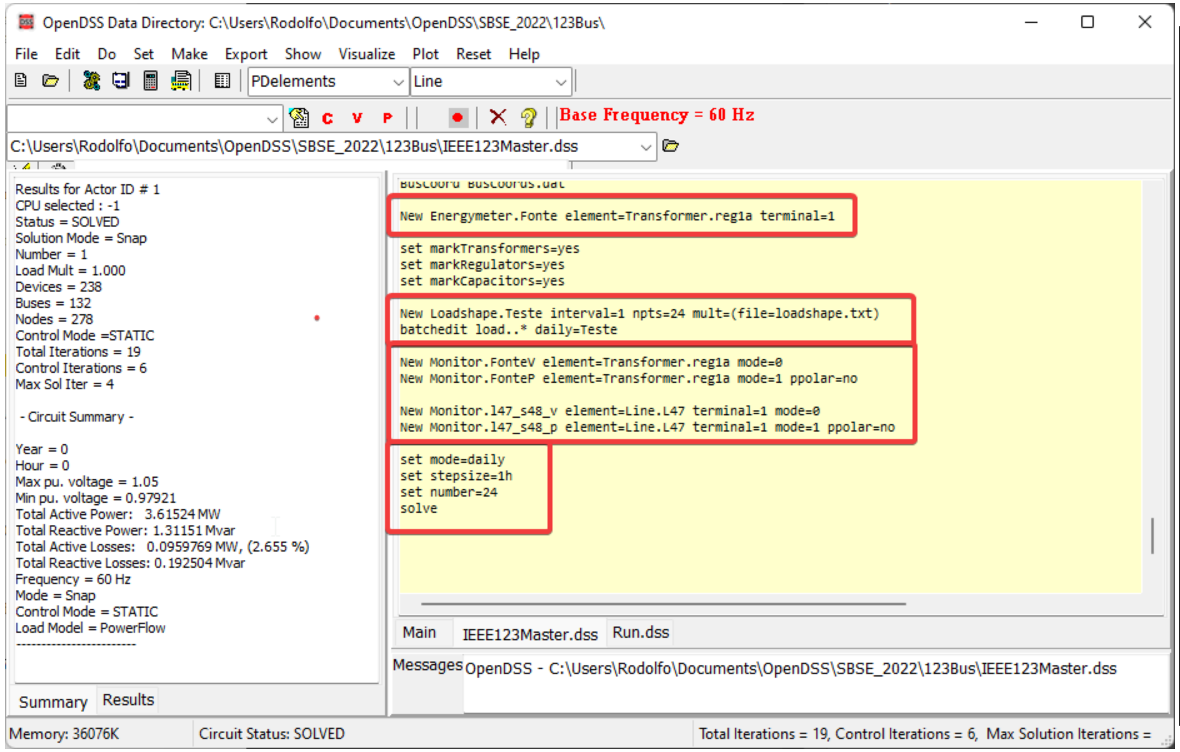
Dicas – Simulação Estática – Mapas







Dicas – Simulação Temporal




```python
import py_dss_interface
import pathlib

script_path = os.path.dirname(os.path.abspath(__file__))

dss_file = pathlib.Path(script_path).joinpath("../feeders", "123Bus",
"IEEE123Master.dss")

dss = py_dss_interface.DSSDLL()

dss.text(f"compile [{dss_file}]")

dss.text("New Loadshape.my_loadshape interval=1 npts=24
mult=(file=loadshape.txt)")

dss.text(f"batchedit load..* daily=my_loadshape")

dss.text("set mode=daily")
dss.solution_solve()
```

Dicas – Simulação Temporal - Energia

Criar:

- EnergyMeter
- Monitors

OpenDSS

```
Show meters
```

Python

```python
dss.meters_write_name("Feeder")
register_names = dss.meters_register_names()
register_values = dss.meters_register_values()
```

ieee123_EMout.Txt - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
ENERGY METER VALUES

Registers:
Reg 1 = kWh
Reg 2 = kvarh
Reg 3 = Max kW
Reg 4 = Max kVA
Reg 5 = Zone kWh
Reg 6 = Zone kvarh
Reg 7 = Zone Max kW
Reg 8 = Zone Max kVA
Reg 9 = Overload kWh Normal
Reg 10 = Overload kWh Emerg
Reg 11 = Load EEN
Reg 12 = Load UE
Reg 13 = Zone Losses kWh
Reg 14 = Zone Losses kvarh
Reg 15 = Zone Max kW Losses
Reg 16 = Zone Max kvar Losses
Reg 17 = Load Losses kWh
Reg 18 = Load Losses kvarh
Reg 19 = No Load Losses kWh
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

ieee123_EMout.Txt - Bloco de notas

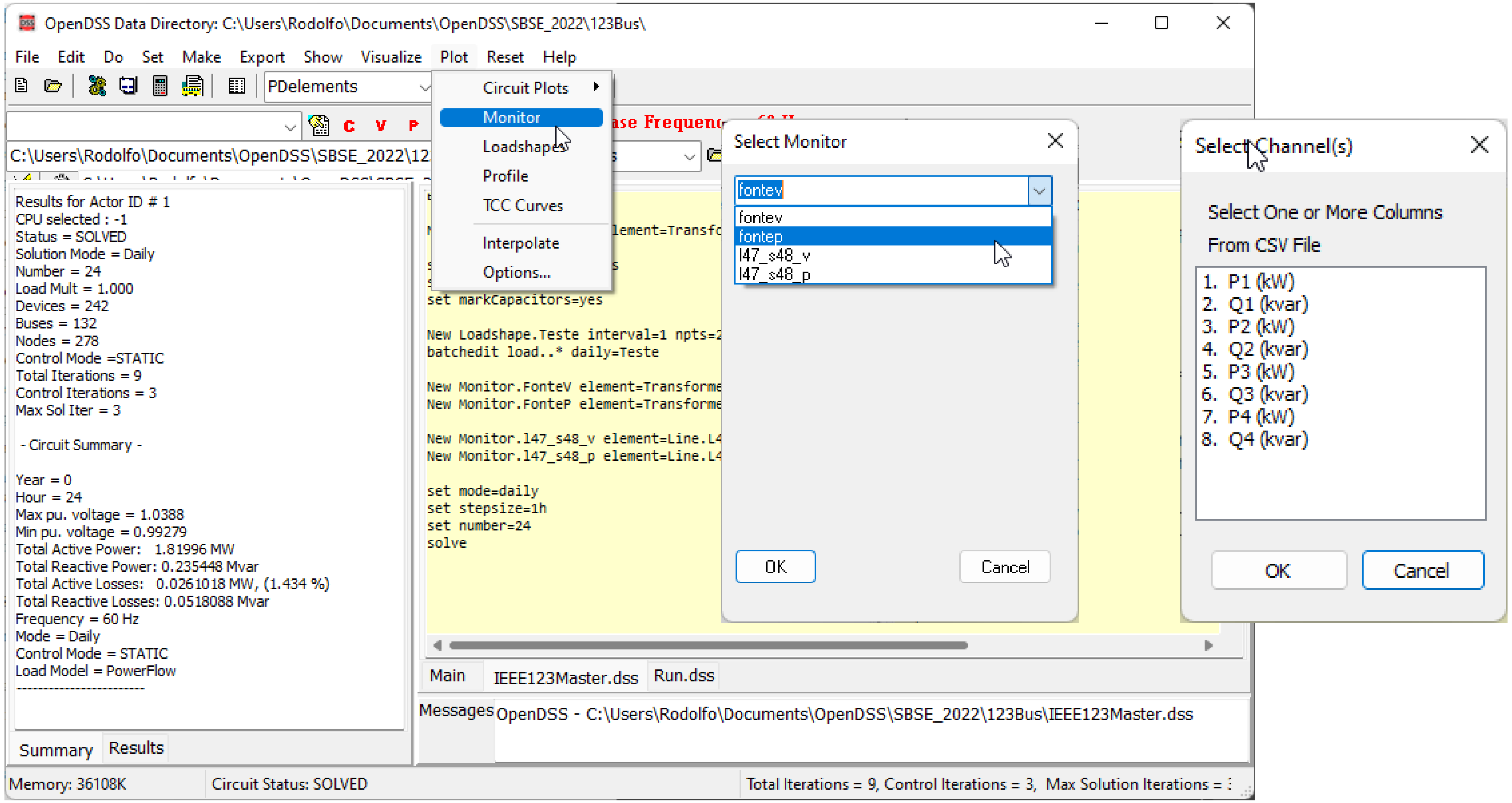
Arquivo Editar Formatar Exibir Ajuda

```
Reg 50 = Aux13
Reg 51 = Aux18
Reg 52 = Aux23
Reg 53 = Aux28
Reg 54 = 4.16 kV No Load Loss
Reg 55 = Aux4
Reg 56 = Aux9
Reg 57 = Aux14
Reg 58 = Aux19
Reg 59 = Aux24
Reg 60 = Aux29
Reg 61 = 4.16 kV Load Energy
Reg 62 = Aux5
Reg 63 = Aux10
Reg 64 = Aux15
Reg 65 = Aux20
Reg 66 = Aux25
Reg 67 = Aux30
```

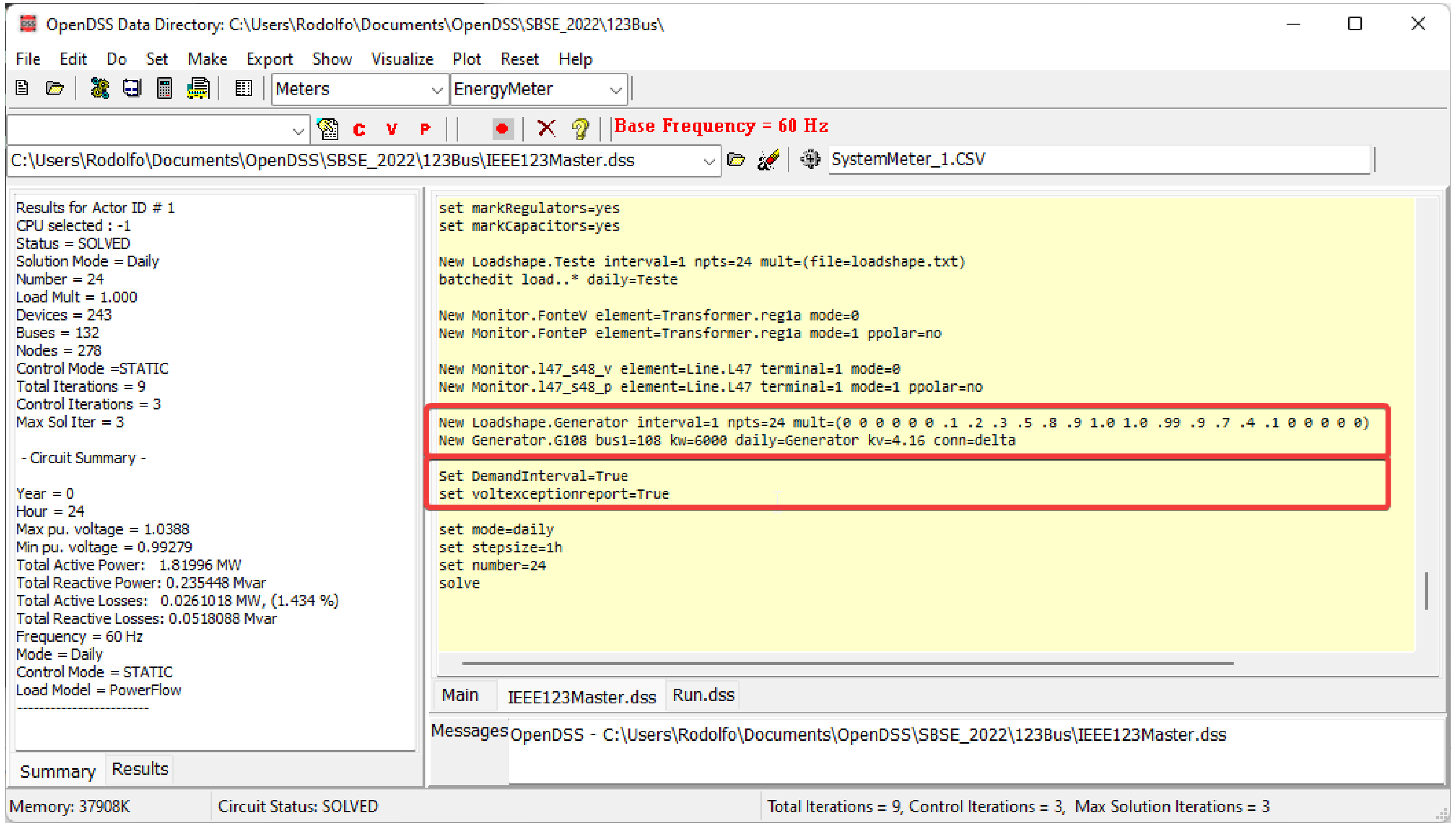
Meter	Reg 1	Reg 2	Reg 3	Reg 4	Reg 5
fonte	48834	8663	3613	3845	47964

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Dicas – Simulação Temporal – Plot



Dicas – Simulação Temporal – Sobretensões



OpenDSS

```
edit generator.G108 pf=-valor
```

Python

```
dss.generators_write_pf(-valor)
```