Introduction to The Next Generation of Distribution Analysis Tools

Summer course D2

Davis Montenegro Martinez Ph.D.
Engineer/Scientist III

Universidad de los Andes
Bogotá, Colombia
June 4 – 7, 2019

Electric Power Research Institute - EPRI

Together...Shaping the Future of Electricity

Advancing safe, reliable, affordable, and environmentally responsible electricity for society through global collaboration, thought leadership, and science and technology innovation

Areas of focus

3420 Hillview Avenue, Palo Alto, California 94304-1338 • PO Box 10412, Palo Alto, California 94303-0813 USA
800.313.3774 • 650.855.2121 • askepri@epri.com • www.epri.com

The instructor

Davis Montenegro, *Member, IEEE*

Davis Montenegro-Martinez serves as Engineer/Scientist III at the Electric Power Research Institute (EPRI) in the areas of power system modeling, analysis and high performance computing. He received his B.Sc. degree in electronics engineering from Universidad Santo Tomás, Bogotá, Colombia (2004); he is M.Sc. in electrical engineering from Universidad de los Andes, Bogotá , Colombia (2012). He received his Ph.D. in electrical engineering from Universidad de los Andes (2015), and a Ph.D. in electrical engineering from the University Grenoble-Alpes, France (2015).

Before joining EPRI, Davis served for 10 years as a lecturer for Universidad Santo Tomas in Colombia, during this time he was also technology consultant in the areas of industrial automation, software and electronic hardware design focused in the electric power industry, specifically in monitoring and control for meter calibration laboratories. His expertise in parallel computing techniques is being used at EPRI for incorporating multi-core processing to power system analysis methods such as QSTS, reducing the computational time required to perform these analysis using standard computing architectures.

Dr. Montenegro is also a member of the International Council on Large Electric Systems CIGRE, he was awarded with the IEEE 2016 I&CPS Ralph H. Lee Department Prize Paper Award at the 2017 I&CPS Technical Conference Awards luncheon in Niagara Falls, ON, Canada, for the paper titled "Energy Storage Modeling for Distribution Planning." He was also awarded an IEEE recognition in 2017 for notable services and contributions towards the advancement of IEEE and the engineering professions chairing of IM09 (Instrumentation and Measurements) Society Chapter, Colombian Section 2015–2017.

Custom scripting

Custom scripting

- Snapshot solution mode
- This is the default solution mode
- Attempts one solution for each “solve”
- Solves the circuit “as is”
- If you want something done, you have to specifically tell it
  - Set Load and Generator kW, etc.
    - Load.MyLoad.kW=125
    - Loadshapes are not used in this mode!
  - Sample Monitors and meters
    - Solve
    - Sample

Custom scripting

- Snapshot solution mode

```
! Start the ramp down at 1 sec
Set sec=1
Generator.PV1.kW=(2500 250 -)
Solve
Sample
Set sec=2
Generator.PV1.kW=(2500 500 -)
Solve
Sample

Set sec = 2.020834372 ! Unit 1
storage.jo0235001304 state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.022028115 ! Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.023158858 ! Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.024604602 ! Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
Sample
Set sec = 2.025738325 ! Unit 5
storage.jo0235000268_2.dispmode=discharging %discharge=11.9
Solve
Sample

Etc.
```

Set Gen kW explicitly each time step

Solve and Sample explicitly at each step

Set each Storage unit discharge rate explicitly each time step

(time is used for recording purposes only)

Custom scripting

- Time solution mode
  - Similar to Snapshot mode EXCEPT:
    - Loads, Generators can follow a selected Loadshape
      - Duty, Daily, or Yearly
    - Monitors are automatically sampled
      - But not Energymeters; do that explicitly if desired
    - Time is automatically incremented AFTER solve

Custom scripting

- Time solution mode

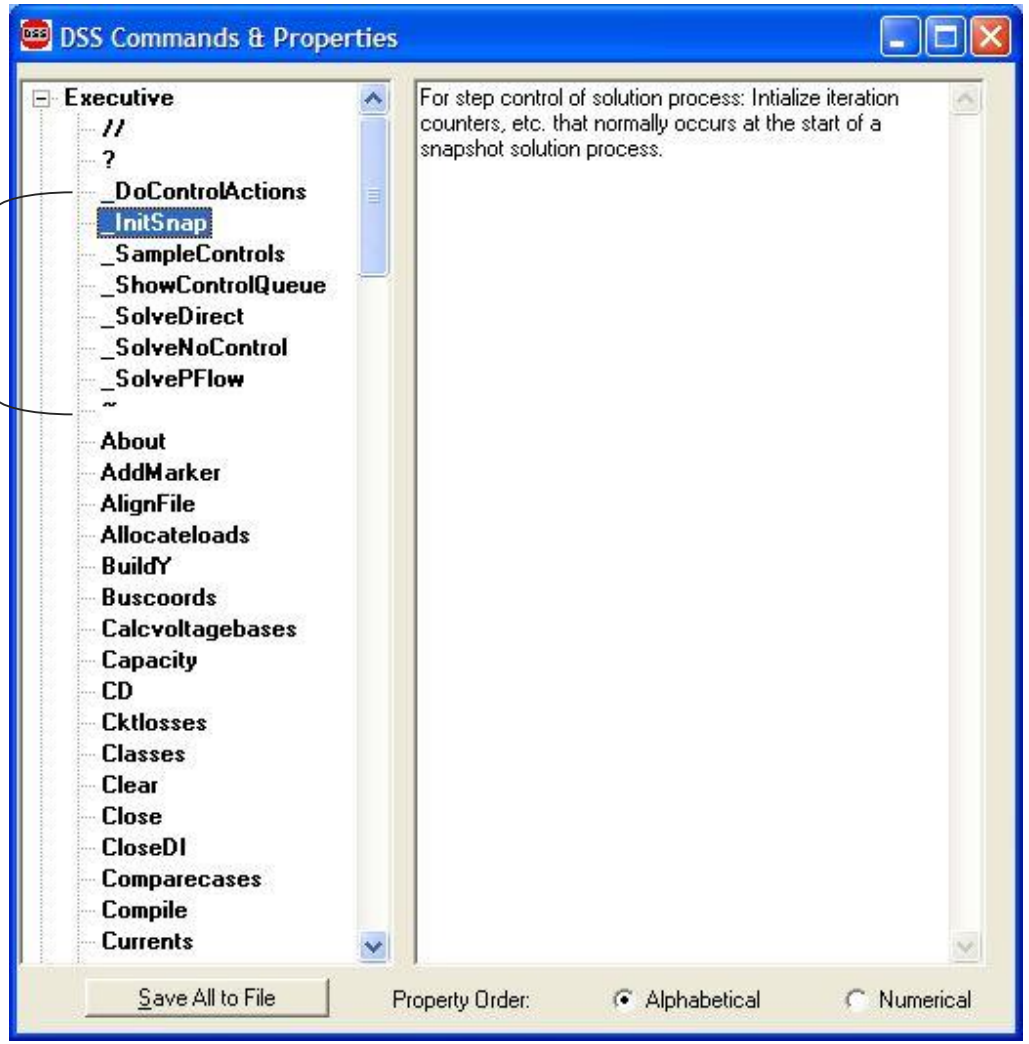
```
! Start the ramp down at 1 sec
Set mode=time loadshapeclass=duty
set stepsize=1s


Solve   ! Base case t=0
Solve    ! t=t+1 = 2
Solve    ! t=2 second solution; t=t+1 = 3

Set sec = 2.020834372 ! Unit 1 (reset t)
storage.jo0235001304.state=discharging %discharge=11.9
Solve
Set sec = 2.022028115 ! Unit 2
storage.jo0235000257.state=discharging %discharge=11.9
Solve
Set sec = 2.023158858 ! Unit 3
storage.jo0235000265.state=discharging %discharge=11.9
Solve
Set sec = 2.024604602 ! Unit 4
storage.jo0235000268_1.state=discharging %discharge=11.9
Solve
Set sec = 2.025738325 ! Unit 5
storage.jo0235000268_2.dispmode=discharging %discharge=11.9
Solve

Etc.

Set sec=3
Solve   ! t=3 solution; t=t+1 = 4

….
```

All Loads, Generators will follow assigned Duty cycle loadshape

Custom scripting

- Rolling Your Own Solution Algorithm

These commands allow step-by-step control of the solution process



Custom scripting

- Rolling Your Own Solution Algorithm
  - The basic Snapshot solution process:
    - Initialize Snapshot (**_InitSnap**)
    - Repeat until converged:
      - Solve Circuit (**_SolveNoControl**)
      - Sample control devices (**_SampleControls**)
      - Do control actions, if any (**_DoControlActions**)
  - You may wish, for example, to interject custom control actions after the _SolveNoControl step

Custom scripting

- Custom Simulation Scripting, cont'd
  - Via COM/Direct DLL interface
    - Whatever you want (if you can write code)
  - See Examples Folder on Sourceforge site
    - Excel: SampleDSSDriver.xls
    - Matlab:
      - VoltageProfileExample.m
      - DSSMonteCarlo.m

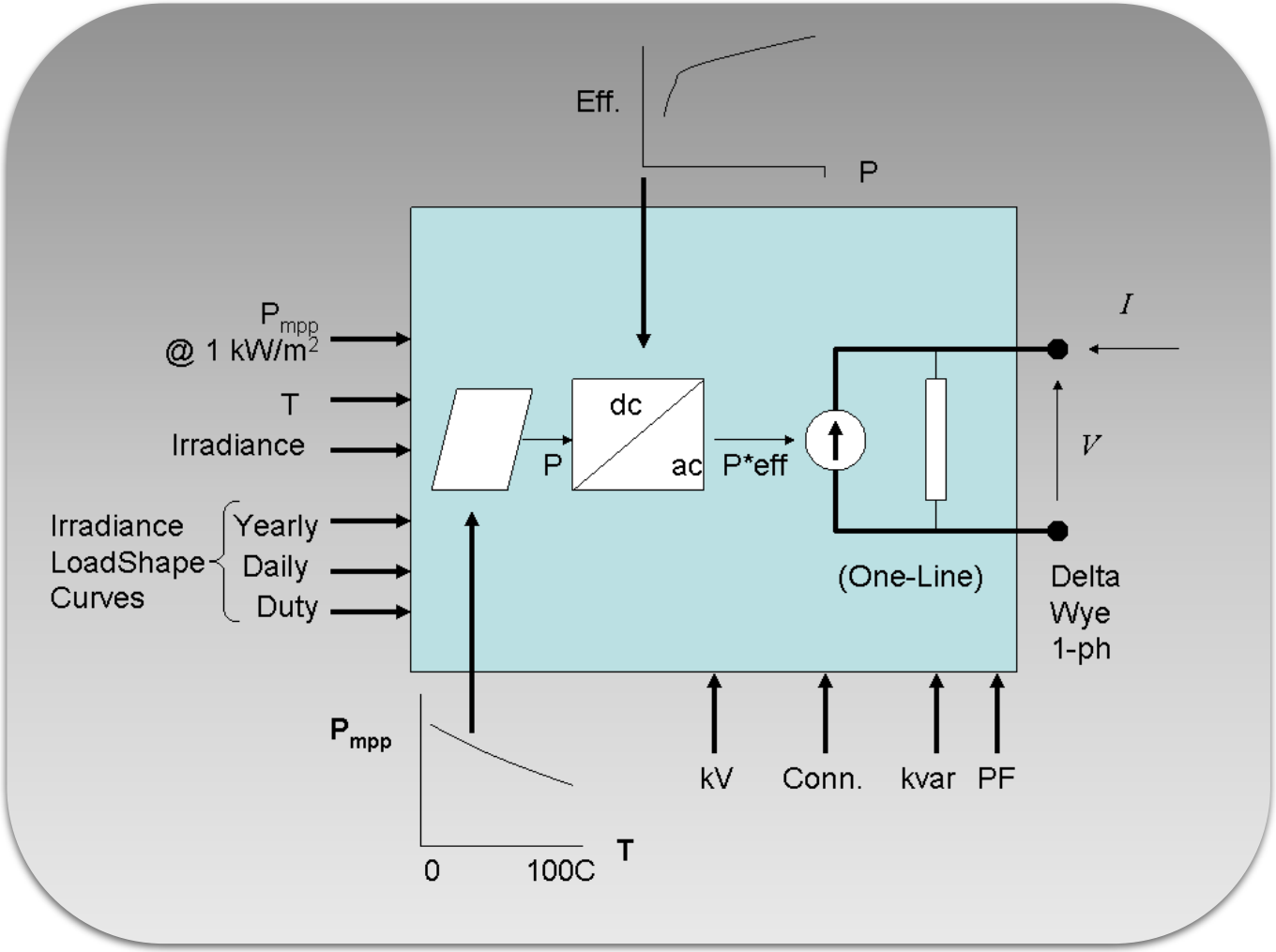
- See "OpenDSS Custom Scripting.Doc"
  - (Sourceforge site and "Doc" Folder)

Special models in OpenDSS

Special models in OpenDSS

- Modeling solar PV

The PVSystem model combines a model of the PV array and the PV inverter into one convenient model to use for distribution system impacts studies



Special models in OpenDSS

- Modeling solar PV (example)

- This example defines a PV system with a panel Pmpp of 500 kW at 1 kW/m^2 irradiance and a panel temperature of 25°C. The inverter is rated at 500 kVA. A PF of 1.0 is assumed for this example.

- Can also be used with the InvControl control object that implements advanced ('smart') inverter functions such as volt-var, volt-watt, and dynamic reactive current.

- InvControl usage to be covered later today

Special models in OpenDSS

- Modeling solar PV (example)

```
clear

New Circuit.PVSystem  basekv=12.47  Isc3=1000 Isc1=900

// P-T curve is per unit of rated Pmpp vs temperature
// This one is for a Pmpp stated at 25 deg
New XYCurve.MyPvsT npts=4  xarray=[0  25  75  100]  yarray=[1.2 1.0 0.8  0.6]

// efficiency curve is per unit eff vs per unit power
New XYCurve.MyEff npts=4  xarray=[.1  .2  .4  1.0]  yarray=[.86  .9  .93  .97]

// per unit irradiance curve (per unit if "irradiance" property)
New Loadshape.MyIrrad npts=24 interval=1 mult=[0 0 0 0 0 0 .1 .2 .3  .5  .8  .9  1.0  1.0  .99  .9  .7  .4  .1 0  0  0  0  0]

// 24-hr temp shape curve
New Tshape.MyTemp npts=24 interval=1 temp=[25, 25, 25, 25, 25, 25, 25, 25, 35, 40, 45, 50  60 60  55 40  35  30  25 25 25 25 25 25]

// **** plot tshape object=mytemp

// take the default line
New Line.line1 Bus1=sourcebus bus2=PVbus  Length=2
```

Special models in OpenDSS

- Modeling solar PV (example) cont'd

```
! PV definition
New PVSystem.PV phases=3 bus1=PVbus kV=12.47  kVA=500  irrad=0.8  Pmpp=500
~ temperature=25 PF=1  effcurve=Myeff  P-TCurve=MyPvsT
~ Daily=MyIrrad  TDaily=MyTemp

set voltagebases=[12.47]
calcv

solve  ! solves at the specified irradiance and temperature

new monitor.m1 PVSystem.PV  1 mode=1 ppolar=no
new monitor.m2 PVSystem.PV  1

solve
solve mode=daily

show mon m1
show mon m2
```

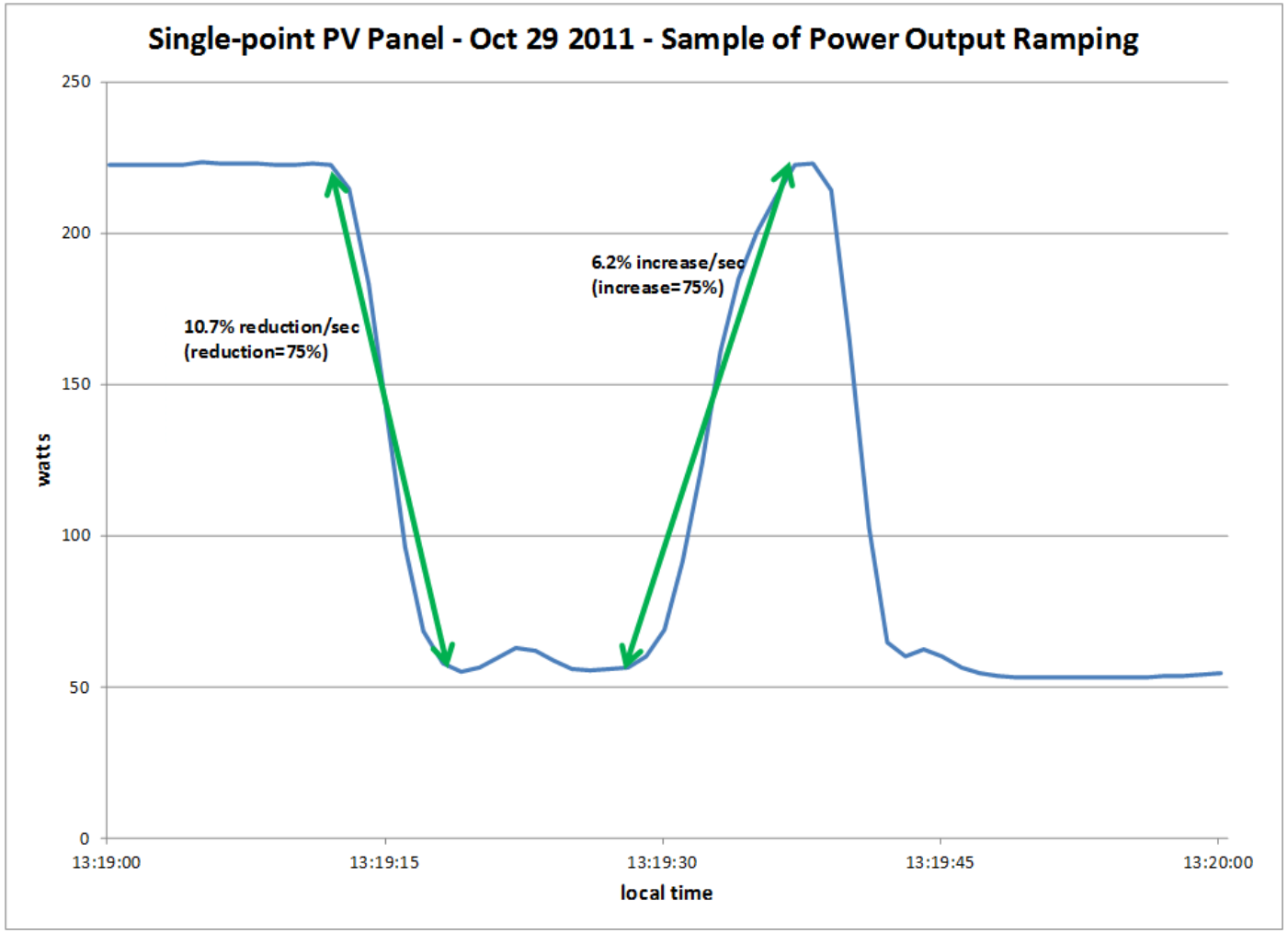
Special models in OpenDSS

- Modeling solar PV (example) cont'd

```
Export monitors m1
Plot monitor object= m1 channels=(1 )
Export monitors m2
Plot monitor object= m2 channels=(1 ) base=[7200]
Export monitors m2
Plot monitor object= m2 channels=(9 )
```

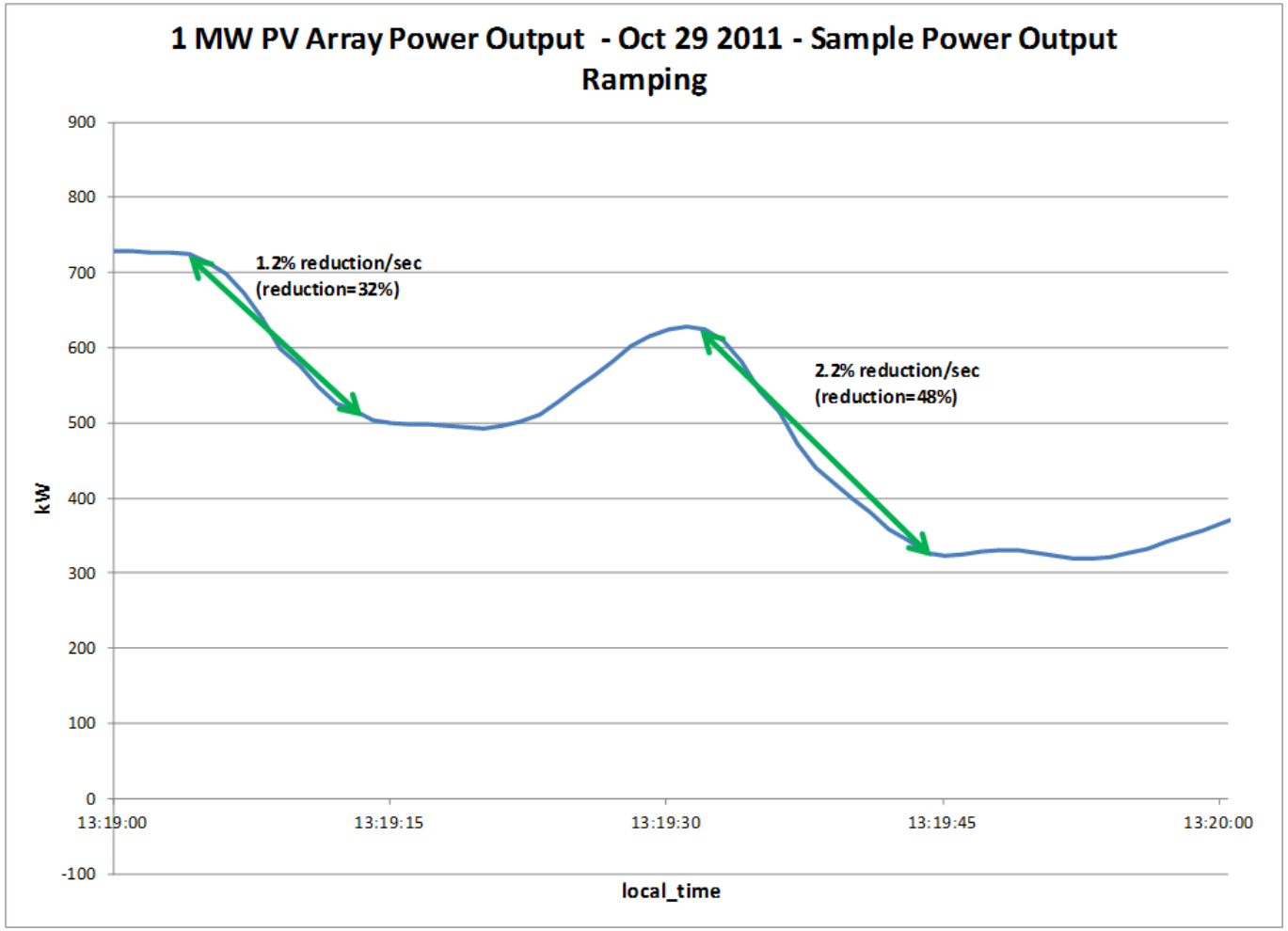
Special models in OpenDSS

- Modeling solar PV



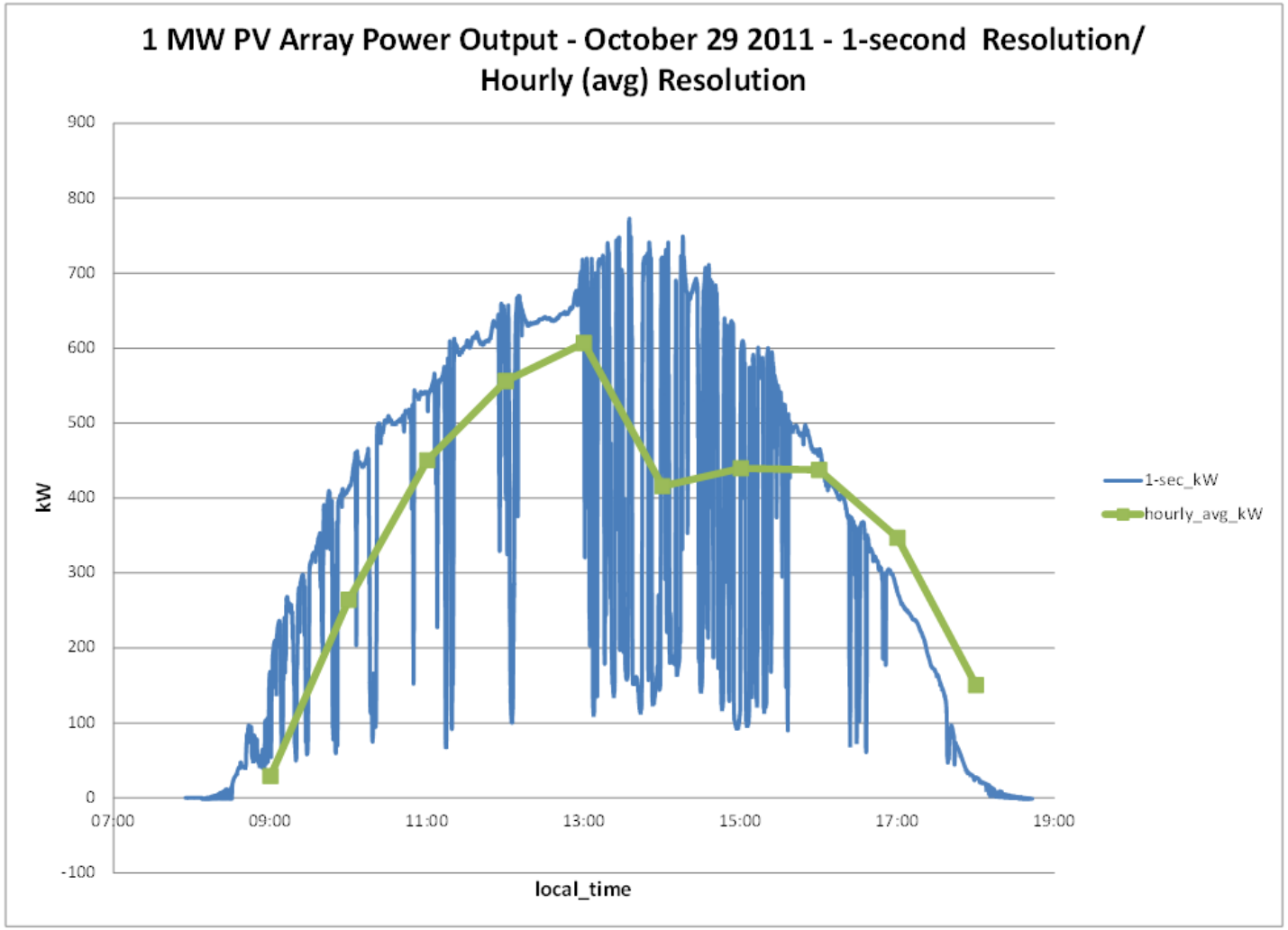
Special models in OpenDSS

- Modeling solar PV



Special models in OpenDSS

- Modeling solar PV



Special models in OpenDSS

- Modeling solar PV (app cases)

- Fault current contribution
- Conservative Rule-of-thumb: 2 x Full Output Rating of Inverter for 1 cycle (three-phase fault)
- Other testing has been performed by Southern California Edison, NREL, PV inverter manufacturers, etc
- Inverters generally shut down at 1.2 pu of rated current
  - May be 3-4 times pre-fault current
- Irregular behavior on voltage sags
  - Tries to hold constant power (current increases)
  - May become discontinuous

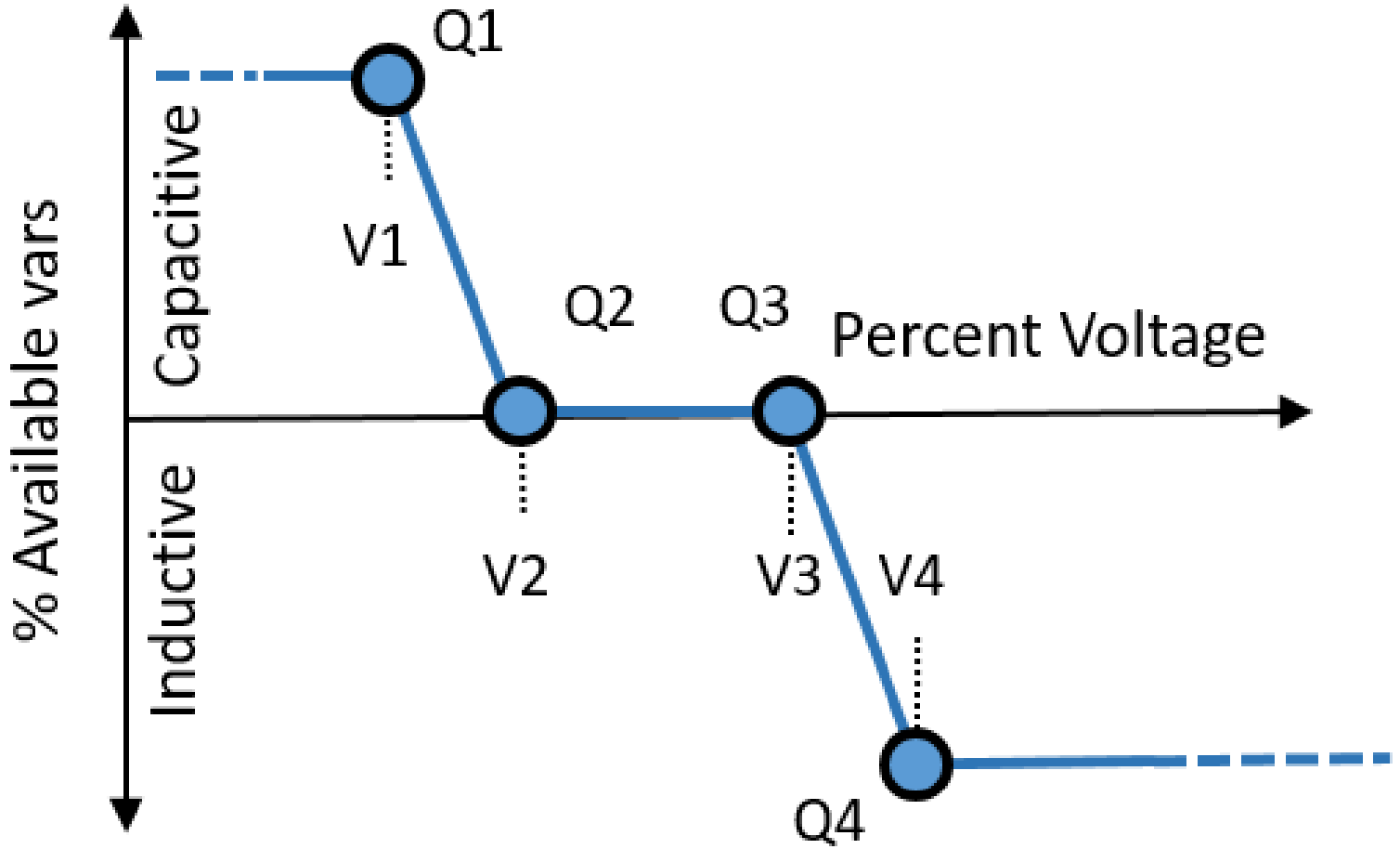
Special models in OpenDSS

Smart inverter

- Works in conjunction with PVSystem object(s) to control the PVSystem(s) output according to 'smart' inverter functions
- Three modes currently available:
  - Volt-var
    - Follows a voltage versus reactive power curve and changes the reactive power generation (capacitive) or reactive power absorption (inductive) according to the terminal voltage at each PVSystem
  - Volt-watt
    - Follows a voltage versus active power curve and changes the active power output according to the terminal voltage at each PVSystem (within the limits of the present irradiance)
  - Dynamic Reactive Current (DRC)
    - Has several settings that change the reactive power generation or absorption in response to fast changes in terminal voltage (e.g., during a sag or swell)

Special models in OpenDSS

- Smart inverter – Control curve – Volt-var



Special models in OpenDSS

- Smart inverter – Control curve – Volt-var

New PVSystem.**3P_ExistingSite4** phases=3 bus1=B51854_sec kV=0.4157 kVA=523
~ irradiance=1 Pmpp=475 pf=1 %cutin=0.1 %cutout=0.1 yearly=PV_ls

New PVSystem.**3P_ExistingSite1** phases=3 bus1=X_5865228330A kV=0.4157 kVA=314
~ irradiance=1 Pmpp=285 pf=1 %cutin=0.1 %cutout=0.1 yearly=PV_ls

New PVSystem.**3P_ExistingSite3** phases=3 bus1=X_5891328219_Cust1 kV=0.4157
~ kVA=836 irradiance=1 Pmpp=760 pf=1 %cutin=0.1 %cutout=0.1 yearly=PV_ls

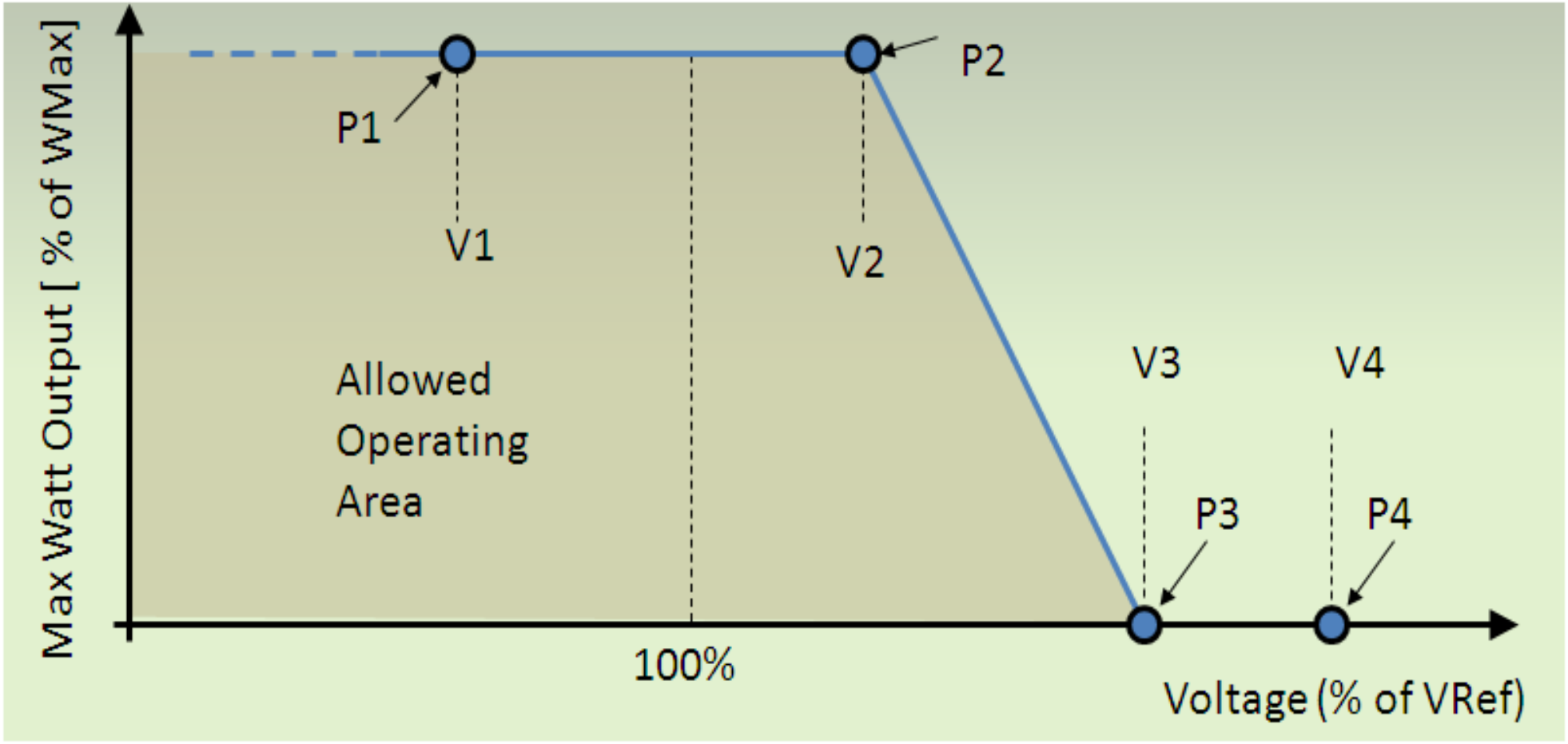
New PVSystem.**3P_ExistingSite2** phases=3 bus1=B4832_sec kV=0.4157 kVA=209
~ irradiance=1 Pmpp=190 pf=1 %cutin=0.1 %cutout=0.1 yearly=PV_ls

New XYCurve.vv_curve npts=4 Yarray=(1.0,1.0,-1.0,-1.0)
~ XArray=(0.5,0.95,1.05,1.5)

New InvControl.InvPVCtrl mode=VOLTVAR voltage_curvex_ref=rated
~ vvc_curve1=vv_curve EventLog=yes

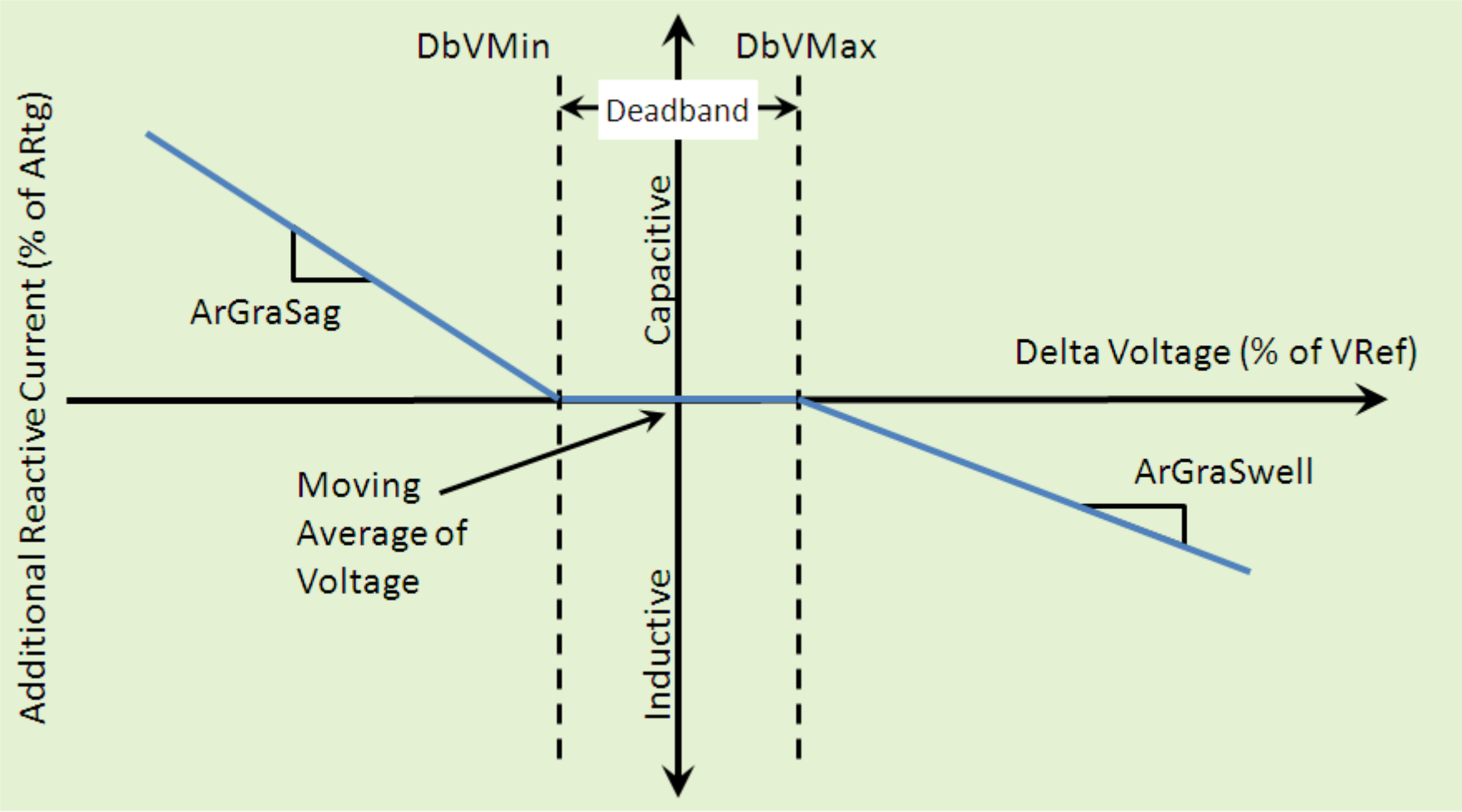
Special models in OpenDSS

- Smart inverter – Control curve – Volt-watt



Special models in OpenDSS

- Smart inverter – DRC Control curve



Special models in OpenDSS

- Storage devices
  - Storage is the frequently-proposed solution to renewable generation issues
  - States and provinces are requiring large amounts of storage
    - (Cal: 1.3 GW by 2020; 425 MW on distribution)
  - Significant amounts expected on Distribution systems controlled for benefit of Transmission
  - Distribution planners are accustomed to static power flow calculations
  - Accurate analysis of storage required sequential-time simulation (“QSTS”)
  - Summary of EPRI research into modeling energy storage for planning …

Special models in OpenDSS

- Storage devices – Applications in distribution
  - Smoothing solar PV power output
  - Extending solar PV output into the evening
  - Support of the Transmission grid
  - Extending capacity of existing assets
  - Supporting alternate feeds during reconfig.
  - Controlling frequency of a microgrid
  - Increasing short circuit strength of a microgrid

Special models in OpenDSS

- Storage devices – Planning Issues Introduced by Storage
  - Overvoltages while discharging
  - Low voltages while charging
  - Voltage regulation while compensating for transmission grid support
  - Interference with overcurrent protection scheme
  - Insufficient short-circuit capacity in microgrid

Special models in OpenDSS

- Storage devices – Planning Issues Introduced by Storage
  - All DER disrupt the normal load shape
  - A single static power flow does not give a good answer
    - Have to simulate over a significant time period
  - Sequential-time power flow is now accepted practice in advanced distribution planning
  - Storage is a variable resource but it is also Limited
    - Planning tools must account for energy stored
    - Has a limited ramp rate
    - Has to be recharged at some other time
  - Has losses
    - Charge-discharge cycle (~15-20%)
    - Idling losses (temperature dependent)

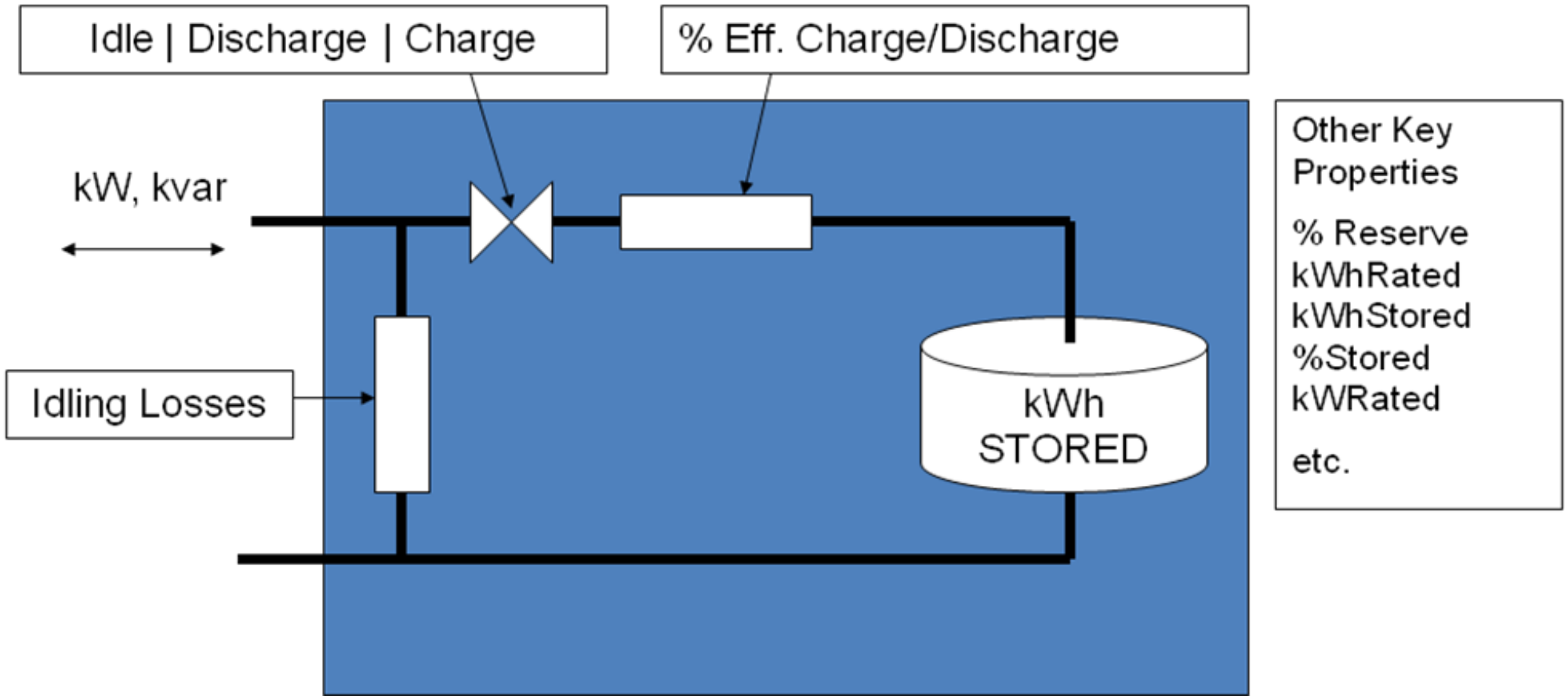
Special models in OpenDSS

- 6 Simulation Modes Have Been Identified and Implemented in OpenDSS Program

1. Static (charge or discharge at specific rate)
2. Time (charge or discharge at specific time)
3. Peak Shave
4. Load Following
5. Loadshape Following (define a loadshape)
6. Dynamics (i.e., electromechanical transients)

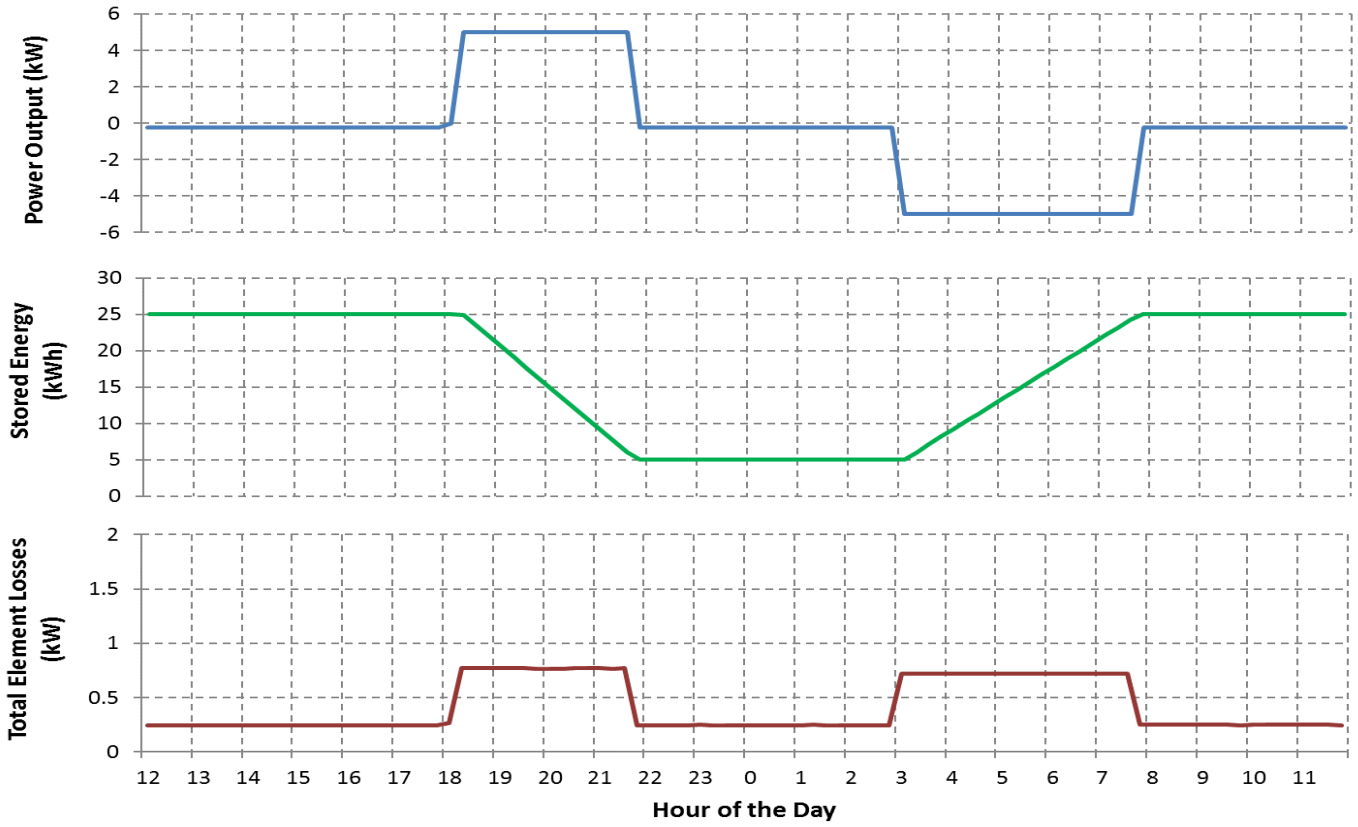
Special models in OpenDSS

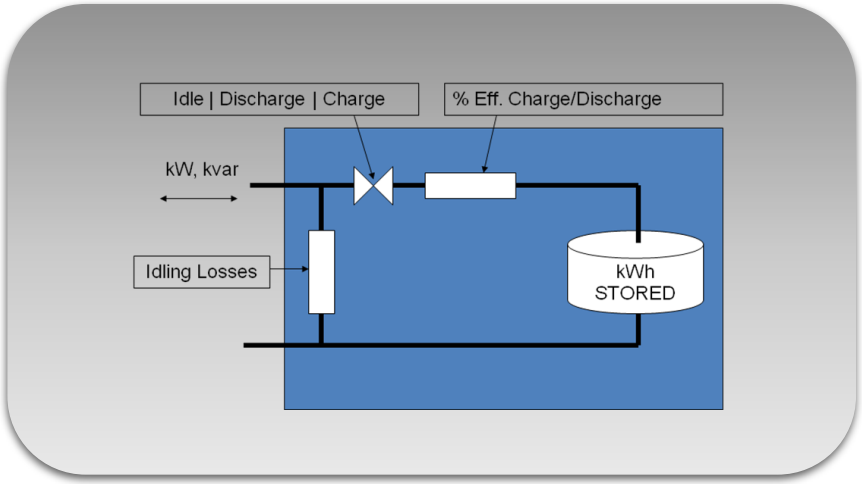
- The storage device model



Special models in OpenDSS

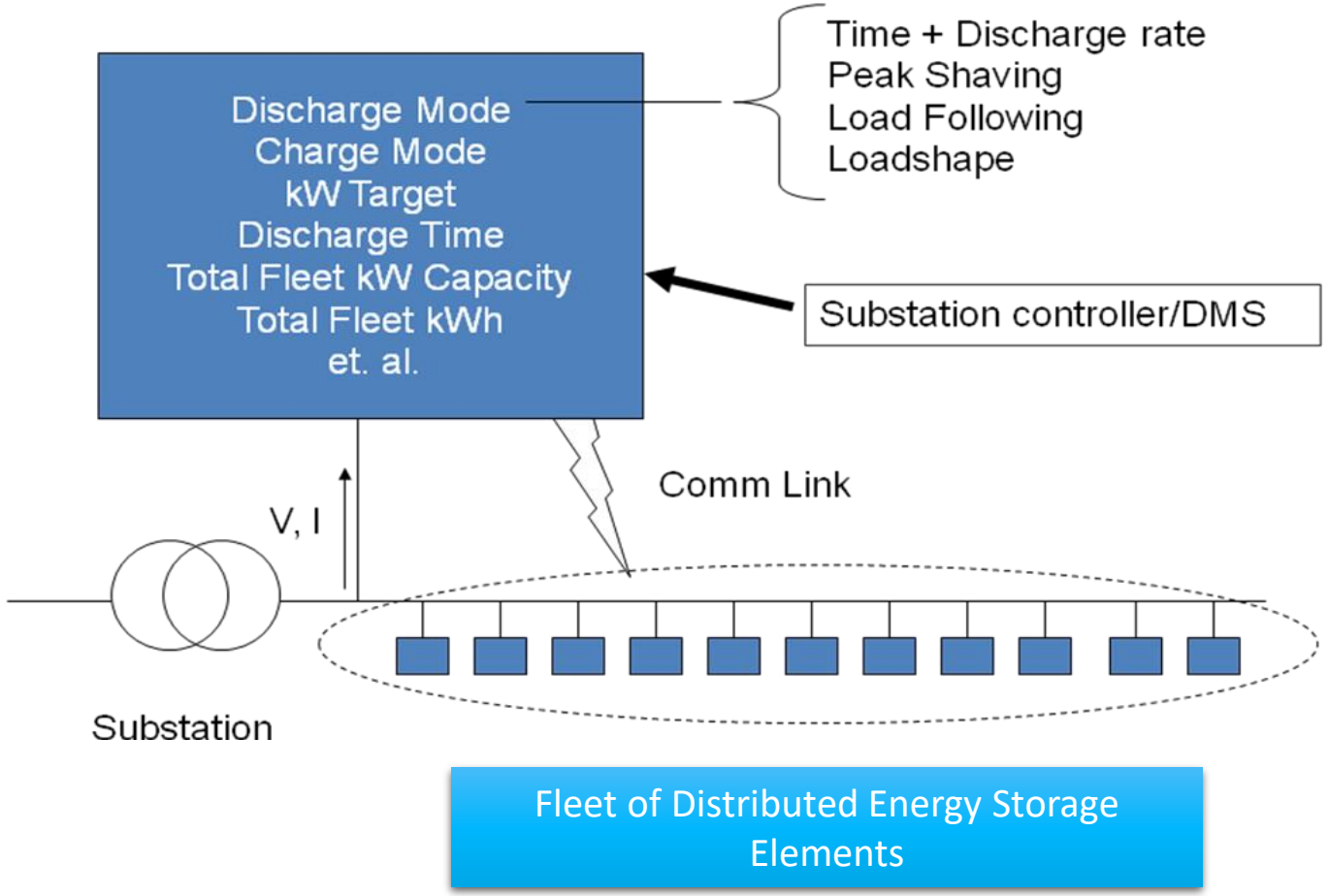
- The storage device model





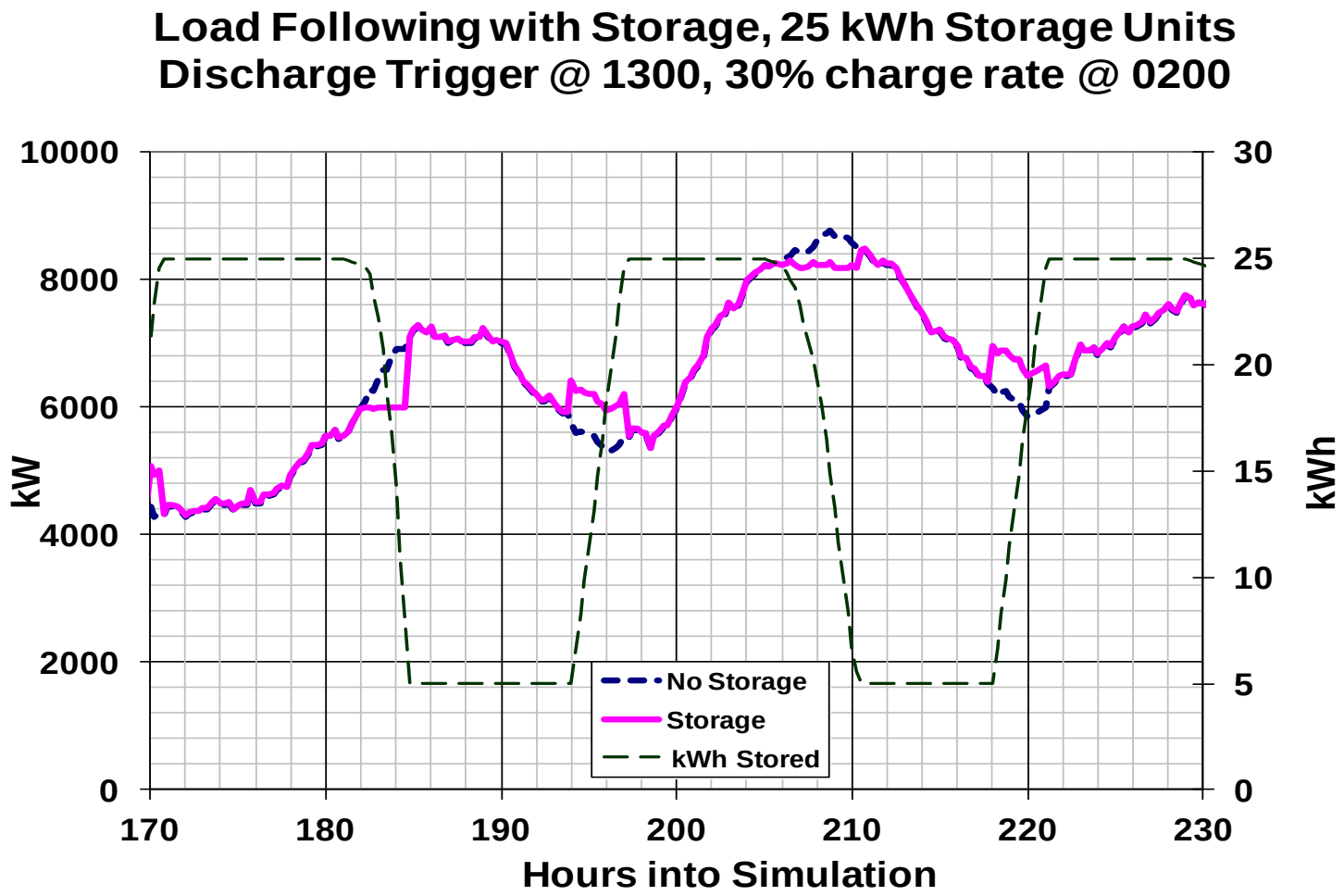
Special models in OpenDSS

- The storage controller model



Special models in OpenDSS

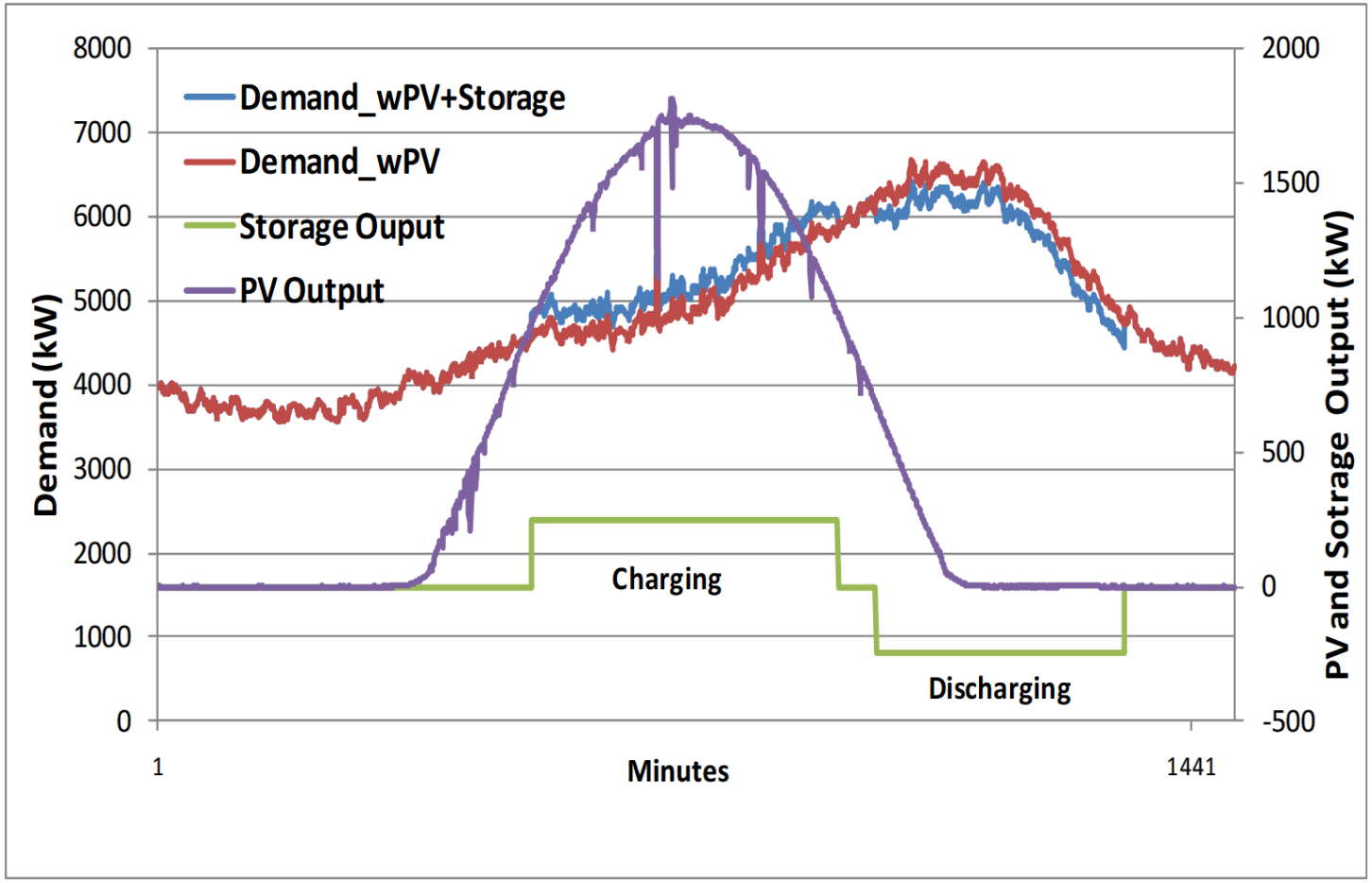
- The storage controller model



Time step size:
15-60 min

Special models in OpenDSS

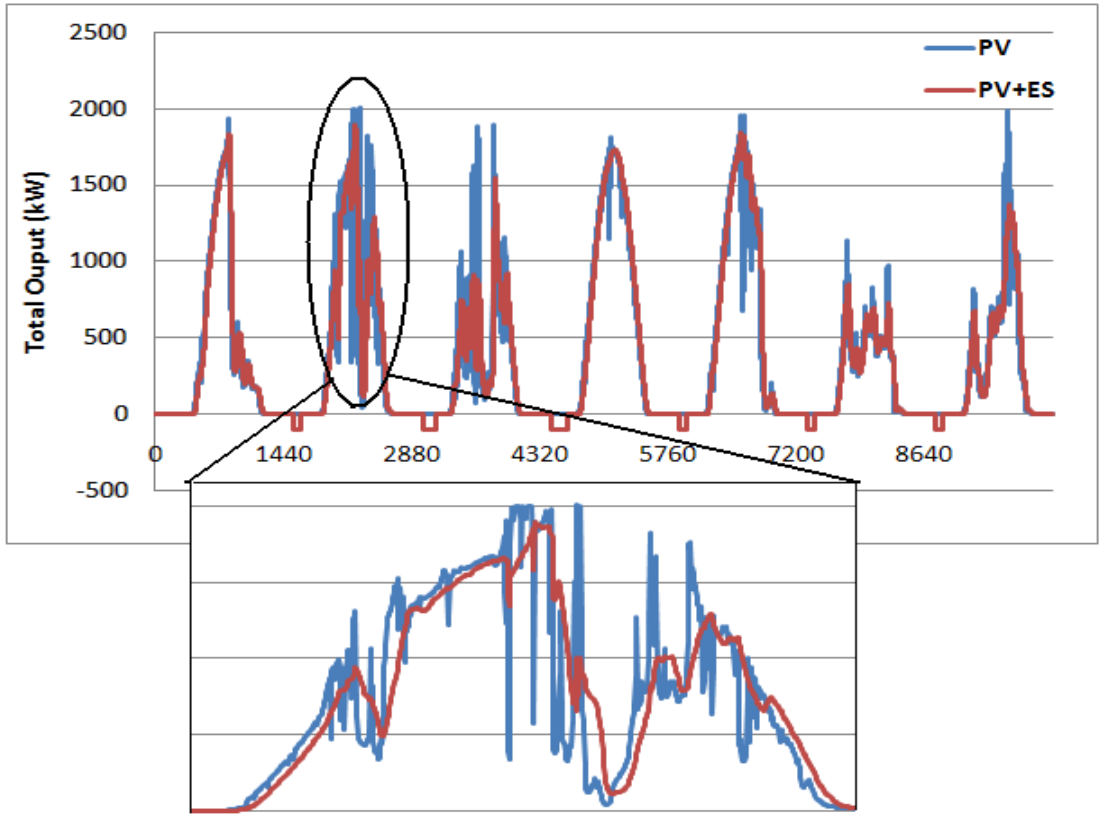
- The storage controller model



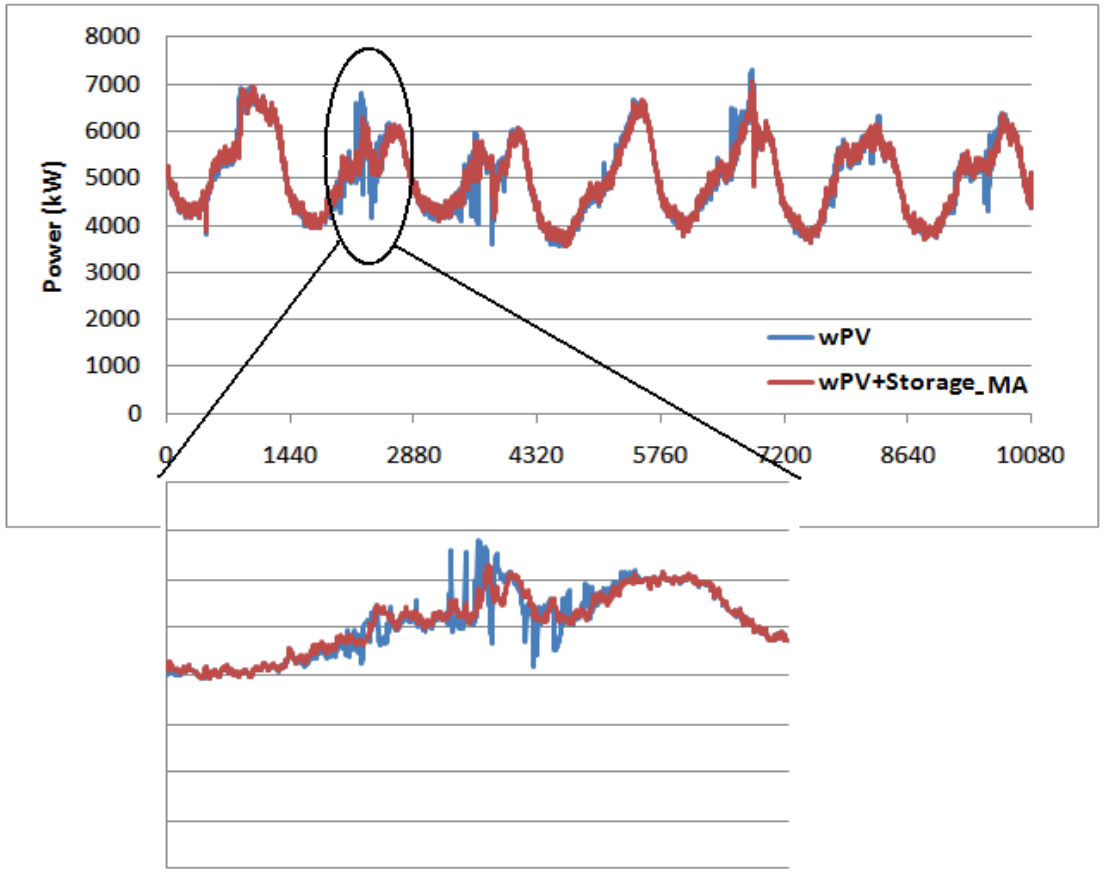
Extend solar PV output into evening peak.

Special models in OpenDSS

- The storage controller model – smooth output from DER
  - PV Output

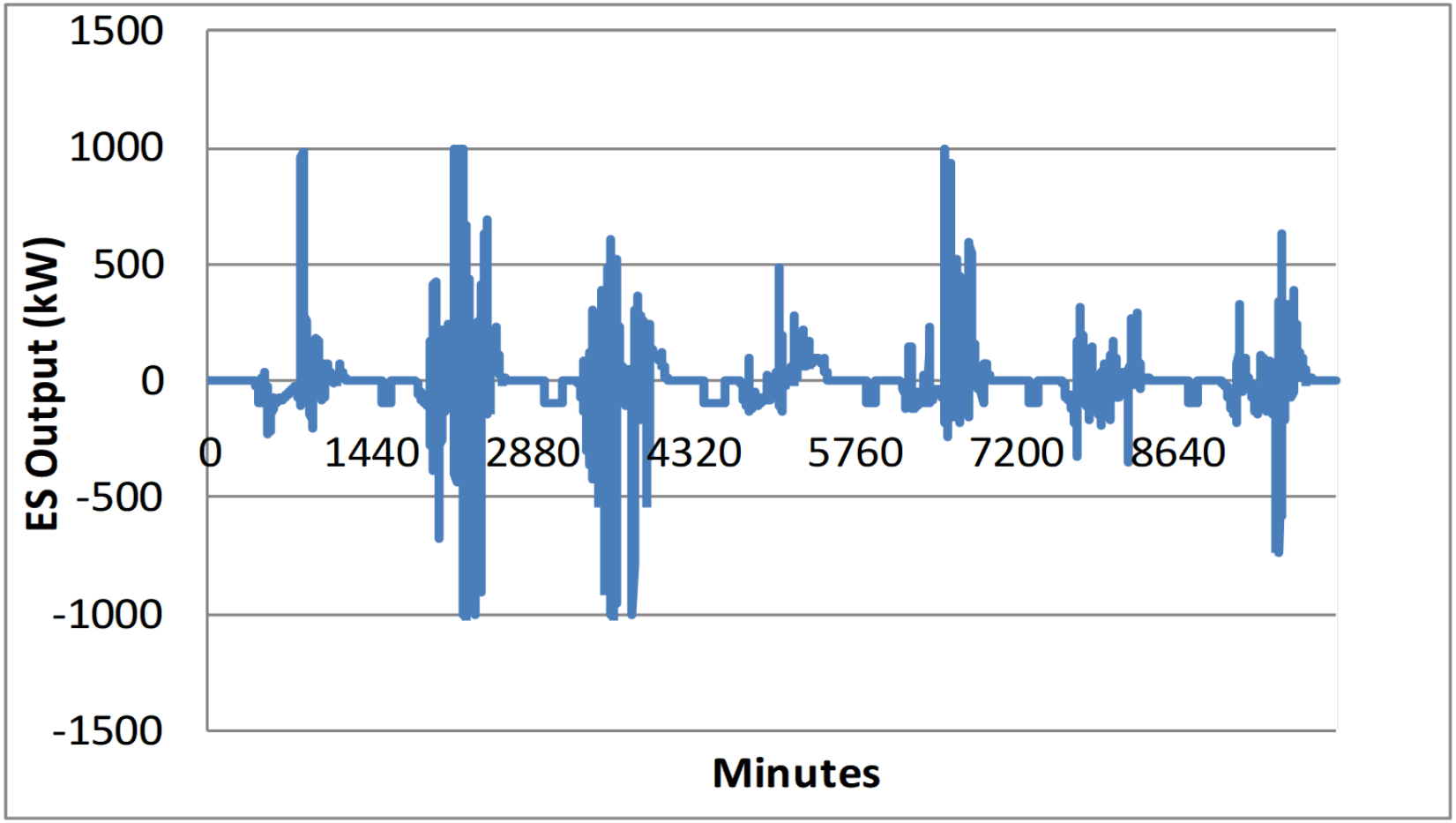


  - Network Response



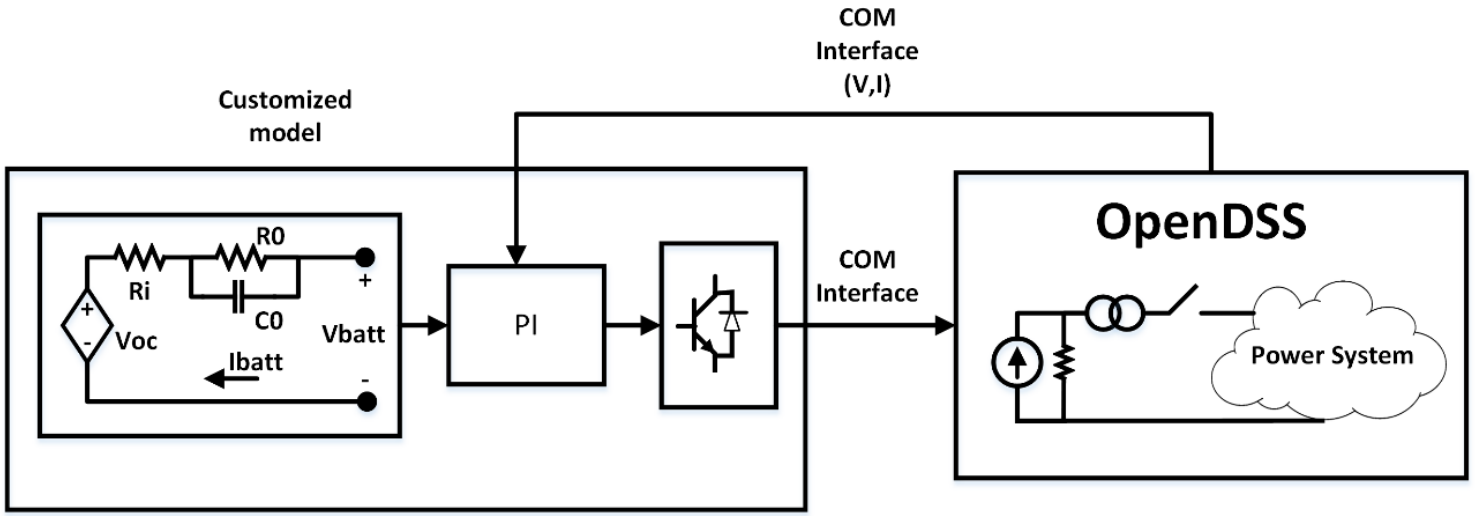
Special models in OpenDSS

- The storage controller model – smooth output from DER

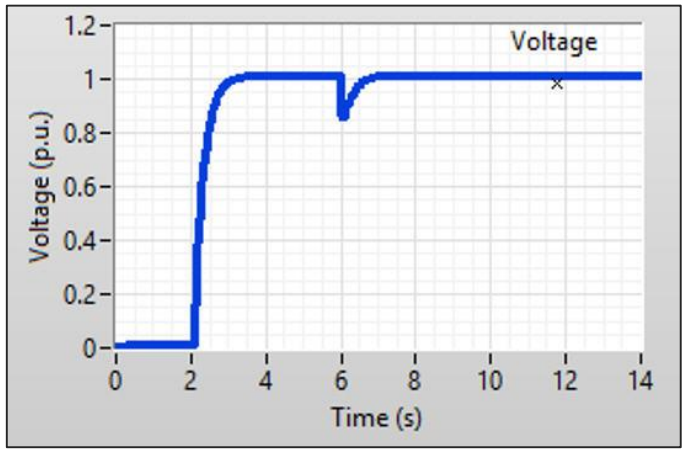


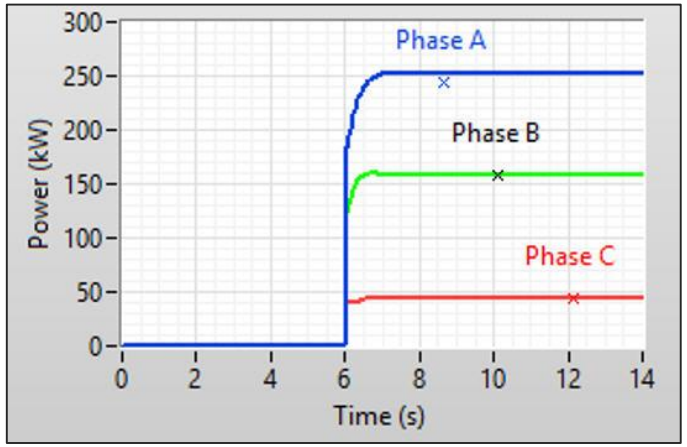
Special models in OpenDSS

- The storage controller model – Dynamics



Model may require more than 30 parameters.





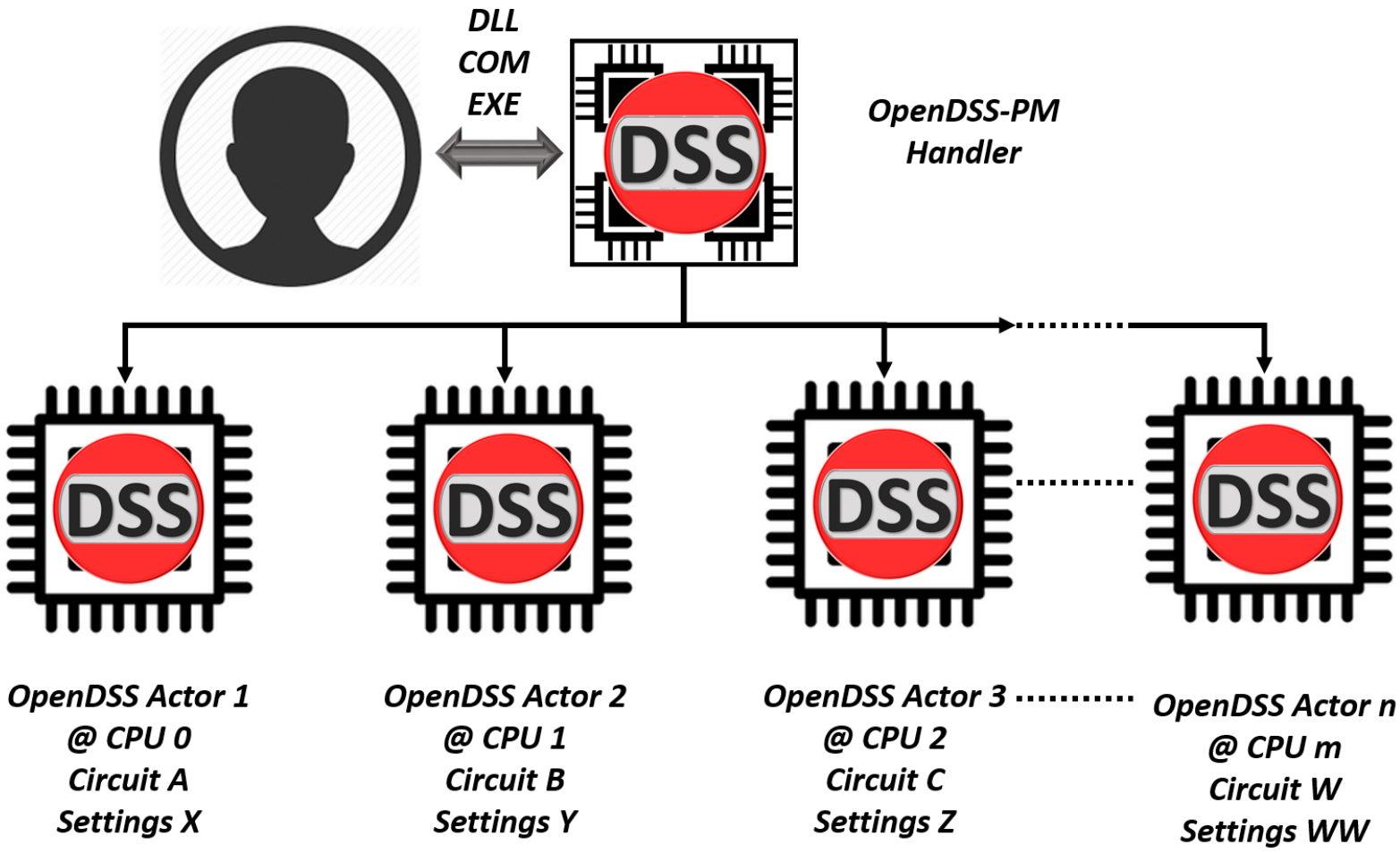
Special models in OpenDSS

- How to Support Vendor-Supplied Models for Complex Storage System Models
  - Establish common software interface (DLL?)
  - Variants for QSTS, Dynamics, and EMT
  - Windows dominant platform in US for distribution
  - DSA vendors will have to support
  - Storage system vendors will want protection

Introduction to parallel processing

Introduction to parallel processing

- The evolution of OpenDSS into a parallel computing machine



Introduction to parallel processing

- The evolution of OpenDSS into a parallel computing machine

These features have been implemented into the existing interfaces preserving the format and adding instructions to the instruction set

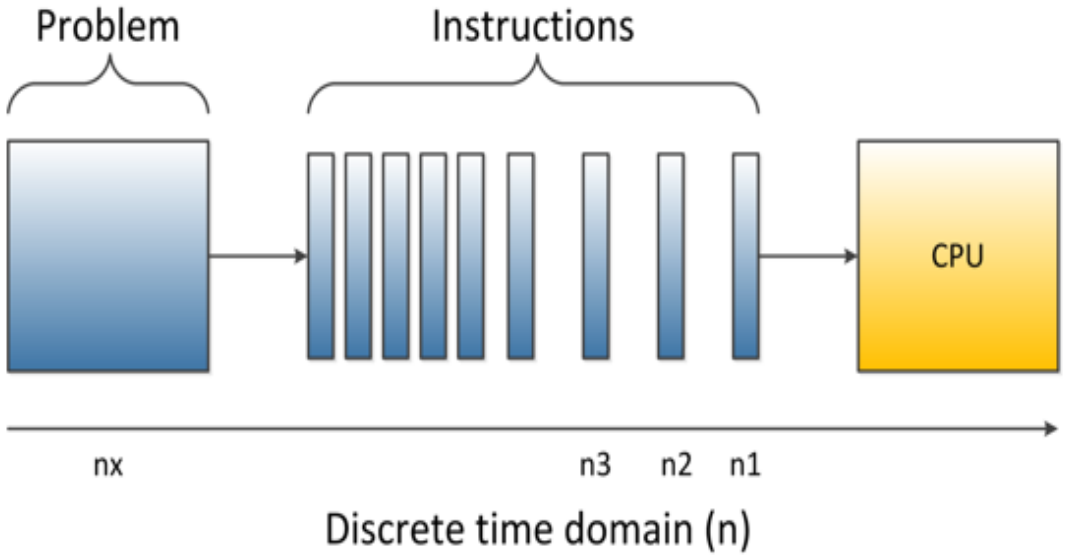
Introduction to parallel processing

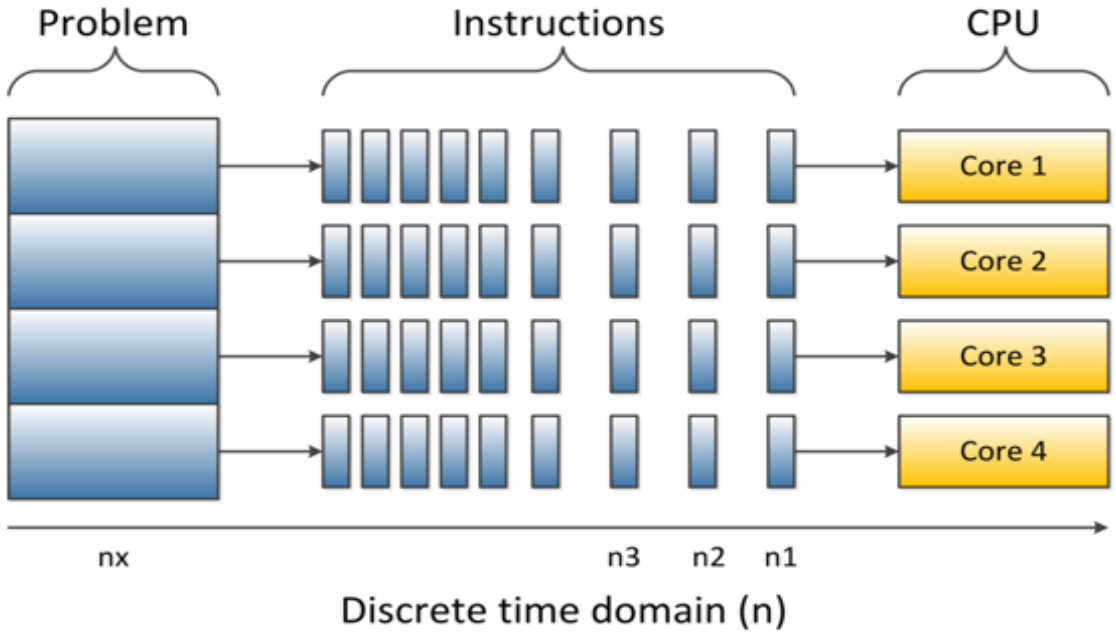
- There is a new set of functions and instructions related with parallel processing, topology analysis and more that can be used using the same DSS command environment

NumCPUs	ClearAll	Tear_Circuit
NumCores	Wait	Export IncMatrix
NewActor	Parallel	Export IncMatrixRows
NumActors	SolveAll	Export IncMatrixCols
ActiveActor	ConcatenateReports	Export BusLevels
CPU	CalcIncMatrix	ADiakoptics
ActorProgress	CalcIncMatrix_o	Refine_BusLevels
Abort	CalcLaplacian	Export Laplacian
Clone	LinkBranches	Contours
ZLL	ZCC	Y4
Num_SubCircuits		

Introduction to parallel processing

- What to expect from parallel processing

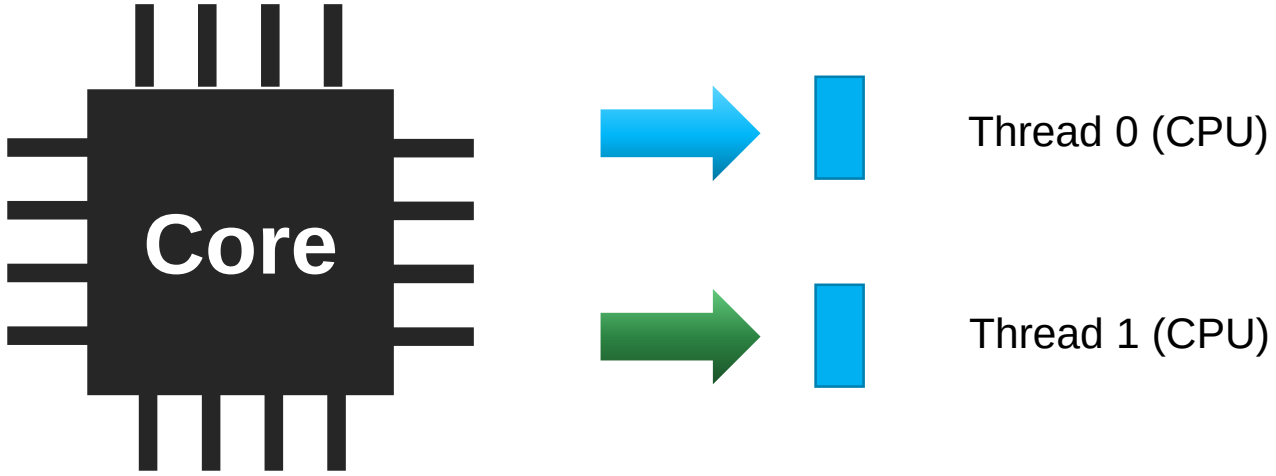




Depending on your computer features and the understanding that you have about them it will depend how far can you go with parallel processing

Introduction to parallel processing

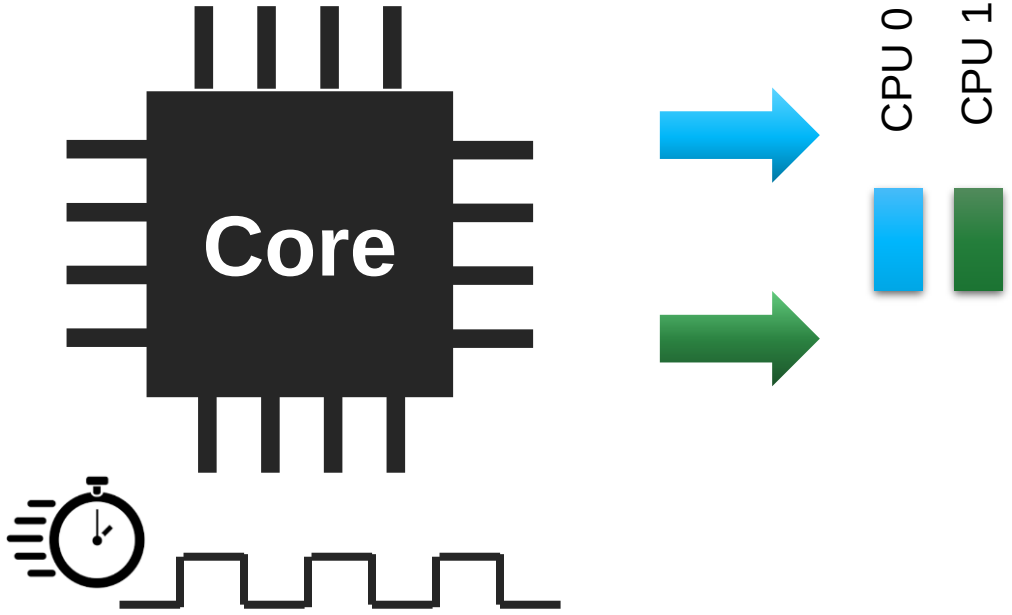
- What to expect from parallel processing



Depending on your computer features and the understanding that you have about them it will depend how far can you go with parallel processing

Introduction to parallel processing

- What to expect from parallel processing



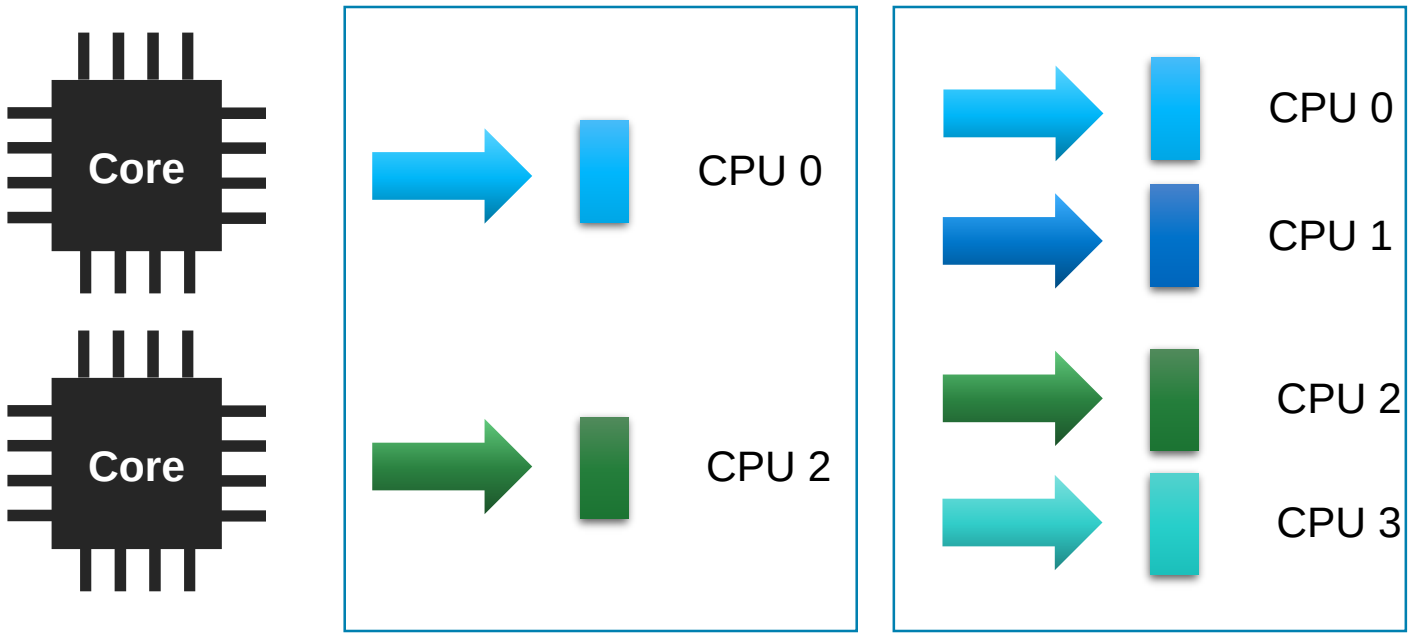
DESKTOP	Core i3	Core i5	Core i7
No. of cores	2	4	4
Frequency range	3.4-4.2GHz	2.4-3.8GHz	2.9-4.2GHz
Turbo Boost	No	Yes	Yes
Hyper-Threading	Yes	No	Yes
Cache	3-4MB	6MB	8MB

http://www.trustedreviews.com/opinion/best-intel-processor-core-i3-i5-i7-2937692

Depending on your computer features and the understanding that you have about them it will depend how far can you go with parallel processing

Introduction to parallel processing

- What to expect from parallel processing



Amdahl's law

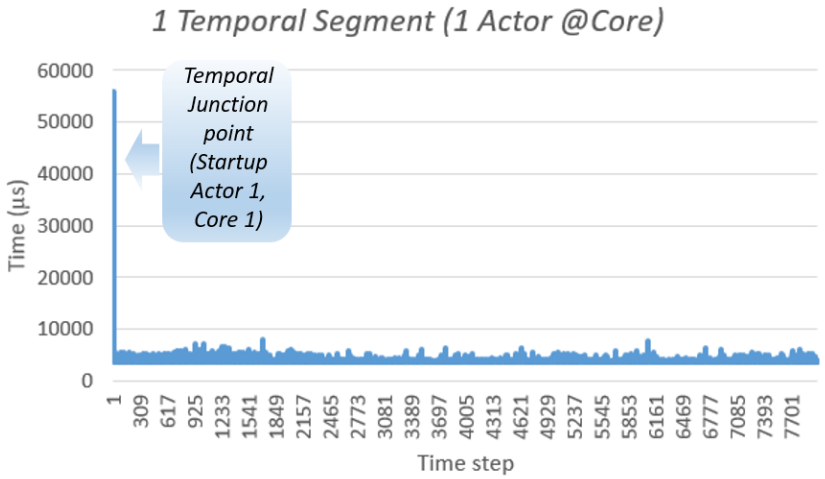
Gustafson's law

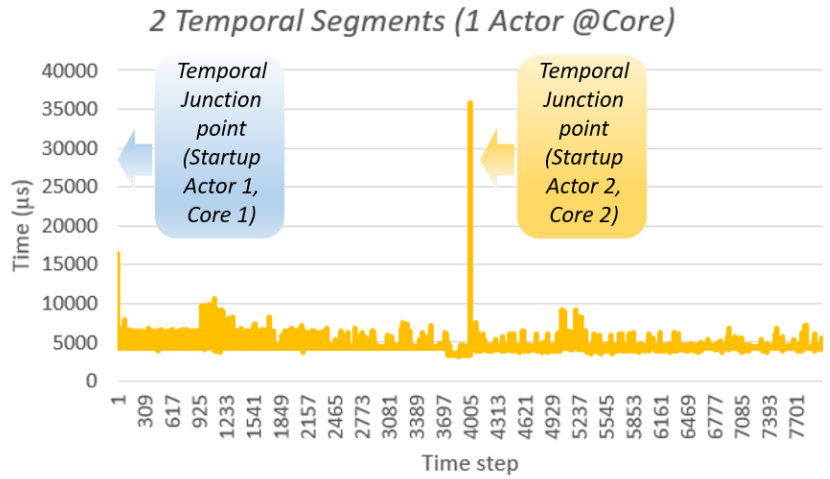
Depending on your computer features and the understanding that you have about them it will depend how far can you go with parallel processing

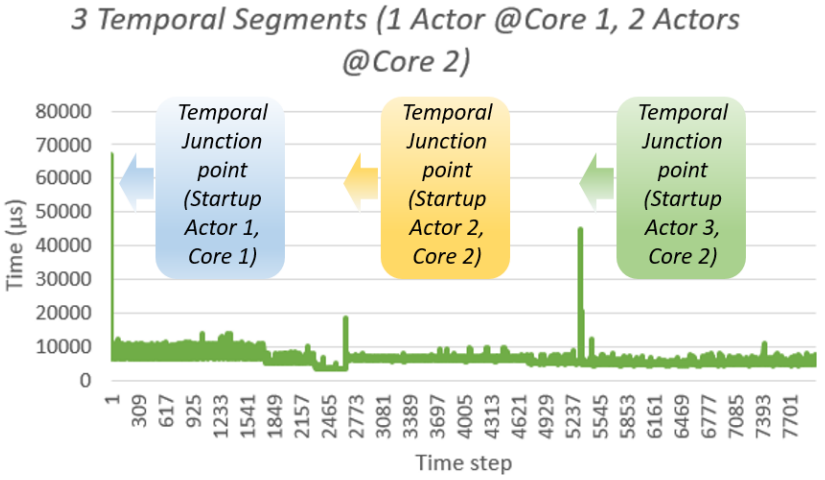
Temporal parallelization

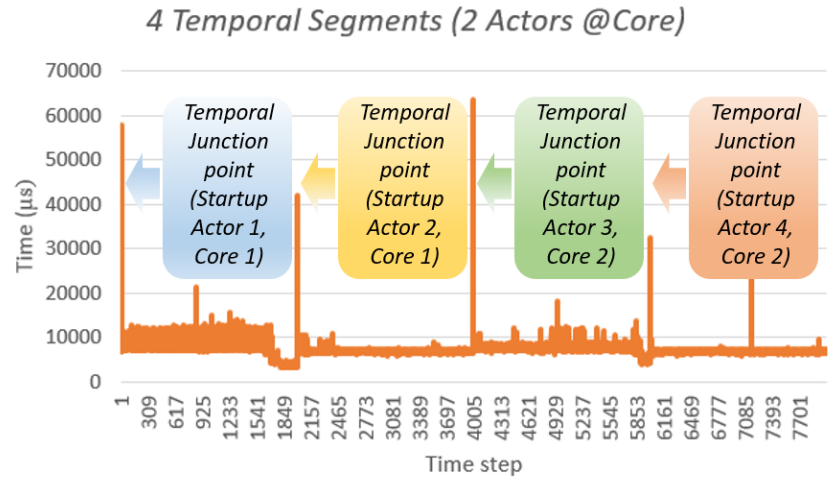
Temporal parallelization

- "The computational burden is on the simulation intensity"





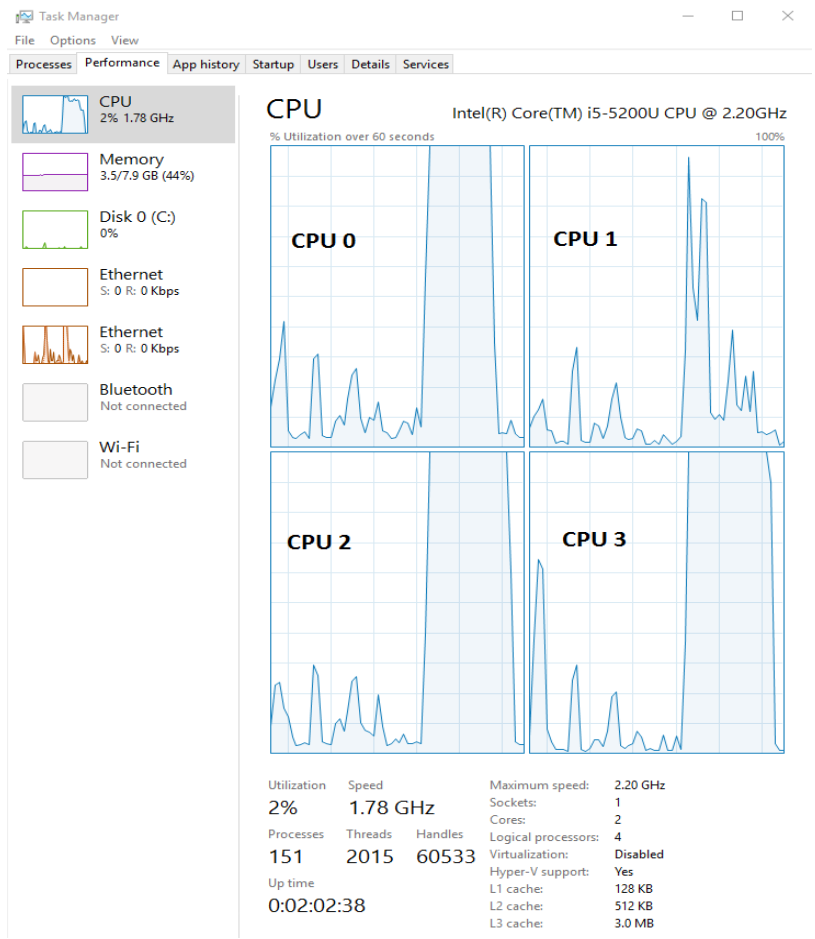




Temporal parallelization

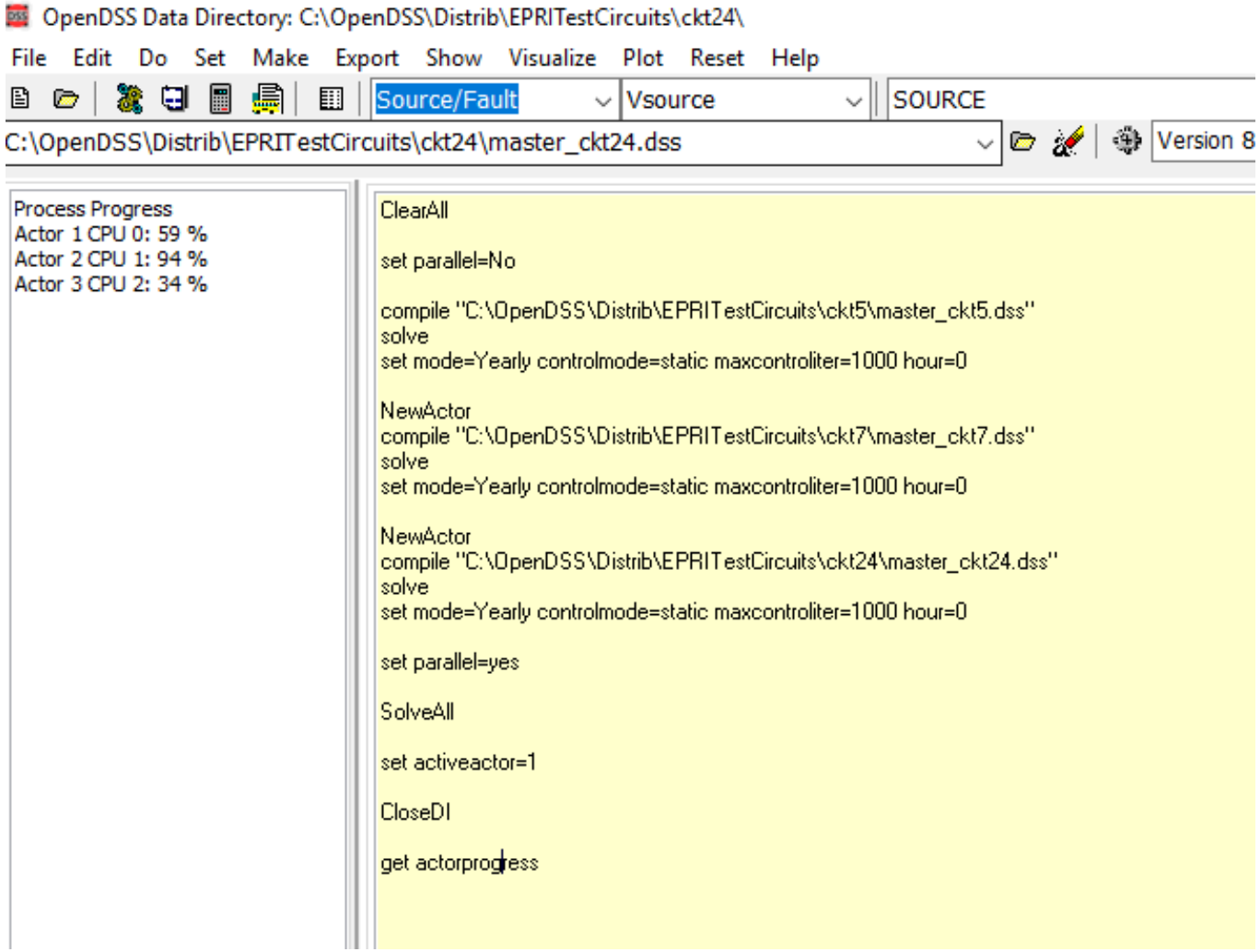
- "The computational burden is on the simulation intensity"

```
clearAll
set parallel=No

compile "C:\Program Files\OpenDSS\EPRITestCircuits\ckt5\Master_ckt5.dss"
set CPU=0
Solve

Clone 2

set parallel=Yes

set activeActor=*
set mode=yearly number=2000 totaltime=0
Set ActiveActor=1
Set hour=0
set activeActor=2
set hour = 2000
set activeActor=3
set hour = 4000

SolveAll
Wait
get actorprogress

set ConcatenateReports=Yes
show monitor MS2
```



Temporal parallelization

- "The computational burden is on the simulation intensity"



Temporal parallelization

- "The computational burden is on the simulation intensity"

The examples for parallel processing can be downloaded from:

https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Version8/Distrib/Examples/Parallel_Processing

These examples involves MATLAB and python.

Other examples for LabVIEW through the VI Package Manager

Temporal parallelization

- "The computational burden is on the simulation intensity"

Example # 1 MATLAB

```
clc;
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
DSSCircuit      =   DSSObj.ActiveCircuit;
DSSText.Command =   'ClearAll';             % Clears all instances of OpenDSS-PM
DSSText.Command =   'Set Parallel=Yes';             % Clears all instances of OpenDSS-PM

DSSParallel     =   DSSCircuit.Parallel;    % Habdler for Parallel processing functions
CPUs            =   DSSParallel.NumCPUs;    % Gets how many CPUs this PC has
% By default one actor is created by default, if you want more than one
% parallel instance you will have to create them. Try to leave at least
% One CPU available to handle the rest of windows, otherwise will block
% Everything
for i=1:CPUs-1,
    if i ~= 1,
        DSSParallel.CreateActor; % Creates additonal actors
    end;
    DSSText.Command =   'compile (C:\Program Files\OpenDSS\EPRITestCircuits\ckt5\Master_ckt5.DSS)';
    DSSCircuit.Solution.Solve;
    DSSParallel.Wait;   % for the first solve, it is needed to wait before creating other actor
    DSSText.Command =   'set mode=Time stepsize=1h number=16000';
end;
% Now the actors are solved

DSSCircuit.Solution.SolveAll;
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

Example # 1 MATLAB

```
25
26  pause(0.1);
27  BoolStatus      =   0;
28  while BoolStatus == 0,
29      ActorStatus     =   DSSParallel.ActorStatus;
30      BoolStatus      =   all(ActorStatus & 1); %Checks if everybody has ended
31      ActorProgress   =   DSSParallel.ActorProgress;
32      clc;
33      for i=1:CPUs-1,
34          fprintf('Actor %i Progress(%%) @ CPU %i : %i\n',i,i-1,ActorProgress(i));
35      end;
36      pause(0.5);  %  A little wait to not saturate the Processor
37  end;
38  disp('Simulation finished by all the actors');
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

Example # 2 MATLAB

```
clc;
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
DSSCircuit      =   DSSObj.ActiveCircuit;
DSSText.Command =   'ClearAll';             % Clears all instances of OpenDSS-PM
DSSText.Command =   'Set Parallel=No';      % Deactivates parallel processing

DSSParallel     =   DSSCircuit.Parallel;    % Habdler for Parallel processing functions
CPUs            =   DSSParallel.NumCPUs;    % Gets how many CPUs this PC has
% By default one actor is created by default, if you want more than one
% parallel instance you will have to create them. Try to leave at least
% One CPU available to handle the rest of windows, otherwise will block
% Everything
% Prepares everything for a yearly simulation using temporal parallelization
YDelta  =   8760/(CPUs-1);
disp('Compiling and creating Actors');
for i=1:CPUs-1,
    if i ~= 1,
        DSSParallel.CreateActor; % Creates additonal actors
    end;
    DSSText.Command =   'compile (C:\Program Files\OpenDSS\EPRITestCircuits\ckt7\Master_ckt7.DSS)';
    DSSCircuit.Solution.Solve;
    if i == (CPUs-1),
        YDelta = 8760 - (CPUs-2)*YDelta;
    end;
    DSSText.Command =   ['set mode=Yearly number=',int2str(YDelta),' hour=',int2str((i-1)*YDelta)];
end;
% Now the actors are solved
DSSText.Command =   'Set Parallel=Yes';             % Activates parallel processing
DSSCircuit.Solution.SolveAll;
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

Example # 2 MATLAB

```
30
31  pause(0.1);
32  BoolStatus      =   0;
33  while BoolStatus == 0,
34      ActorStatus     =   DSSParallel.ActorStatus;
35      BoolStatus      =   all(ActorStatus & 1); %Checks if everybody has ended
36      clc;
37      % Prints the current time on each simulation
38      for i = 1:(CPUs-1),
39          DSSParallel.ActiveActor = i;
40          CHour   =   DSSCircuit.Solution.dblhour;
41          fprintf('Actor %i Time(hours) : %f\n',i,CHour);
42      end;
43      pause(0.5);  %  A little wait to not saturate the Processor
44  end;
45  disp('Simulation finished by all the actors');
46
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

Example # 3 MATLAB

```
clc;
[DSSStartOK, DSSObj, DSSText] = DSSStartup;
DSSCircuit      =   DSSObj.ActiveCircuit;
DSSText.Command =   'ClearAll';             % Clears all instances of OpenDSS-PM
DSSText.Command =   'Set Parallel=Yes';             % Clears all instances of OpenDSS-PM

DSSParallel     =   DSSCircuit.Parallel;    % Habdler for Parallel processing functions
CPUs            =   DSSParallel.NumCPUs;    % Gets how many CPUs this PC has
% By default one actor is created by default, if you want more than one
% parallel instance you will have to create them. Try to leave at least
% One CPU available to handle the rest of windows, otherwise will block
% Everything
disp('Creating Actors');
for i=1:CPUs-1,
    if i ~= 1,
        DSSParallel.CreateActor; % Creates additonal actors
    end;
    DSSText.Command =   'compile (C:\Program Files\OpenDSS\EPRITestCircuits\ckt5\Master_ckt5.DSS)';
    DSSCircuit.Solution.Solve;
    DSSParallel.Wait;   % for the first solve, it is needed to wait before creating other actor
    DSSText.Command =   'set mode=Time stepsize=1h number=16000';
end;
% Now the actors are solved
disp('Simulation Started');
DSSCircuit.Solution.SolveAll;
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

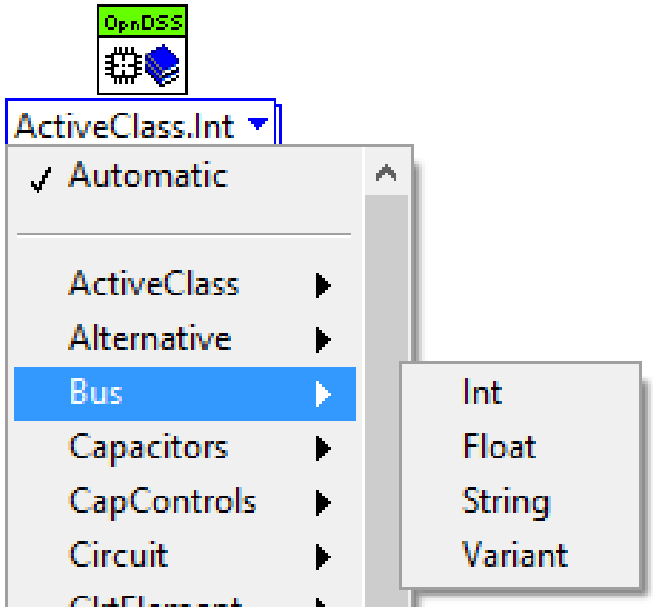
Example # 3 MATLAB

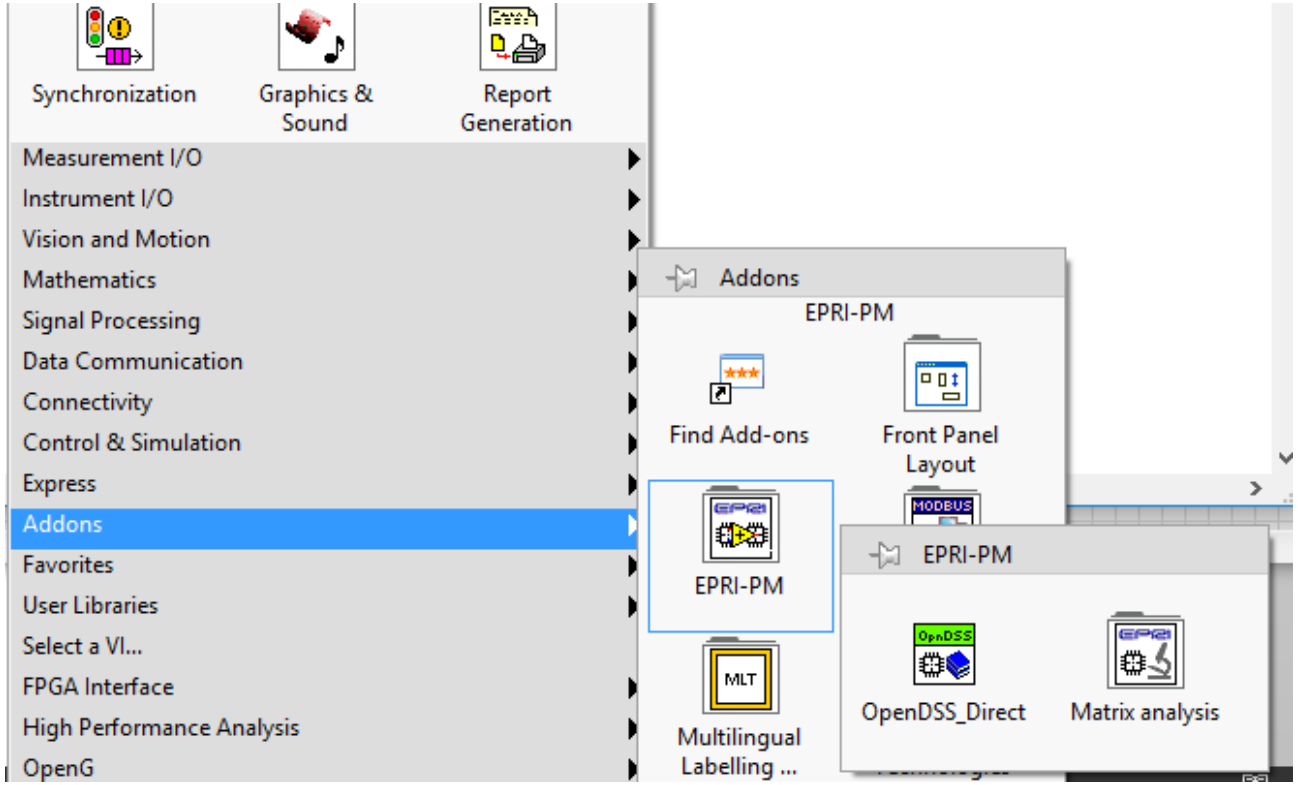
```
26  pause(0.1);
27  hold on;
28  BoolStatus      =   0;
29  while BoolStatus == 0,
30      ActorStatus     =   DSSParallel.ActorStatus;
31      BoolStatus      =   all(ActorStatus & 1); %Checks if everybody has ended
32      ActorProgress   =   DSSParallel.ActorProgress;
33      bar(ActorProgress);
34      axis([0 (CPUs) 0 100]);
35      xlabel('Actor #');
36      ylabel('Actor progress (%)');
37      pause(0.5);  %  A little wait to not saturate the Processor
38  end;
39  disp('Simulation finished by all the actors');
```

Temporal parallelization

- "The computational burden is on the simulation intensity"

Using the OpenDSS-PM library for NI LabVIEW



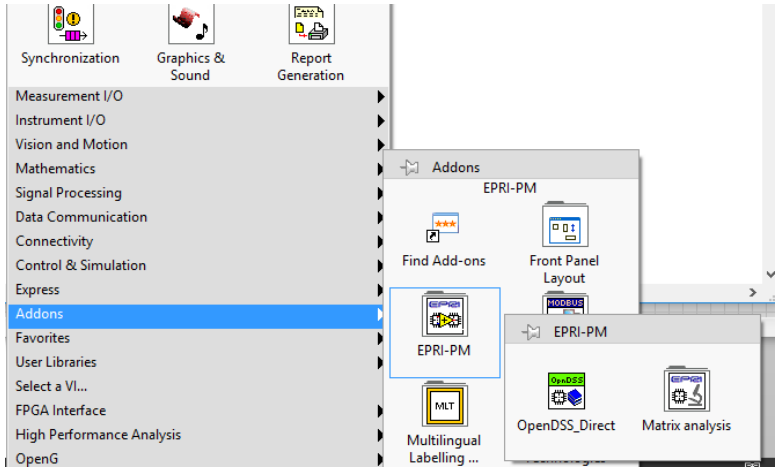


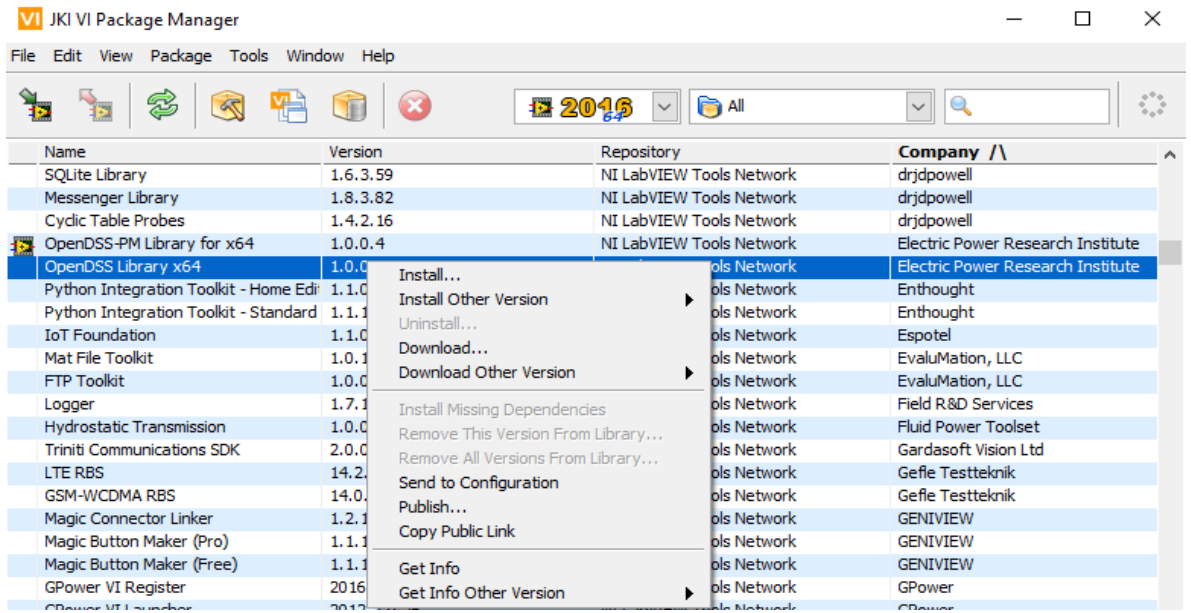
Temporal parallelization

- "The computational burden is on the simulation intensity"

Using the OpenDSS-PM library for NI LabVIEW

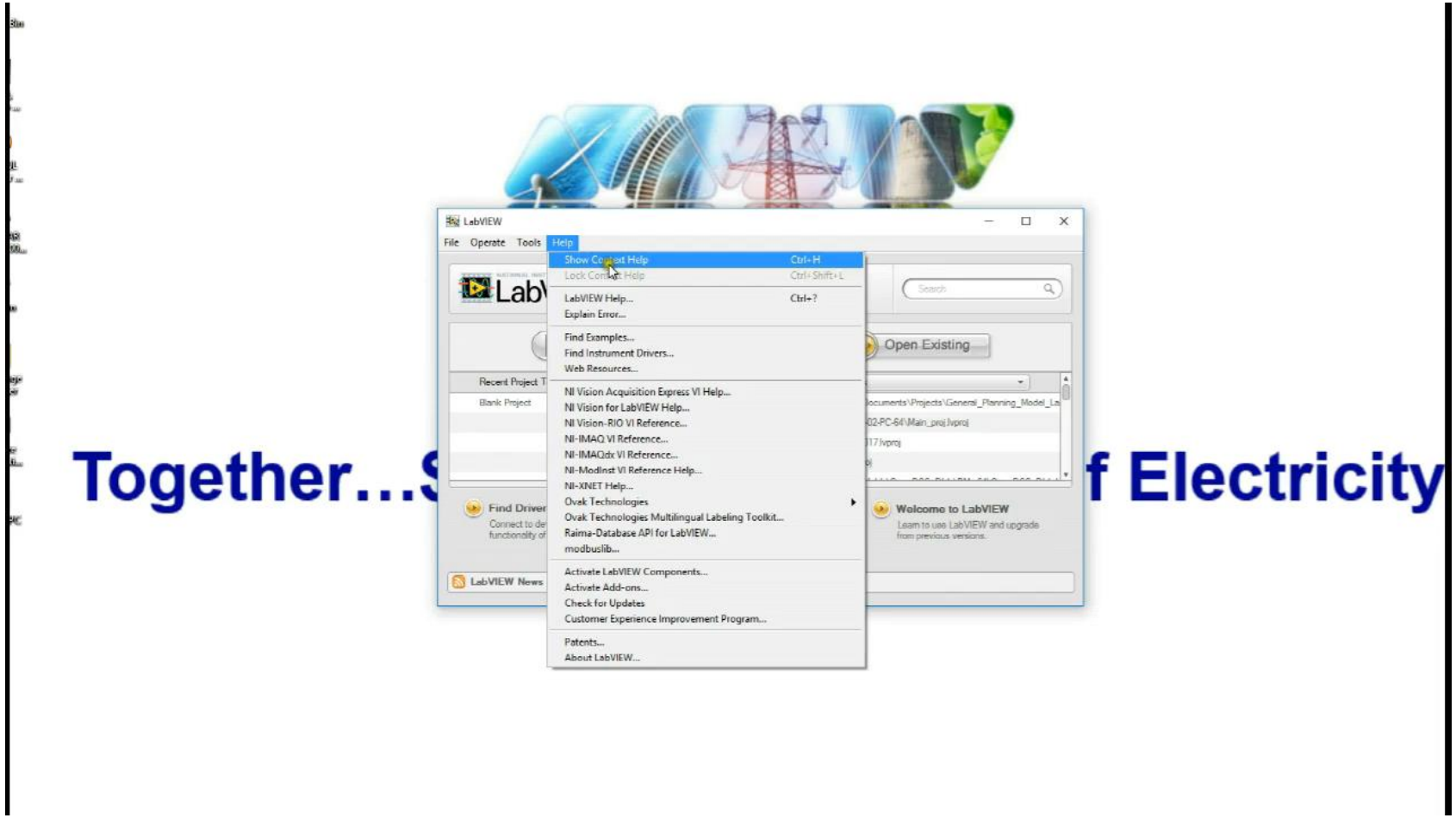
VI Package™ Manager





Temporal parallelization

- “The computational burden is on the simulation intensity”



Temporal parallelization

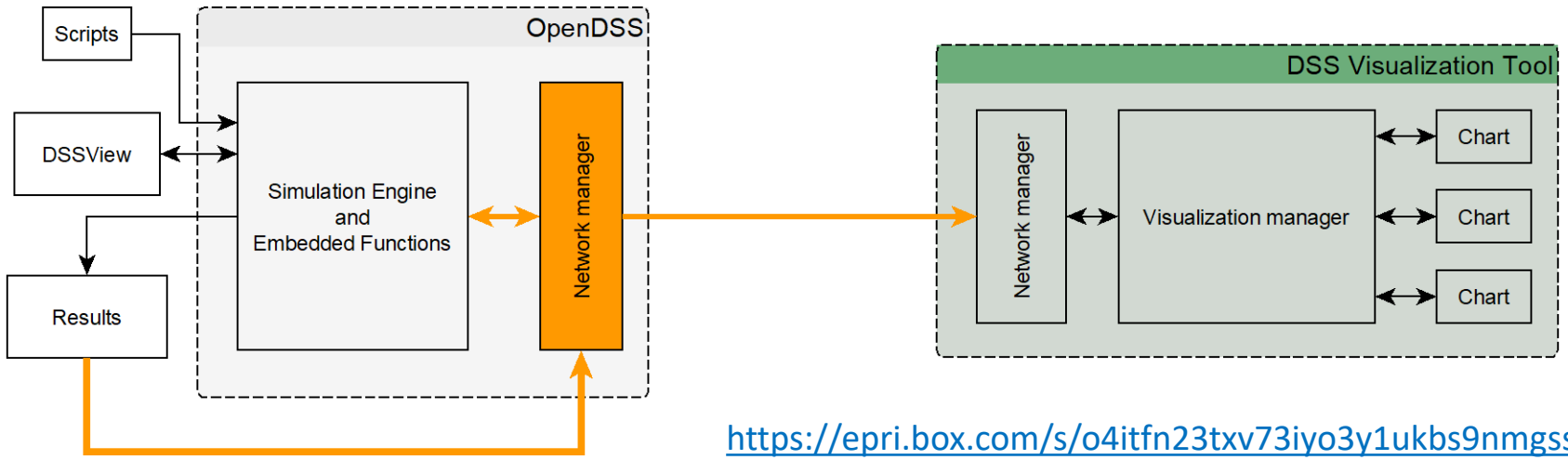
- “The computational burden is on the simulation intensity”

Temporal parallelization

- Complementary tools

Advanced Graphics Module for OpenDSS (OpenDSS-Viewer)

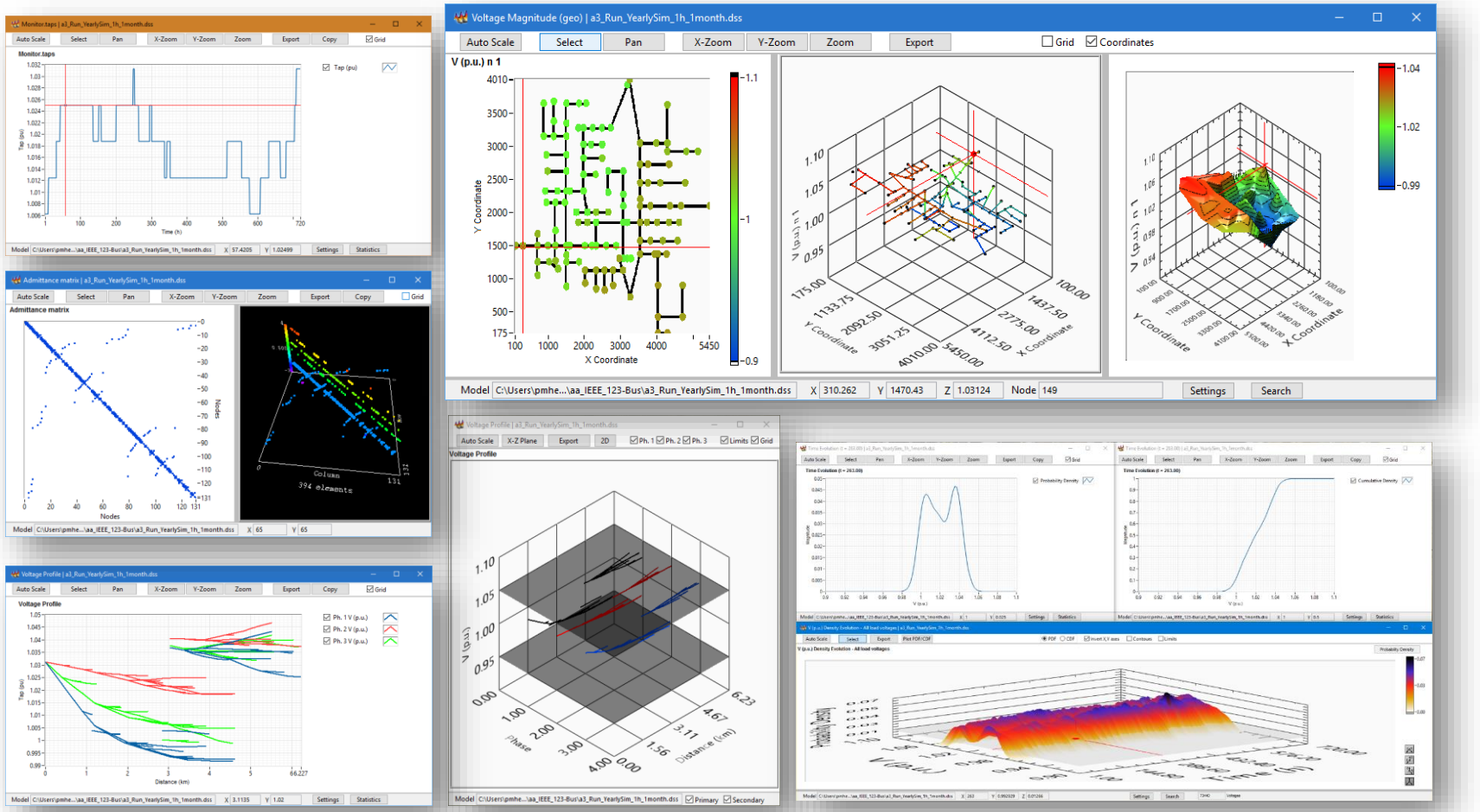
Developed by Miguel Hernandez (EPRI). Enhance the visualization of Distribution System Simulations with a **flexible**, **scalable** and **meaningful** approach.



https://epri.box.com/s/o4itfn23txv73iyo3y1ukbs9nmgsspgo

Temporal parallelization

- Complementary tools

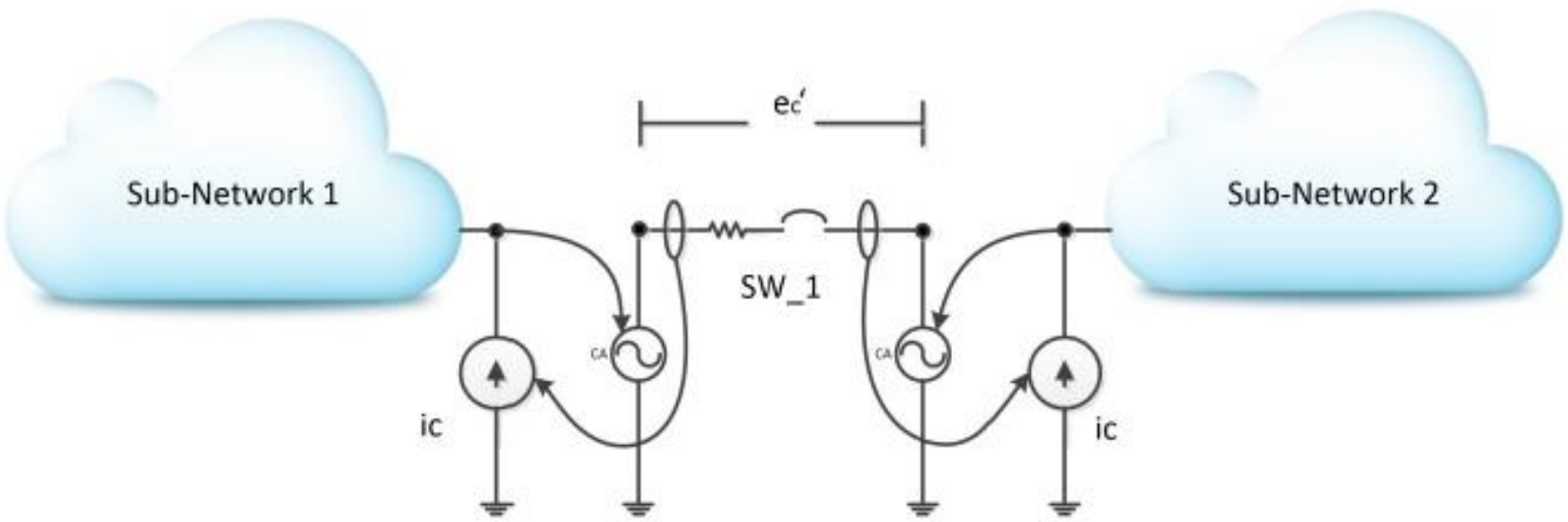


Actor based Diakoptics (A-Diakoptics)

A-Diakoptics

- "The computational burden is on the model complexity"

$$\begin{bmatrix} E_T \\ 0 \end{bmatrix} = \begin{bmatrix} Z_{TT} & Z_{TC} \\ Z_{CT} & Z_{CC} \end{bmatrix} \begin{bmatrix} I_0 \\ i_C \end{bmatrix}$$



A-Diakoptics

- "The computational burden is on the model complexity"

  - *Diakoptics proposed by G. Kron in the 50's.*
  - *Authors have used it for interconnecting independent system in several domains.*
  - *H. Happ proposed a systematic approach based on the Z matrix (Z-Diakoptics).*
  - *This approach was initially conceived was to cover systems with multiple excitations.*

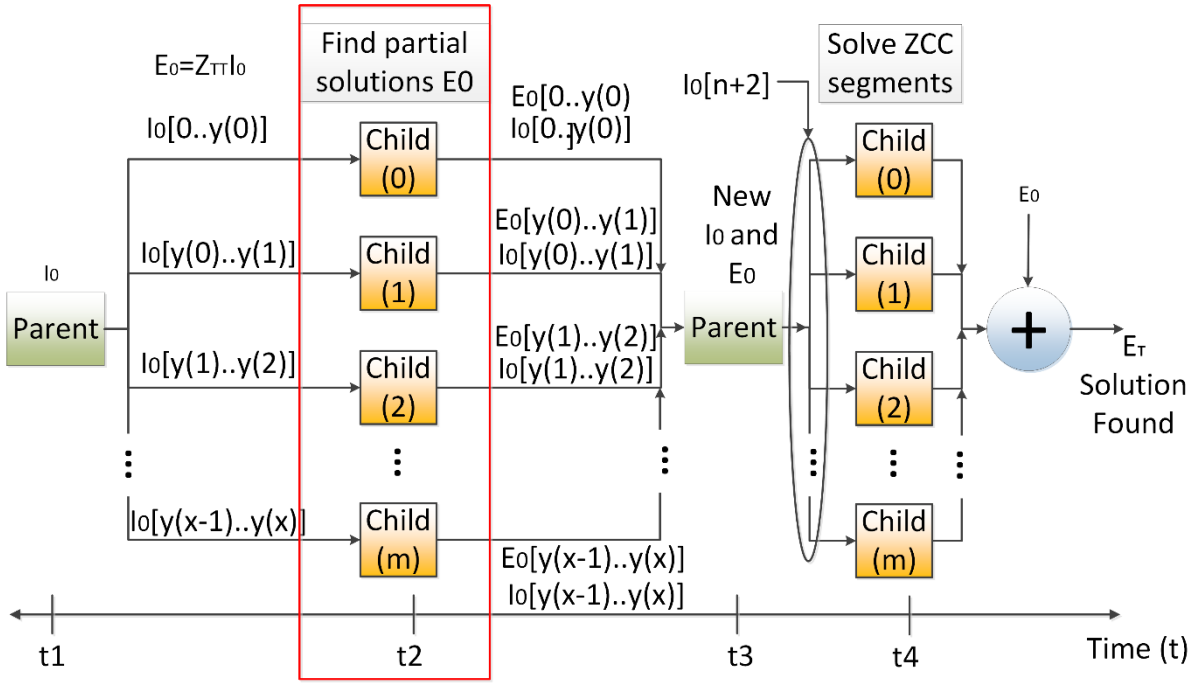
A-Diakoptics

- "The computational burden is on the model complexity"

  - *There are modern methods to solve the power flow problem…*
  - *This strategy can be used to generate several solvers working together concurrently…*
  - *Each primitive can be solved independently…*
  - *This means that the reference vector (currents) will change during the solving process…*

A-Diakoptics

- "The computational burden is on the model complexity"



"Multilevel A-Diakoptics for the Dynamic Power Flow Simulation of Hybrid Power Distribution Systems". D. Montenegro, G. Ramos, S. Bacha.

IEEE Transactions on Industrial Informatics.

"Iterative method for detecting and localizing islands within sparse matrixes using DSSim-RT". D. Montenegro, G. Ramos.

IEEE PEPQA 2015.

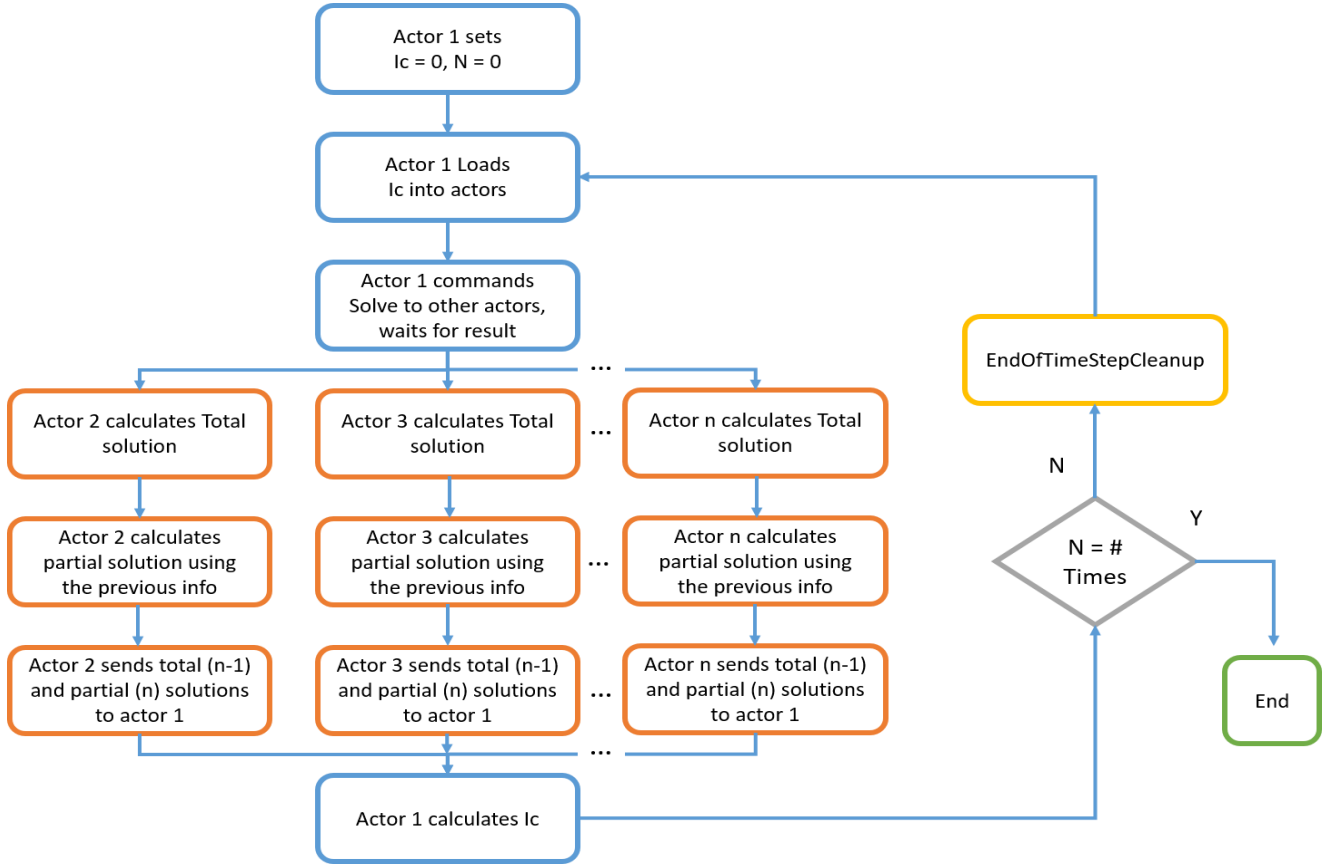
$$E_T = \sum_{l=1}^{k}\left(Z_{TT(l-1)}I_{0(n)} - Z_{TC(l)}Z_{CC(l)}^{-1}Z_{CT(l)}I_{0(n+m)}\right) \qquad 3.30$$

Simplified A-Diakoptics (New)

- "The computational burden is on the model complexity"

$$(E_T = [Y_{TT}]^{-1} I_{0(n)} + [Y_{TT}]^{-1} I_c)$$

$$I_c = -C Z_{CC}^{-1} C^T E_{T(0)}$$



https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Version8/Distrib/Doc/A_Diakoptics_Suite.pdf

Simplified A-Diakoptics (New)

- "The computational burden is on the model complexity"

$$Circuit\ reduction\ (\%) = \left(1 - \frac{\max(actor_m.NumNodes|_{m=2}^{N})}{Total\_NumNodes}\right)100 \qquad N = \#\ sub - circuits$$

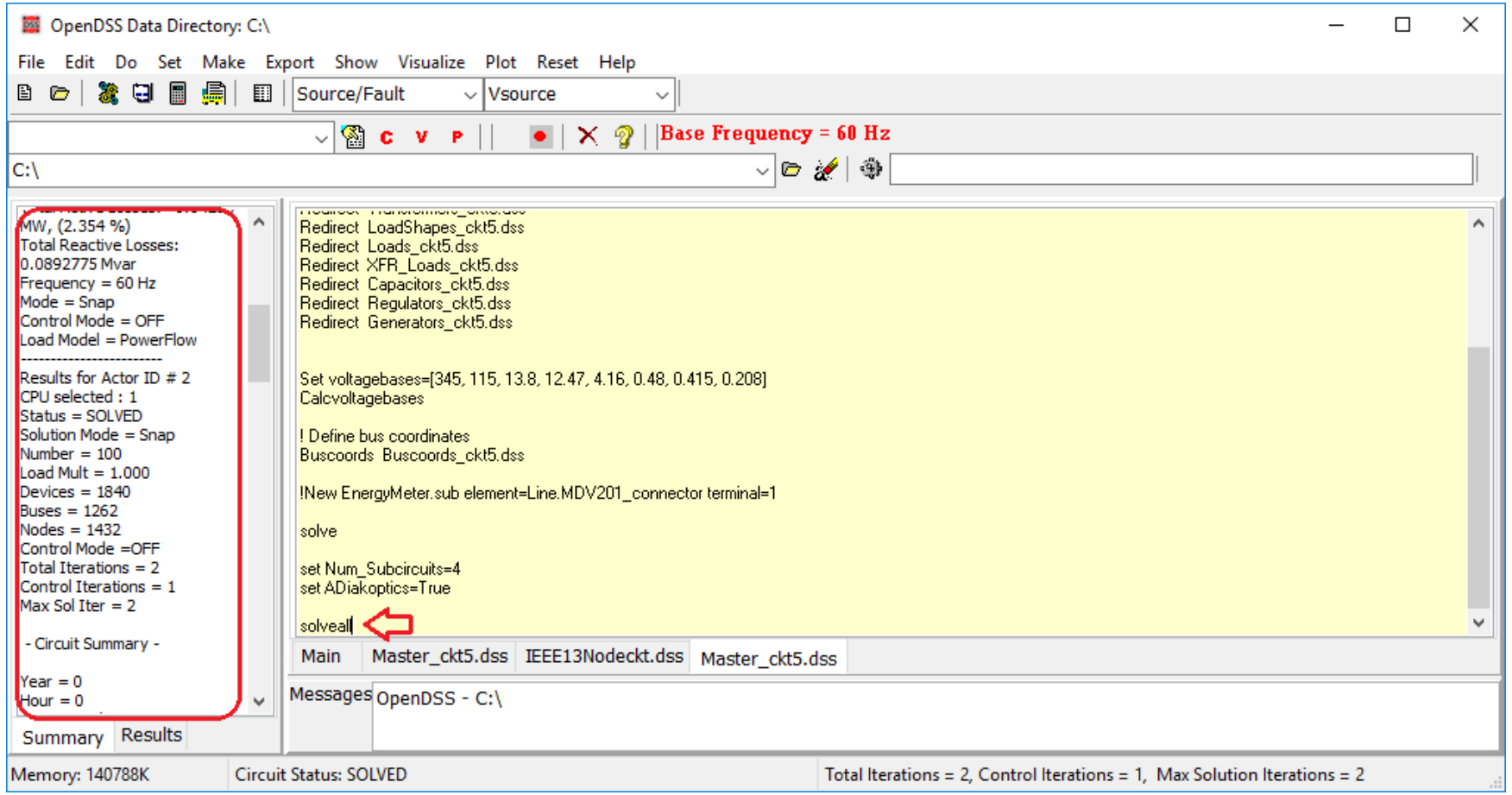
$$Max\ imbalance\ (\%) = \left(1 - max\left(\frac{actor_m.NumNodes}{Largest.NumNodes}\Big|_{m=2}^{N}\right)\right)100 \qquad N = \#\ sub - circuits$$

$$avg\ imbalance\ (\%) = mean\left(\left[1 - \frac{actor_m.NumNodes}{Largest.NumNodes}\right]\Big|_{m=2}^{N}\right)100 \qquad N = \#\ sub - circuits$$

Simplified A-Diakoptics (New)

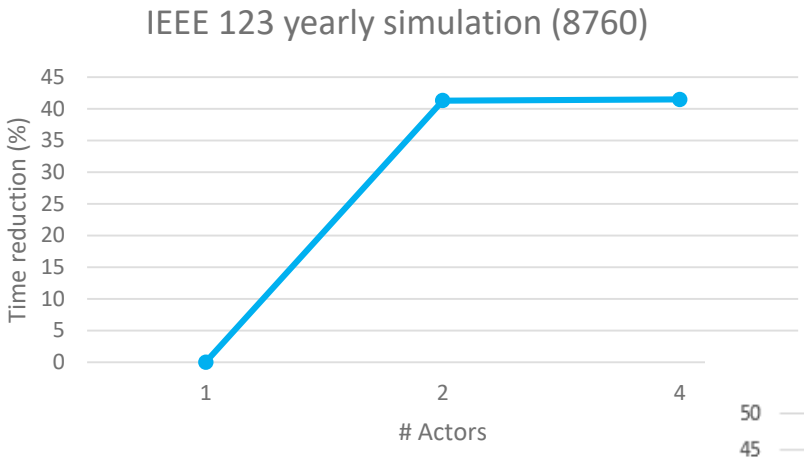
- "The computational burden is on the model complexity"

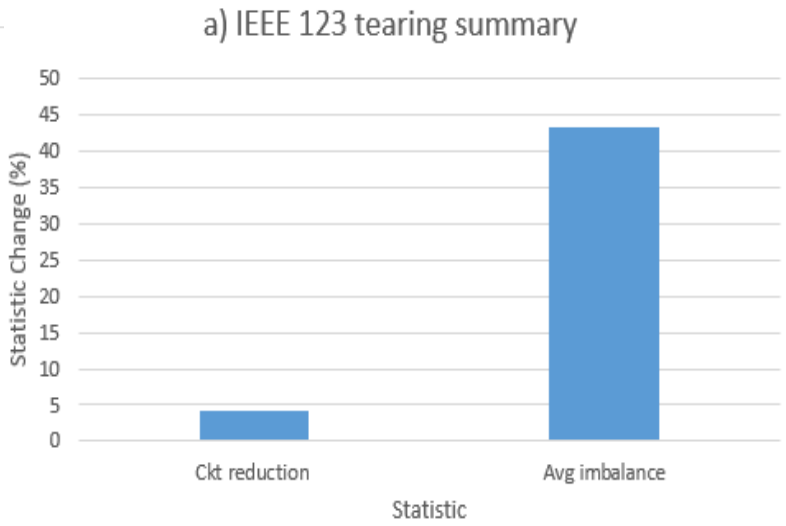
```
Circuit reduction     (%): 68.52
Max imbalance         (%): 42.33
Average imbalance     (%): 20.52


Circuit reduction     (%): 46.34
Max imbalance         (%): 13.64
Average imbalance     (%): 6.818


Circuit reduction     (%): 60.95
Max imbalance         (%): 94.04
Average imbalance     (%): 48.67
```

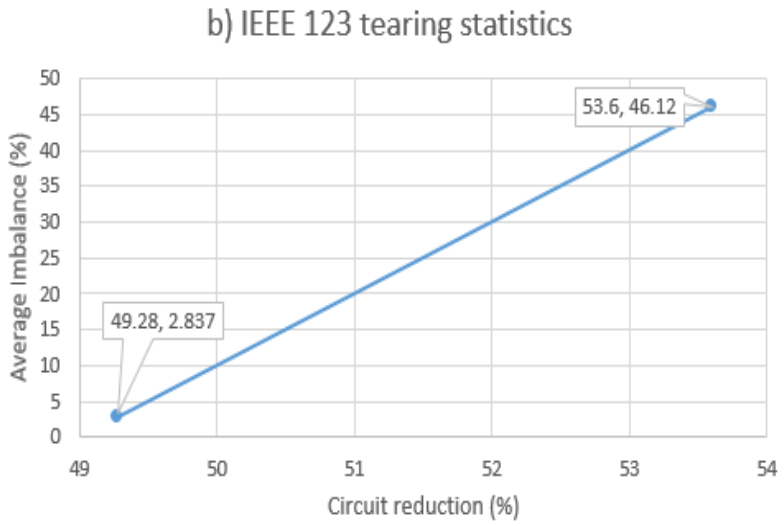


Simplified A-Diakoptics (New)

- "The computational burden is on the model complexity"

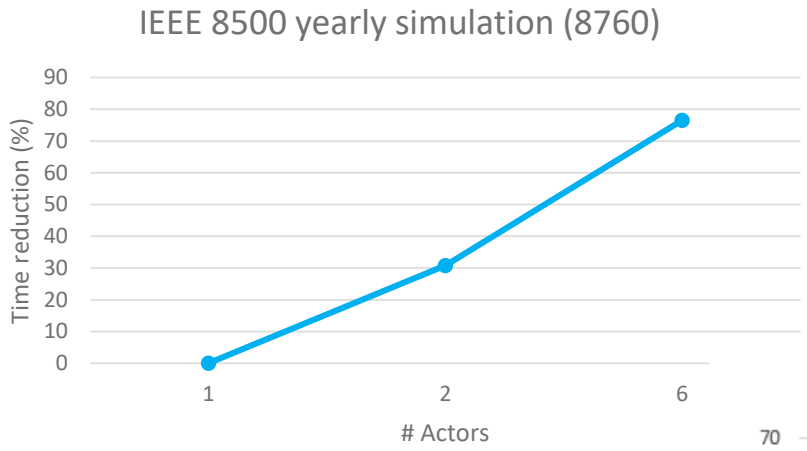


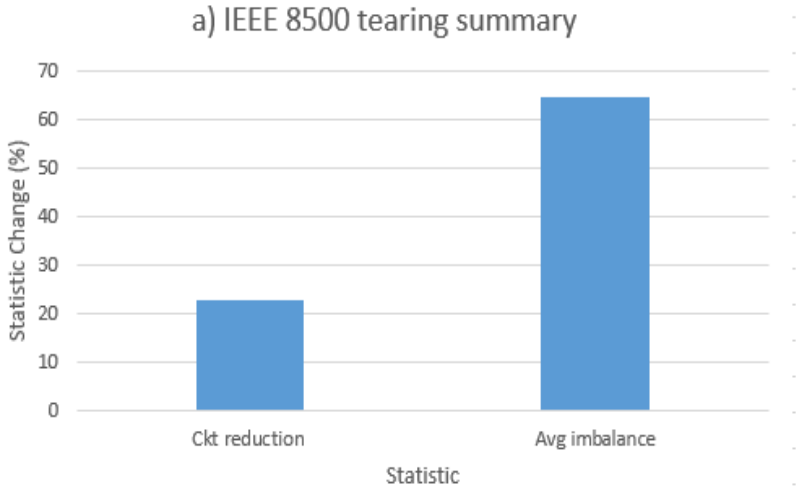


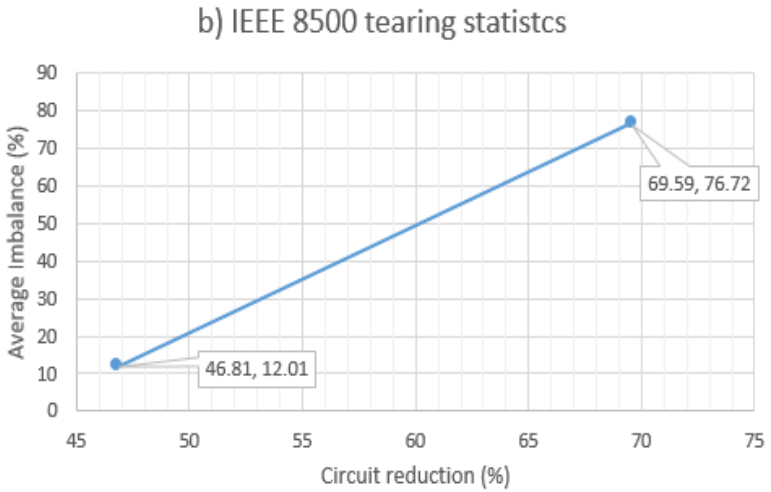


Simplified A-Diakoptics (New)

- "The computational burden is on the model complexity"



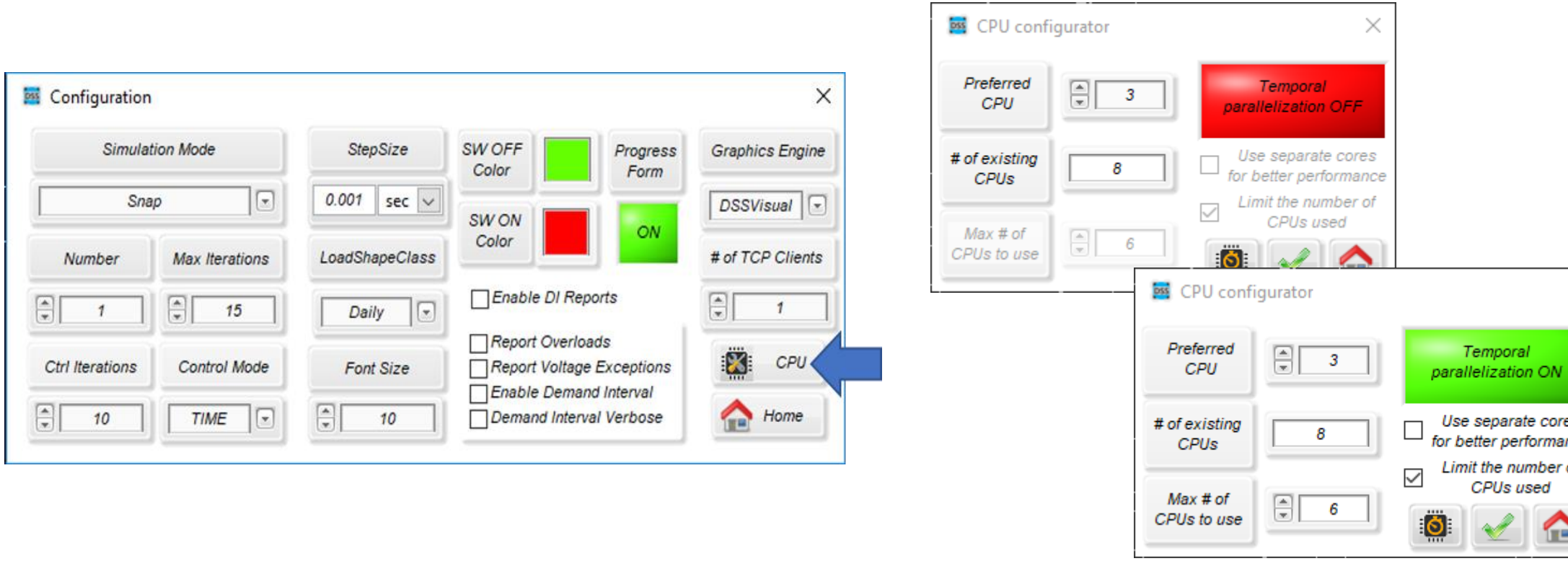




Parallel processing using OpenDSS-G

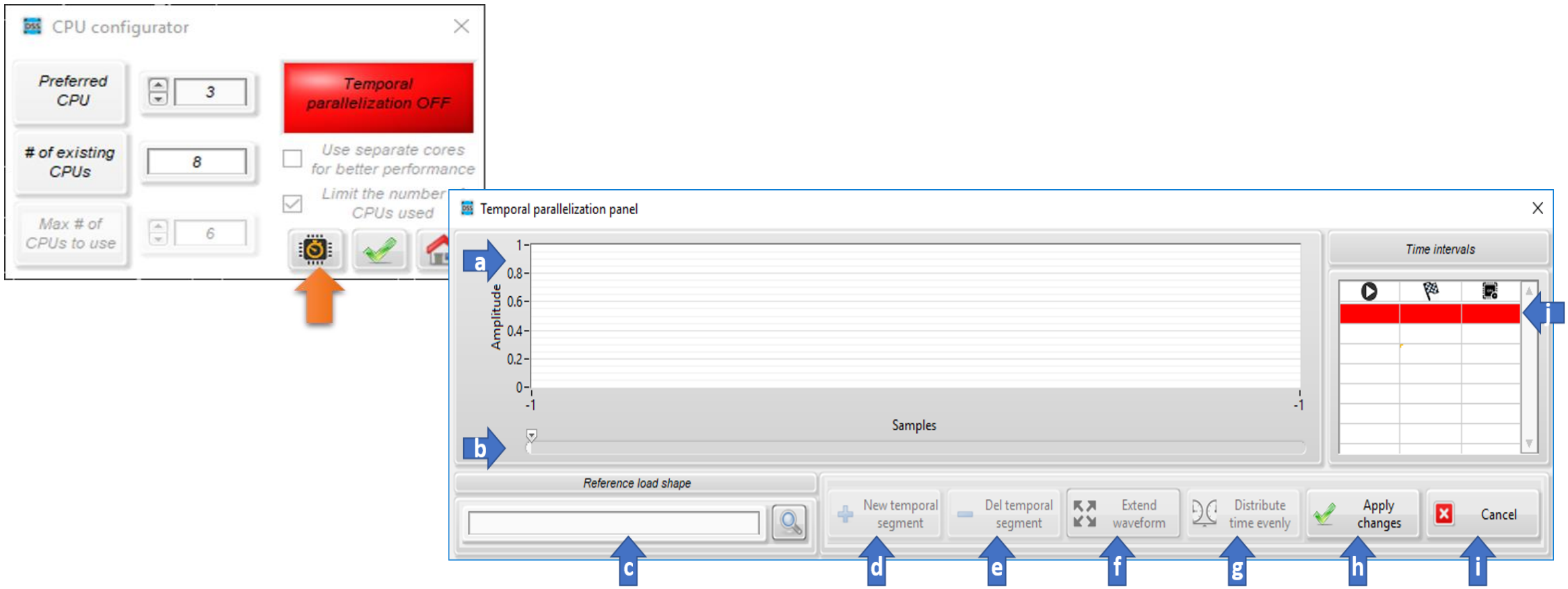
Parallel processing using OpenDSS-G

- OpenDSS-G includes parallel processing capabilities that can be used through the *Configuration* menu.



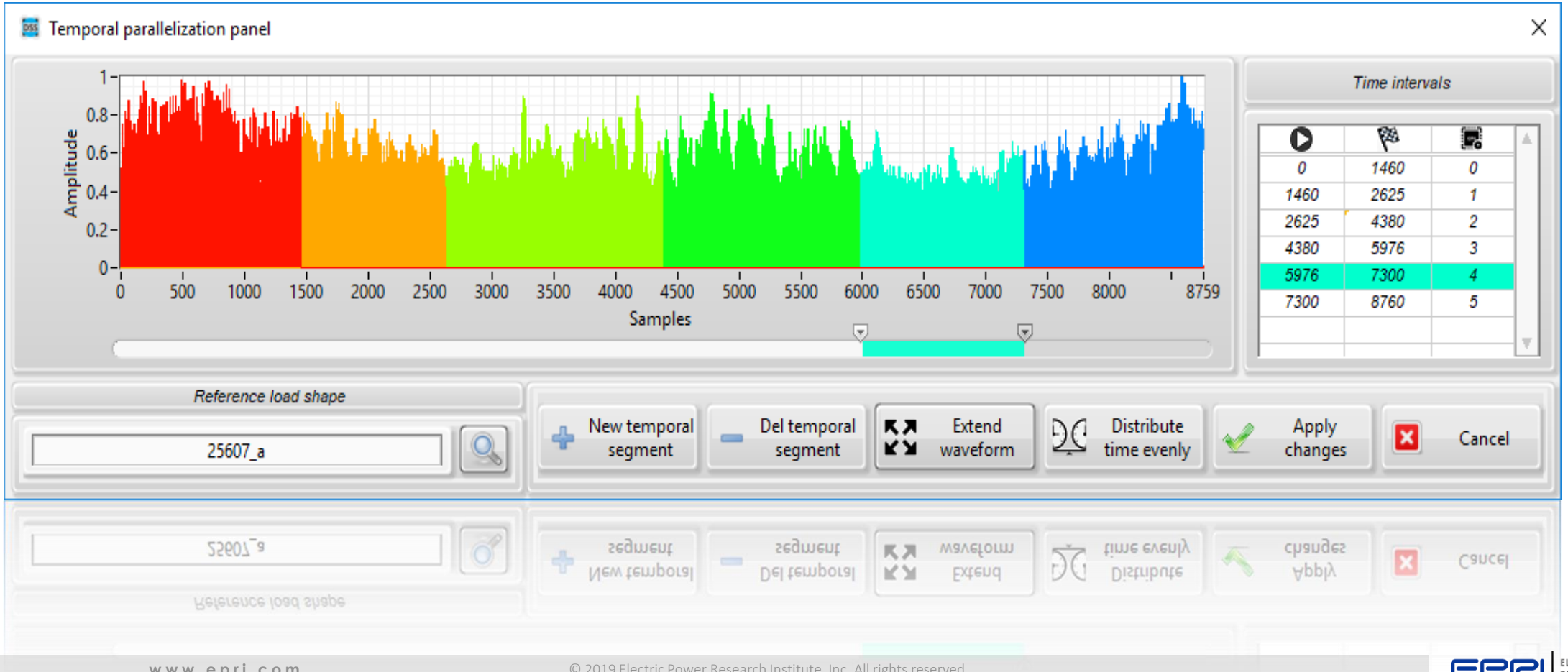
Parallel processing using OpenDSS-G

- OpenDSS-G includes parallel processing capabilities that can be used through the *Configuration* menu.



Parallel processing using OpenDSS-G

- OpenDSS-G includes parallel processing capabilities that can be used through the *Configuration* menu.



Parallel processing using OpenDSS-G

- OpenDSS-G includes parallel processing capabilities that can be used through the *Configuration* menu.

Today's challenge

Today's challenge

Given the system at
https://sourceforge.net/p/dssimpc/code/HEAD/tree/trunk/Distribution/Examples/IEEE_123_Smart_Inverters/

Add a smart inverter to the PV systems on site to evaluate 20 different control curves using daily simulations with 1 sec simulation time resolution. The analysis cannot take more than 3 minutes.

Together...Shaping the Future of Electricity