PySAM Workshop

Darice Guittet
2020 SAM Webinars
Oct 14, 2020
SAM Webinars for 2020

Introduction to SAM Workshop July 22  
PV Systems in SAM 2020.2.29 Aug 5  
Batteries in SAM 2020.2.29: 
  Focus on Battery Technology Aug 19  
  Behind-the-Meter Systems Sep 2  
  Front-of-Meter Systems Sep 16  
PySAM Workshop Oct 14

This webinar will be recorded and posted on the SAM website at
https://sam.nrel.gov/
Questions and Answers

Desktop application

Instant Join Viewer
<table>
<thead>
<tr>
<th></th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>How to Get Started</td>
</tr>
<tr>
<td>3</td>
<td>Model Initialization</td>
</tr>
<tr>
<td>4</td>
<td>Getting Module Information</td>
</tr>
<tr>
<td>5</td>
<td>Detailed PV-Battery Commercial Example</td>
</tr>
<tr>
<td>6</td>
<td>Helper Functions</td>
</tr>
<tr>
<td>7</td>
<td>Questions / More Examples</td>
</tr>
</tbody>
</table>
1 Introduction
2 How to Get Started
3 Model Initialization
4 Getting Module Information
5 Detailed PV-Battery Commercial Example
6 Helper Functions
7 Questions / More Examples
What is SAM?

Weather Data + System Specs = System Losses

+ Costs + Compensation = Financing + Incentives

= Results

Annual, Monthly, and Hourly Output, Capacity Factor, LCOE, NPV, Payback, Revenue
Photovoltaics
  Detailed & PVWatts
  High Concentration PV
Battery Storage
  Detailed & PVWatts
  Generic System
Concentrating solar power
Wind
Fuel Cell
Geothermal
Solar water heating
Biomass
Marine Energy

Distributed
  Residential
  Commercial
  Third-party ownership
Power Purchase Agreements
  Single owner
  Equity flips
  Sale-leaseback
Merchant Plant
Host/Developer
Simple LCOE calculator
What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Unit modules called compute_modules in the SSC code
What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Unit modules called compute_modules in the SSC code

A single simulation is a process chaining together multiple unit modules

- Order
- Information needs to be passed from one to the next
What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM
  • Unit modules called compute_modules in the SSC code

A single simulation is a process chaining together multiple unit modules
  • Order
  • Information needs to be passed from one to the next
  • Assembled behind the scenes in SAM user interface

PySAM, and SAM's other software development kits, expose these unit modules so that they can be customized and embedded in software applications
What is PySAM?

Python package that enables you to run the underlying modules that make up a simulation in SAM

- Unit modules called compute_modules in the SSC code

A single simulation is a process chaining together multiple unit modules

- Order
- Information needs to be passed from one to the next
- Assembled behind the scenes in SAM user interface

PySAM, and SAM's other software development kits, expose these unit modules so that they can be customized and embedded in software applications

PySAM does NOT contain all the features in the SAM GUI
PySAM Versions

Official

Release Notes

Version 2.1.4, June 8, 2020 ~ SAM 2020.2.29 r2, SSC Version 240
- Pwattsv5_ips bug fix
- Self-shading calculation speed-up for Pwattsv7, Pvsamv1 & Pwattsv5

- SAM Release fixes for revision 2

Version 2.1.1, May 15, 2020 ~ SAM 2020.2.29 r1, SSC Version 238
- recpt size post bug
- ssc_sim_from_dict bug fix
- Version attribute: PySAM.__version__

Development

Version 2.2.0
- Rename Stand Alone Battery to Battery

Version 2.1.5.dev3, Sep 3, 2020 ~ SAM 2020.2.29 r3, SSC Version 242
- Price Signals Dispatch
- Bug fix in PVWattsBatteryCommercial and PVBatteryCommercial incentives defaults

Version 2.1.5.dev2, Aug 10, 2020 ~ SAM 2020.2.29 r3, SSC Version 242
- BatteryStateful bug fixes: current

Version 2.1.5.dev1, Aug 3, 2020 ~ SAM 2020.2.29 r3, SSC Version 242
- BatteryStateful bug fixes: thermal, voltage
- Stub files syntax fix

New version of SAM and PySAM in mid-November
https://github.com/NREL/pysam/blob/master/Examples/PySAMWorkshop.ipynb
Install Python or Anaconda, a Python distribution platform
  • 64-bit Python 3.5-3.8 for Linux, Mac and Windows

pip install nrel-pysam

conda install -u nrel nrel-pysam nrel-pysam-stubs
  • Note the name has NREL prefixed
  • Nrel-pysam-stubs is automatically downloaded using pip
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>How to Get Started</td>
</tr>
<tr>
<td>3</td>
<td>Model Initialization</td>
</tr>
<tr>
<td>4</td>
<td>Getting Module Information</td>
</tr>
<tr>
<td>5</td>
<td>Detailed PV-Battery Commercial Example</td>
</tr>
<tr>
<td>6</td>
<td>Helper Functions</td>
</tr>
<tr>
<td>7</td>
<td>Questions / More Examples</td>
</tr>
<tr>
<td></td>
<td>Section</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>How to Get Started</td>
</tr>
<tr>
<td>3</td>
<td>Model Initialization</td>
</tr>
<tr>
<td>4</td>
<td>Getting Module Information</td>
</tr>
<tr>
<td>5</td>
<td>Detailed PV-Battery Commercial Example</td>
</tr>
<tr>
<td>6</td>
<td>Helper Functions</td>
</tr>
<tr>
<td>7</td>
<td>Questions / More Examples</td>
</tr>
</tbody>
</table>
Getting Module Information

Explore the user interface:
- What technology and financial simulations are available
- Categories of inputs
- Inputs’ data requirements
- Interdependent inputs

PySAM requires users to maintain consistency
- Changes in the correct order
- Data flow from one unit module to the next
In SAM, a set of input pages is loaded to present all the inputs that are needed for the whole simulation.

A list of all simulation configurations and their unit models in order can be found on the PySAM documentation site under “SAM Simulation Configurations.”
Getting Module Information

• For each unit module, PySAM inputs are categorized into groups. These groups roughly correspond to the SAM UI pages.

• But sometimes they don’t.
Getting Module Information

• For each unit module, its inputs are categorized into groups. These groups roughly correspond to the SAM UI pages.

• But sometimes they don’t.

• SDKtool, variable tables in SSC source code
Getting Module Information

• For each unit module, its inputs are categorized into groups. These groups roughly correspond to the SAM UI pages.

• But sometimes they don’t.

• SDKtool, variable tables in SSC source code
Getting Module Information

Input consistency

• “Variable may need to be updated if the values of the following have changed”
• Intra-module input dependencies
• In SAM, automatically handled
• In PySAM, up to user

\[\text{inverter\_count} \]
\begin{align*}
\text{Number of inverters} \\
\text{Constraints: INTEGER,POSITIVE} \\
\text{Required: True}
\end{align*}

This variable may need to be updated if the values of the following have changed:

• 6par\_imp
• 6par\_vmp
• 6par\_voc
• cec\_i\_mp\_ref
• cec\_v\_mp\_ref

\[\text{6par\_imp} \]
\begin{align*}
\text{Imp [A]} \\
\text{Required: True if module\_model=2}
\end{align*}

Changes to this variable may require updating the values of the following:

• inverter\_count
• subarray1\_modules\_per\_string
• subarray1\_nstrings
• subarray2\_enable
• subarray3\_enable
• subarray4\_enable
• system\_capacity
Input consistency

• “Variable may need to be updated if the values of the following have changed”
• Intra-module input dependencies
• In SAM, automatically handled
• In PySAM, up to user
1 Introduction
2 How to Get Started
3 Model Initialization
4 Getting Module Information
5 Detailed PV-Battery Commercial Example
6 Helper Functions
7 Questions / More Examples
Using a set of PSM weather files for different years, calculate how the average net present value (NPV) of the default Detailed PV-Battery – Commercial owner system changes with the size of a four-hour battery.

lexington_or_45.446370_-119.687903_psmv3_30_1998.csv
lexington_or_45.446370_-119.687903_psmv3_30_1999.csv
lexington_or_45.446370_-119.687903_psmv3_30_2000.csv
lexington_or_45.446370_-119.687903_psmv3_30_2001.csv
lexington_or_45.446370_-119.687903_psmv3_30_2002.csv
lexington_or_45.446370_-119.687903_psmv3_30_2003.csv
lexington_or_45.446370_-119.687903_psmv3_30_2004.csv
lexington_or_45.446370_-119.687903_psmv3_30_2005.csv
lexington_or_45.446370_-119.687903_psmv3_30_2006.csv
lexington_or_45.446370_-119.687903_psmv3_30_2007.csv
lexington_or_45.446370_-119.687903_psmv3_30_2008.csv
lexington_or_45.446370_-119.687903_psmv3_30_2009.csv
lexington_or_45.446370_-119.687903_psmv3_30_2010.csv
lexington_or_45.446370_-119.687903_psmv3_30_2011.csv
lexington_or_45.446370_-119.687903_psmv3_30_2012.csv
lexington_or_45.446370_-119.687903_psmv3_30_2013.csv
lexington_or_45.446370_-119.687903_psmv3_30_2014.csv
lexington_or_45.446370_-119.687903_psmv3_30_2015.csv
lexington_or_45.446370_-119.687903_psmv3_30_2016.csv
lexington_or_45.446370_-119.687903_psmv3_30_2017.csv
lexington_or_45.446370_-119.687903_psmv3_30_2018.csv
## Detailed PV-Battery - Commercial

<table>
<thead>
<tr>
<th>Detailed PV Model – Commercial Owner</th>
<th>Photovoltaic system using detailed photovoltaic model with separate module and inverter component models. Renewable energy system displaces commercial building electric load</th>
<th>Pvsamv1, Grid, Utilityrate5, Cashloan</th>
</tr>
</thead>
</table>

### Choose a performance model, and then choose from the available financial models.

- **Pvsamv1**
  - Photovoltaic (detailed)
  - Detailed photovoltaic system model with separate components for module and inverter

- **Grid**
  - Grid
  - Interconnect and Curtailment limits

- **Utilityrate5**
  - Residential, Commercial, Third Party, Host Developer
  - Retail electricity bill calculator

- **Cashloan**
  - Residential and Commercial
  - Financial model for residential and commercial behind-the-meter projects
Detailed PV-Battery - Commercial

Pvsamv1

Wrapper for SAM Simulation

Creating an Instance

There are three methods to create a new class with attributes and method configuration:

- PySSC

Pvsamv1 model description

Detailed photovoltaic system

PySAM.Pvsamv1.defa

Use financial configuration:

- "FlatPlatePVA"
- "FlatPlatePVC"
- "FlatPlatePVH"
- "FlatPlatePVL"
- "FlatPlatePVL"
- "FlatPlatePVW"
- "FlatPlatePVV"
- "FlatPlatePVR"
- "FlatPlatePVS"
- "FlatPlatePVT"
- "PVBatteryAll"
- "PVBatteryCo"
- "PVBatteryHo"
- "PVBatteryLc"
- "PVBatteryMe"
- "PVBatteryRe"
Helper Functions

ResourceTools
- TMY_CSV_to_solar_data
  - *TMY csv file as 'solar_resource_data' dictionary for Pvsamv1, Pvwattsv5, Pvwattsv7, ...*
- SRW_to_wind_data
  - *SRW csv file as 'wind_resource_data' dictionary for Windpower*
- URDBv7_to_ElectricityRates
  - *Utility Rate Database API version 7 response as Utilityrate5 inputs*
- FetchResourceFiles
  - *Downloader for National Solar Radiation Database and Wind Toolkit*

BatteryTools
- battery_model_sizing
  - *Modifies model for desired power and capacity*
1 Introduction
2 How to Get Started
3 Model Initialization
4 Getting Module Information
5 Detailed PV-Battery Commercial Example
6 Helper Functions
7 Questions / More Examples