

pbsPowerGridLab

Version: 1.0

Kamjoo Bayat – kb@pbscontrol.com

Date: September 2026

Table of Contents**1 – Introduction****2 – Installation****3 – Basic Power Grid Concepts****4 – Defining a new Simple model****5 - Complex System network Model for Power Grid****6 - Power Grid Operation Simulations****7 - Power Grid Model Network Metrics**

1 - Introduction

pbsPowerGridLab is an advanced software environment for generating power-grid models based on multiple network-topology frameworks and analyzing them using modern complex-systems methods. The platform enables users to construct synthetic electrical networks, explore their structural and dynamic behavior, and apply a wide range of analytical tools derived from complex-systems science.

pbsPowerGridLab also serves as a powerful data-generation engine for machine-learning applications across different grid-operation modes. Users can simulate normal, stress, cascade, and blackout scenarios, producing high-quality datasets suitable for training and evaluating ML models in grid-stability assessment, anomaly detection, and failure-prediction tasks.

pbsPowerGridLab performs all grid simulations and data-generation tasks using AC Power Flow calculations. To ensure accurate and efficient computation, the platform integrates the `power_grid_model` Python library, a high-performance engine for steady-state power-flow analysis.

For correct operation of pbsPowerGridLab, users must have Python installed on their system along with the `power_grid_model` package. These components provide the numerical backend required for building network models, solving AC power flow, and exporting data files for further analysis or machine-learning workflows.

2 - Installation

To install pbsPowerGridLab, run the pbsPowerGridLabSetupV1_0.exe installer. The setup program will guide you through the installation process and place all required components on your system.

After installation is complete, a shortcut icon will appear on your desktop:



pbsPowerGridLab requires .NET Framework 4.7.2, which is included by default in all modern Windows 10 and Windows 11 installations. In most cases, no additional setup is needed, and the application will run immediately after installation.

Download the Windows version of Python from the official website at python.org and install it on the same PC where pbsPowerGridLab is installed. At the time of writing this manual, the latest available 64-bit release is Python 3.13.15.

You can download the installer directly using the following link:

<https://www.python.org/ftp/python/3.13.15/python-3.13.15-amd64.exe>

After installation, verify that Python is correctly installed by running `python --version` in a Command Prompt window.

After installing Python, you must install the `power_grid_model` library, which provides the numerical engine used by pbsPowerGridLab for AC Power Flow calculations. You can find detail of `power_grid_model` from following link :

<https://power-grid-model.readthedocs.io/en/stable/index.html>

Open a Command Prompt window and run the following command:

```
pip install power-grid-model
```

This will download and install the latest compatible version of the library from the Python Package Index (PyPI).

When you launch pbsPowerGridLab, the interface is organized into three primary windows, each serving a distinct purpose in the modeling workflow:

- **Left Window — Power Grid Model List** Displays all defined power-grid models. Users can select, load, or manage existing models from this panel.
- **Right Window — Grid Elements and Properties** Shows the list of available power-grid elements (nodes, loads, generators, lines, etc.) along with their editable properties and parameters.
- **Middle Window — Model Content and Parameters** Presents the detailed structure of the selected model, including its components, configuration, and operational parameters. This is the main workspace for building and editing grid models.

In the figure-1 you can see a sample page that shows 2 defined models .

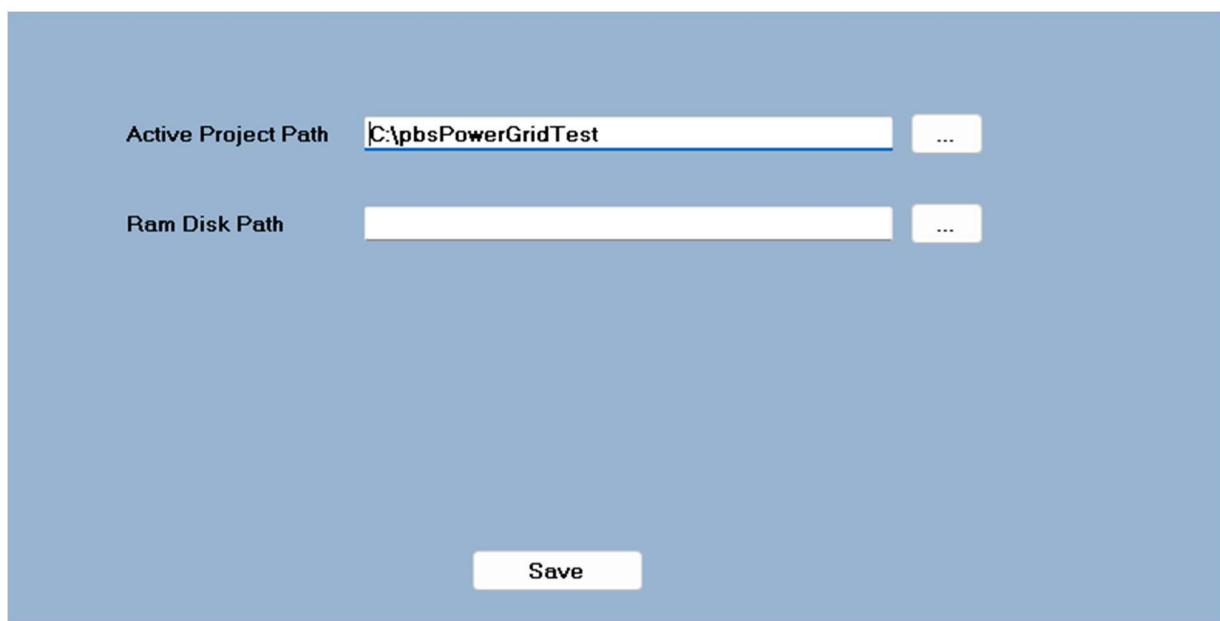


Figure - 1

Settings menu

In the File menu, open the Settings panel. Two configuration parameters can be defined here: Figure-2

- **Active Project Path** This directory is used by pbsPowerGridLab to store all model configurations, JSON files, and generated data outputs. Every project you create or load will reference this path.
- **RAM Disk Path** Because pbsPowerGridLab exchanges JSON files with the *power_grid_model* Python library, using a RAM Disk can significantly improve performance for large-scale simulations. If you prefer not to use a RAM Disk, simply leave this field empty. The software will then use the standard storage path defined by your system.



Active Project Path ...

Ram Disk Path ...

Save

Figure-2

Click the Save button to store your configuration. pbsPowerGridLab writes all settings to the application's root directory in the options.xml file. This ensures that your project paths and RAM Disk preferences are automatically loaded the next time you start the software.

3 - Basic Power Concepts

Active , Reactive and Power Factor Concepts

In AC electrical systems, voltage and current are sinusoidal and may not be perfectly in phase. The relationship between them defines how effectively electrical energy is converted into useful work.

- **Active Power (P)** represents the real power that performs mechanical or thermal work. It is measured in watts (W) and calculated as:

$$P = V_{rms}I_{rms}\cos(\phi)$$

where V_{rms} and I_{rms} are the root-mean-square voltage and current, and ϕ is the phase angle between them.

- **Reactive Power (Q)** represents the oscillating power exchanged between inductive and capacitive components, measured in volt-amperes reactive (VAR). It is given by:

$$Q = V_{rms}I_{rms}\sin(\phi)$$

Reactive power does not perform useful work but is essential for maintaining magnetic and electric fields in motors, transformers, and transmission lines.

- **Power Factor (PF)** expresses the efficiency of power usage and is defined as the ratio of active power to apparent power:

$$PF = \cos(\phi) = \frac{P}{S}$$

where $S = V_{rms}I_{rms}$ is the apparent power. A power factor close to 1 indicates efficient energy conversion, while a low power factor implies higher losses and reduced system capacity.

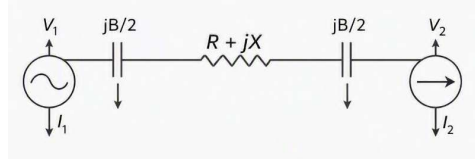
Together, these quantities describe the complete power behavior of an AC network. pbsPowerGridLab uses them to analyze system efficiency, voltage stability, and reactive-power compensation strategies.

π -model Transmission Line model

pbsPowerGridLab is using π -Model for Transmission Lines.

The π -model (Pi Model) is the most widely used equivalent circuit for representing medium-length transmission lines in steady-state power-flow analysis. It captures both the series impedance and the shunt admittance of the line, providing an accurate balance between simplicity and precision. The line is modeled as a series impedance $Z = R + jX$ between the sending and receiving ends, with half of

the total shunt admittance $Y = jB$ placed at each end to represent the line's capacitive charging effect.



The π -model allows calculation of the sending-end and receiving-end voltages and currents using the standard ABCD parameters:

$$A = D = 1 + \frac{YZ}{2}, \quad B = Z, \quad C = Y \left(1 + \frac{YZ}{4} \right)$$

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix}$$

and are exactly what pbsPowerGridLab can use internally for line modeling and validation.

These parameters form the foundation for voltage-regulation, power-flow, and fault-analysis studies. **Benefits:** The π -model accurately represents voltage drop, reactive charging current, and phase shift for typical transmission distances. **Limitations:** It assumes uniform line parameters and neglects distributed effects significant in very long lines, where the long-line model or distributed parameter model becomes necessary.

Despite these limitations, AC Power Flow remains the most widely used method for analyzing the operational state of electrical grids and is the core computational engine used inside pbsPowerGridLab.

The π -model provides an accurate representation of transmission lines whose length is less than approximately 250 km (for 50 Hz systems) or less than 150 km (for 60 Hz systems). Within this range, the distributed effects of resistance, inductance, and capacitance can be safely approximated by lumped parameters. The model assumes that voltage and current are uniform along the line segment, allowing the use of a single series impedance $Z = R + jX$ and two shunt admittances $Y/2 = jB/2$ at each end.

Admittance Matrix

The Admittance Matrix, commonly called the Y-bus, is the fundamental mathematical representation of an electrical power network used in AC Power Flow calculations. Each transmission line or transformer contributes a complex admittance $Y = G + jB$ between two nodes, where G is conductance and B is susceptance. The Y-bus is constructed by summing all admittances connected to each node:

$$Y_{ii} = \sum_{k \in \text{connected to } i} Y_{ik}$$

$$Y_{ij} = -Y_{ij} \ (i \neq j)$$

The diagonal elements Y_{ii} represent the total admittance connected to node i , while the off-diagonal elements Y_{ij} represent the negative admittance of the branch between nodes i and j . Once the Y-bus is built, it becomes the core matrix used in solving the nonlinear AC power-flow equations.

Simple Example Network

Suppose there is a 3 Node , Triangle Power grid , minimal 3-bus system Y-bus matrix becomes

$$Y_{\text{bus}} = \begin{bmatrix} Y_{11} & -Y_{12} & -Y_{13} \\ -Y_{12} & Y_{22} & -Y_{23} \\ -Y_{13} & -Y_{23} & Y_{33} \end{bmatrix}$$

Where:

- $Y_{11} = Y_{12} + Y_{13}$
- $Y_{22} = Y_{12} + Y_{23}$
- $Y_{33} = Y_{13} + Y_{23}$

This simple example demonstrates how each line contributes to the diagonal and off-diagonal elements of the Y-bus.

AC Power Flow Calculation

AC Power Flow calculation determines the steady-state operating point of an electrical power system by solving a set of nonlinear algebraic equations that describe the relationship between bus voltages, power injections, and network impedances. For each bus i , the active and reactive power injections are computed using:

$$P_i = \sum_{j=1}^N |V_i| |V_j| (G_{ij} \cos \theta_{ij} + B_{ij} \sin \theta_{ij})$$

$$Q_i = \sum_{j=1}^N |V_i| |V_j| (G_{ij} \sin \theta_{ij} - B_{ij} \cos \theta_{ij})$$

where G_{ij} and B_{ij} are the real and imaginary parts of the admittance matrix, V_i and V_j are bus voltages, and θ_{ij} is the voltage angle difference. Solving these equations provides the complete system state: bus voltages, line flows, transformer loading, and reactive-power balance.

AC Power Flow offers several benefits:

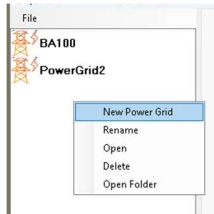
- It provides accurate representation of both active and reactive power behavior.
- It captures voltage magnitude and angle relationships essential for stability analysis.
- It supports detailed modeling of transformers, line impedances, and reactive compensation devices.
- It is the foundation for contingency analysis, grid planning, and machine-learning dataset generation.

However, AC Power Flow also has limitations:

- The equations are nonlinear, requiring iterative numerical methods (Newton–Raphson, Gauss–Seidel), which may fail to converge in stressed or unstable grid conditions.
- It represents only steady-state behavior and does not capture transient dynamics or protection-system actions.
- Large networks can be computationally expensive, especially when performing thousands of simulations for machine-learning applications.

4 - Defining a new Simple model

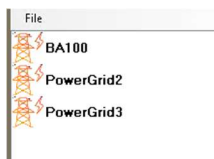
When you right-click inside the Left Panel (the *List of Models*), pbsPowerGridLab displays a context menu containing the following items:



Select “New Power Grid” from the context menu. pbsPowerGridLab scans the model folders located in the Active Project Path and automatically determines the next available model name using the format PowerGrid{N}, where N is an incremental number.

The naming logic only checks folders that follow the exact pattern `PowerGrid{N}`. If you manually rename a model to a custom name, `pbsPowerGridLab` will not consider that name when generating the next model index.

As shown in the example, when you run New Power Grid, the software detects existing folders *BA100* and *PowerGrid2*, and therefore proposes *PowerGrid3* as the name for the new model.



After defining new model , you can see figure 4-1:

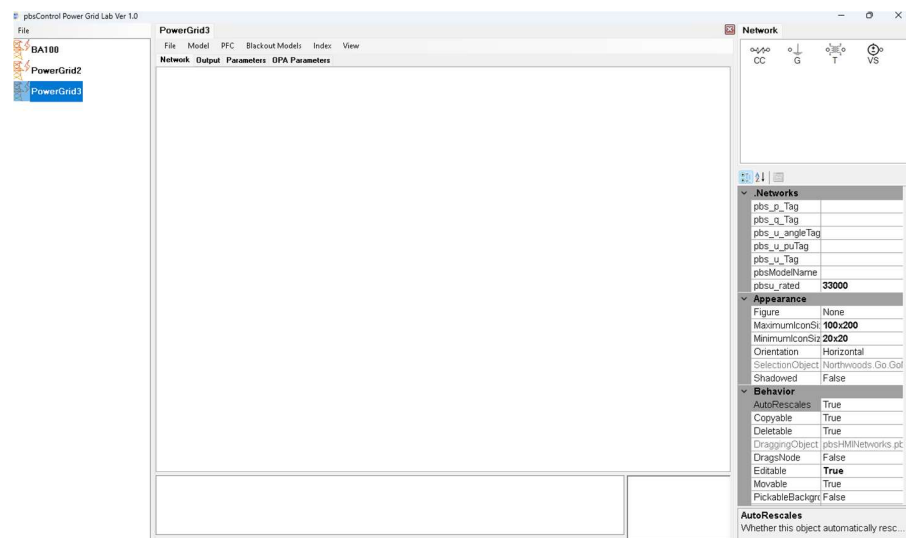


Figure 4-1

When you double-click a model in the *List of Models* panel, pbsPowerGridLab checks whether the model is already open:

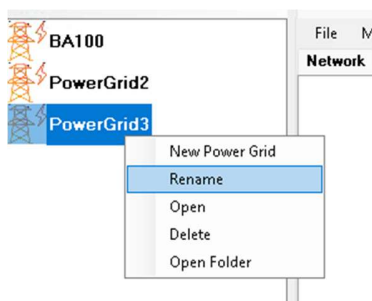
- If the model is not open, the software loads it and displays it as the active workspace.
- If the model is already open, pbsPowerGridLab simply switches focus to its existing page and marks it as the active panel.

Each model is displayed on its own independent page, with its own menu, parameters, and configuration environment.

When you create a new model, pbsPowerGridLab automatically generates a folder—named exactly the same as the model—in the Active Project Path. When you save project by File menu – Save item, Inside this folder, the software creates two initial files containing the model’s parameters and structural definitions. These files form the base configuration used for editing, simulation, and data generation.

You may rename any model to a custom name, provided that the new name does not already exist in the Active Project Path. When a model is renamed, pbsPowerGridLab automatically updates the corresponding folder and all configuration files to match the new name. For this reason, the target name must be unique within the project directory to avoid conflicts with existing model folders.

Always ensure that no folder with the same name is present in the Active Project Path before applying a rename operation.



Always rename your model before generating simulation files for Normal, Stress, Cascade, or Blackout operation modes. Simulation output files are created using the current model name, and these JSON files will not be renamed automatically if you later change the model’s name.

If you rename the model *after* generating simulation files, the folder and configuration files will be updated, but the previously generated JSON simulation files will still keep the old name, causing inconsistencies in your project structure.

In this section, we will manually design a very simple power-grid model, run an AC Power Flow calculation, and demonstrate the resulting outputs. This introductory example is intended to illustrate the fundamental power-grid elements and the core concepts used throughout pbsPowerGridLab.

Every power-grid model in pbsPowerGridLab is composed of the following element types:

- **Power Source** – Represents generators or supply points that inject active and reactive power into the network.
- **Load** – Represents consumer demand, including industrial, commercial, or residential consumption.
- **Transformer** – Provides voltage conversion between different grid levels and introduces impedance characteristics.
- **Grid Node** – Structural nodes such as substations, feeders, or junction points that define the network topology.
- **Power Transmission Line** – Connects nodes using a Pi-model representation of line impedance and admittance.

Using these components, we will construct a minimal grid, configure its parameters, and execute a power-flow simulation to demonstrate how pbsPowerGridLab processes and analyzes electrical networks.

You can drag and drop any network element from the component list into the network design area. In this example, we construct a simple grid consisting of one Power Source, one Load, two Grid Nodes, and three transmission lines connecting them. This minimal configuration is sufficient to demonstrate how pbsPowerGridLab handles node placement, element linking, and basic power-flow behavior.

Figure 4-2

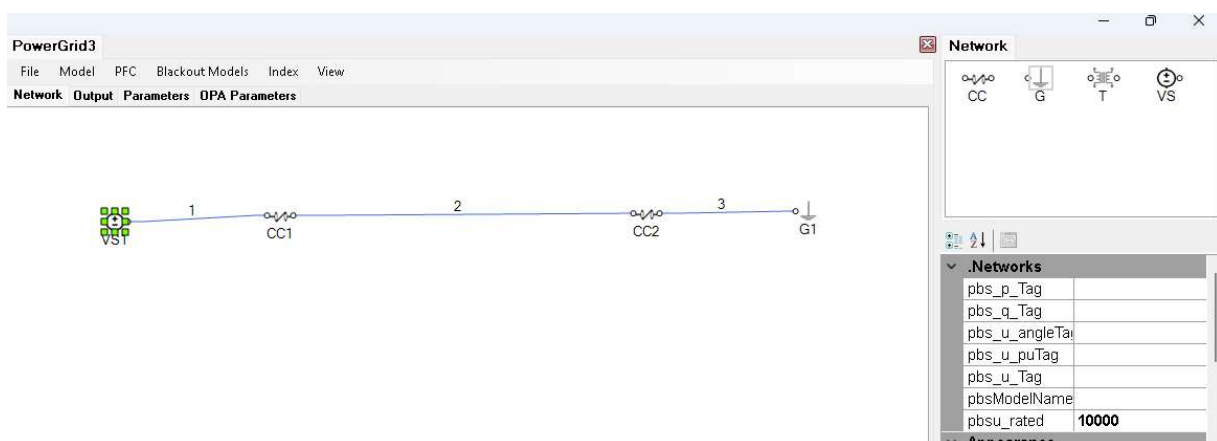


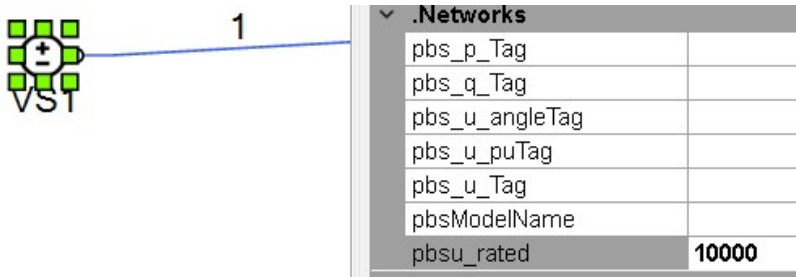
Figure 4-2

The Power Source injects active power (P) and reactive power (Q) into the network and serves as the main supply point for the grid model. It has one static configuration parameter: Base Voltage, which defines the nominal operating voltage level of the source and its connected node. This parameter is essential for per-unit calculations and for maintaining consistent voltage levels across the network.

Below is the Power Source icon used in pbsPowerGridLab:



You can set the Base Voltage of the Power Source using the pbsu_rated parameter in the *Property Grid* located on the right panel. This value defines the nominal operating voltage of the source and ensures that all connected elements operate within the correct per-unit voltage level.



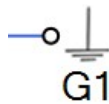
The pbsu_rated parameter defines the Base Voltage of the Power Source, and its unit is Volt. In the example above, the Base Voltage is set to 10 kV. All other parameters shown for the Power Source element belong to pbsHMI, where they are used for real-time tag connections and runtime integration. These additional parameters are not used inside the pbsPowerGridLab environment and can be ignored when designing offline grid models. Three additional parameters are available for the Power Source element: Text, ToolTipText, and PartID. PartID is automatically generated by the network and provides a unique identifier for each element. This ID is used internally during Power Flow Calculation to track and reference elements within the simulation engine. The Text and ToolTipText fields are optional metadata fields. They allow you to attach extra descriptive information to the element, but they are not used in any electrical calculation and have no effect on the simulation results.

Text	PowerSrc1
ToolTipText	Power Voltage Source
UserFlags	0
UserObject	
Ownership	
Document	pbsHMInetworks.GraphDoc
Layer	Northwoods.Go.GoLayer
Parent	
PartID	0

The Load element represents consumer demand in the network and uses two electrical parameters: Active Power (P) and Reactive Power (Q). By adjusting these values, you can simulate different categories of loads such as residential, commercial, industrial, agricultural, or EV charging stations. In the following table, you can see several typical load types that are commonly used in power-system modeling.

Load Type	Active Power (P)	Reactive Power (Q)	Notes
Residential Load	2–10 kW	1–5 kVAR	Small appliances, lighting, HVAC.
Commercial Load	20–200 kW	10–80 kVAR	Shops, offices, elevators, cooling.
Industrial Load	200–2000 kW	100–1000 kVAR	Motors, machinery, heavy equipment.
Agricultural Load	5–50 kW	3–20 kVAR	Pumps, irrigation systems.
EV Charging Load	7–150 kW	0–10 kVAR	PF often close to unity.
Street Lighting	1–5 kW	0.5–2 kVAR	Public lighting circuits.
Data Center Load	500–5000 kW	0–200 kVAR	PF close to 1 due to UPS and PFC.

In pbsPowerGridLab , Load is showing with following icon:



You can simulate different types of loads by configuring the `pbsp_specified`, `pbsq_specified`, and `pbsu_rated` properties of the Load element.

- `pbsp_specified` defines the Active Power (P) in watt.
- `pbsq_specified` defines the Reactive Power (Q) in var.
- `pbsu_rated` sets the Base Voltage of the load, and its unit is volt.

These parameters allow pbsPowerGridLab to model a wide range of load behaviors, from small residential consumers to large industrial demands, using realistic electrical characteristics.

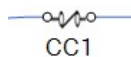
pbs_p_Tag	
pbs_q_Tag	
pbs_u_angleTag	
pbs_u_puTag	
pbs_u_Tag	
pbsModelName	
pbsp_specified	250000
pbsq_specified	60000
pbsu_rated	10000

When generating data files for machine-learning applications, pbsPowerGridLab automatically varies the load parameters pbsp_specified (active power) and pbsq_specified (reactive power) within predefined ranges. For each randomized load configuration, the system executes a full Power Flow Calculation on the grid and records the resulting electrical states. By repeating this process many times, pbsPowerGridLab produces a rich dataset containing diverse operating conditions, which is ideal for training machine-learning models for state estimation, anomaly detection, or blackout forecasting.

Node Element : The Node element represents any intermediate power-grid equipment between Sources, Loads, and Transformers, such as substations, feeders, distribution posts, and other connection points. Every electrical element in the network must be connected through a Node. A Power Source, Load, or Transformer cannot be directly connected to each other; they must always link through a Node element. This ensures a clear network structure and allows pbsPowerGridLab to correctly build the connectivity graph required for Power Flow Calculation.

The only property that the user must configure for a Node element is pbsu_rated, which defines the rated voltage of the Node in volt. During Power Flow Calculation, pbsPowerGridLab evaluates each Node in the network and computes its resulting Active Power (P), Reactive Power (Q), and Power Factor (PF). These values are derived from the connected elements and the overall grid topology, allowing each Node to represent a realistic electrical state within the simulation.

Node icon is shown in the following figure:



Power Line Element:

We use Pi Model for power line in pbsPowerGridLab. Following properties can be set for power line :

pbsLen	50
pbsLineC0	1E-06
pbsLineC1	1E-06
pbsLineI_n	2000
pbsLineR0	0.25
pbsLineR1	0.25
pbsLineTan0	0.01
pbsLineTan1	0.01
pbsLineX0	0.2
pbsLineX1	0.2

Power Line Parameters

pbsLen

Line length in kilometers (km). This value defines the physical length of the transmission or distribution line. All electrical parameters (R, X, C, Tan) are interpreted per kilometer and scaled internally using this length.

pbsLineR0

Zero-sequence resistance per kilometer (Ω/km). Represents the resistive component of the zero-sequence impedance. Used for modeling unbalanced faults, ground return paths, and asymmetrical operating conditions.

pbsLineR1

Positive-sequence resistance per kilometer (Ω/km). Defines the resistive component of the positive-sequence impedance. This is the primary resistance used in balanced steady-state power-flow calculations.

pbsLineX0

Zero-sequence reactance per kilometer (Ω/km). Inductive reactance associated with the zero-sequence network. Important for fault studies and modeling earth-return inductive effects.

pbsLineX1

Positive-sequence reactance per kilometer (Ω/km). Inductive reactance of the positive-sequence network. This value dominates the line's impedance in normal balanced operation and directly affects voltage drops and reactive power flow.

pbsLineC0

Zero-sequence shunt capacitance per kilometer (F/km). Represents the capacitive coupling of the line to ground in the zero-sequence network. Used for modeling unbalanced capacitive effects and ground-related charging currents.

pbsLineC1

Positive-sequence shunt capacitance per kilometer (F/km). Defines the line's positive-sequence capacitive charging. This parameter influences reactive power generation on long lines and affects voltage profiles in steady-state operation.

pbsLineTan0

Zero-sequence loss tangent ($\tan \delta$). Represents dielectric losses in the zero-sequence network. Used to model energy dissipation in insulation and ground-return paths.

pbsLineTan1

Positive-sequence loss tangent ($\tan \delta$). Defines dielectric losses in the positive-sequence network. This parameter contributes to the real power losses associated with line capacitance.

pbsLineI_n

Nominal current rating of the line (A). Maximum continuous current the line can carry under normal operating conditions. pbsPowerGridLab uses this value to calculate line loading percentage and to detect overload conditions during simulation.

pbsPowerGridLab uses per-kilometer electrical parameters to construct the total series impedance and shunt admittance of the line. For a line of length pbsLen (in km), the following relations apply.

Series Impedance Z

The total series impedance of the line is:

$$Z = R + jX$$

Where:

- $R = \text{pbsLineR1} \times \text{pbsLen}$ (positive-sequence resistance)
- $X = \text{pbsLineX1} \times \text{pbsLen}$ (positive-sequence reactance)

Thus:

$$Z = (\text{pbsLineR1} \cdot \text{pbsLen}) + j(\text{pbsLineX1} \cdot \text{pbsLen})$$

Zero-sequence impedance is computed the same way using pbsLineR0 and pbsLineX0.

Series Reactance X

Reactance is simply the imaginary part of the series impedance:

$$X = \text{pbsLineX1} \cdot \text{pbsLen}$$

This value determines the reactive voltage drop and reactive power flow along the line.

Shunt Admittance Y

The shunt admittance is derived from the per-kilometer capacitance and loss tangent:

$$Y = G + jB$$

Where:

- B = capacitive susceptance
- G = dielectric loss conductance

For a line of length pbsLen, the total shunt admittance is:

$$Y = (j\omega C + \omega C \cdot \tan \delta) \cdot \text{pbsLen}$$

Mapping to your parameters:

- $C = \text{pbsLineC1}$
- $\tan \delta = \text{pbsLineTan1}$

Thus:

$$Y = (j\omega \cdot \text{pbsLineC1} + \omega \cdot \text{pbsLineC1} \cdot \text{pbsLineTan1}) \cdot \text{pbsLen}$$

Where:

- $j\omega C \rightarrow$ susceptance
- $\omega C \tan \delta \rightarrow$ conductance (dielectric losses)

Zero-sequence admittance uses pbsLineC0 and pbsLineTan0.

Quantity	Formula	Based on Parameters
R	$\text{pbsLineR1} \cdot \text{pbsLen}$	Positive-sequence resistance
X	$\text{pbsLineX1} \cdot \text{pbsLen}$	Positive-sequence reactance
Z	$R + jX$	Series impedance
G	$\omega C \tan \delta \cdot \text{pbsLen}$	Dielectric loss conductance
B	$\omega C \cdot \text{pbsLen}$	Capacitive susceptance
Y	$G + jB$	Shunt admittance

In three-phase power-system analysis, transmission lines are represented using symmetrical components: positive-sequence, negative-sequence, and zero-sequence networks. For steady-state AC power-flow, only the positive-sequence network is used, because it represents the balanced operating condition of a healthy three-phase system. The positive-sequence resistance (R1), reactance (X1), and capacitance (C1) describe how voltage and current behave when all three phases carry equal sinusoidal currents displaced by 120°. In contrast, the zero-sequence network—defined by R0, X0, and C0—models unbalanced conditions such as ground faults, single-phase loads, or asymmetrical conductor arrangements. Zero-sequence parameters are essential for protection studies and fault analysis, but they do not participate in normal power-flow calculations. Therefore, pbsPowerGridLab uses positive-sequence parameters for all AC power-flow simulations, while zero-sequence parameters remain available for advanced studies and future extensions involving fault modeling or asymmetrical grid behavior.

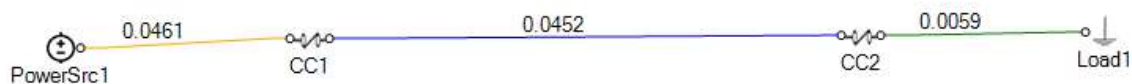
In the following table provides representative per-kilometer electrical parameters for a range of common underground cables and overhead transmission lines. These values illustrate how conductor type, insulation, voltage level, and installation method influence the electrical behavior of a line. Underground cables typically exhibit higher capacitance and lower reactance due to their close conductor spacing and dielectric insulation, while overhead lines show lower capacitance and higher reactance because of wider phase separation and air insulation. Zero-sequence parameters are generally larger than positive-sequence values, reflecting the higher impedance associated with ground-return paths. Nominal current ratings increase with conductor size and voltage level, highlighting the thermal limits of each line type. Although actual parameters vary by manufacturer and installation conditions, the ranges shown here provide a practical reference for configuring synthetic networks and validating power-flow simulations within pbsPowerGridLab.

Cable / Line Type	R1 (Ω/km)	X1 (Ω/km)	C1 (μF/km)	Tan1	R0 (Ω/km)	X0 (Ω/km)	C0 (μF/km)	Tan0	I _n (A)
-------------------	--------------	--------------	---------------	------	--------------	--------------	---------------	------	-----------------------

LV Cable 0.4 kV (Cu, 150 mm ²)	0.206	0.08	0.30	0.01	0.40	0.25	0.20	0.02	250
MV Cable 11 kV (XLPE, 185 mm ²)	0.12	0.09	0.25	0.01	0.30	0.27	0.20	0.02	300
MV Cable 33 kV (XLPE, 240 mm ²)	0.09	0.10	0.20	0.01	0.25	0.30	0.15	0.02	400
Overhead Line 33 kV (ACSR)	0.28	0.40	0.01	0.005	0.70	1.20	0.005	0.01	350
Overhead Line 110 kV (ACSR)	0.15	0.35	0.01	0.005	0.40	1.00	0.005	0.01	600
Transmission Line 220 kV (ACSR)	0.08	0.30	0.012	0.005	0.25	0.90	0.010	0.01	900
Transmission Line 400 kV (ACSR)	0.05	0.25	0.015	0.005	0.20	0.80	0.012	0.01	1200

The following diagram illustrates a basic three-segment power grid used to demonstrate AC power-flow calculation in pbsPowerGridLab. PowerSource1 supplies the network and is connected to node CC1 through a 1 km cable. From CC1, the grid continues to node CC2 via a 50 km cable, which represents the main transmission section of the model. Finally, CC2 is connected to Load1 through a 1 km line, completing the path from source to load. The load is configured as 200 kW and 80 kVAr, representing a typical commercial customer. A 10 kV base voltage is used for all elements in this example.

To run the power-flow calculation, open the PFC menu and select “Run AC Power Flow Calculation”. Figure4-3 .Python and the power-grid-model library must be installed for the PFC engine to operate correctly. The numerical values displayed along each line segment represent the line loading, calculated as the actual line current divided by the nominal current rating of that line. This provides a clear indication of how heavily each segment is utilized during the simulation.



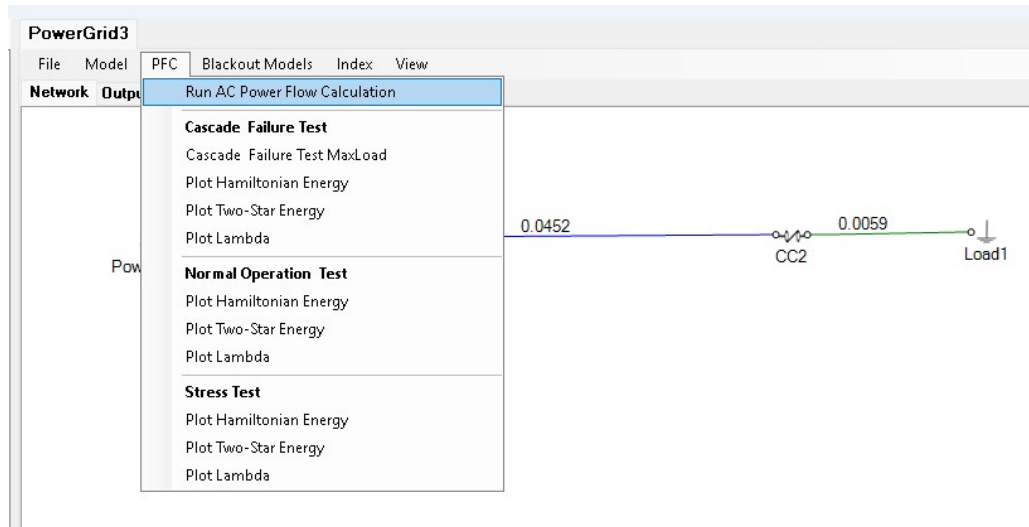


Figure 4-3

After running the AC power-flow calculation, the results can be viewed in the Output tab under the Power Flow page. This page displays a structured JSON file that contains the complete set of calculated values for the network, including node voltages, line currents, loading percentages, and power injections. The JSON shown here corresponds to the simple example network described above. It represents the final output generated by the PFC engine and provides a clear, machine-readable summary of all electrical quantities computed during the simulation. This JSON format is used internally by pbsPowerGridLab for visualization, analysis, and further processing of power-flow results.

```
{
  "version": "1.0",
  "type": "sym_output",
  "is_batch": false,
  "attributes": {},
  "data": {
    "node": [
      {
        "id": 0, "energized": 1, "u_pu": 1.0001530855835195, "u": 10001.530855835195, "u_angle": -4.5917446314367816e-05, "p": 304615.39468885225, "q": -1569198.7302909933},
      {
        "id": 29, "energized": 1, "u_pu": 1.0481058900296836, "u": 10481.058900296835, "u_angle": -0.12218824709517156, "p": -199999.99999999997, "q": -80000},
      {
        "id": 2, "energized": 1, "u_pu": 1.0025086163579731, "u": 10025.08616357973, "u_angle": -0.0045266358489482834, "p": 0, "q": 0},
      {
        "id": 25, "energized": 1, "u_pu": 1.0487031074994417, "u": 10487.031074994416, "u_angle": -0.12196672709717532, "p": 0, "q": 0}
    ],
    "line": [
      {
        "id": 5, "energized": 1, "loading": 0.046137412986133615, "p_from": 304615.39468885225, "q_from": -1569198.7302909936, "i_from": 92.274825972267223, "s_from": 1598491.5369898933, "p_to": -298037.28232542239, "q_to": 1542709.5774276941, "i_to": 90.488287947269328, "s_to": 1571234.8843960466},
      {
        "id": 6, "energized": 1, "loading": 0.0059, "p_from": 0, "q_from": 0, "i_from": 0, "s_from": 0, "p_to": 0, "q_to": 0, "i_to": 0, "s_to": 0}
    ]
  }
}
```

```

{"id": 28, "energized": 1, "loading": 0.045244143973634569, "p_from": 298037.28232539562, "q_from": -1542709.5774276964, "i_from": 90.488287947269143, "s_from": 1571234.8843960438, "p_to": -200445.45644714628, "q_to": -45549.237043169938, "i_to": 11.316605550725843, "s_to": 205555.622656058},

{"id": 31, "energized": 1, "loading": 0.0059328477792286206, "p_from": 200445.45644708865, "q_from": 45549.2370430827, "i_from": 11.316605550721688, "s_from": 205555.62265598247, "p_to": -199999.99999998085, "q_to": -80000.000000032436, "i_to": 11.865695558457242, "s_to": 215406.59228537444}

],

"source": [

{"id": 33, "energized": 1, "p": 304615.39468885225, "q": -1569198.7302909933, "i": 92.274825972267223, "s": 1598491.5369898931, "pf": 0.19056428366362899}

],

"sym_load": [

{"id": 32, "energized": 1, "p": 199999.99999999997, "q": 80000, "i": 11.865695558457556, "s": 215406.59228538015, "pf": 0.92847669088525919}

]

}

}

```

How pbsPowerGridLab Executes Power-Flow Calculations Using power-grid-model

When the user runs a power-flow calculation from the PFC menu, pbsPowerGridLab prepares a complete JSON description of the current network model, including all nodes, lines, sources, and loads. This JSON file contains every parameter required by the power-grid-model engine to solve the AC power-flow. The file is written to the RAMDISK for fast access. If the model name is BA100, pbsPowerGridLab generates a file named BA100_Output.json and places it in the RAMDISK. The power-grid-model engine then reads this file, processes the network, and attempts to solve the AC power-flow. If the calculation is successful, the engine produces a second file named BA100_Input.json, which contains the full set of computed results—node voltages, line currents, loading factors, power injections, and other electrical quantities. pbsPowerGridLab then loads this result file and displays the final power-flow solution in the Output Power Flow page. The JSON example shown below demonstrates the structure of a typical output file generated by power-grid-model for a simple three-segment network.

The JSON file generated for power-flow calculation follows a structured and predictable layout, where each section describes one category of power-grid elements. At the top level, the file contains metadata such as version, type, and attribute definitions. Under the data section, each element type—node, line, sym_load, source, transformer, and optional sensor objects—is represented as an array of lists. Each list contains the values for that element in the exact order defined in the attributes section. For example, a line entry includes its ID, from-node, to-node, status flags, and electrical parameters (R1, X1, C1, Tan1, and nominal current). Nodes include their ID and rated voltage, loads include their node and specified P/Q values, and sources include their reference voltage. This consistent structure allows power-grid-model to parse the network efficiently, reconstruct the full electrical model, and perform the AC power-flow calculation. The JSON format is intentionally

compact, machine-readable, and designed for fast exchange between pbsPowerGridLab and the solver.

When using pbsPowerGridLab, you do not need to understand the internal JSON structures or the detailed exchange mechanism between pbsPowerGridLab and the power-grid-model solver. These technical files are handled automatically in the background. As a user, your main focus is simply on the final PFC results, which pbsPowerGridLab presents through its visualization tools and analysis pages. Once the AC power-flow is solved, you can apply complex-systems concepts—such as network topology analysis, cascading behavior, stress evaluation, and synthetic grid studies—directly to the computed results without dealing with the underlying solver data formats. The system is designed so that you can concentrate entirely on interpreting the electrical behavior of the grid, not on the internal mechanics of the calculation.

```
{
  "version": "1.0",
  "type": "input",
  "is_batch": false,
  "attributes": {
    "node": ["id", "u_rated"],
    "line": ["id", "from_node", "to_node", "from_status", "to_status", "r1", "x1", "c1", "tan1", "i_n"],
    "sym_load": ["id", "node", "status", "type", "p_specified", "q_specified"],
    "source": ["id", "node", "status", "u_ref"],
    "transformer": ["id", "from_node", "to_node", "from_status", "to_status",
      "u1", "u2", "sn", "uk", "pk", "i0", "p0",
      "winding_from", "winding_to", "clock",
      "tap_side", "tap_pos", "tap_min", "tap_max", "tap_size"],
    "sym_voltage_sensor": ["id", "measured_object", "u_sigma", "u_measured"],
    "sym_power_sensor": ["id", "measured_object", "measured_terminal_type",
      "power_sigma", "p_measured", "q_measured"]
  },
  "data": {
    "node": [ [0, 10000.0], [29, 10000.0], [2, 10000.0], [25, 10000.0] ],
    "line": [ [5, 0, 2, 1, 1, 0.25, 0.20, 0.000001, 0.01, 2000.0], [28, 2, 25, 1, 1, 12.50, 10.00, 0.00005, 0.01, 2000.0], [31,
    25, 29, 1, 1, 0.25, 0.20, 0.000001, 0.01, 2000.0] ],
```

```

"sym_load": [
  [32, 29, 1, 0, 200000.0, 80000.0] ],
"source": [ [33, 0, 1, 1.0] ],
"transformer": [],
"sym_voltage_sensor": [],
"sym_power_sensor": []
}
}

```

In the following figure, two additional loads of 250 kW and 60 kVAr have been connected to node CC2, representing extra commercial consumers supplied from the same node. After running the AC power-flow calculation (PFC), the line loading values along each segment change accordingly, reflecting the increased power demand and higher currents flowing through the lines. The updated loading indicators shown on the lines in the figure illustrate how additional loads at CC2 redistribute stress across the network, allowing you to visually compare the utilization of each line before and after the new loads are added. Figure 4-4

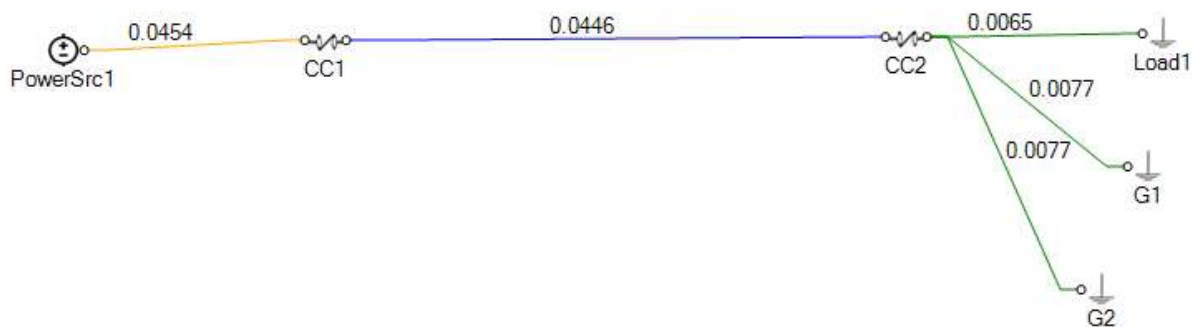


Figure 4-4

In figure 4-5 , we added a new power source to the network and run PFC .

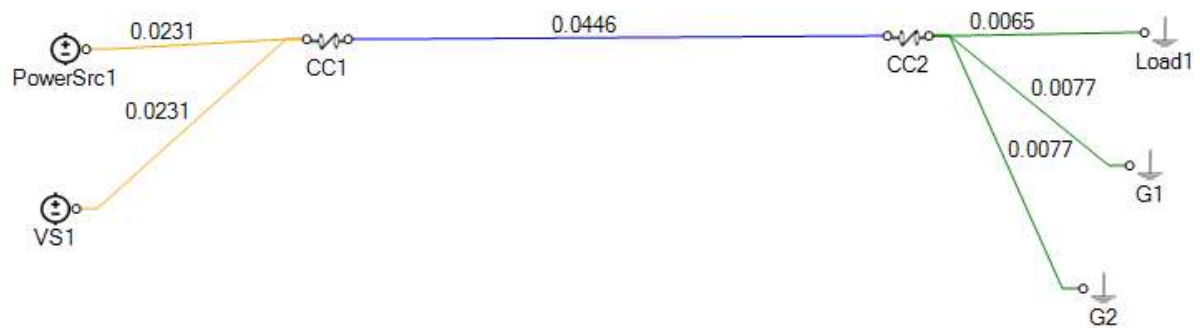
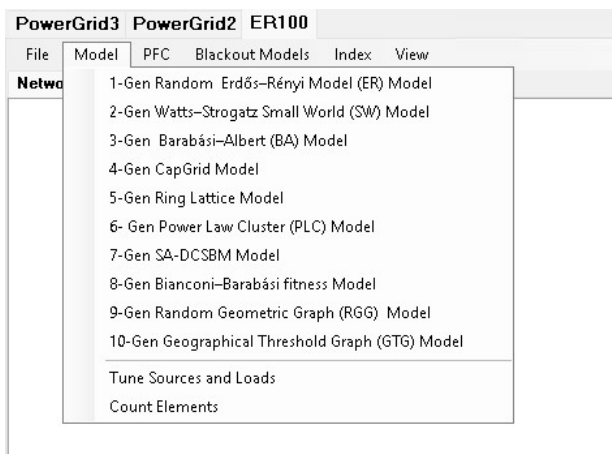


Figure 4-5

5-Complex System network Model for Power Grid

pbsPowerGridLab allows users to generate synthetic power-grid topologies using a wide range of complex-network models, each capturing different structural characteristics found in real or theoretical systems. The platform supports ten distinct models: the Erdős–Rényi (ER) random graph, the Watts–Strogatz Small-World (SW) model, the Barabási–Albert (BA) scale-free model, the CapGrid capacity-driven model, the Ring Lattice regular network, the Power-Law Cluster (PLC) model, the SA-DCSBM community-structured model, the Bianconi–Barabási fitness-based model, the Random Geometric Graph (RGG) model, and the Geographical Threshold Graph (GTG) model. Together, these models provide a rich toolbox for exploring how topology influences electrical behavior, stability, and emergent phenomena in power systems. Users can select any model, configure its parameters, and instantly generate a corresponding grid ready for AC power-flow analysis.



By opening the *Model* menu, users can choose from a wide range of network-generation models. For each model, the user specifies the network parameters—such as the number of nodes, the number of sources and loads, and the method used to attach these elements to the network. When generating a new power grid, pbsPowerGridLab first creates a pure topological network consisting only of Node elements. It then applies the selected model to generate the links between nodes according to the model’s rules (ER, BA, SW, CapGrid, etc.). After the network structure is formed, the system assigns sources and loads to nodes. Users can choose between two connection strategies: (1) randomly connecting sources and loads to any nodes, or (2) connecting sources to high-degree nodes and loads to low-degree nodes, which produces a more realistic grid structure where generation tends to cluster around central nodes and consumption spreads toward peripheral nodes. This flexible workflow allows users to build synthetic power grids that reflect both theoretical network models and practical power-system behavior.

In this section, we generate an ER network and run a power-flow calculation (PFC) for it. Although each network model has its own structural parameters, several configuration options are shared across all models in pbsPowerGridLab. Users must specify the number of sources and loads, the rated kV for the entire grid, and the active (p) and reactive (q) power ranges from which individual load values are randomly selected. The line length range is also defined, allowing the system to assign a random physical length to each line segment. Finally, users can enable automatic power-flow calculation and set the number of retries. After a network topology is generated, pbsPowerGridLab immediately attempts to run PFC; if the solver cannot converge, the system regenerates the network and retries until a valid configuration is found. This ensures that every generated grid—whether ER, BA, SW, CapGrid, or others—is not only structurally correct but also electrically solvable.

The methodology used in pbsPowerGridLab combines principles from complex-network theory with practical power-system modeling to create synthetic yet electrically meaningful power grids. Each grid begins as a purely topological structure: the system first generates a set of nodes and then applies the selected network model to create links between them. Once the topology is established, electrical attributes are added by assigning sources, loads, and line lengths according to user-defined ranges. Users can choose to connect sources and loads either randomly or based on node degree, placing generation at highly connected nodes and consumption at peripheral nodes—an approach that mirrors real-world grid behavior. After the grid is assembled, the platform automatically runs a power-flow calculation and, if necessary, regenerates the network until a solvable configuration is found. This methodology ensures that every generated network is both structurally valid and electrically functional, providing a robust foundation for exploring how different network models influence grid performance.

5-1 Random Network Model

In this section, we create a Random Network Model using 100 nodes and a connection probability of 0.05. In an Erdős–Rényi (ER) network, the topology is formed by first placing all nodes in the system without any links. The generator then examines every possible pair of nodes and produces a random number between 0 and 1; if this number is less than the selected probability p , a link is created between the two nodes. By repeating this process for all node pairs, the ER model produces a network with statistically uniform randomness, where each link exists independently with probability 0.05. This simple but powerful mechanism allows pbsPowerGridLab to generate large synthetic grids whose connectivity patterns follow classical random-graph behavior, providing a foundation for further electrical assignment and power-flow analysis.

In pbsPowerGridLab, the parameter $\frac{\log(N)}{N}$ is used as the critical probability for Erdős–Rényi (ER) networks. This value represents the connectivity threshold at which an ER network becomes fully connected with high probability. In the classical ER model $G(N, p)$, each possible link between two nodes is created independently with probability p . When p is below the critical value, the network may contain isolated nodes or small disconnected clusters. However, when

$$p_c = \frac{\log(N)}{N},$$

the likelihood of isolated nodes drops sharply, and the network transitions into a single connected component. This threshold is stronger than the giant-component threshold $1/N$, ensuring not just large clusters but complete connectivity across all nodes. Using $\frac{\log(N)}{N}$ as the critical probability helps pbsPowerGridLab generate ER networks that are structurally robust and far more likely to support successful power-flow calculations without disconnected or unreachable elements.

Random Network Configuration

Number of Nodes: 100

pc: 0.0461

p: 0.05

☐ Connect Source and Loads Randomly

Number of Sources: 25

Number of Loads: 40

Rated KV: 33

Rated p Min KW: 20

Rated p Max KW: 2000

Rated q Min KVar: 10

Rated q Max KVar: 500

Min Length (KM): 10

Max Length (KM): 100

☐ Run Power Flow Calculation

Number of PFC Run: 10

Generate Cancel

Figure 5-1

Select Run Power Flow Calculation and click the Generate button.(Figure 5-1) . pbsPowerGridLab will begin generating ER networks using 100 nodes and the selected probability value. For each attempt, the system creates the ER topology, connects sources and loads according to the chosen method, and immediately runs a power-flow calculation (PFC). If the PFC does not converge, the system automatically regenerates a new ER network and repeats the process until a valid, solvable configuration is found. Once PFC runs successfully, the generation process stops. In the resulting figure, you can see an example of a generated ER network using these parameters: sources are placed on the left side, loads on the right side, and the ER-generated nodes and links form the central structure of the grid.(figure 5-2)

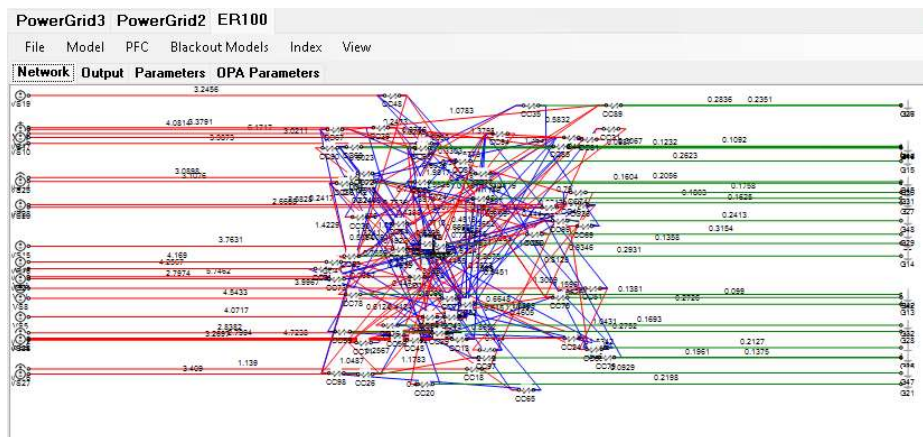


Figure 5-2

As shown in the figure 5-2, some lines appear in red, indicating that their loading exceeds 0.9, meaning these branches are heavily stressed under the initial power-flow solution. Before using this network for further analysis, we refine it by opening the Model menu and selecting Tune Sources and Loads. This tuning process adjusts the placement and power levels of sources and loads to achieve a more balanced operating condition. After tuning, save the model, reopen it, and run the power-flow calculation manually. You will observe that all line colors return to normal, indicating acceptable loading levels. In the final tuned network, sources are displayed in brown, loads in green, and all intermediate nodes remain connected through the ER-generated topology, forming a stable and electrically consistent grid ready for analysis. Figure 5-3

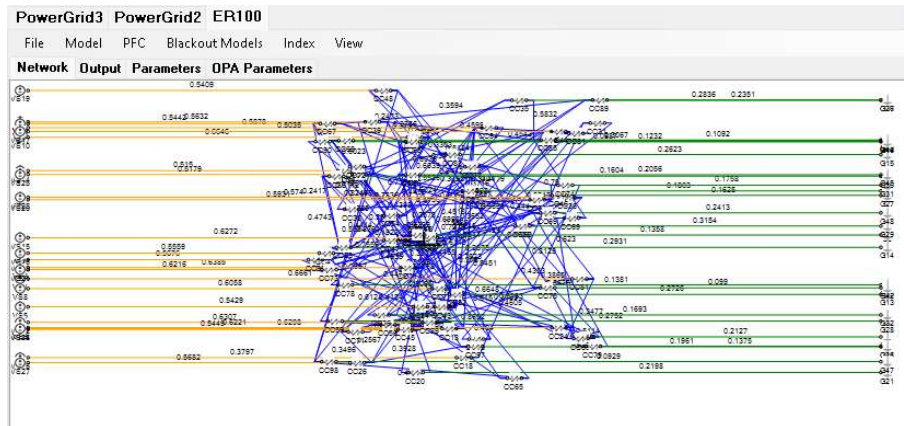


Figure 5-3

To confirm that the generated network behaves as an Erdős–Rényi (ER) random graph, users can analyze its Probability Degree Distribution (PDD). From the Index menu, select Cal PDD to calculate the degree distribution for the current network. This process computes the probability $P(k)$ of nodes having degree k and plots the results. Open the Output tab and navigate to the Degree Distribution page to view the generated PDD diagram. For an ER100 network, the resulting plot should resemble the example shown below—peaking near the average degree and following a Poisson-like curve. This confirms that the network’s connectivity pattern matches the statistical properties expected from an ER model, validating that the random generation process is functioning correctly. Figure 5-4

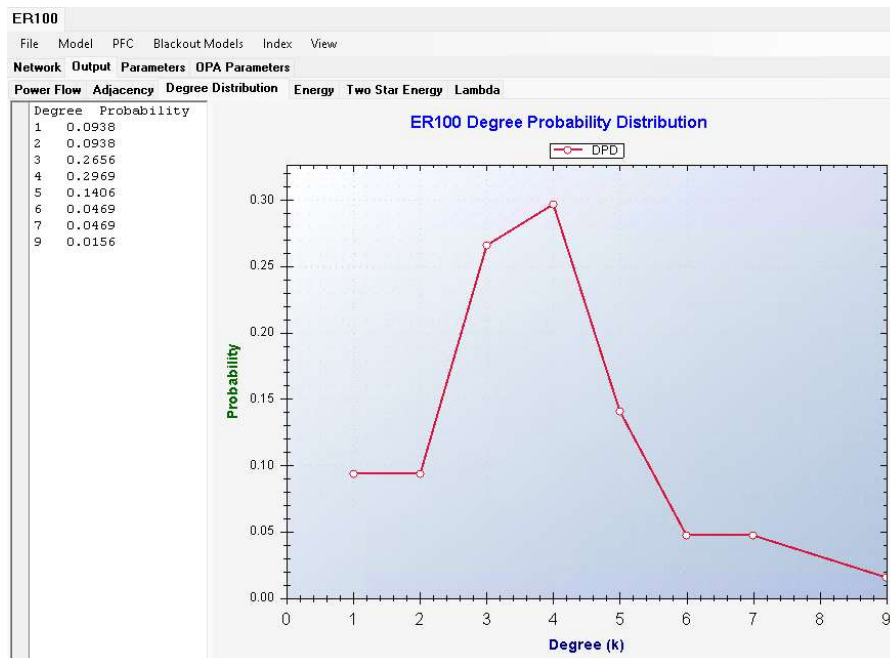


Figure 5-4

In an Erdős–Rényi network $G(N, p)$, every possible pair of nodes is connected independently with probability p . Because each node has $N - 1$ possible neighbors, the expected degree of a node is

$$\langle k \rangle = p(N - 1) \approx pN.$$

For an ER100 network with $p \approx 0.05$, the expected degree is

$$\langle k \rangle \approx 0.05 \times 100 = 5.$$

This means most nodes should have degrees clustered around 4–6, and the probability degree distribution (PDD) should form a Poisson-like curve peaking near this value. The PDD generated shows a clear peak around degree 4 and a smooth exponential decay for higher degrees, which matches the theoretical Poisson distribution of an ER network. This confirms that your ER100 network is behaving exactly as expected and that the PDD calculation in pbsPowerGridLab is correct.

pbsPowerGridLab can also display the Probability Degree Distribution (PDD) in log–log form, which is useful for visually confirming whether a network follows the statistical behavior of an Erdős–Rényi (ER) model. From the Index menu, select Cal PDD log-log to generate this diagram. For an ER100 network, the resulting plot should show a smooth curve that peaks near the expected degree and then drops rapidly as the degree increases. Because ER networks follow a Poisson-like degree distribution, the log–log plot does not form a straight line (as in scale-free networks) but instead bends downward, reflecting the exponential decay of higher-degree probabilities. The diagram you obtained matches this pattern exactly: a clear peak around degrees 3–4 and a steep decline for larger degrees. This confirms that your generated ER100 network is statistically consistent with the theoretical ER model and that the PDD calculation is working correctly. Figure 5-5

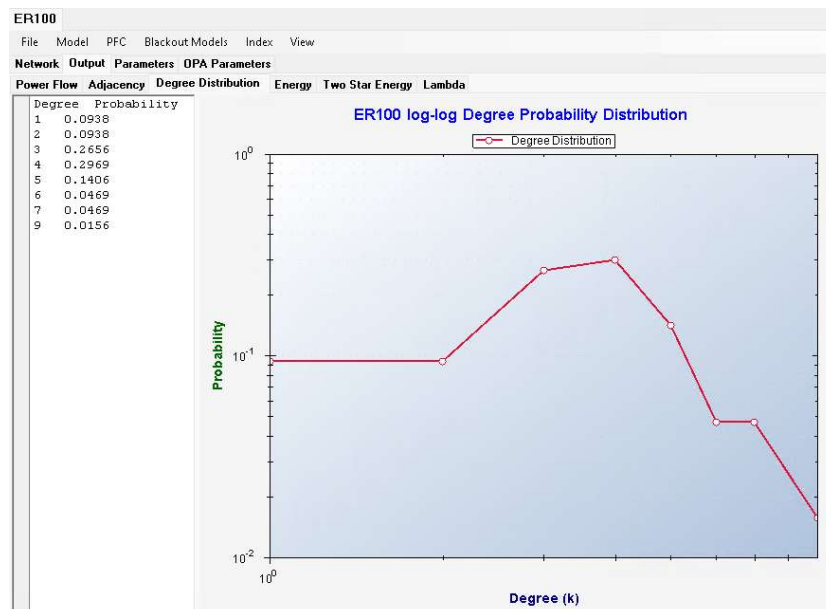


Figure 5-5

pbsPowerGridLab can also generate the Adjacency Matrix (shown as an adjacency list) for any created network, allowing users to inspect the exact pattern of connections between nodes. From the Index menu, select Cal Adjacency to compute this structure. For each node, the software lists all directly connected neighbors and the total number of links. In an ER100 network, this adjacency list should appear completely unstructured, with connections scattered randomly across nodes and no visible clustering or repeating patterns. The example shown for ER100 matches this behavior perfectly: each node has a small number of neighbors, typically around 4–6, consistent with the expected degree $\langle k \rangle \approx pN$. The irregular, pattern-free distribution of links confirms that the generated network follows the statistical properties of an Erdős–Rényi random graph and that the adjacency calculation is functioning correctly. Figure 5-6

ER100

File

Model

PFC

Blackout Models

Index

View

Network

Output

Parameters

OPA Parameters

Power Flow

Adjacency

Degree Distribution

Energy

Two Star Energy

Lambda

Node 12: (5) 14 32 55 58 93

Node 14: (4) 34 38 68 12

Node 17: (1) 30

Node 19: (2) 53 10

Node 23: (6) 26 28 68 78 82 6

Node 25: (4) 44 70 80 96

Node 26: (3) 47 72 23

Node 28: (4) 83 93 6 23

Node 30: (3) 70 6 17

Node 32: (4) 43 64 98 12

Node 33: (4) 34 37 43 96

Node 34: (3) 84 14 33

Node 35: (3) 37 52 72

Node 36: (5) 43 55 59 72 6

Node 37: (3) 76 33 35

Node 38: (4) 39 84 94 14

Node 39: (7) 46 47 58 74 95 10 38

Node 40: (5) 42 66 85 88 95

Node 42: (7) 56 69 70 93 94 95 40

Node 43: (5) 70 6 32 33 36

Node 44: (5) 45 50 66 94 25

Node 45: (2) 55 44

Node 46: (4) 66 67 93 39

Node 47: (5) 50 74 97 26 39

Node 50: (5) 62 83 93 44 47

Node 52: (3) 67 94 35

Node 53: (3) 89 93 19

Node 55: (4) 76 12 36 45

Node 56: (1) 42

Node 58: (2) 12 39

Node 59: (2) 71 36

Node 62: (3) 99 10 50

Figure 5-6

When a power-grid network is generated in pbsPowerGridLab—regardless of the selected model (ER, BA, SW, CapGrid, etc.)—the system automatically removes any nodes that have no connections. This ensures that the final network contains only electrically meaningful components and avoids isolated nodes that cannot participate in power-flow calculations. Because of this cleanup step, the actual number of nodes in the final network may be less than the initially requested value. To see the true size of the generated grid, open the Model menu and select Count Elements. This function counts and displays the real number of nodes, links, sources, and loads in the network after all disconnected nodes have been removed. This provides an accurate summary of the final grid structure and helps verify that the generated model is ready for further analysis.

5-2 Small-World Network Model

The Small-World (SW) network model combines the regular structure of a ring lattice with the randomness of rewired links, producing networks that have high clustering and short average path lengths—two key characteristics of real-world power grids. In pbsPowerGridLab, the SW model is defined by two parameters: m and p . The parameter m specifies how many nearest neighbors each node is initially connected to on the ring (typically two on each side, giving degree $2m$). After building this regular lattice, each link is then rewired with probability p , where $p \in [0, 1]$. When $p = 0$, the network remains a pure lattice with long path lengths; when $p = 1$, the network becomes fully random like an ER model. For intermediate values of p , the network exhibits the classic small-world behavior: strong local clustering from the lattice structure and short global distances created by the rewired shortcuts. This makes the SW model useful for generating synthetic grids that preserve locality while still allowing efficient long-range connectivity.

In figure 5-7 you can see parameters for Small-World network generation :

Small World Parameters

Number of Nodes	100
m	3
p	0.1
☐ Connect Source and Loads Randomly	
Number of Sources	25
Number of Loads	40
Rated KV	33
Rated p Min KW	20
Rated p Max KW	2000
Rated q Min KVar	10
Rated q Max KVar	500
Min Length (KM)	10
Max Length (KM)	100
☒ Run Power Flow Calculation	
Number of PFC Run	10
Generate Cancel	

Figure 5-7

With the selected parameters, the generated network forms a Small-World power grid, shown in the figure 5-8

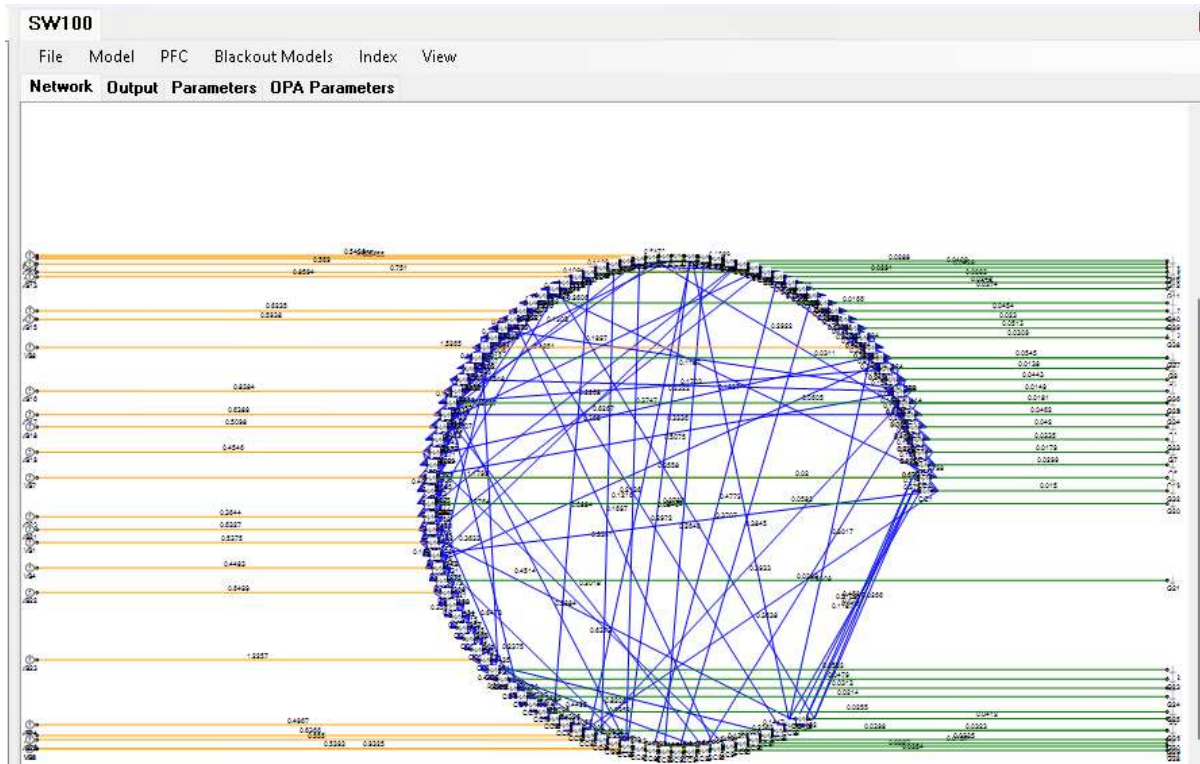


Figure 5-8

The Probability Degree Distribution (PDD) generated for the SW100 network is fully consistent with the expected behavior of a Small-World model. Because each node begins with a fixed number of local connections determined by the parameter m , most nodes retain degrees close to this initial value. The rewiring probability p introduces a small number of additional long-range links, creating slight variations in degree but keeping the overall distribution narrow. The PDD shown here reflects exactly this pattern: a strong peak around degrees 6 and 7, followed by a rapid drop for higher degrees. This confirms that the generated network maintains the characteristic structure of a Small-World topology and that the PDD calculation is functioning correctly. Figure 5-9

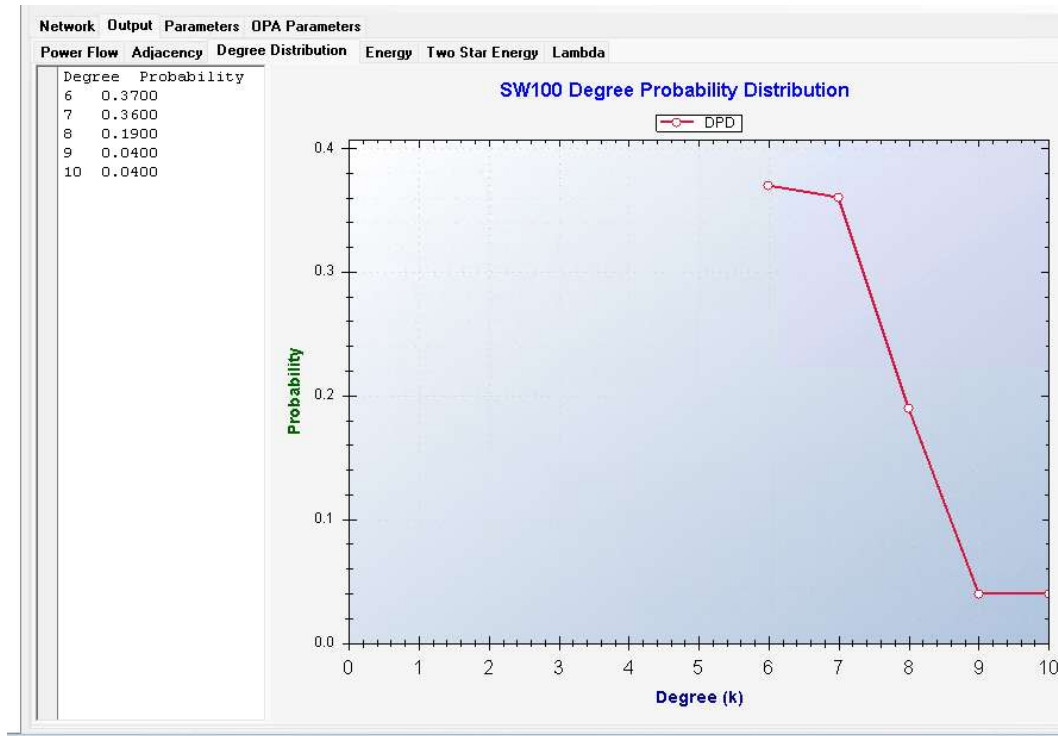


Figure 5-9

5-3 Barabási–Albert (BA) Network Model

The Barabási–Albert (BA) model generates a scale-free power-grid topology where a small number of nodes naturally become highly connected hubs. As the network grows, each new node attaches to existing nodes with a probability proportional to their current degree, creating a structure dominated by preferential attachment. This produces a characteristic degree distribution with many low-degree nodes and a few high-degree hubs, forming a long-tailed pattern that is fundamentally different from ER or SW networks. BA networks are useful for studying robustness and vulnerability because their hub-based structure strongly influences how failures propagate through the grid. In figure 5-10 you can see parameters for defining a power grid with BA network model .

Barabási-Albert Network Parameters

Number of Nodes	100
m	3
☐ Connect Source and Loads Randomly	
Number of Sources	25
Number of Loads	40
Rated KV	33
Load Rated p Min KW	20
Load Rated p Max KW	2000
Load Rated q Min KVar	10
Load Rated q Max KVar	500
Min Length (KM)	10
Max Length (KM)	100
☒ Run Power Flow Calculation	
Number of PFC Run	10

Generate Cancel

Figure 5-10

In figure 5-11 you can see a generated BA network with 100 Nodes and m = 3 .



Figure 5-11

PDD for above network is as following figure 5-12:

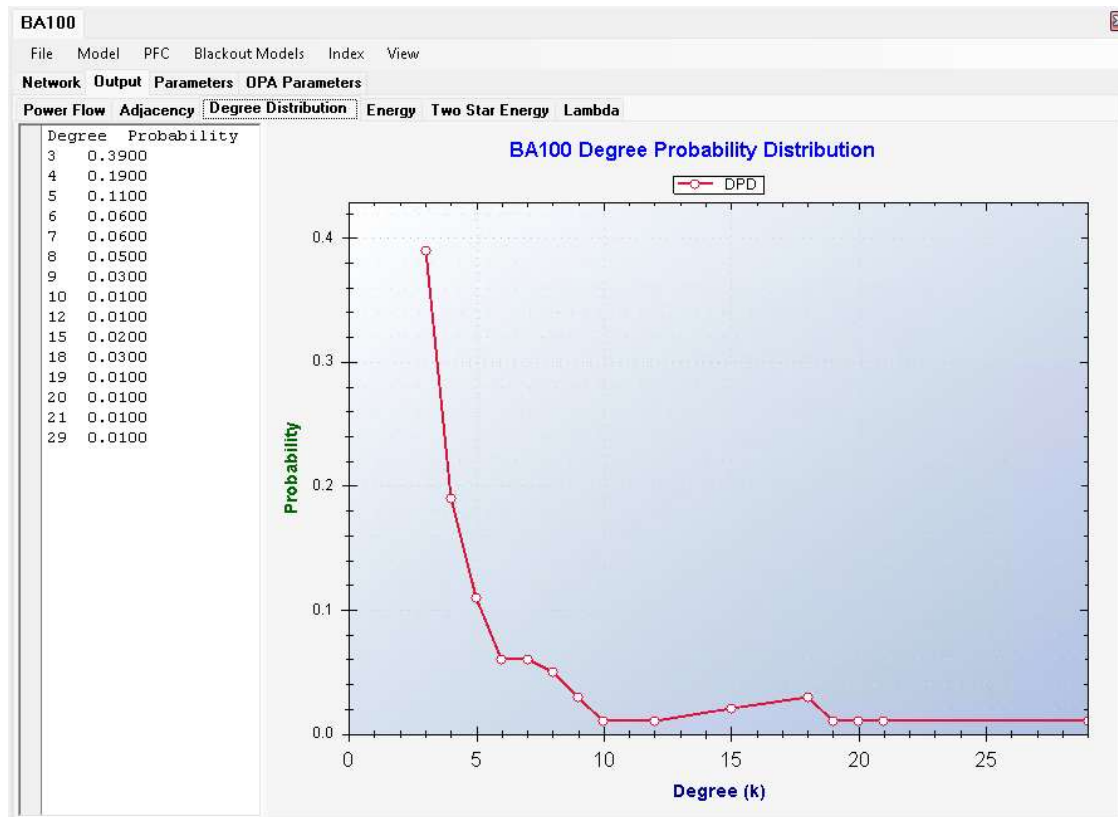


Figure 5-12

The Probability Degree Distribution (PDD) generated for the BA100 network shows the characteristic long-tailed pattern of a scale-free topology. In the BA model, new nodes preferentially attach to existing nodes with higher degrees, causing a few nodes to become highly connected hubs while most nodes remain lightly connected. The PDD displayed here reflects this behavior precisely: a high concentration of nodes with small degrees (3–5), followed by a gradual decline and the presence of several nodes with significantly larger degrees (up to 29). This heavy-tailed distribution confirms that the generated BA100 network follows the expected preferential-attachment dynamics and that the PDD calculation in pbsPowerGridLab is correct.

5-4 SA-DCSBM (Spatially Aware Degree-Corrected Stochastic Block Model)

The Spatially Aware Degree-Corrected Stochastic Block Model (SA-DCSBM) is a powerful framework for generating synthetic power grids that closely resemble real-world transmission networks. It extends the classic Stochastic Block Model (SBM) by incorporating:

- Community structure (modularity)
- Degree heterogeneity (realistic node connectivity)
- Spatial constraints (distance-aware edge formation)

This model is particularly suited for simulating regional grids, interconnected utility zones, and multi-voltage-layer systems, where both structural and geographic realism are essential.

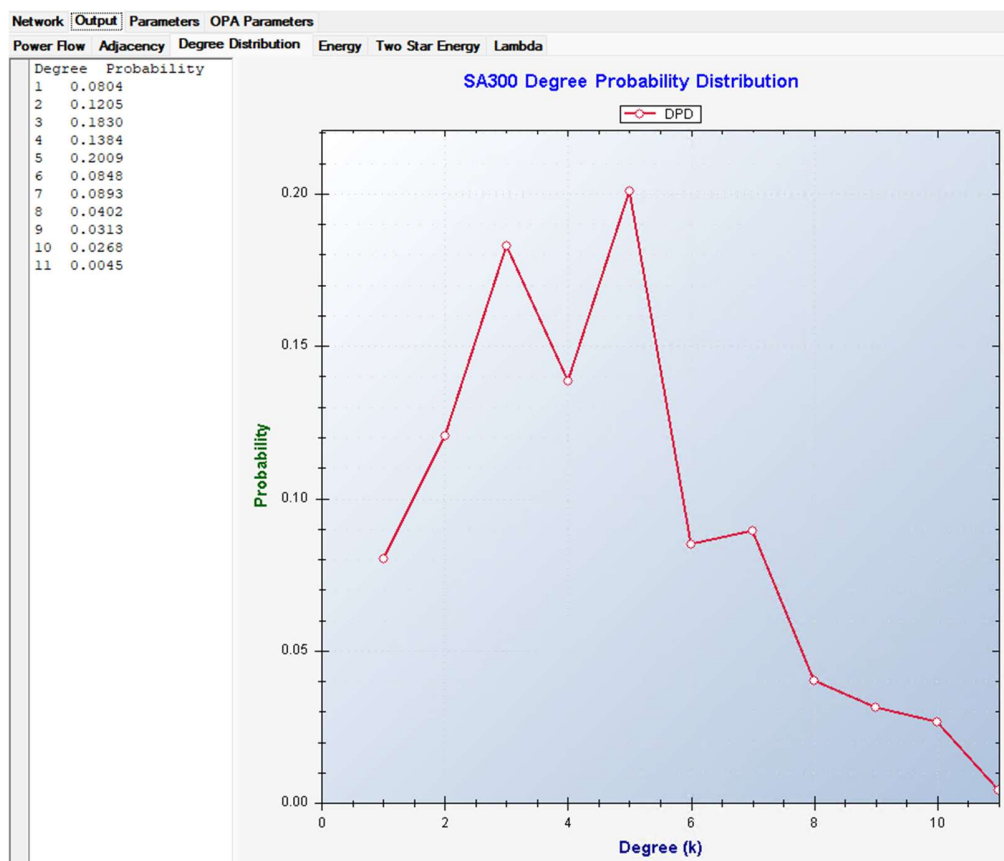


Figure 5-13 Degree Probability Distribution for a sample SA-DCSMB Network with 500 Nodes

To generate a SA-DCSBM grid, open the Model menu and select SA-DCSBM. The configuration dialog is shown in Figure 5-14.

Figure 5-14 allows users to generate synthetic power grid topologies using the Spatially-Aware Degree-Corrected Stochastic Block Model (SA-DCSBM). The model simulates realistic network growth

by combining modular community structure, geographic constraints, and degree heterogeneity. Users can customize the topology, electrical parameters, and simulation settings to benchmark diagnostics or run power flow calculations.

Topology Parameters

- **Number of Nodes:** Total number of buses (nodes) in the network.
- **Number of Clusters:** Number of spatially distributed communities or zones.
- **Spatial Decay:** Controls how edge probability decreases with distance between nodes.
- **P Intra Community:** Base probability of connection between nodes within the same cluster.
- **P Inter Community:** Base probability of connection between nodes in different clusters.

Degree Distribution Settings

- **DegreeDistributionType:** Selects how node connectivity (θ_i) is sampled. Options include:
 - **Uniform:** Evenly distributed degrees within a range.
 - **TruncatedNormal:** Bell-shaped distribution with bounded extremes.
 - **PowerLaw:** Heavy-tailed distribution for hub-dominated networks.
 - **LogNormal:** Skewed distribution for moderate heterogeneity.
- **thetaMin / thetaMax:** Sets the minimum and maximum values for node degree parameters.
- **powerLawExponent:** Controls the steepness of the power-law distribution (recommended range: 2.2–3.2).
- **normalizeTheta:** Ensures degree parameters are scaled to maintain consistent edge density.

Source and Load Placement

- **Connect Source and Loads Randomly:** If checked, sources and loads are assigned randomly across the network.
- **Number of Sources / Loads:** Specifies how many generation and consumption nodes to place.
- **Rated KV:** Voltage level for all buses.
- **Load Rated p Min/Max KW:** Range of real power demand for loads.
- **Load Rated q Min/Max KVar:** Range of reactive power demand for loads.
- **Min/Max Length (KM):** Range of line lengths for edge assignment.

Simulation Settings

- **Run Power Flow Calculation:** Enables power flow simulation after topology generation.
- **Number of PFC Run:** Number of iterations to test convergence and stability.

Actions

- **Generate:** Builds the network and optionally runs power flow.
- **Cancel:** Clears the configuration or exits the setup.

SA-DCSMB network Model Configuration

Number of Nodes	300	DegreeDistributionType	Uniform
Number of Clusters	6	thetaMin	0.5
Spatial decay	0.1	thetaMax	1.5
P Intra Community	0.1	powerLawExponent	2.5
P Inter Community	0.01	☐ normalize Theta	

$$\mathbf{B} = \begin{pmatrix} \rho_{intra} & \rho_{inter} \\ \rho_{inter} & \rho_{intra} \end{pmatrix}.$$

☐ Connect Source and Loads Randomly

Number of Sources	50	Load Rated q Min KVar	10
Number of Loads	150	Load Rated q Max KVar	500
Rated KV	15	Min Length (KM)	30
Load Rated p Min KW	20	Max Length (KM)	100
Load Rated p Max KW	2000		

☐ Run Power Flow Calculation

Number of PFC Run

- **Figure 5-14** network model parameters for SA-DCSBM model

The Spatially-Aware Degree-Corrected Stochastic Block Model (SA-DCSBM) is designed to replicate the structural and spatial features of real-world transmission grids. Based on published studies and synthetic grid benchmarks, here are the recommended parameter ranges for realistic and robust grid generation:

Parameter	Typical Range	Purpose
Number of Nodes	200–500	Defines grid size; larger networks improve spectral benchmarking
Number of Clusters	3–8	Models regional zones or utility areas
Intra-Community p_{intra}	0.05–0.2	Controls density of connections within clusters
Inter-Community p_{inter}	0.005–0.02	Controls connectivity between clusters
Spatial Decay	0.05–0.2	Determines how edge probability decreases with distance

The Spatially-Aware Degree-Corrected Stochastic Block Model (SA-DCSBM) extends the classic SBM by incorporating degree heterogeneity and spatial constraints. The probability of forming an edge between two nodes i and j is given by:

$$P_{ij} = \theta_i \theta_j B_{g_i g_j} f(d_{ij})$$

Where:

- θ_i, θ_j : Degree parameters for nodes i and j , controlling expected connectivity. θ_i, θ_j are random number that generated by DegreeDistributionType
- g_i, g_j : Community assignments of nodes i and j
- $B_{g_i g_j}$: Block matrix entry defining base connection probability between communities g_i and g_j
- $f(d_{ij})$: Spatial decay function based on distance d_{ij} between nodes. A common choice is exponential decay:

$$f(d_{ij}) = e^{-\alpha d_{ij}}$$

Where:

- d_{ij} : Euclidean distance between nodes i and j

- α : Spatial decay coefficient (e.g., 0.05–0.2)

This ensures that closer nodes are more likely to connect, reflecting geographic constraints in real power grids.

Block Matrix B

For a two-community system, the block matrix might look like:

$$B = \begin{pmatrix} p_{\text{intra}} & p_{\text{inter}} \\ p_{\text{inter}} & p_{\text{intra}} \end{pmatrix}$$

- p_{intra} : Probability of intra-community connections
- p_{inter} : Probability of inter-community connections

This modular structure allows pbsHMI to simulate regional zones with dense internal links and sparse external ties.

In the SA-DCSBM model, each node i is assigned a degree parameter θ_i , which represents its expected relative connectivity within the network. These parameters are typically sampled from a controlled distribution—such as uniform, truncated normal, or power-law—to reflect the natural heterogeneity found in real power grids. Nodes with higher θ_i values are more likely to form connections and act as hubs (e.g., substations), while nodes with lower values tend to be peripheral (e.g., feeders or local loads). When calculating the probability of an edge between two nodes i and j , the product $\theta_i \theta_j$ scales the base community-level probability $B_{g_i g_j}$, modulated further by a spatial decay function $f(d_{ij})$.

This mechanism allows pbsHMI to generate modular, spatially realistic networks with variable node degrees, enabling more accurate simulation of hierarchical and regionally clustered grid topologies.

$$P_{ij} = \theta_i \theta_j B_{g_i g_j} e^{-\alpha d_{ij}}$$

if $P_{ij} > \text{rnd}(0,1)$ link between i, j is establish

You can see a sample generated a power grid with SA-DCSBM network model in figure 5-15.

In Figure 5-16 , PDD for a Sample SA-DCSBM with 500 Nodes is shown .

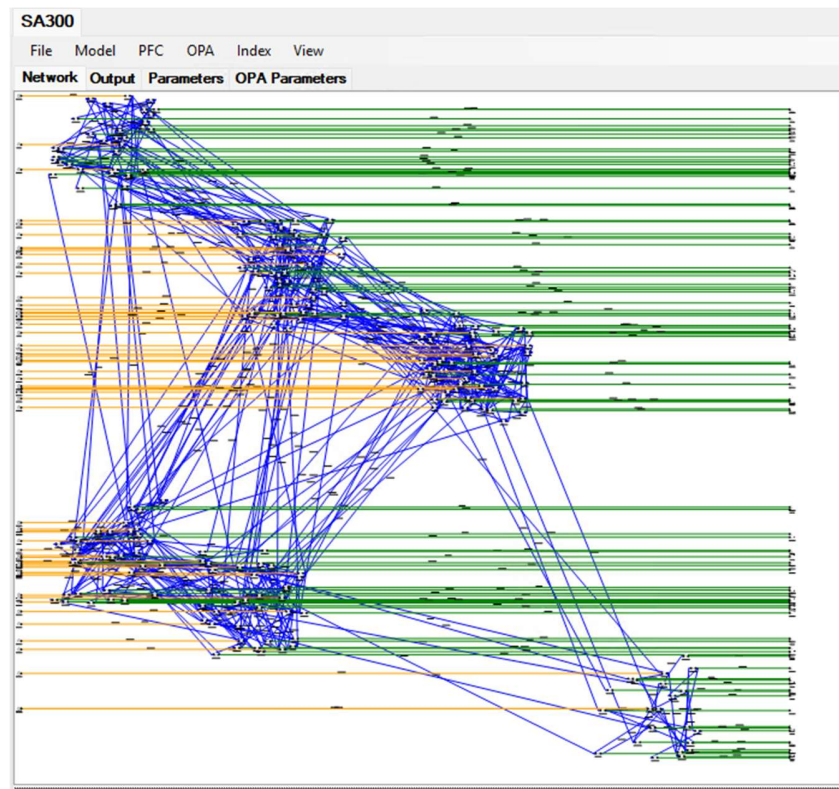


Figure 5-15

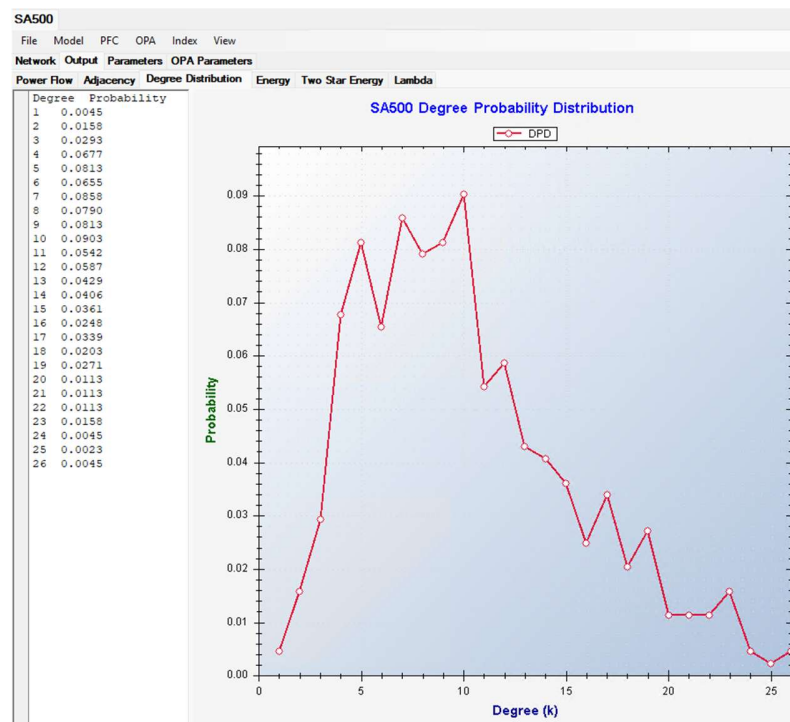


Figure 5-16

5-5 Power Law Clustering (PLC)

The Power Law Clustering (PLC) model extends scale-free networks by introducing triadic closure. After preferential attachment, additional edges are added between neighbours of a node to increase clustering. The generative rule combines:

- Preferential attachment: $P(i) \propto k_i$
- Triadic closure: connect neighbours of newly added node with probability p_c . PLC networks retain hub dominance but improve local coherence, making them suitable for modal diagnostics that reward clustering.

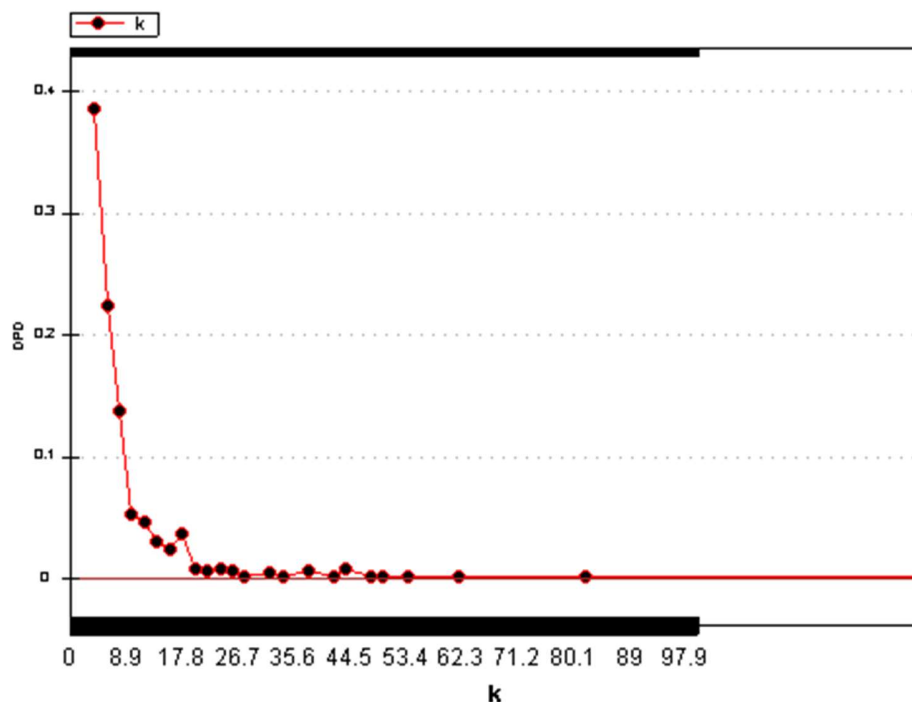


Figure 5-17 Degree Probability Distribution for a sample PLC Network with 500 Nodes

Power-law clustering refers to network models that combine heavy-tailed degree distributions with localized triangle formation, producing realistic topologies with hubs and modular structure.

What Is Power-Law Clustering?

Power-law clustering arises when a network exhibits both:

- Power-law degree distribution: A few nodes (hubs) have very high degree, while most have low degree. This follows $P(k) \sim k^{-\gamma}$, typically with $\gamma \in [2.2, 3.0]$.
- High local clustering: Nodes tend to form triangles, meaning neighbours of a node are also connected to each other.

This combination is seen in many real-world systems, including social networks, and biological systems.

How It's Modelled

One popular model is the Holme–Kim algorithm, which extends the Barabási–Albert (BA) preferential attachment model by adding triangle formation:

- Each new node connects to *m* existing nodes preferentially (like BA).
- With probability p , it also connects to a neighbor of one of those nodes, forming a triangle.

This creates a scale-free network with tuneable clustering.

Another approach is the spatial preferential attachment model, where:

- Nodes are placed in space.
- Connection probability Favors short distances and high degrees.
- This leads to power-law degree distribution and positive clustering coefficient, with a phase transition around $\gamma = 3$

PLC Network Model Configuration

Figure 5-18 allows users to generate synthetic networks using the Power-Law Clustering (PLC) model, which combines preferential attachment with triangle formation to simulate realistic topologies. The PLC model is ideal for testing diagnostics, simulating power flow, and benchmarking spectral metrics in hub-dominated but locally clustered networks.



Topology Parameters

- **Number of Nodes:** Total number of buses (nodes) in the network.
- **m:** Number of edges each new node creates upon joining the network.
- **p:** Triangle formation probability. With probability p , a new edge connects to a neighbor of an existing node, forming a triangle. This controls the clustering coefficient.

Source and Load Placement

- **Connect Source and Loads Randomly:** If checked, sources and loads are assigned randomly across the network.
- **Number of Sources / Loads:** Specifies how many generation and consumption nodes to place.
- **Rated KV:** Voltage level for all buses.
- **Rated p Min/Max KW:** Range of real power demand for loads.

- **Rated q Min/Max KVar:** Range of reactive power demand for loads.
- **Min/Max Length (KM):** Range of line lengths for edge assignment.

Simulation Settings

- **Run Power Flow Calculation:** Enables power flow simulation after topology generation.
- **Number of PFC Run:** Number of iterations to test convergence and stability.

Actions

- **Generate:** Builds the network and optionally runs power flow.
- **Cancel:** Clears the configuration or exits the setup.

PLC network Model Configuration

Number of Nodes	100
m	4
p	0.1
☐ Connect Source and Loads Randomly	
Number of Sources	25
Number of Loads	40
Rated KV	15
Rated p Min KW	20
Rated p Max KW	2000
Rated q Min KVar	10
Rated q Max KVar	500
Min Length (KM)	10
Max Length (KM)	100
☐ Run Power Flow Calculation	
Number of PFC Run	10
Generate Cancel	

- Figure 5-18 network model parameters for PLC model

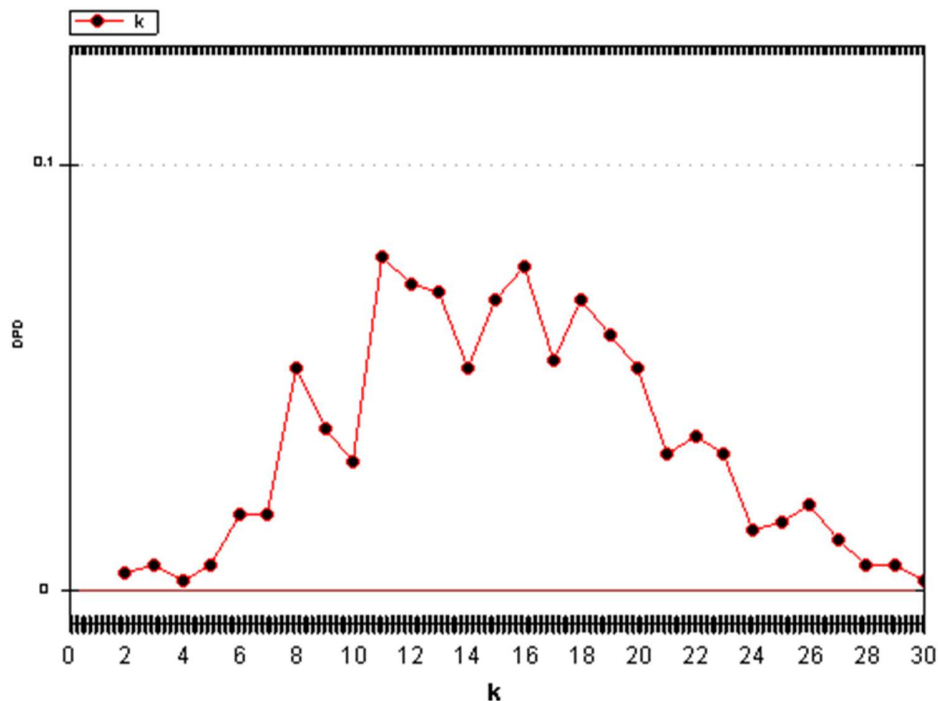
5-6 Random Geometric Graphs (RGG)

The Random Geometric Graph (RGG) model places n nodes uniformly in a metric space (typically 2D) and connects pairs whose Euclidean distance is less than a threshold r . Formally:

$$RGG A_{ij} = 1 \quad \text{if } \|x_i - x_j\| < r$$

$$Ring RGG A_{ij} = 1 \quad \text{if } r < \|x_i - x_j\| < r_2$$

RGGs exhibit spatial locality, high clustering, and smooth modal transitions. They are robust under random failures and serve as a benchmark for spatially embedded grid diagnostics.



5-19 Degree Probability Distribution for a sample RGG Network with 500 Nodes

Random Geometric Graphs (RGG) are spatial network models where nodes are placed randomly in a geometric space (typically 2D), and edges are formed between nodes that lie within a specified distance of each other. This simple yet powerful model captures the essence of spatial locality, making it highly relevant for simulating physical infrastructure like power grids, sensor networks, and communication systems.

RGGs are ideal for modelling:

- **Distribution networks with localized connections**
- **Microgrids and systems where physical proximity governs connectivity**
- **Spatial clustering and modularity without requiring community labels**

Unlike scale-free models (e.g., BA or PLC), RGGs emphasize distance-based constraints, which reflect real-world limitations in cable length, signal propagation, and installation cost.

Core Principles

- **Node Placement:** Nodes are uniformly distributed in a bounded 2D space (e.g., 100×100 km).
- **Connection Radius r :** Two nodes are connected if their Euclidean distance is less than or equal to r .
- **Edge Formation:** Deterministic based on distance; no probabilistic edge sampling.
- **Clustering:** Naturally high due to spatial proximity; triangle formation is common.

RGG Network Model Configuration

In Figure 5-20 users can generate synthetic networks using the Random Geometric Graph (RGG) model. RGGs simulate spatially constrained topologies by connecting nodes based on physical proximity, making them ideal for modelling distribution grids, microgrids, and sensor networks.

Topology Parameters

- **Number of Nodes:** Total number of buses (nodes) in the network.
- **Connection Radius (KM):** Maximum distance between two nodes for an edge to be formed. Nodes within this radius are connected, creating a spatially localized network.

Source and Load Placement

- **Connect Source and Loads Randomly:** If checked, sources and loads are assigned randomly across the network.
- **Number of Sources / Loads:** Specifies how many generation and consumption nodes to place.
- **Rated KV:** Voltage level for all buses.
- **Rated Min/Max KW:** Range of real power demand for loads.
- **Rated Min/Max KVar:** Range of reactive power demand for loads.

- **Min/Max Length (KM):** Range of line lengths for edge assignment.

Simulation Settings

- **Run Power Flow Calculation:** Enables power flow simulation after topology generation.
- **Number of PFC Run:** Number of iterations to test convergence and stability.

Actions

- **Generate:** Builds the network and optionally runs power flow.
- **Cancel:** Clears the configuration or exits the setup.

RGG Network Model Configuration

Number of Nodes

ConnectionRadius (r)

ConnectionRadius (r2) ☐ Ring RGG

$$RGG A_{ij} = 1 \quad \text{if } \|x_i - x_j\| < r$$

$$Ring RGG A_{ij} = 1 \quad \text{if } r < \|x_i - x_j\| < r2$$

☐ Connect Source and Loads Randomly

Number of Sources

Number of Loads

Rated KV

Rated Min KW

Rated Max KW

Rated Min KVar

Rated Max KVar

☐ Run Power Flow Calculation

Number of PFC Run

- Figure 5-20 network model parameters for RGG model

5- 7 Geographical Threshold Graphs (GTG)

The Geographical Threshold Graph (GTG) model extends RGG by assigning each node a weight w_i , and connecting nodes i, j if:

$$\frac{w_i + w_j}{d_{ij}^\beta} \geq \theta$$

- w_i, w_j : node weights (uniform)
- d_{ij} : Euclidean distance
- β : spatial decay
- θ : threshold

This introduces heterogeneity and allows for hub formation based on both spatial proximity and node importance. GTGs are ideal for modelling infrastructure networks where geography and capacity jointly influence connectivity. θ is the threshold that must be exceeded for an edge to exist

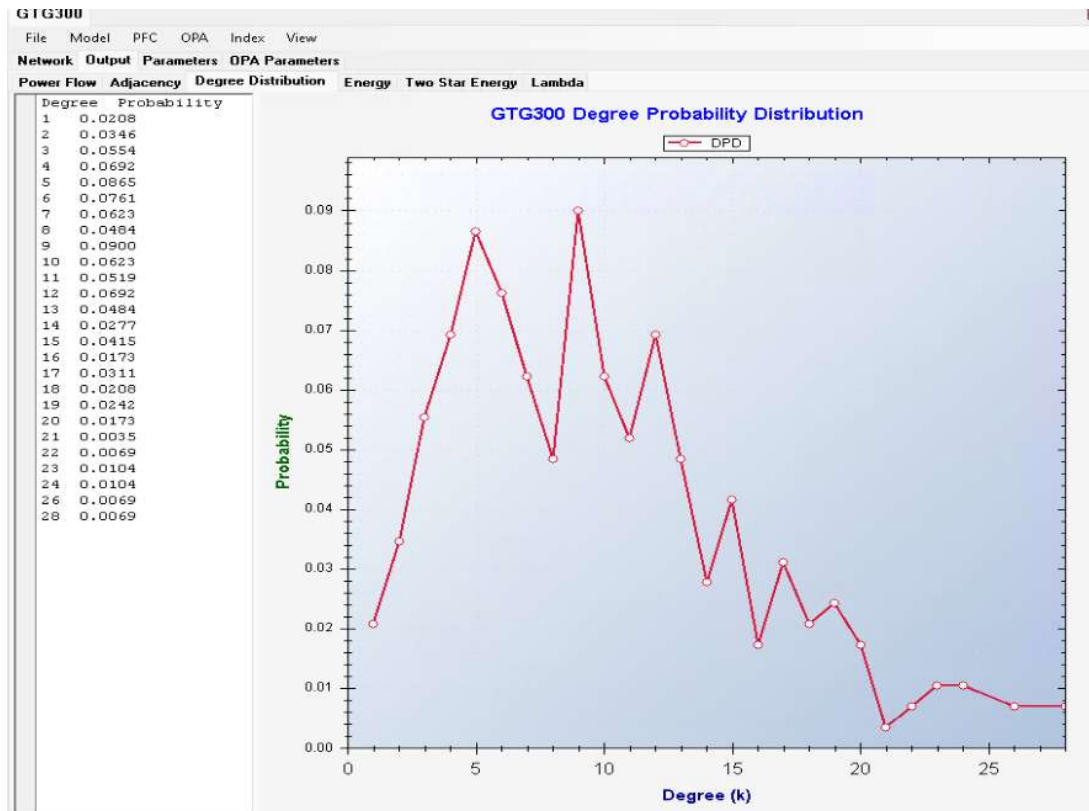
GTGs are spatial network models where nodes are placed in a geometric space and assigned individual weights. An edge is formed between two nodes if a function of their weight and distance exceeds a predefined threshold. This model generalizes Random Geometric Graphs (RGG) by introducing node-level heterogeneity, making it suitable for simulating networks where both proximity and node influence matter. In figure 5-21 you can see a typical PDD for a GTG network with 300 Nodes .

Core Principles

- **Node Placement:** Nodes are randomly distributed in a 2D Euclidean space.
- **Node Weights:** Each node receives a weight w_i , representing influence, capacity, or transmission power.
- **Edge Formation Rule:** Nodes i and j are connected if:

$$f(w_i, w_j, d_{ij}) \geq \theta$$

where d_{ij} is the Euclidean distance and θ is the threshold parameter.



5-21 Degree Probability Distribution for a sample GTG Network with 300 Nodes

A common form is:

$$\frac{w_i + w_j}{d_{ij}^\beta} \geq \theta$$

where β controls spatial decay.

beta (spatial decay): Typical range 1–3; larger means stronger distance penalty and more local edges.

Typical Ranges for θ

- Weights sampled uniformly in [0,1]:
 - If $(w_i + w_j)/d_{ij}^\beta \geq \theta$, then θ usually falls in the range 0.01 – 1.0.
 - Smaller $\theta \rightarrow$ denser network (more edges).
 - Larger $\theta \rightarrow$ sparser network (fewer edges).
- Effect of β (spatial decay exponent):
 - With $\beta \approx 1-2$, θ around 0.1–0.5 often yields a connected but not overly dense graph.

- With higher β (≥ 3), you'll need smaller θ (≈ 0.01 – 0.1) to avoid disconnecting the network.
- Scaling with node count:
 - For 100–500 nodes, $\theta \approx 0.1$ – 0.3 is a good starting point.
 - For larger networks, θ may need to be reduced to maintain connectivity.

Rule of Thumb

- $\theta \approx 0.1$ – 0.3 with $\beta = 2 \rightarrow$ balanced connectivity for medium-sized networks.
- $\theta < 0.1 \rightarrow$ dense, almost RGG-like.
- $\theta > 0.5 \rightarrow$ sparse, hub-dominated.

GTG parameter choices for a 500×500 coordinate space

When node coordinates are uniform in $[0, 500] \times [0, 500]$, Euclidean distances d_{ij} range from 0 up to $\sqrt{500^2 + 500^2} \approx 707$. Because the GTG rule uses $(w_i + w_j)/d_{ij}^\beta \geq \theta$ with $w_i \in [0, 1]$, typical θ depends strongly on β and this distance scale.

Recommended starting points

- Beta ≈ 1 :
 - Typical edge scores near: $(w_i + w_j)/d_{ij} \sim \mathcal{O}(10^{-3} \text{ to } 10^{-1})$ depending on distance.
 - Theta: start at 0.02–0.08.
 - Effect: moderate spatial decay; connectivity spans medium distances.
- Beta ≈ 2 :
 - Score decays faster: d_{ij}^{-2} on the order of 10^{-6} at long range, 10^{-4} – 10^{-3} mid range.
 - Theta: start at 0.002–0.02.
 - Effect: strongly local edges; good for distribution-like locality.
- Beta ≈ 3 :
 - Very steep decay; long-range edges are rare.
 - Theta: start at 5×10^{-4} – 5×10^{-3} .

- Effect: highly clustered, very local networks.

In Figure 5-22 users can generate synthetic networks using the Geographical Threshold Graph (GTG) model, which combines spatial locality with node-level heterogeneity. GTGs are ideal for simulating power systems where both physical proximity and node influence affect connectivity — such as wireless PLC networks, distributed grids, or sensor-based control zones.

Topology Parameters

- **Number of Nodes:** Total number of buses (nodes) to place in a 2D spatial domain (default: 500×500 km).
- **Threshold (θ):** Controls edge formation. Nodes i and j are connected if:

$$\frac{w_i + w_j}{d_{ij}^\beta} \geq \theta$$

where w_i and w_j are node weights (sampled uniformly in $[0,1]$), d_{ij} is Euclidean distance, and β is the spatial decay exponent.

- **Beta (β):** Governs how strongly distance penalizes connectivity. Higher values favor local clustering and suppress long-range edges.

Source and Load Placement

- **Connect Source and Loads Randomly:** If checked, sources and loads are assigned randomly across the network.
- **Number of Sources / Loads:** Specifies how many generation and consumption nodes to place.
- **Rated KV:** Voltage level for all buses.
- **Rated Min/Max KW:** Range of real power demand for loads.
- **Rated Min/Max KVar:** Range of reactive power demand for loads.
- **Min/Max Length (KM):** Range of line lengths for edge assignment (optional override of geometric distance).

Simulation Settings

- **Run Power Flow Calculation:** Enables power flow simulation after topology generation.
- **Number of PFC Run:** Number of iterations to test convergence and stability.

Actions

- **Generate:** Builds the GTG network and optionally runs power flow.

- **Cancel:** Clears the configuration or exits the setup.

Notes on Tuning

- For a 500×500 km space and uniform weights:
 - Start with $\theta = 0.01$, $\beta = 1$ for moderate connectivity.
 - Lower θ for denser graphs; raise θ for sparser graphs.
 - Increase β to emphasize local clustering and suppress long-range edges.

Geographical Threshold Graphs (GTGs)

Number of Nodes	100
Threshold	0.01
betta	1
☐ Connect Source and Loads Randomly	
Number of Sources	25
Number of Loads	40
Rated KV	15
Rated Min KW	20
Rated Max KW	2000
Rated Min KVar	10
Rated Max KVar	500
Min Length (KM)	10
Max Length (KM)	100
☐ Run Power Flow Calculation	
Number of PFC Run	10

- Figure 5-22 network model parameters for GTG model

5-8 Bianconi–Barabási model

The Bianconi–Barabási (BB) model, introduced by physicist Ginestra Bianconi in 2001, extends the classic Barabási–Albert preferential attachment framework by incorporating node fitness — a parameter η_i that represents each node’s intrinsic ability to attract links. In the BB model, the probability that a new node connects to an existing node i is proportional to both its degree k_i and fitness η_i :

$$\Pi_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j}$$

This formula captures the competition between “the earliest” (nodes that arrived first and accumulated degree) and “the fittest” (nodes with higher intrinsic attractiveness). Conceptually, the BB model explains why late-arriving but high-fitness nodes can rival or surpass early hubs, producing heavier tails in degree distributions, stronger inequality, and multiple dominant hubs. In power grid applications, this framework is valuable for modeling heterogeneous substations and feeders, where fitness can represent physical attributes such as transformer capacity, geographic importance, or reliability rating. By applying BB dynamics, operators can anticipate hub competition, identify nodes likely to become critical bottlenecks, and design redundant pathways to mitigate risks of over-centralization. This makes the BB model a powerful tool for both theoretical benchmarking and practical SCADA/HMI operator training in modern grid expansion.

Comparative Analysis of BA and BB Network Indexes (N=400, m=4)

The Barabási–Albert (BA) model and the Bianconi–Barabási (BB) model represent two paradigms of scale-free network growth. BA networks grow through preferential attachment, where early nodes accumulate links and dominate the structure. BB networks extend this by introducing fitness, allowing late-arriving but high-fitness nodes to compete with early hubs. The indexes below highlight how these dynamics manifest in measurable network properties.

- **Mean Degree:** Both BA and BB networks yield a corrected mean degree of ~ 7.95 , consistent with the theoretical expectation of $2m = 8$. This confirms that fitness heterogeneity in BB does not alter the average connectivity, but instead reshapes its distribution.
- **Density:** Both models produce sparse networks (0.04), typical of infrastructure grids. Despite BB’s hub reinforcement, overall density remains low, reflecting realistic feeder-hub structures.
- **Clustering Coefficient:** BA’s clustering is 0.0669, while BB’s is higher at 0.114. This indicates that BB hubs attract denser neighborhoods, creating more local redundancy, whereas BA remains tree-like.

- **Gini Coefficient:** BA shows moderate inequality (0.3617), while BB's inequality is stronger (0.3966). Fitness heterogeneity intensifies hub dominance, producing sharper disparities in node connectivity.
- **Assortativity:** BA is mildly disassortative (-0.0915), while BB is more strongly disassortative (-0.2087). This reflects BB's sharper hub–feeder polarization, where high-fitness hubs pull in many low-degree nodes.
- **Local Efficiency:** BA's local efficiency is 0.0869, while BB's is higher at 0.152. BB's fitness hubs create localized redundancy, improving resilience in neighborhoods, while BA remains sparse.
- **Eigenvector Centrality:** BA produces a single dominant hub (1 node >0.8, 371 nodes <0.2), reflecting the early-hub effect. BB produces multiple dominant hubs (2 nodes >0.8, 374 nodes <0.2), showing fitness-driven late hubs can rival early ones.
- **K-Core Decomposition:** BA's backbone is thinner (25 nodes in 10-core, 375 nodes in 8-core), while BB's backbone is broader (40 nodes in 10-core, 360 nodes in 8-core). Fitness heterogeneity allows more nodes to join the dense hub cluster.
- **Spectral Radius:** BA's spectral radius is 15.372, while BB's is higher at 16.767. BB's hubs amplify connectivity more strongly, reflecting sharper hub concentration.
- **Algebraic Connectivity:** BA's algebraic connectivity is 4.182, slightly lower than BB's 4.443. BB's fitness hubs improve global cohesion and resilience against fragmentation.
- **BA networks** are dominated by early hubs, with a thinner backbone, lower clustering, and moderate inequality. They are robust but less redundant, with one critical hub acting as a single point of failure.
- **BB networks** introduce fitness heterogeneity, producing multiple dominant hubs, heavier tails in degree distribution, stronger inequality, and higher clustering. They are more cohesive and locally redundant, but also more polarized, with sharper hub–feeder dependency.

For power grid applications, BA represents systems where early substations dominate connectivity, while BB models capture the reality of heterogeneous substations with varying capacities and importance. BB's indexes highlight risks of hub over-centralization but also opportunities for redundant pathways and fitness-driven resilience planning.

Comparative BA vs BB Network Metrics (N=400, m=4)

Tabulated Indexes

Metric	BA Network	BB Network	Operator-facing note
Mean degree	7.95 (corrected for in/out ports)	7.95 (corrected)	Both align with theory ($2m \approx 8$); differences emerge in distribution, not average
Density	0.04	0.04	Sparse networks typical of infrastructure grids; hubs dominate but feeders remain minimally connected
Clustering coefficient	0.0669	0.114	BB shows more local redundancy near hubs; BA remains tree-like
Gini coefficient	0.3617	0.3966	BB has stronger inequality due to fitness-driven hubs; BA is more balanced
Assortativity (Newman)	-0.0915	-0.2087	Both disassortative; BB more polarized hub–feeder structure
Local efficiency	0.0869	0.152	BB improves local redundancy; BA remains sparse and hub-centric
Eigenvector centrality	Max=1, Min=0.015; 1 node >0.8; 371 nodes <0.2	Max=1, Min=0.018; 2 nodes >0.8; 374 nodes <0.2	BA produces a single dominant hub; BB allows multiple hubs due to fitness heterogeneity
K-core decomposition	Max=10 (25 nodes); Min/Mean=8 (375 nodes)	Max=10 (40 nodes); Min/Mean=8 (360 nodes)	Both have narrow backbones; BB's is slightly broader
Spectral radius	15.372	16.767	BB's hubs amplify connectivity more strongly; BA's early hubs dominate but less extremely
Algebraic connectivity	4.182	4.443	BB slightly improves global cohesion; BA remains robust but less reinforced

The Barabási–Albert (BA) model represents networks where *the earliest nodes dominate*. Early hubs accumulate links predictably, producing a thinner backbone, lower clustering, and moderate inequality. Operators should note that BA networks often hinge on a single critical hub, making them vulnerable to founder-hub failure but otherwise balanced in connectivity.

The Bianconi–Barabási (BB) model, introduced by Ginestra Bianconi, extends BA by adding fitness — the intrinsic attractiveness of nodes. This allows late-arriving but high-fitness nodes to rival early hubs. The result is a network with multiple dominant hubs, heavier tails in degree distribution, stronger inequality, and higher clustering. Operators should recognize that BB networks are more cohesive and locally redundant, but also more polarized, with sharper hub–feeder dependency.

For power grid applications, BA captures systems where early substations dominate connectivity, while BB reflects the reality of heterogeneous substations with varying capacities and importance. BB's indexes highlight risks of hub over-centralization but also opportunities for redundant pathways and fitness-driven resilience planning. Manuals should emphasize monitoring both early hubs (BA) and fitness-driven late hubs (BB), ensuring operators anticipate where bottlenecks and vulnerabilities may arise. In Figure 5-23 you can see a typical BB, PDD for a network with 500 Nodes and $m = 2$

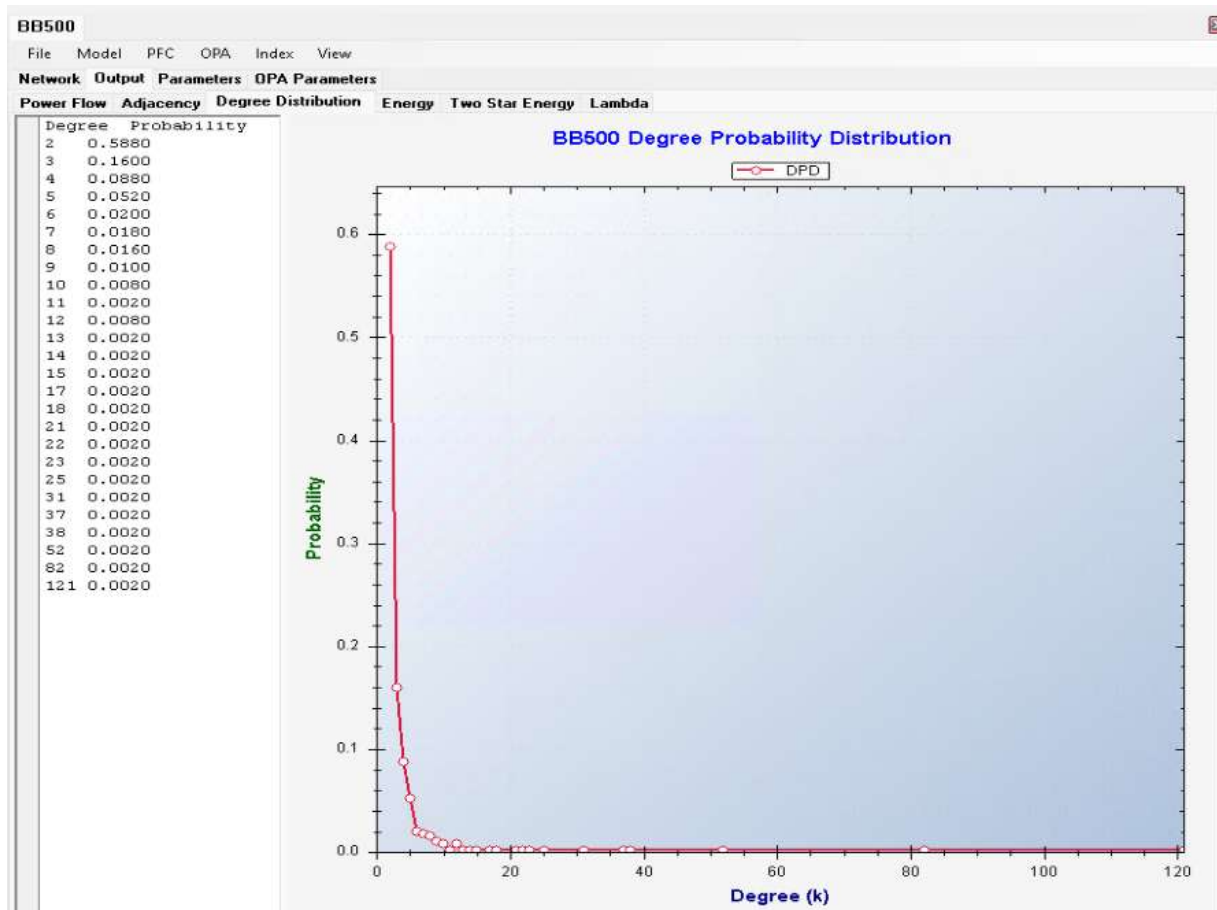


Figure 5-23

5-9 Comparison of Network Models for Power Grid Simulation

#	Model	Type	Capacity-Aware	Hub Generation	Operator Relevance
1	ER (Erdős–Rényi)	Random	Optional	✗ Minimal	Baseline randomness; useful for stress testing
2	SW (Watts–Strogatz)	Small-World	Optional	✗ Weak	High clustering; short paths; good for local zones
3	BA (Barabási–Albert)	Scale-Free	Optional	✓ Strong	Hubs via preferential attachment; abstract but insightful
4	CapGrid	Custom	✓ Yes	✓ Strong	Realistic hub emergence via capacity; ideal for operator benchmarking
5	Ring Lattice	Regular	Optional	✗ None	Pure locality; useful for control experiments
6	PLC (Power Law Cluster)	Scale-Free + Clustering	Optional	✓ Strong	Triadic closure + hubs; mimics real-world clustering
7	SA-DCSBM	Community + Degree-Corrected	✓ Yes	✓ Moderate	Captures modular zones and operator regions
8	Bianconi–Barabási	Fitness-Based Scale-Free	✓ Yes	✓ Strong	Capacity-sensitive hubs; realistic under load
9	RGG (Random Geometric Graph)	Spatial	✓ Yes	✓ Moderate	Distance-based edges; good for physical layout modeling
10	GTG (Geographical Threshold Graph)	Spatial + Threshold	✓ Yes	✓ Strong	Combines geography and capacity; excellent for grid realism
11	Cap-SW	Hybrid (SW + Capacity)	✓ Yes	✓ Strong	Local clustering + emergent hubs; operator-friendly
12	Cap-Random	Hybrid (ER + Capacity)	✓ Yes	✓ Moderate	Random topology with realistic flow constraints

6 - Power Grid Operation Simulations

Normal Operation of Power Grid

In this section, we describe how *pbsPowerGridLab* simulates normal operating conditions for a power grid and generates JSON data files for machine-learning applications. After creating a power grid using one of the available network models, the system performs iterative modifications on the loads' active power (P) and reactive power (Q) values. For each iteration, a Power Flow Calculation (PFC) is executed, and the results are saved in JSON format.

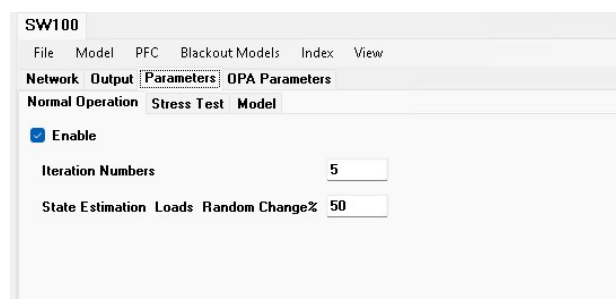
Each iteration produces two JSON files:

- **{ModelName}_output_{N}.json** — Contains the full power-grid structure, parameters, and design information used as input for the PFC.
- **{ModelName}_input_{N}.json** — Contains the resulting power-flow solution generated by the PFC for the corresponding output file.

The user can configure the number of iterations and the maximum random variation applied to load active and reactive powers. These parameters are available under: Parameters / Normal Operation Page. The configurable fields are:

- **Iteration Numbers** — Defines how many PFC iterations will be executed, and consequently how many pairs of JSON data files will be generated.
- **State Estimation Loads Change %** — Specifies the maximum percentage of random variation applied to each load's active and reactive power during every iteration.

After adjusting the parameters, right-click on the page and select Save Parameters to store the configuration.



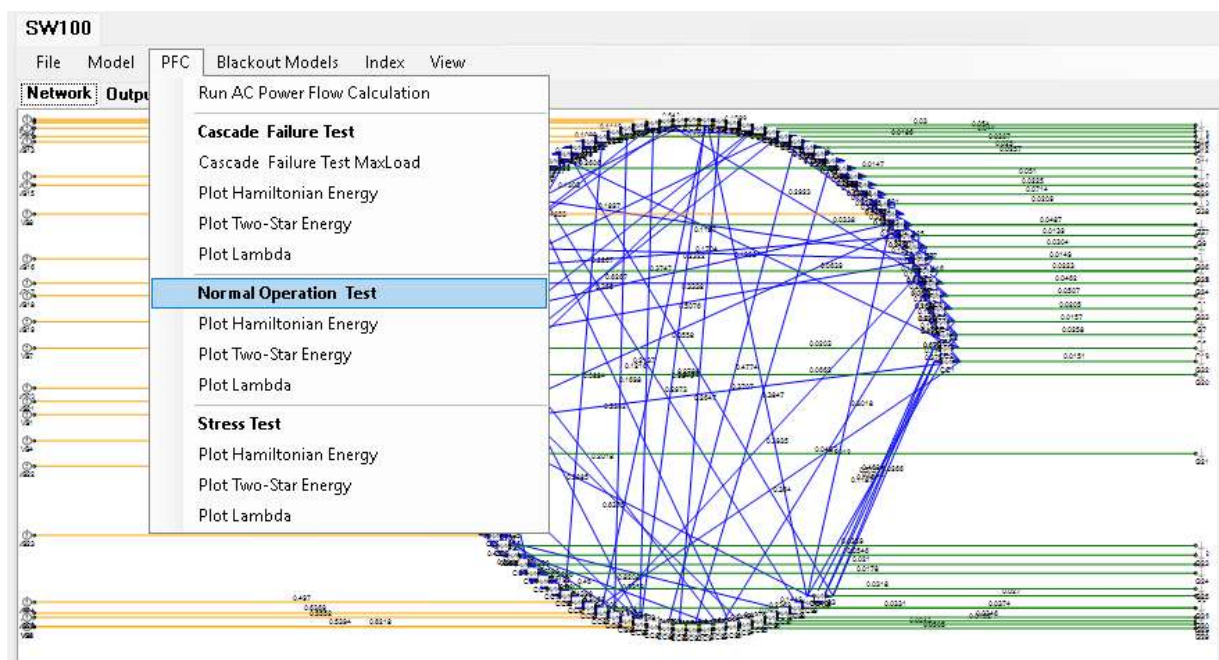
To begin generating Normal Operation data files, open the PFC menu and select Normal Operation Test. When executed, *pbsPowerGridLab* automatically creates a directory named *StateEstimation* inside the current project folder. Within this directory, three subfolders are generated:

- Train
- Predict
- Validation

After the simulation completes, the generated JSON files are distributed across these folders according to the following ratios:

- 85% of the files are copied into the Train folder
- 10% are placed in the Validation folder
- 5% are assigned to the Predict folder

This structure ensures a clean separation between training, validation, and prediction datasets for machine-learning workflows.



Stress Test

To evaluate the power grid under stressed operating conditions, *pbsPowerGridLab* provides a dedicated Stress Test mode. In this mode, the system applies incremental changes to the grid and executes a Power Flow Calculation (PFC) at each iteration. The following stress factors can be applied:

- Reduce line nominal current to simulate thermal or aging constraints
- Disable power sources to emulate generator outages
- Increase load active (P) and reactive (Q) power at each iteration

During the stress test, if a transmission line exceeds the specified loading threshold, *pbsPowerGridLab* automatically disables that line and continues the simulation. This allows users to observe how the network behaves under progressive overload conditions and partial grid failures.

Stress test parameters can be configured from: “Parameters” , “Stress Test” Page .

The configuration page allows you to define all limits and random-change ranges used during the simulation, as shown in the following image.

The Stress Test configuration page allows you to define how the grid will be stressed during each iteration of the simulation. The following parameters are available:

- **Normal Operation Iteration Number** : Specifies how many times the system will run the Normal Operation process *before* starting the Stress Test. These preliminary iterations allow the grid to stabilize under normal operating conditions so that key network properties—such

as energy levels, line loading, and baseline power-flow characteristics—are established. This ensures that the Stress Test begins from a realistic and well-defined operating state.

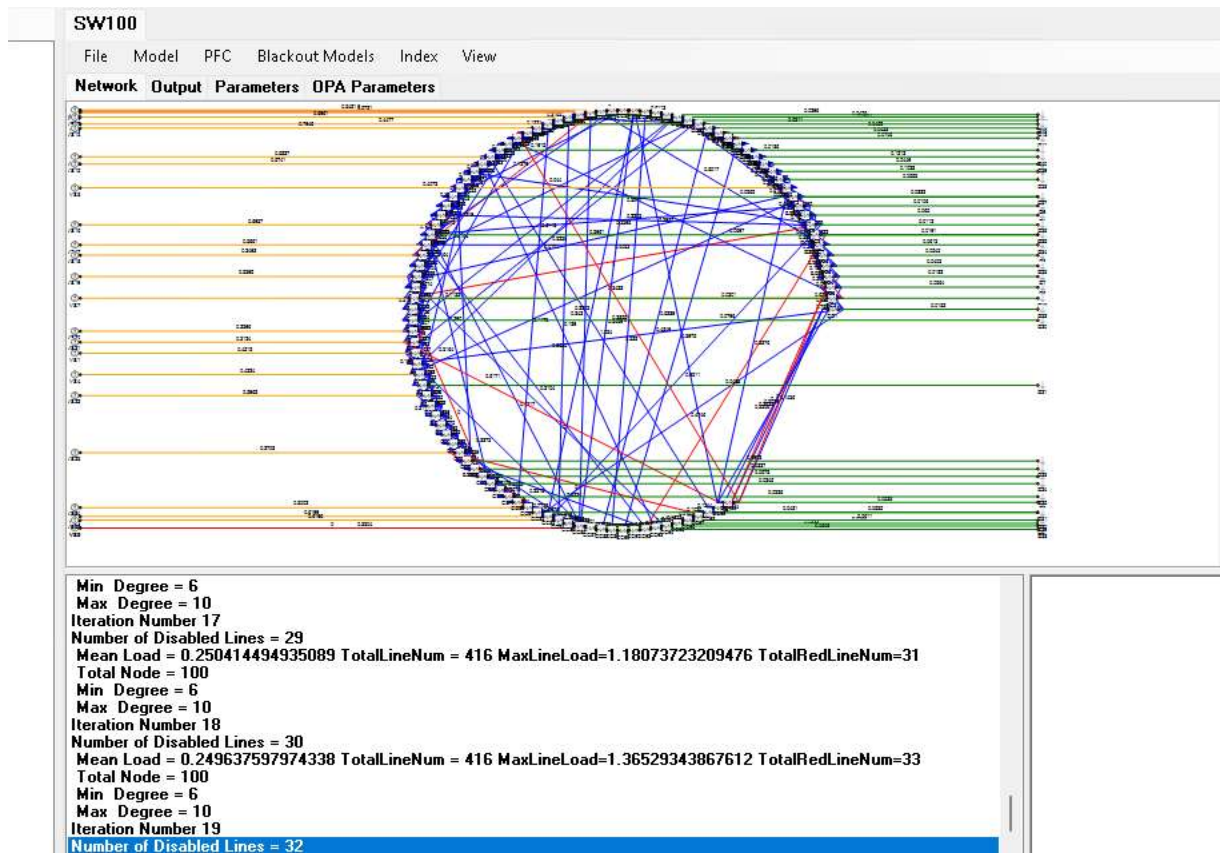
- **% of Lines Select** Indicates the percentage of transmission lines that will be selected for nominal-current reduction. These selected lines will have their nominal current decreased according to the stress test settings.
- **% of Sources** Defines the percentage of power sources (generators) that will be randomly selected and disabled during each iteration.
- **% of Loads Select** Determines the percentage of loads that will be selected for increasing their active (P) and reactive (Q) power values.
- **Disable Line if Load >** After each PFC execution, if a line's loading exceeds this threshold, the system will automatically disable that line from the network for subsequent iterations.

To begin the Stress Test simulation, open the PFC menu and select Stress Test. The system first executes the configured number of Normal Operation iterations, allowing the grid to reach a stable baseline state. After these initial iterations are completed, *pbsPowerGridLab* applies all parameters defined in the Stress Test settings page and begins the stress-based simulation.

During each stress iteration, the system evaluates line loading after the PFC is executed. If any transmission line exceeds the specified loading threshold, that line is automatically disabled. Disabled lines are visually highlighted in red on the network diagram, enabling the user to quickly identify which lines have been removed from service due to overload conditions.

The Stress Test continues until the PFC fails to converge for 10 consecutive iterations, indicating that the network can no longer operate under the applied stress conditions.

All generated JSON files are saved inside the project directory under the StressTest folder.



Cascade Failure Test

The Cascade Failure Test evaluates how the power grid behaves when sequential line outages occur. *pbsPowerGridLab* provides two modes for performing this analysis:

1. Normal Cascade Mode

In Normal Cascade Mode, the system randomly selects one transmission line, disables it, and then executes a PFC. The resulting JSON files are saved in the project directory under the LRT folder. This process repeats automatically: after each PFC execution, another random line is disabled, and the simulation continues. The cascade test stops when the PFC fails to converge for 10 consecutive iterations, indicating that the network can no longer sustain additional failures.

2. Max Mode

In Max Mode, the system identifies the transmission line with the highest loading and disables it. A PFC is then executed, and the results are saved. This approach simulates a worst-case cascade scenario, where the most heavily loaded line fails first, potentially triggering further overloads in subsequent iterations.

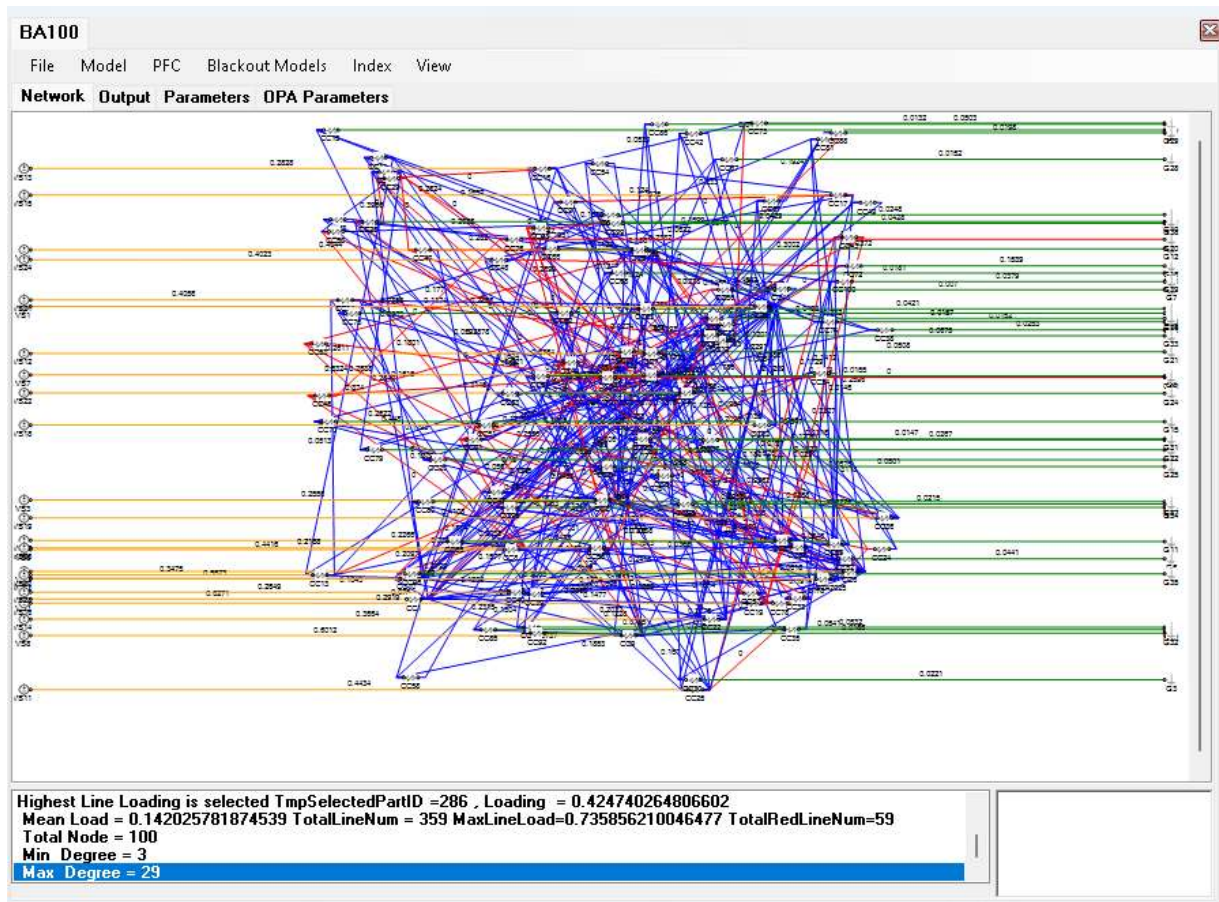
User Parameters

The Cascade Failure Test does not require any user-defined parameters. All selection logic, line disabling, and stopping conditions are handled automatically by *pbsPowerGridLab*.

During each iteration of the Cascade Failure Test, any line that becomes disabled is immediately highlighted in red on the network diagram. This allows the user to visually track the progression of the cascade and see how many lines have been removed from service in total.

In both Normal Cascade Mode and Max Mode, after the PFC is executed, *pbsPowerGridLab* also checks the loading of all remaining active lines. Any line whose loading reaches or exceeds 0.9 (90%) is automatically disabled as well. This mechanism ensures that secondary failures caused by overload propagation are accurately captured, providing a realistic simulation of cascading outages.

In the following image, you can see a BA network with 100 nodes undergoing a Cascade Failure Test. The display shows the total number of failed lines as the cascade progresses, along with the maximum line loading observed during each iteration. This visual summary helps the user understand how failures propagate through the network and how close the remaining lines are to their loading limits.



The testing framework in *pbsPowerGridLab* is built around a clear scientific principle: a power grid must be understood not only in its stable state, but also in the way it responds to stress, overload, and failure propagation. Normal Operation establishes a realistic baseline for energy flow and network behavior. The Stress Test then applies controlled disturbances—such as load increases, source outages, and line-capacity reductions—to reveal how the grid absorbs and distributes stress. Finally, the Cascade Failure Test models the chain-reaction dynamics that occur when overloaded lines fail and trigger subsequent failures. Together, these tests provide a comprehensive view of grid resilience, enabling engineers and researchers to study stability limits, vulnerability patterns, and collapse mechanisms using both simulation results and machine-learning-ready datasets.

OPA Blackout Model

The OPA model (Oak Ridge–PSerc–Alaska) provides a foundational framework for understanding the long-term behavior of power grids as complex adaptive systems. Unlike traditional engineering approaches that analyze individual contingencies, OPA captures the interaction between slow structural evolution and fast cascading failures. In this model, system load grows gradually over many planning cycles, pushing the grid toward higher stress levels and reduced operating margins. Periodically, a random disturbance initiates a cascade in which overloaded lines trip, flows redistribute, and further overloads may occur, potentially leading to widespread blackouts. After each cascade, the system adapts through local upgrades—typically increasing the capacity of stressed lines—which temporarily restores stability. This makes OPA a powerful conceptual and analytical tool for studying blackout risk, criticality, and long-term grid resilience, and it forms the foundation for integrating complex-systems behavior into pbsPowerGridLab.

Self-Organized Criticality (SOC) describes the tendency of large, interconnected systems—such as power grids—to naturally evolve toward a critical operating point where small disturbances can trigger disproportionately large events. In this state, the system balances on the edge between stability and instability, not because of poor design, but because long-term pressures such as load growth, economic optimization, and operational constraints continually push it toward high utilization. Occasional cascades act as “stress-release” events that temporarily move the system away from this critical point, but ongoing growth and adaptation gradually return it to the same sensitive regime. As a result, blackout sizes follow heavy-tailed, power-law distributions, meaning that while most events are small, large blackouts remain an inherent possibility. SOC provides a powerful conceptual lens for understanding why power grids exhibit complex, scale-free blackout behavior and why eliminating all large failures is fundamentally challenging.

In the OPA framework, the cascade represents the fast, short-timescale response of the power grid to an initiating disturbance. Once a line or component is removed—typically due to a random outage or fault—the system immediately redistributes power flows across the remaining network. If any of these new flows exceed the thermal or operational limits of other lines, those lines are tripped as well, causing further redistribution and potentially triggering additional overloads. This process continues iteratively, forming a sequence of cascading stages in which each stage consists of all components that fail due to the overloads created in the previous one. The cascade ends only when the network reaches a stable configuration with no remaining overloads or when the system fragments into islands that cannot sustain their load. The OPA cascade is intentionally modeled using static power flow and simple overload-trip rules, capturing the essential mechanism of cascading failures without requiring full dynamic simulation. This abstraction allows OPA to focus on the statistical and structural properties of cascades, making it ideal for long-term studies of blackout risk and criticality.

In the OPA model, load growth represents the slow, long-term pressure that continually pushes the power system toward higher stress levels. At each planning iteration, the total demand in the network is increased by a small percentage, reflecting real-world trends such as population growth, economic expansion, and the ongoing electrification of society. This incremental rise in load gradually reduces

operating margins, increases line flows, and brings the system closer to its thermal and structural limits. Importantly, OPA treats load growth as a persistent background force: it does not cause immediate failures, but it steadily elevates the system's vulnerability to cascading events. Over many iterations, this slow accumulation of stress is what drives the grid toward a critical operating point, setting the stage for the emergence of self-organized criticality and the heavy-tailed blackout statistics observed in real power systems.

In pbsPowerGridLab, the OPA model is implemented as an iterative simulation that captures the long-term evolution of grid stress and the short-term dynamics of cascading failures. At each iteration, pbsPowerGridLab increases system load by a small increment to represent gradual demand growth, then solves a power-flow model to determine the resulting line loadings. Lines whose loading exceeds their thermal or operational limits are treated as failed components and are tripped, initiating a cascade. The system then recomputes power flow, identifies new overloads, and continues this stage-by-stage failure process until no further violations occur. pbsPowerGridLab records the number of failures in each cascade stage and uses these counts to compute the branching factor λ , which indicates how close the system is to the critical regime where large blackouts become likely. After each cascade, the model applies local upgrades to the most stressed lines, reflecting real-world reinforcement practices. Through repeated cycles of load growth, cascading, and upgrades, pbsPowerGridLab reproduces the self-organized critical behaviour characteristic of the OPA framework and provides users with detailed blackout indexes and criticality indicators.

In pbsPowerGridLab OPA implementation, the branching factor λ is used to quantify how strongly failures propagate during a cascade. To compute λ , pbsPowerGridLab first groups all line outages into discrete cascade stages: stage 0 contains the initially failed or overloaded lines that trigger the cascade, stage 1 contains all lines that fail due to overloads caused by stage 0, stage 2 contains failures caused by stage 1, and so on, until the cascade stops. Let n_k denote the number of line failures in stage k . The empirical branching factor for a single cascade is then defined as

$$\lambda = \frac{\sum_k n_{k+1}}{\sum_k n_k},$$

where the sums run over all stages for which both n_k and n_{k+1} are nonzero. Intuitively, λ measures how many new failures each failed component produces on average in the next stage. In pbsPowerGridLab, λ can be computed for each individual cascade and then averaged over a sliding window of recent events, allowing users to monitor whether the system is in a subcritical regime ($\lambda < 1$), near the critical point ($\lambda \approx 1$), or entering a supercritical regime ($\lambda > 1$) where large blackouts become more probable.

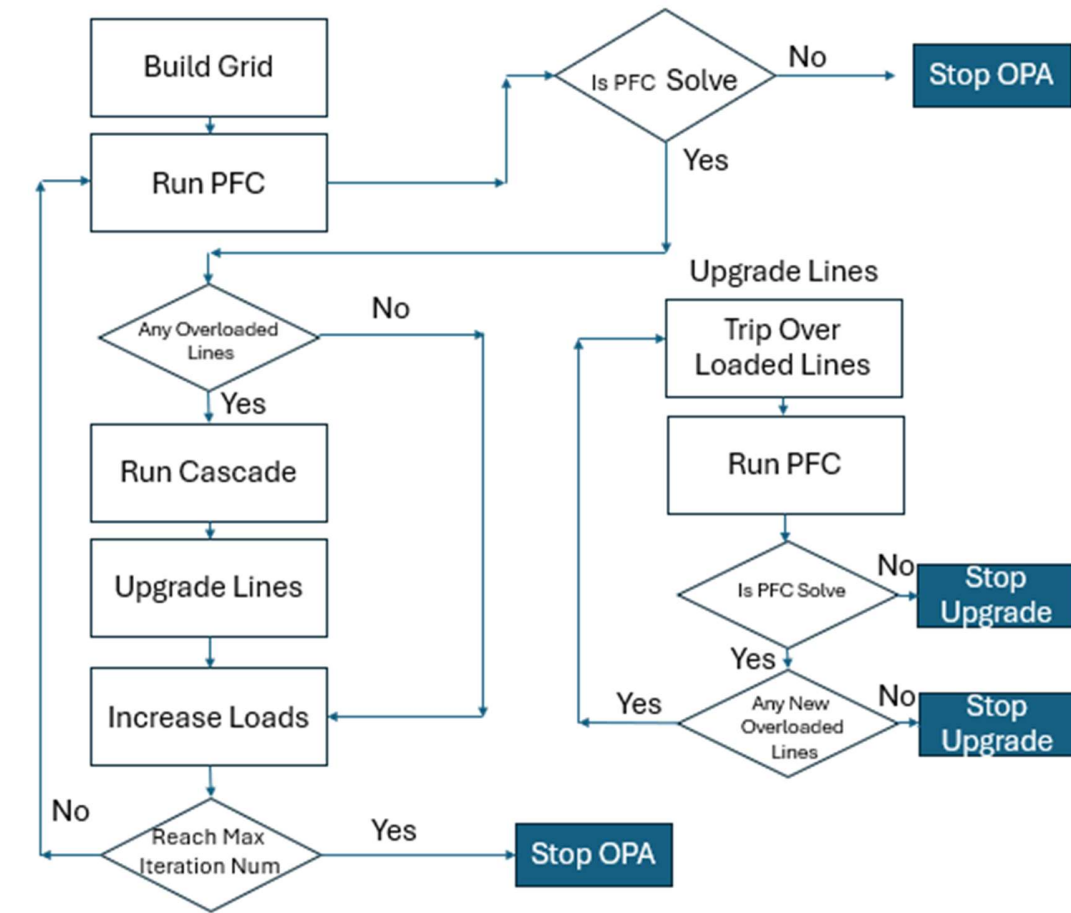
pbsPowerGridLab supports two complementary approaches for modelling load growth in OPA simulations, allowing users to choose between classical OPA behaviour and more realistic, spatially diverse scenarios. In the **standard mode**, pbsPowerGridLab applies a single global growth factor ϵ at each iteration, uniformly scaling the active and reactive power of all load buses according to

$$P_i^{\text{new}} = (1 + \epsilon) P_i^{\text{old}}, Q_i^{\text{new}} = (1 + \epsilon) Q_i^{\text{old}}$$

which preserves each load’s power factor and reflects system-wide demand growth consistent with the original OPA formulation. In the **advanced mode**, pbsPowerGridLab allows heterogeneous growth by assigning different ϵ values to individual loads or regions, enabling certain areas to grow faster, slower, or not at all. This optional mode supports more realistic planning studies—such as industrial expansion, urban growth, or localized electrification—while still maintaining compatibility with the OPA cascade and upgrade mechanisms. Together, these two methods give pbsPowerGridLab the flexibility to model both idealized and real-world load-evolution patterns.

When the power flow calculation (PFC) detects one or more overloaded transmission lines, pbsPowerGridLab enters the cascade phase to simulate failure propagation. The cascade proceeds in discrete stages: in each stage, all lines whose loading exceeds the trip threshold are removed from the network, and PFC is re-run to determine the new system state. If additional overloads appear, they are tripped in the next stage, and the process repeats until the system stabilizes, the maximum cascade depth is reached, or PFC fails due to islanding or disconnection. Each stage’s failures are recorded to compute cascade metrics such as depth, total line failures, load shed, and the branching factor λ , which quantifies how strongly failures propagate. Once the cascade ends, pbsPowerGridLab applies line upgrades and load growth before returning to the next OPA iteration.

After the cascade process finishes, pbsPowerGridLab enters the upgrade stage, where the network is reinforced based on the stresses observed during the cascade. In this step, selected transmission lines—typically those that failed or operated near their loading limits—have their nominal current ratings increased to improve future resilience. The upgrade is applied using the rule $I_{\text{nominal}}^{\text{new}} = (1 + \gamma)I_{\text{nominal}}^{\text{old}}$, where γ is the user-defined upgrade factor. Increasing the nominal rating reduces the loading ratio of the line in subsequent iterations, making it less likely to overload and trip. This stage represents the long-term adaptive response of the grid, capturing how operators reinforce infrastructure after experiencing stress or failures. Once upgrades are applied, the simulation proceeds to the load-growth step and continues with the next OPA iteration.



The OPA algorithm in pbsPowerGridLab continues iterating through power-flow calculations, cascade simulation, line upgrades, and load growth until one of its termination conditions is reached. The simulation ends when the power-flow solver can no longer find a feasible solution, typically due to severe islanding or structural collapse caused by accumulated line outages. This represents a point where the grid can no longer support the imposed load growth or withstand further disturbances. Alternatively, the algorithm stops when the user-defined maximum number of OPA iterations is reached, ensuring controlled simulation length for long-term studies. At termination, pbsPowerGridLab reports the final system state, including the last successful power flow, the extent of the final cascade (if any), and the evolution of key metrics such as load growth, line upgrades, and the branching factor λ . This provides a complete picture of how the network evolved toward criticality or collapse over the course of the OPA process.

Cascade Step in the OPA Model

In the **OPA Blackout Model**, a *cascade step* is part of the **fast-dynamics loop** where the system evaluates line overloads and removes failed components. Each cascade step represents one round of failures triggered by the previous outages. The process is quasi-steady-state: after each failure, a new power flow is solved, and additional overloads may appear.

A cascade step proceeds as follows:

1. Solve Power Flow After Previous Outages

The model computes a new power-flow solution reflecting the current network topology (with previously failed lines removed). This is the “fast dynamics” part of OPA.

2. Check Line Loading Against Trip Threshold

Every transmission line is evaluated. If a line’s loading exceeds the **trip condition threshold** (e.g., 0.9), it is considered overloaded and is **tripped** (disabled). This matches the classical OPA mechanism where overloads trigger outages.

3. Remove Overloaded Lines

All overloaded lines identified in this step are removed from service. This changes the network topology and may cause new overloads in the next iteration.

4. Continue to the Next Cascade Step

After removing overloaded lines, the model repeats the process:

- Solve power flow again
- Check for new overloads
- Trip overloaded lines

This loop continues until **no new lines fail** or until the cascade reaches the configured **Cascade Maximum Run** limit.

5. End of Cascade

The cascade stops when:

- No additional overloads occur, **or**
- The maximum number of cascade steps is reached (e.g., 10 steps)

This behavior is consistent with the OPA fast-dynamics loop described in research, where cascading failures are simulated as repeated overload-and-trip cycles.

OPA Parameters in pbsPowerGridLab

The OPA Blackout Model uses a set of configurable parameters that control how load growth, line tripping, and network upgrades evolve during each iteration of the simulation. The Increase load P and Q (epsilon) value defines the percentage by which active and reactive power are increased at each step, following the relation $P_t^{new} = (1 + \epsilon)P_t^{old}$ and $Q_t^{new} = (1 + \epsilon)Q_t^{old}$. Users may choose a random and uniform increase by specifying a minimum and maximum epsilon range, allowing the model to apply stochastic load growth. The Line Loading Trip Condition sets the threshold above which a line is considered overloaded and is automatically tripped. The Cascade Maximum Run parameter determines how many consecutive cascade steps are allowed before the model stops. The Upgrade Line Loading Factor (Landa) controls how much the nominal current of a line increases after an upgrade event, using $I_{nominal}^{new} = (1 + \gamma)I_{nominal}^{old}$. Finally, OPA Maximum Iteration defines the total number of long-term OPA cycles, combining load growth, cascading failures, and network upgrades into a complete blackout-risk simulation.

OPA

Increase load P and Q (epsilon)

☐ Random and uniform increase of P and Q

Max epsilon

Min epsilon

$P_t^{new} = (1 + \epsilon) P_t^{old}, Q_t^{new} = (1 + \epsilon) Q_t^{old}$

Line loading Trip condition >

Cascade Maximum Run

Upgrade Line Loading factor (Landa)

$I_{nominal}^{new} = (1 + \gamma) I_{nominal}^{old}$

OPA Maximum Iteration

The OPA simulation function performs one complete long-term iteration of the OPA blackout model. In each cycle, the system increases all load values according to the selected epsilon growth factor, then executes a power-flow calculation to determine the network's operating state. If any lines become overloaded, the model enters a cascade phase where overloaded lines are progressively removed and the power flow is recalculated until the cascade stabilizes or reaches its maximum depth. Once the cascade finishes, the system upgrades the capacity of lines that experienced overloads, reflecting long-term infrastructure reinforcement. Finally, the model evaluates how much of the total demand was successfully served and records the corresponding lambda value for blackout-risk analysis. This iterative process captures the essential dynamics of load growth, cascading failures, and network adaptation that define the OPA model.

7 - Power Grid Model Network Metrics

The **pbsPowerGridLab** provides a comprehensive suite of structural and spectral indexes to evaluate the topology, efficiency, and resilience of power grid models. These indexes enable operators and researchers to quantify connectivity patterns, detect vulnerabilities, and benchmark network growth dynamics. The supported metrics include:

- **Average Clustering Coefficient** – measures the tendency of nodes to form tightly connected neighbourhoods.
- **Gini Coefficient** – quantifies inequality in the degree distribution, highlighting disparities in node connectivity.
- **Pearson Degree Correlation** – captures assortativity, indicating whether nodes of similar degree preferentially connect.
- **Mean Degree** – the average number of connections per node, reflecting overall network density.
- **Local Efficiency** – evaluates fault tolerance by measuring how efficiently information is exchanged in local subgraphs.
- **Eigenvector Centrality** – identifies influential nodes based on their connections to other highly connected nodes.
- **K-Core Number** – determines the maximal subgraph in which each node has at least k connections, useful for resilience analysis.
- **Density** – expresses the ratio of actual to possible connections, indicating overall network compactness.
- **Algebraic Connectivity (Fiedler Value)** – the second-smallest eigenvalue of the Laplacian matrix, measuring robustness and connectivity strength.
- **Spectral Radius** – the largest eigenvalue of the adjacency matrix, linked to network stability and dynamic processes.
- **Number of Triangles** – counts closed triplets, providing insight into clustering and redundancy in the network.

Average Clustering Coefficient

The **clustering coefficient** measures how strongly nodes in a network tend to form tightly connected groups (triangles). The **average clustering coefficient** is the mean of all individual node clustering coefficients, giving a global measure of local interconnectedness.

For a single node i with degree k_i :

$$C_i = \frac{2 \cdot e_i}{k_i(k_i - 1)}$$

- e_i = number of edges between the neighbors of node i
- k_i = degree of node i

The **average clustering coefficient** across the network is:

$$C = \frac{1}{N} \sum_{i=1}^N C_i$$

- N = total number of nodes

Conceptual Meaning

- High C : Nodes form many closed triangles → strong local redundancy.
- Low C : Neighbors are loosely connected → weak local cohesion.
- It reflects **local resilience**: if one connection fails, alternative paths may exist.

Application in Power Grids

- **Fault tolerance**: A higher clustering coefficient indicates that substations or buses are interconnected in ways that provide backup paths, improving reliability.
- **Redundancy analysis**: Helps identify areas where local loops exist, which can prevent cascading failures.
- **Grid design**: Engineers may aim for moderate clustering — too low means fragile local structures, too high may imply costly redundancy.
- **Benchmarking models**: Comparing clustering across different network models, or real-world grids shows how well models capture local resilience.

For an ER graph $G(N, p)$, the **expected clustering coefficient** is simply:

$$C \approx p$$

because the probability that two neighbours of a node are connected is exactly p .

For a **Watts–Strogatz small-world model**:

- Start with a regular ring lattice where each node is connected to m neighbors.
- The clustering coefficient of the initial lattice is high (close to $\frac{3(m-2)}{4(m-1)}$ for large N).
- As edges are rewired with probability p , clustering decreases but remains significantly higher than in an equivalent ER random graph.

For a regular ring lattice, the expected clustering coefficient is:

$$C = \frac{3(k-2)}{4(k-1)}$$

The comparative analysis of clustering coefficients across canonical network models highlights the distinct structural signatures that each topology introduces into power grid modeling. The **Erdős–Rényi (ER)** random graph exhibits very low clustering, consistent with its purely probabilistic edge formation. In contrast, the **Small-World (SW)** model maintains strong local clustering even after rewiring, reflecting its balance between randomness and lattice-like order. The **Barabási–Albert (BA)** scale-free network shows modest clustering, typical of preferential attachment structures where hubs dominate connectivity. The **Ring Lattice** achieves the highest clustering among deterministic models, closely matching analytic expectations. The **Power-Law Cluster (PLC)** model improves upon BA by introducing triad formation, yielding moderate clustering values. The **Random Geometric Graph (RGG)** demonstrates spatially driven clustering, sensitive to mean degree and boundary conditions, while the **Geometric Threshold Graph (GTG)** produces near-lattice levels of clustering due to its threshold-based neighborhood reinforcement. Together, these results illustrate how clustering serves as a diagnostic lens for resilience, redundancy, and local cohesion in power grid networks, providing operators and researchers with a benchmark for evaluating model behavior against real-world grid structures.

Cumulative results table

Network model	Nodes (N)	Parameters	Expected CC (qualitative)	Measured CC	Notes
ER random graph	400	$p = 0.02$	$\approx p$	0.0175	Matches ER expectation
SW small-world	400	$m = 4, p = 0.1$	High ($\gg$ ER)	0.4729	Strong clustering typical of SW
BA scale-free	400	$m = 4$	Low, decreases with N	0.0557	Typical BA clustering

Ring lattice	400	m = 4 per side (k = 8)	≈ 0.6429 (analytic)	0.6443	Matches analytic value
PLC (Holme–Kim)	400	m = 4, p = 0.1	Moderate, above BA	0.0904	Consistent with triad formation
RGG (2D)	400	mean degree = 5.25	Moderate; boundary-sensitive	0.3974	Plausible for non-periodic boundaries
GTG (Geometric threshold)	400	$\theta = 0.02, \beta = 1$	Very high local cohesion	0.6218	Strong clustering, near lattice-level

Gini Coefficient

The **Gini Coefficient** is a measure of inequality in a distribution. Applied to networks, it quantifies how unevenly node degrees are distributed.

- $G = 0$: perfect equality (all nodes have the same degree).
- $G = 1$: maximum inequality (one node has all connections, others have none).

For a network with N nodes and degree sequence $\{k_1, k_2, \dots, k_N\}$, sorted in non-decreasing order:

$$G = \frac{\sum_{i=1}^N \sum_{j=1}^N |k_i - k_j|}{2N \cdot \sum_{i=1}^N k_i}$$

Conceptual Meaning

- **Low Gini (≈ 0):** Degrees are evenly distributed → homogeneous network.
- **High Gini (≈ 1):** Degrees are highly unequal → presence of hubs and peripheral nodes.
- It captures **heterogeneity** in connectivity, complementing mean degree and clustering.

Application in Power Grids

- **Resilience analysis:** A high Gini indicates hub-dominated structures (like BA networks), which may be vulnerable to targeted attacks but robust against random failures.
- **Equity of load distribution:** Helps assess whether connectivity is balanced across substations or concentrated in a few critical nodes.

- **Benchmarking models:** Comparing Gini across ER, SW, BA, and real grids shows how well models capture heterogeneity.
- **Design guidance:** Operators may prefer moderate Gini values — too low implies uniformity (less realistic), too high implies fragility.

Theoretical Expectation for ER Network

- In ER graphs, degrees follow a **binomial distribution** (or Poisson for large N and small p).
- Degree variance is relatively low compared to scale-free networks.
- Thus, the **Gini coefficient** should be small, reflecting near-uniform connectivity.

For $N = 400$, $p = 0.02$:

- Expected mean degree: $\langle k \rangle = p(N - 1) \approx 0.02 \times 399 \approx 7.98$.
- Degree distribution is narrow around 8, so Gini ≈ 0.15 – 0.20 is typical.

The comparative Gini results across canonical network models reveal how degree inequality varies with topology. The **Ring Lattice** and **Small-World** networks show near-zero Gini values, reflecting their almost uniform degree distributions, with only slight variation introduced by rewiring in the SW case. The **Erdős–Rényi (ER)** random graph produces a modest Gini, consistent with its narrow binomial degree spread. In contrast, the **Barabási–Albert (BA)** and **Power-Law Cluster (PLC)** models exhibit high Gini values, highlighting the hub-dominated structure created by preferential attachment, where a few nodes accumulate disproportionate connectivity. The **Random Geometric Graph (RGG)** demonstrates intermediate inequality: at lower mean degree (≈ 5.25) Gini is around 0.23, while at higher mean degree (≈ 12.85) it rises to 0.30, showing how boundary effects and spatial heterogeneity intensify with density. Finally, the **Geometric Threshold Graph (GTG)** is expected to yield strong clustering and moderate-to-high Gini, depending on threshold parameters, as local reinforcement creates both cohesive neighborhoods and degree disparity. Together, these results illustrate how Gini complements clustering by quantifying inequality in connectivity, offering operators and researchers a diagnostic lens for resilience, vulnerability, and load distribution in power grid networks.

Gini table

Network model	Nodes (N)	Parameters	Expected Gini (qualitative)	Measured Gini	Notes
ER random graph	400	$p = 0.02$	Low ($\sim 0.15\text{--}0.20$)	0.1727	Narrow, near-Poisson degrees
SW small-world	400	$m = 4, p = 0.1$	Very low (< 0.1)	0.0555	Near-regular; mild rewiring variance
BA scale-free	400	$m = 4$	High ($\approx 0.30\text{--}0.45$)	0.3541	Hubs drive inequality
Ring lattice	400	k uniform ($k = 8$)	~ 0	0.0037	Uniform degree distribution
PLC (Holme–Kim)	400	$m = 4, p = 0.1$	High ($\approx$ BA)	0.3598	Heavy-tailed; triads keep inequality high
RGG (2D)	400	mean degree = 12.85	Moderate–high	0.3004	Boundary vs. interior disparity increases with radius

Pearson Degree Correlation (Assortativity)**1. Definition**

The **Pearson degree correlation coefficient** measures the correlation between the degrees of connected nodes.

- Positive values ($r > 0$): **Assortative mixing** — high-degree nodes tend to connect to other high-degree nodes.
- Negative values ($r < 0$): **Disassortative mixing** — high-degree nodes tend to connect to low-degree nodes.
- Values near zero: No significant correlation.

For a network with edges (i, j) and degrees k_i, k_j :

$$r = \frac{\sum_{(i,j)} (k_i - \mu)(k_j - \mu)}{\sum_{(i,j)} (k_i - \mu)^2}$$

where μ is the mean degree over all edge ends. This is essentially the Pearson correlation coefficient applied to the degree sequence at either end of each edge

Conceptual Meaning

- **Assortative networks ($r > 0$):** Common in social networks, where well-connected individuals link to each other.
- **Disassortative networks ($r < 0$):** Common in technological and biological networks, where hubs connect to many low-degree nodes (e.g., BA scale-free).
- **Neutral ($r \approx 0$):** Random graphs often show little correlation.

Application in Power Grids

- **Resilience:** Assortative grids may be more robust to random failures, since hubs reinforce each other.
- **Vulnerability:** Disassortative grids (hub–periphery structures) can be efficient but fragile if hubs fail.
- **Benchmarking:** Comparing assortativity across ER, SW, BA, RGG, GTG, etc., helps operators understand whether their grid resembles a hub–periphery or a more balanced structure.
- **Design guidance:** Moderate assortativity may balance efficiency and resilience.

Newman’s assortativity coefficient is a specialized measure of degree correlation in networks, designed to capture whether nodes of similar degree tend to connect. It is defined as

$$r = \frac{\sum_{jk} jk (e_{jk} - q_j q_k)}{\sigma_q^2}$$

where e_{jk} is the joint probability distribution of the degrees at either end of an edge, q_k is the distribution of *excess degree* (degree minus one, since one edge is already “used” when arriving at a node), and σ_q^2 is the variance of that distribution. This formulation differs from a simple Pearson correlation in two important ways: (1) it uses **excess degree** rather than raw degree, ensuring that the measure reflects the connectivity structure of edges rather than node attributes; and (2) it weights the calculation by **edge ends**, treating each stub as part of the distribution, rather than averaging over unique nodes or edges. A plain Pearson correlation between node degrees at edge endpoints can give misleading results, often inflating values in heterogeneous networks like BA, whereas Newman’s definition corrects for this bias and yields values consistent with theoretical expectations (negative for BA, neutral for ER, positive for RGG).

The comparative results highlight how Newman’s assortativity definition provides a more reliable measure of degree–degree correlation across canonical network models. In homogeneous structures such as the Erdős–Rényi random graph, the small-world model, and the ring lattice, both Pearson and

Newman values collapse near zero, confirming their essentially neutral mixing patterns. In preferential-attachment networks (BA, PLC, and CapGrid), raw Pearson correlations misleadingly suggest positive assortativity, while Newman's formulation correctly reveals mild to strong disassortativity driven by hub–periphery connections. By contrast, spatially embedded models (RGG and GTG) exhibit genuine assortativity: RGG shows strong positive correlation due to clustering in dense regions, while GTG produces only weakly positive values, reflecting threshold-based homophily. Taken together, the table demonstrates that Newman's edge-end, excess-degree approach eliminates biases present in raw Pearson calculations and yields results consistent with theoretical expectations, making it the preferred standard for operator manuals and academic benchmarking.

Network model	Nodes (N)	Parameters	Pearson (raw degree)	Newman (excess degree)	Notes
ER random graph	400	$p = 0.02$	-0.0463	-0.0481	Near-neutral; finite-size bias
SW small-world	400	$m = 4, p = 0.1$	-0.0426	-0.0434	Mild disassortativity from rewiring
BA scale-free	400	$m = 4$	+0.3201	-0.0572	Pearson inflated positive; Newman shows mild disassortativity
CapGrid	400	$m = 4$	-0.1042	-0.6666	Strong hub–periphery disassortativity
Ring lattice	400	$k = 2m$ (regular)	0.0000	0.0000	Strictly regular; variance collapses
PLC (Holme–Kim)	400	$m = 4, p = 0.1$	+0.3289	-0.0525	Newman corrects Pearson's positive bias
RGG (2D)	400	mean degree ≈ 14.8	+0.5804	+0.5749	Both strong positive; genuine spatial assortativity
GTG (Threshold)	400	mean degree $\approx 14\text{--}15$	+0.1852	+0.1834	Both weakly positive; near-neutral

Local efficiency

Local efficiency measures how well a node's neighbors communicate when that node is removed. Formally, for each node i , you compute the efficiency of the subgraph induced by its neighbors:

$$E_{loc}(i) = \frac{1}{k_i(k_i - 1)} \sum_{\substack{u, v \in N(i) \\ u \neq v}} \frac{1}{d_{uv}}$$

- k_i : degree of node i
- $N(i)$: set of neighbors of node i
- d_{uv} : shortest path length between neighbors u and v in the induced subgraph

The **global local efficiency** is the average over all nodes:

$$E_{loc} = \frac{1}{N} \sum_{i=1}^N E_{loc}(i)$$

Why it matters

- **Operator manuals:** Local efficiency quantifies **redundancy and resilience** — how well local clusters remain connected if a node fails.
- **Model benchmarking:**
 - ER: low local efficiency (neighbors rarely connected).
 - SW: high local efficiency (neighbors form tight clusters).
 - BA/PLC: hubs have low local efficiency (neighbors are peripheral), but peripheral nodes can have higher values.
 - CapGrid: efficiency reflects capacity-driven clustering — useful diagnostic.
 - RGG/GTG: typically high local efficiency due to spatial clustering.
 - Ring lattice: maximum local efficiency (neighbors are fully interconnected in the local subgraph).

Application in power grids

- **Resilience to node failure:** Local efficiency quantifies how well neighboring substations or buses remain connected if one node fails. A high local efficiency means that even if a node is removed, its neighbors can still exchange power through alternate paths.

- **Redundancy assessment:** Operators use it to evaluate whether local clusters of substations have enough backup connections to avoid isolation.
- **Planning and design:** Helps compare radial vs. meshed layouts. Radial distribution grids typically have low local efficiency (neighbors are not interconnected), while transmission grids with multiple tie-lines show higher values.
- **Reliability benchmarking:** Local efficiency complements clustering coefficient and assortativity in describing how robustly information (or power flow) can be rerouted in case of outages

Typical values

From studies in complex networks and power grid modeling:

- **Radial distribution grids:** Local efficiency is often < 0.2 , reflecting poor redundancy.
- **Transmission grids (meshed):** Values typically fall in the **0.3–0.6** range, depending on interconnection density.
- **Small-world or spatial models (RGG, GTG):** Can reach **0.5–0.7**, since neighbors are densely clustered.
- **Ring lattice (ideal regular):** Approaches **1.0**, because neighbors are fully interconnected.
- **Real-world European grids:** Empirical studies report local efficiency around **0.4–0.5**, consistent with meshed transmission design

Operator-facing interpretation

For manuals and training:

- **Low local efficiency (<0.2):** Vulnerable clusters; a single node failure isolates neighbors.
- **Moderate local efficiency (0.3–0.5):** Balanced redundancy; typical of transmission grids.
- **High local efficiency (>0.6):** Strong resilience; neighbors remain well connected even under failures.

Erdős–Rényi (ER) Local efficiency is very low (0.0253). Neighbors of a node are rarely interconnected, so when a node fails its immediate neighborhood has almost no alternative paths. For operators, this highlights the fragility of random connectivity and the need for deliberate meshing.

Small-World (SW) Local efficiency is high (0.6234). The rewiring preserves clustering while introducing shortcuts, so neighbors remain well interconnected. This demonstrates how modest rewiring yields both short global paths and strong local redundancy.

Barabási–Albert (BA) Local efficiency is low (0.0813). Hubs dominate connectivity, but their neighbors are peripheral and sparsely linked. This produces global efficiency but local fragility, underscoring the need for meshed interconnections.

Ring Lattice Local efficiency is high (0.7611). The regular lattice ensures neighbors are strongly interconnected, providing robust redundancy. This serves as a benchmark for maximum neighborhood resilience.

Power-Law Cluster (PLC) Local efficiency is low (0.1277). Triangle formation improves clustering compared to BA, but hub-periphery structure still dominates. The result is slightly more resilient than BA but still fragile locally.

Random Geometric Graph (RGG) Local efficiency is high (0.7376). Spatial embedding ensures neighbors are close and interconnected, so redundancy is strong. This illustrates how spatial constraints naturally enhance local robustness.

Geometric Threshold Graph (GTG) Local efficiency is high (0.7732). Threshold-driven connectivity concentrates links among nearby or high-weight nodes, forming dense neighborhoods. This balances global reach with strong local resilience.

CapGrid Local efficiency is very high (0.8437). Capacity-driven clustering fosters dense local neighborhoods, ensuring neighbors remain richly interconnected. This suggests strong redundancy under localized failures, making CapGrid a candidate design reference for meshed transmission systems.

Model	Local Efficiency	Operator-Facing Note
ER	0.0253	Very low redundancy; random connectivity fragile at neighborhood level
SW	0.6234	High clustering; rewiring yields robust local backup paths
BA	0.0813	Hub-dominated; globally efficient but locally fragile
Ring Lattice	0.7611	Strong redundancy; benchmark for maximum neighborhood resilience
PLC	0.1277	Slightly better than BA; still weak local backup paths
RGG	0.7376	Spatial clustering ensures strong local robustness
GTG	0.7732	Threshold connectivity produces dense, resilient neighborhoods
CapGrid	0.8437	Capacity-driven meshing yields very high local redundancy

Eigenvector Centrality in Network Analysis

◆ Definition

Eigenvector centrality is a measure of node importance in a network. Unlike degree centrality, which counts direct connections, eigenvector centrality assigns higher scores to nodes that are connected to other well-connected nodes. In other words, a node's influence grows not only with the number of its neighbors but also with the quality of those neighbors.

Eigenvector centrality is defined as the solution to the eigenvalue problem:

$$Ax = \lambda x$$

- A = adjacency matrix of the network
- x = vector of centrality scores
- λ = largest eigenvalue of A

Each node's score x_i satisfies:

$$x_i = \frac{1}{\lambda} \sum_j A_{ij} x_j$$

This recursive definition means a node is important if it is connected to other important nodes.

Concepts

- **Recursive influence:** A node's centrality depends on the centrality of its neighbors.
- **Global measure:** Unlike local metrics (degree, clustering), eigenvector centrality captures influence across the entire network.
- **Hub identification:** Nodes with high eigenvector centrality are often hubs or connectors to hubs.
- **Distribution insight:** The spread of eigenvector scores reveals whether influence is concentrated (few dominant hubs) or distributed (many moderately important nodes).

Applications in Power Grids

- **Critical substation identification:** Substations with high eigenvector centrality are structurally important because they connect to other influential substations. Their failure can disproportionately affect grid stability.
- **Vulnerability analysis:** Comparing eigenvector centrality with load or capacity highlights nodes that are both structurally and operationally critical.

- **Resilience planning:** Operators can prioritize reinforcement or redundancy around high-centrality nodes to prevent cascading failures.
- **Design optimization:** Eigenvector centrality helps evaluate whether a grid design concentrates influence in a few hubs (risk of fragility) or distributes it across many nodes (greater resilience).
- **Integration with other metrics:** Used alongside assortativity and local efficiency, eigenvector centrality provides a holistic view of both influence and redundancy in grid topology.

When we compute **eigenvector centrality**, we don't explicitly solve the full eigenvalue/eigenvector problem with matrix decomposition (like you would in linear algebra coursework). Instead, we approximate the **principal eigenvector of the adjacency matrix** using iterative methods (most commonly **power iteration**).

Why we don't directly calculate eigenvalues/eigenvectors

- **Full decomposition** (e.g., via QR algorithm or SVD) is computationally expensive for large graphs.
- **Eigenvector centrality only needs the principal eigenvector** (the one associated with the largest eigenvalue).
- Iterative methods converge quickly to that vector without computing all eigenvalues/eigenvectors.

Computational Procedure

Since directly solving the eigenvalue problem can be expensive for large graphs, we typically use **power iteration**:

1. **Initialize:** Start with a vector $x^{(0)}$ of ones (or random values).
2. **Iterate:** Multiply by the adjacency matrix repeatedly:

$$x^{(k+1)} = Ax^{(k)}$$
3. **Normalize:** After each multiplication, normalize the vector (e.g., divide by its norm).
4. **Converge:** Repeat until $x^{(k)}$ stabilizes. The result approximates the eigenvector corresponding to the largest eigenvalue.
5. **Interpret:** The entries of the final vector are the eigenvector centrality scores.

Eigenvector Centrality Profiles by Model

- **Erdős–Rényi (ER):** Influence is skewed by chance. A few nodes (5) emerge as highly influential, while many (32) remain weak. Even in random designs, hidden hubs appear, creating fragility if those chance hubs fail.
- **Small-World (SW):** Influence is more evenly spread, with 18 nodes scoring high and none weak. This reflects clustered neighborhoods connected by shortcuts, producing redundancy and distributed influence. Operators benefit from multiple overlapping hubs.
- **Barabási–Albert (BA):** Influence is extremely concentrated. Only 2 nodes dominate, while 368 remain weak. This scale-free hub structure is efficient but fragile: resilience depends heavily on protecting the few critical hubs.
- **Ring Lattice:** Influence is perfectly uniform. All nodes score equally (1.0), reflecting the regularity of the lattice. While robust in theory, such uniformity is unrealistic for real grids, which require hierarchy and redundancy.
- **Power-Law Cluster (PLC):** Adds clustering to BA, but hub dominance persists. Three nodes dominate influence, while 363 remain weak. Clustering improves local redundancy, but fragility remains if hubs fail.
- **Random Geometric Graph (RGG):** Influence is moderately distributed. Fourteen nodes score high, while 360 remain weak. Dense spatial clusters create local hubs, while boundary nodes remain peripheral. Operators should reinforce cluster hubs and monitor isolated nodes.
- **Geometric Threshold Graph (GTG):** Threshold rules concentrate influence in 11 nodes, while 317 remain weak. These threshold-qualified hubs act as structural keystones. Resilience depends heavily on their stability, making them priority targets for reinforcement.

Model	Max Score	Min Score	Nodes > 0.8	Nodes < 0.2	Distribution Note
ER	1.0	0.06	5	32	Skewed; few chance hubs, many weak nodes
SW	1.0	0.0	18	0	Distributed influence across clustered neighborhoods
BA	1.0	0.013	2	368	Strong hub dominance; majority structurally weak
Ring Lattice	1.0	1.0	All	0	Uniform; perfectly distributed influence

PLC	1.0	0.016	3	363	Hub-dominated; clustering adds redundancy but fragility persists
RGG	1.0	0.0	14	360	Spatial clusters yield local hubs; majority remain weak
GTG	1.0	0.0	11	317	Threshold hubs dominate; majority peripheral

In eigenvector centrality analysis, nodes with scores **greater than 0.8** are classified as **critical hubs**. These nodes exert strong influence across the network because they are connected to other highly influential nodes, and their failure would disproportionately reduce connectivity and resilience. Conversely, nodes with scores **below 0.2** are considered **peripheral nodes**. They contribute little to global influence, often lying at the boundaries of clusters or in sparse regions, and their removal has minimal impact on overall stability. By applying these thresholds, operators can quickly distinguish between structurally important nodes that require reinforcement and monitoring, and peripheral nodes that, while less critical globally, may still matter locally. This classification provides a practical framework for prioritizing resilience planning and training.

k-Core Number

The **k-core number** of a node is a measure of its structural embeddedness in a network. Formally, a **k-core** is the maximal subgraph in which every node has degree at least k . The **k-core number** of a node is the highest value of k for which the node belongs to that subgraph. Mathematically, if $d(v)$ is the degree of node v , then node v is in the k -core if $d(v) \geq k$ after recursively removing all nodes with degree less than k . Conceptually, this metric identifies how deeply a node is integrated into the network's backbone: nodes with high k-core numbers are surrounded by dense, redundant connections, while nodes with low k-core numbers are peripheral and easily isolated.

In **power grid applications**, k-core analysis highlights the difference between **critical backbone substations** and **fragile peripheral nodes**. High k-core nodes form the resilient core of the grid, where redundancy ensures stability even under stress. Low k-core nodes, by contrast, are vulnerable to disconnection and can represent weak points in network expansion or rural distribution. For operators, k-core classification provides a practical framework for resilience planning: reinforcing high k-core nodes maintains backbone integrity, while monitoring low k-core nodes prevents local isolation. Combined with eigenvector centrality, k-core analysis helps distinguish between nodes that are globally influential and those that are structurally embedded, offering a dual perspective for robust grid design.

In the pbsHMI interface, the k-core diagnostic panel presents a concise set of values that summarize the structural depth of the network. Operators see the **maximum k-core value**, identifying the deepest backbone nodes; the **minimum k-core value**, highlighting peripheral nodes most vulnerable to isolation; and the **mean k-core value**, rounded for readability, which reflects the average embeddedness of the system. Alongside these, pbsHMI shows the **counts of nodes** at each extreme (maximum, minimum, and mean), as well as the **most common k-core value** and its frequency, which indicates the dominant structural layer of the grid. Together, these metrics allow operators to quickly distinguish backbone substations from peripheral ones, assess how influence is distributed across the network, and prioritize reinforcement or monitoring based on structural classification.

pbsHMI K-Core Variables :

- **K_Core_MaxValue – Maximum k-core value** The highest k-core number found in the network. It identifies the deepest backbone nodes that are most structurally embedded.
- **K_Core_MinValue – Minimum k-core value** The lowest k-core number in the network. It highlights peripheral nodes that are least connected and most vulnerable to isolation.
- **K_Core_MeanValue – Mean (rounded) k-core value** The average k-core number across all nodes, rounded to the nearest integer. It provides a quick measure of the overall embeddedness of the network.
- **K_Core_Max_Num – Count of nodes with max value** The number of nodes that achieve the maximum k-core value. This shows how many substations or nodes form the deepest structural backbone.
- **K_Core_Min_Num – Count of nodes with min value** The number of nodes that fall at the minimum k-core level. These are the peripheral nodes at the edge of the network.
- **K_Core_Mean_Num – Count of nodes with mean value** The number of nodes whose k-core number equals the rounded mean. This indicates how many nodes sit near the “average” level of embeddedness.
- **K_Core_SameValue – Most frequent k-core value** The k-core number that occurs most often in the network. It represents the dominant structural layer of the grid.
- **K_Core_SameNum – Count of nodes with that value** The number of nodes that share the most frequent k-core value. This shows the size of the largest structural group in the network.

Relationship Between Degree and k-Core Value

- **Degree ($d(v)$):** The number of direct neighbors a node has.
- **k-Core value ($c(v)$):** The highest k such that the node belongs to the k -core subgraph, where every node has degree $\geq k$ within that subgraph.

So:

- A node's **degree** is its raw connectivity.
- A node's **k-core value** is its *structural depth*, determined by recursively peeling away nodes with degree less than k .

ER Network k-Core Analysis 400 Nodes , $p = 0.02$

- **Maximum k-core value = 6 (225 nodes):** A large backbone exists in the ER network, with 225 nodes embedded at depth 6. These nodes form the resilient structural core, providing strong redundancy and stability.
- **Minimum k-core value = 2 (1 node):** Only one node lies at the shallowest level, meaning it is highly peripheral and vulnerable to isolation. Its removal would not affect the backbone but highlights the presence of weak outliers.
- **Mean k-core value = 5 (154 nodes):** The average embeddedness is high, with 154 nodes clustered around k-core 5. This shows that most of the network is well integrated, supporting resilience.
- **Most frequent k-core value = 6 (225 nodes):** The dominant structural layer is the 6-core, which contains the largest group of nodes. This confirms that the ER network's connectivity is broadly distributed across a strong backbone rather than concentrated in a few hubs.

Metric	Value	Node Count	Interpretation
Max k-core value	6	225	Deep backbone; highly resilient nodes
Min k-core value	2	1	Single peripheral node; vulnerable to isolation
Mean k-core value	5	154	Average embeddedness; majority well integrated
Most frequent k-core	6	225	Dominant layer; backbone widely shared

This ER profile shows a **robust backbone (k=6)** with most nodes embedded deeply, only a single weak peripheral node, and strong redundancy overall.

SW Network k-Core Analysis Nodes = 400 , m = 4 , p = 0.1

- **Maximum k-core value = 14 (4 nodes):** A small set of nodes reach the deepest structural layer, forming the strongest backbone of the SW network. These 4 nodes are highly embedded and provide critical redundancy.
- **Minimum k-core value = 12 (1 node count = 257):** The lowest k-core level is still relatively high (12), showing that even peripheral nodes are well connected. With 257 nodes at this level, the network maintains strong uniform embeddedness.
- **Mean k-core value = 12 (257 nodes):** The average embeddedness aligns with the minimum, meaning most nodes cluster around k-core 12. This reflects the SW property of dense local neighborhoods reinforced by shortcut links.
- **Most frequent k-core value = 12 (257 nodes):** The dominant structural layer is k-core 12, containing the majority of nodes. This confirms that the SW topology distributes resilience broadly, with most nodes sharing a strong level of embeddedness.

Metric	Value	Node Count	Interpretation
Max k-core value	14	4	Deepest backbone; highly embedded critical nodes
Min k-core value	12	257	Even periphery is strongly connected
Mean k-core value	12	257	Average embeddedness; majority clustered at k=12
Most frequent k-core	12	257	Dominant layer; resilience broadly distributed

This SW profile shows **uniformly high embeddedness**, with most nodes at k=12 and a small backbone at k=14. Unlike BA or PLC, resilience is not concentrated in a few hubs but spread across many nodes, making SW networks structurally robust.

BA Network k-Core Analysis Nodes = 400 , m = 4

- **Maximum k-core value = 10 (59 nodes):** A relatively small group of nodes reach the deepest structural layer. These 59 hubs form the backbone of the BA network, reflecting the scale-free property where a few nodes are highly embedded and critical for resilience.
- **Minimum k-core value = 8 (341 nodes):** The majority of nodes lie at the shallowest level of embeddedness. With 341 nodes at k-core 8, most of the network is peripheral, connected through hubs but not deeply redundant.

- **Mean k-core value = 8 (341 nodes):** The average embeddedness aligns with the minimum, showing that most nodes cluster around k-core 8. This indicates limited redundancy for the majority of the network.
- **Most frequent k-core value = 8 (341 nodes):** The dominant structural layer is k-core 8, containing the largest group of nodes. This confirms the BA topology's hub-and-spoke nature: a few nodes are deeply embedded, while most remain shallow.

Metric	Value	Node Count	Interpretation
Max k-core value	10	59	Deep backbone hubs; critical for resilience
Min k-core value	8	341	Majority peripheral; shallow embeddedness
Mean k-core value	8	341	Average matches periphery; limited redundancy
Most frequent k-core	8	341	Dominant layer; hub-and-spoke structure confirmed

This BA profile shows **hub concentration with shallow periphery**: resilience depends heavily on the 59 backbone hubs, while most nodes remain weakly embedded.

CapGrid Network k-Core Analysis Nodes = 400 , m = 4

- **Maximum k-core value = 14 (4 nodes):** A small backbone exists at the deepest structural layer, with 4 nodes embedded at k-core 14. These represent the most resilient substations, carrying the highest redundancy and capacity integration.
- **Minimum k-core value = 8 (389 nodes):** The majority of nodes lie at the shallowest level of embeddedness. With 389 nodes at k-core 8, most of the network is peripheral, but still structurally stronger than BA or PLC peripheries, since the minimum is relatively high.
- **Mean k-core value = 8 (389 nodes):** The average embeddedness aligns with the minimum, showing that most nodes cluster at k-core 8. This reflects CapGrid's design principle: capacity-driven expansion produces a broad, stable base layer.
- **Most frequent k-core value = 8 (389 nodes):** The dominant structural layer is k-core 8, containing the largest group of nodes. This confirms that CapGrid distributes resilience across a wide base, with only a small set of nodes forming the deepest backbone.

Metric	Value	Node Count	Interpretation
Max k-core value	14	4	Deep backbone; highly resilient capacity hubs
Min k-core value	8	389	Majority peripheral; shallow but structurally stable
Mean k-core value	8	389	Average matches periphery; broad base layer
Most frequent k-core	8	389	Dominant layer; resilience distributed across many nodes

This CapGrid profile shows **broad resilience at k=8** with a small but strong backbone at k=14. Unlike BA or PLC, where resilience is concentrated in a few hubs, CapGrid spreads structural strength across a wide base, aligning with its **capacity-driven design philosophy**.

Ring Lattice Network k-Core Analysis nodes = 400 , m = 4

- **Maximum k-core value = 16 (all nodes):** Every node in the ring lattice is embedded at the same depth, k-core 16. This reflects the perfectly regular structure of the lattice, where each node has identical degree and redundancy.
- **Minimum k-core value = 16 (all nodes):** There are no peripheral nodes in this topology. Even the “least connected” nodes share the same embeddedness as the backbone.
- **Mean k-core value = 16 (all nodes):** The average equals the maximum and minimum, confirming uniformity across the entire network.
- **Most frequent k-core value = 16 (all nodes):** The dominant structural layer is k=16, and it includes every node. This shows that resilience is evenly distributed, with no structural hierarchy.

Metric	Value	Node Count	Interpretation
Max k-core value	16	All	Uniform backbone; every node equally embedded
Min k-core value	16	All	No periphery; all nodes structurally identical
Mean k-core value	16	All	Average equals max/min; perfect uniformity
Most frequent k-core	16	All	Entire network shares the same structural layer

This Ring Lattice profile shows **perfect uniformity**: every node is equally embedded, with no distinction between backbone and periphery. While this is mathematically elegant, it is unrealistic for real power grids, which require hierarchical differentiation for efficiency and resilience.

PLC Network k-Core Analysis , Nodes = 400 , m = 4 , p = 0.1

- **Maximum k-core value = 12 (2 nodes):** Only two nodes reach the deepest structural layer. These represent highly embedded hubs, critical for maintaining backbone resilience. Their scarcity highlights fragility: if either fails, the deepest redundancy collapses.
- **Minimum k-core value = 8 (129 nodes):** A significant number of nodes lie at the shallowest level of embeddedness. These peripheral nodes are weakly integrated and vulnerable to isolation, reflecting the clustered but hub-dominated nature of PLC networks.
- **Mean k-core value = 9 (0 nodes):** The rounded mean does not correspond to any actual node count, showing that the distribution is skewed. Most nodes are either below ($k=8$) or above ($k=10$), with few at the exact mean.
- **Most frequent k-core value = 10 (269 nodes):** The dominant structural layer is k-core 10, containing the majority of nodes. This confirms that PLC networks distribute resilience across mid-level clusters, but still rely on a few deep hubs for backbone strength.

Metric	Value	Node Count	Interpretation
Max k-core value	12	2	Deepest hubs; critical but scarce
Min k-core value	8	129	Peripheral nodes; vulnerable to isolation
Mean k-core value	9	0	Average embeddedness; skewed distribution
Most frequent k-core	10	269	Dominant layer; majority clustered at mid-level

This PLC profile shows **hub-dominated resilience with clustered redundancy**: most nodes sit at $k=10$, a few hubs reach $k=12$, and many remain shallow at $k=8$. The structure is more resilient than BA (due to clustering) but still fragile if the deepest hubs fail.

RGG Network k-Core Analysis

- **Maximum k-core value = 10 (28 nodes):** A modest backbone exists at depth 10, with 28 nodes forming the most resilient structural layer. These nodes are embedded in dense spatial clusters, acting as local hubs that stabilize connectivity.
- **Minimum k-core value = 3 (5 nodes):** A handful of nodes lie at the shallowest level, reflecting boundary or isolated positions in the geometric layout. These are structurally fragile and vulnerable to disconnection.
- **Mean k-core value = 7 (172 nodes):** The average embeddedness is moderate, with 172 nodes clustered around k-core 7. This shows that most of the network is reasonably well integrated, balancing backbone strength with peripheral fragility.

- **Most frequent k-core value = 7 (172 nodes):** The dominant structural layer is k-core 7, containing the largest group of nodes. This confirms that resilience in RGG networks is distributed across mid-level clusters rather than concentrated in a few deep hubs.

Metric	Value	Node Count	Interpretation
Max k-core value	10	28	Dense cluster hubs; modest backbone strength
Min k-core value	3	5	Boundary/peripheral nodes; vulnerable to isolation
Mean k-core value	7	172	Average embeddedness; majority moderately integrated
Most frequent k-core	7	172	Dominant layer; resilience spread across mid-clusters

This RGG profile shows **cluster-driven resilience**: most nodes sit at k=7, a modest backbone exists at k=10, and only a few weak nodes remain at k=3. Unlike BA or PLC, resilience is not hub-dominated but distributed across spatial clusters.

GTG Network k-Core Analysis

- **Maximum k-core value = 10 (43 nodes):** A modest backbone exists at depth 10, with 43 nodes forming the most resilient structural layer. These nodes represent dense threshold clusters where connectivity is strongest.
- **Minimum k-core value = 2 (3 nodes):** A very small set of nodes lie at the shallowest level, reflecting weakly connected or marginal threshold positions. These are structurally fragile and vulnerable to isolation.
- **Mean k-core value = 8 (165 nodes):** The average embeddedness is relatively high, with 165 nodes clustered around k-core 8. This shows that most of the network is well integrated, balancing backbone strength with peripheral fragility.
- **Most frequent k-core value = 8 (165 nodes):** The dominant structural layer is k-core 8, containing the largest group of nodes. This confirms that resilience in GTG networks is broadly distributed across mid-level clusters, rather than concentrated in a few deep hubs.

Metric	Value	Node Count	Interpretation
Max k-core value	10	43	Dense threshold clusters; modest backbone strength
Min k-core value	2	3	Marginal nodes; structurally fragile and easily isolated
Mean k-core value	8	165	Average embeddedness; majority moderately integrated
Most frequent k-core	8	165	Dominant layer; resilience spread across mid-clusters

This GTG profile shows **balanced resilience**: most nodes sit at $k=8$, a modest backbone exists at $k=10$, and only a few weak nodes remain at $k=2$. Compared to RGG, GTG has slightly deeper backbone nodes but also a few more fragile outliers, reflecting the threshold-driven connectivity rule.

The k-core results reveal distinct structural signatures for each canonical network. **ER networks** show a strong backbone at $k=6$ with most nodes embedded deeply, reflecting random but resilient connectivity. **SW networks** exhibit uniformly high embeddedness, with nearly all nodes clustered at $k=12$ and a small backbone at $k=14$, confirming their balance of local clustering and global shortcuts. **BA networks** highlight scale-free fragility: a few hubs reach $k=10$, but the majority remain shallow at $k=8$, making resilience hub-dependent. **Ring Lattices** are perfectly uniform, with all nodes at $k=16$, showing mathematical symmetry but unrealistic homogeneity for real grids. **PLC networks** combine clustering with hub dominance: most nodes sit at $k=10$, a few hubs reach $k=12$, and many remain shallow at $k=8$, offering more redundancy than BA but still vulnerable to hub loss. **RGG networks** distribute resilience across spatial clusters, with most nodes at $k=7$, a modest backbone at $k=10$, and a few weak boundary nodes at $k=3$. **GTG networks** show similar mid-level resilience, with most nodes at $k=8$, a backbone at $k=10$, and a handful of fragile outliers at $k=2$, reflecting threshold-driven connectivity. Finally, **CapGrid** demonstrates its design philosophy: a broad base at $k=8$ with nearly 400 nodes, and a small but strong backbone at $k=14$, spreading resilience widely while maintaining deep redundancy.

Model	Max k-core (Value, Count)	Min k-core (Value, Count)	Mean k-core (Value, Count)	Most Frequent k-core (Value, Count)	Structural Interpretation
ER	6 (225)	2 (1)	5 (154)	6 (225)	Strong backbone, single weak outlier
SW	14 (4)	12 (257)	12 (257)	12 (257)	Uniformly high embeddedness, broad resilience
BA	10 (59)	8 (341)	8 (341)	8 (341)	Hub-dominated, majority shallow periphery
Ring Lattice	16 (All)	16 (All)	16 (All)	16 (All)	Perfect uniformity, unrealistic homogeneity
PLC	12 (2)	8 (129)	9 (0)	10 (269)	Clustered hubs, skewed distribution
RGG	10 (28)	3 (5)	7 (172)	7 (172)	Cluster-driven resilience, few weak boundary nodes
GTG	10 (43)	2 (3)	8 (165)	8 (165)	Threshold-driven, balanced but with fragile outliers
CapGrid	14 (4)	8 (389)	8 (389)	8 (389)	Broad base resilience, small deep backbone

Operator Takeaway

- **Uniform resilience:** Ring Lattice, SW → every node highly embedded.
- **Hub-dominated resilience:** BA, PLC → few deep hubs, many shallow nodes.
- **Cluster-driven resilience:** RGG, GTG → resilience spread across mid-clusters, with some weak outliers.
- **Capacity-driven resilience:** CapGrid → broad base at k=8 plus small backbone at k=14, balancing redundancy and depth.
- **Random resilience:** ER → strong backbone with one weak outlier.

spectral Radius Index

- The **spectral radius** of a graph is the largest eigenvalue ($\lambda_{\max}$) of its adjacency matrix.
- Mathematically:

$$\rho(G) = \max |\lambda_i|$$

where λ_i are the eigenvalues of the adjacency matrix A .

- Conceptually:
 - It reflects the **overall connectivity intensity** of the network.
 - A higher spectral radius means denser or more influential connections.
 - It is closely tied to **network robustness, epidemic thresholds, and synchronization capacity**.

Why Spectral Radius Matters for Power Grids

- **Grid stability:** The spectral radius indicates how strongly nodes are interconnected. A high value suggests strong coupling, which can improve resilience but also increase vulnerability to cascading failures.
- **Critical thresholds:** In epidemic or diffusion models, the inverse of the spectral radius often defines the critical threshold for spreading processes. In grids, this analogy applies to fault propagation or oscillatory instabilities.
- **Operator insight:**
 - Comparing spectral radius across models shows which topologies are more tightly coupled.
 - Helps balance between **robust backbone** (high spectral radius) and **controlled vulnerability** (moderate spectral radius).
 - Can be benchmarked alongside eigenvector centrality and k-core to give a three-dimensional view: *influence, embeddedness, and coupling strength*.

The spectral radius ranks coupling intensity as CapGrid » GTG > BA ≈ RGG ≈ PLC » SW ≈ ER > Ring lattice. **CapGrid's** exceptionally high value reflects deep backbone plus a wide base, demanding robust sectionalization and protective coordination. **GTG** forms dense cores that push coupling beyond hub-dominated BA/PLC and cluster-driven RGG, increasing propagation speed. **BA/PLC/RGG** share high

radii from hubs or clusters, concentrating risk in central structures and boundaries. **SW/ER** sit in a moderate band—resilient yet manageable. **Ring lattice** is lowest, with uniformity limiting diffusion and cascade potential.

The **largest eigenvalue of a network’s adjacency matrix** — the spectral radius — is a critical diagnostic because it captures the **intensity of global connectivity** in a single scalar. Unlike local measures such as degree or clustering, the spectral radius reflects how strongly the entire network is coupled through its densest pathways. In practical terms, it sets thresholds for dynamic processes: in epidemic or fault-propagation models, the inverse of the largest eigenvalue often defines the critical point at which disturbances spread uncontrollably. A high spectral radius signals that the network can synchronize rapidly and diffuse information or energy efficiently, but it also means that failures can cascade more easily. For operators, monitoring the spectral radius provides a direct measure of **backbone strength versus vulnerability**, helping balance resilience with controllability.

Eigenvalues and Their Importance

Eigenvalues are fundamental quantities that describe how a system behaves under transformation. In networks, they emerge from matrices such as the adjacency matrix (capturing direct connectivity) and the Laplacian matrix (capturing flow and balance). The **largest eigenvalue** (spectral radius) reflects the strongest coupling intensity — essentially how tightly the network is bound together. It governs thresholds for diffusion, synchronization, and cascade propagation. A high spectral radius means fast information or energy spread, but also lower tolerance to disturbances.

Fiedler Value (Algebraic Connectivity)

The **Fiedler value** is the second-smallest eigenvalue of the Laplacian matrix. It measures **algebraic connectivity** — how well the network holds together as a whole.

- **High Fiedler value:** Strong global cohesion, robust against fragmentation, easier to synchronize.
- **Low Fiedler value:** Weak connectivity, vulnerable to splitting into disconnected components, slower recovery after faults.

Mathematically:

$$\lambda_2(L) = \text{Fiedler value}$$

where $L = D - A$ is the Laplacian (degree matrix minus adjacency).

Why Both Matter Together

- **Spectral radius (largest eigenvalue of adjacency):** Tells us how strongly the network can propagate signals or failures.

- **Fiedler value (second-smallest Laplacian eigenvalue):** Tells us how well the network resists fragmentation and maintains cohesion.

Together, they provide a **two-dimensional resilience profile**:

- **High spectral radius + high Fiedler value:** Strong backbone, resilient and cohesive, but must guard against fast cascades (e.g., CapGrid).
- **High spectral radius + low Fiedler value:** Hub-dominated or clustered networks — powerful but fragile if hubs or bridges fail (e.g., BA, PLC, RGG).
- **Low spectral radius + high Fiedler value:** Uniform but modest coupling — stable, slower propagation (e.g., Ring Lattice, SW).
- **Low spectral radius + low Fiedler value:** Weak networks, vulnerable to both fragmentation and slow recovery (rare in engineered grids).

Interpreting the Fiedler Value (Algebraic Connectivity)

- The **Fiedler value** (second-smallest Laplacian eigenvalue) measures **how well the network holds together globally**.
- **Bigger is better** in terms of cohesion:
 - **High Fiedler value:** Strong global connectivity, robust against fragmentation, easier synchronization.
 - **Low Fiedler value:** Fragile cohesion, vulnerable to splitting into disconnected components, slower recovery after faults.

Model	Spectral radius	Fiedler value (λ_2)	Coupling profile	Cohesion profile
ER	9.731	1.480	Moderate, distributed	Moderate cohesion
SW	9.773	1.052	Moderate with shortcuts	Modest cohesion; bridge-sensitive
Ring lattice	8.000	0.015	Low, uniform	Extremely fragile cohesion
BA	15.593	4.140	High, hub-dominated	Strong cohesion via hubs
PLC	15.542	4.614	High, clustered hubs	Very strong cohesion

RGG	15.589	0.331	High, dense local clusters	Weak global cohesion; boundary-fragile
GTG	18.146	0.474	Very high, threshold cores	Low-moderate cohesion; bottleneck-sensitive
CapGrid	34.121	1.307	Extremely high, deep backbone	Modest cohesion; corridor-sensitive

Edge Betweenness Concentration (EBC)

Structural Fragility Metric for Network Topology

Edge Betweenness Concentration (EBC) is a structural index that measures how strongly a network depends on a small number of critical edges to carry most shortest-path traffic. It is a purely topological metric and does not depend on electrical properties, loads, or generation.

High EBC indicates that the network contains “bottleneck” edges that carry a disproportionate amount of flow. When such edges fail, the network fragments rapidly, often causing early collapse during OPA blackout simulations.

Low EBC indicates that shortest paths are distributed more evenly across the network, resulting in higher robustness and slower cascade progression.

Definition

For each edge e , compute its edge betweenness b_e :

- b_e = number of shortest paths in the network that pass through edge e

Then compute:

- $b_{\max}$ = maximum edge betweenness
- $\bar{b}$ = average edge betweenness

The Edge Betweenness Concentration is:

$$EBC = \frac{b_{\max}}{\bar{b}}$$

Interpretation

EBC Value	Meaning	Expected OPA Behavior
2–5	Highly meshed, no dominant bottlenecks	Very robust, slow cascade
5–12	Moderate bottlenecks	Cascades develop gradually
12–18	Strong bottlenecks	Early fragmentation likely
18–25+	Severe bottlenecks, geometric corridors	Collapse in first few OPA steps

EBC is one of the strongest predictors of OPA fragility:

- Networks with **high EBC** (RGG, Ring-RGG, GTG) collapse early because a few edges carry most shortest paths. When OPA redispatch increases loading, these edges overload and trip, fragmenting the network into islands with poor P/Q balance.
- Networks with **low EBC** (CapGrid, ER, PLC) distribute flows more evenly, allowing OPA to run many steps before collapse.

EBC provides a structural explanation for why geometric networks behave differently from engineered or small-world networks.

Typical EBC Values Observed in pbsHMI

Network Type	EBC
CapGrid	2.8
ER	4.17
PLC	4.51
SW	10.44
SA	15
GTG	18.12
RGG	19.93
Ring-RGG	25

These values align closely with observed OPA survivability.

EBC measures how strongly the network depends on a small number of edges to carry shortest-path traffic. A **low EBC** means shortest paths are spread across many edges, with no dominant bottlenecks. This is a hallmark of a well-meshed, engineered grid. A **high EBC** means a few edges act as structural bottlenecks; when they fail, the network fragments quickly, leading to early collapse in OPA simulations.