# senstoolkit

Sensitivity Analysis Toolkit — User Manual

Version 0.1.0

February 21, 2026

## Contents

## 1 Introduction

senstoolkit is a Python package for performing comprehensive sensitivity analysis on simulation or experimental data. It covers the full workflow from experimental design through multi-method analysis:

1. **Design of Experiments (DOE)** — generate space-filling Sobol quasi-random samples over user-defined parameter ranges.
2. **Multi-method sensitivity analysis** — run 12 complementary analysis methods on the filled DOE, producing CSV tables and publication-ready plots.

The toolkit is fully generic: it works with arbitrary parameter names, arbitrary numbers of parameters, and arbitrary response (output) variables. No domain-specific defaults are hard-coded.

## 2 Installation

### 2.1 Basic Installation

```
cd sensitivity_analysis/
pip install -e .
```

This installs the core dependencies: `numpy`, `pandas`, `scipy`, `scikit-learn`, and `matplotlib`.

### 2.2 Full Installation (Recommended)

To enable all analysis methods (XGBoost surrogate, SHAP, Sobol via SALib, Morris screening):

```
pip install -e ".[full]"
```

This additionally installs `xgboost`, `shap`, and `SALib`.

### 2.3 Optional Dependencies and Feature Gating

| **Package** | **Features enabled** | **Fallback** |
|---|---|---|
| `xgboost` | XGBoost surrogate, gain importance, SHAP | `HistGradientBoostingRegressor` |
| `shap` | SHAP values, SHAP interactions | Skipped |
| `SALib` | Sobol indices (SALib), Morris screening | Manual Sobol / skipped |

Table 1: Optional dependencies and their fallback behaviour.

## 3 Quick Start

A typical workflow consists of four steps:

1. **Create a parameter template** and edit it to define your parameters.
2. **Generate a DOE** (Design of Experiments).
3. **Run your simulation** to fill in the response columns.
4. **Run the analysis** to obtain sensitivity results.

## 3.1 Step 1: Create a Parameter File

```
senstoolkit template --output my_params.json
```

This creates a JSON file with example parameters. Edit it to define your own:

```
{
  "temperature": {"min": 300, "max": 800, "scale": "linear"},
  "pressure":   {"min": 0.1, "max": 100, "scale": "log"},
  "flow_rate":  {"min": 0.5, "max": 5.0, "scale": "linear"}
}
```

Each parameter requires:

- `min`, `max` — the lower and upper bounds.
- `scale` — `"linear"` (uniform sampling) or `"log"` (log-uniform sampling; requires `min > 0`).

## 3.2 Step 2: Generate a DOE

```
senstoolkit design --params my_params.json \
                  --n-samples 128 \
                  --seed 42 \
                  --output doe.csv \
                  --response-cols yield efficiency
```

This produces:

- `doe.csv` — a CSV with columns: `id`, parameter columns, and empty response columns (`yield`, `efficiency`).
- `doe.params.json` — a sidecar file storing the parameter bounds, scales, seed, and sample count (needed for extending and analysing).

## 3.3 Step 3: Fill in the Response Values

Open `doe.csv` in a spreadsheet or run your simulation code to populate the `yield` and `efficiency` columns with the corresponding output values for each parameter combination.

## 3.4 Step 4: Run the Analysis

```
senstoolkit analyze --csv doe.csv \
                   --response-cols yield efficiency \
                   --out-dir results/
```

All output files (CSV tables and PNG plots) are written to the `results/` directory.

# 4 Extending an Existing DOE

Sobol sequences are extensible by design. If you need more samples after an initial run, you can extend the DOE without discarding existing simulation results:

```
senstoolkit extend --csv doe.csv --n-new 256
```

This:

- Reads the original seed from the sidecar file (`doe.params.json`).
- Uses `Sobol.fast_forward()` to skip past the existing points.
- Generates 256 new quasi-random samples and appends them to `doe.csv`.
- Preserves all existing rows (including already-filled response values) unchanged.
- Updates the sidecar metadata.

You can also write to a separate file:

```
senstoolkit extend --csv doe.csv --n-new 256 --output doe_extended.csv
```

## 5 Command-Line Interface Reference

### 5.1 `senstoolkit template`

| **Flag** | **Default** | **Description** |
|---|---|---|
| `-output` | `parameters_template.json` | Output file path |

### 5.2 `senstoolkit design`

| **Flag** | **Default** | **Description** |
|---|---|---|
| `-params` | (required) | Path to parameters JSON file |
| `-n-samples` | max(10$p$, 50) | Number of samples |
| `-output` | `DOE_<timestamp>.csv` | Output CSV path |
| `-seed` | None | Random seed for reproducibility |
| `-response-cols` | (empty) | Response column names to add |

### 5.3 `senstoolkit extend`

| **Flag** | **Default** | **Description** |
|---|---|---|
| `-csv` | (required) | Path to the existing DOE CSV |
| `-n-new` | (required) | Number of new samples to add |
| `-output` | (overwrite existing) | Output CSV path |
| `-seed` | (from sidecar) | Override the original seed |

### 5.4 senstoolkit analyze

| Flag | Default | Description |
|---|---|---|
| -csv | (required) | Path to the filled DOE CSV |
| -response-cols | (required) | Response column name(s) |
| -out-dir | outputs | Output directory |
| -seed | 0 | Random seed |
| -cv-folds | 5 | Number of cross-validation folds |
| -r2-threshold | 0.5 | Minimum CV $R^2$ for surrogate methods |
| -no-pdp | | Skip PDP/ICE plots |
| -no-shap | | Skip SHAP analysis |
| -no-group-perm | | Skip grouped permutation importance |
| -no-sobol | | Skip Sobol analysis |
| -no-morris | | Skip Morris screening |
| -no-scatter | | Skip scatter plot grid |

## 6 Python API Reference

All functions are importable from the top-level package:

```python
from senstoolkit import (
    design, extend_design, analyze,
    write_params_template, morris_screening,
    parse_params_json, sobol_sample, apply_scaling, write_doe_csv,
)
```

Submodule imports for advanced use:

```python
from senstoolkit.importance import fit_model, cv_permutation_importance
from senstoolkit.sobol import sobol_on_surrogate
from senstoolkit.correlation import correlation_vector, bootstrap_correlation_ci
from senstoolkit.plotting import scatter_grid, shap_interaction_analysis
```

### 6.1 Key Function Signatures

```python
def design(params_json, n_samples, out_csv,
           seed=None, prefer_power_of_two=False,
           response_cols=()):

def extend_design(existing_csv, n_new,
                  out_csv=None, seed=None):

def analyze(design_csv, response_cols, out_dir="outputs",
            seed=0, cv_folds=5, perm_repeats=20,
            bootstrap_corr=1000, group_corr_threshold=0.9,
            top_k_pdp=6, do_pdp=True, do_shap=True,
            do_group_perm=True, do_sobol=True,
            sobol_samples=2048, do_morris=True,
            do_scatter=True, r2_threshold=0.5):
```

## 7 Output Files Reference

For each response column `<Y>`, the analysis produces the following files in the output directory:

| File | Description |
|---|---|
| `summary.json` | Overall summary: $R^2$ scores, sample counts, surrogate quality |
| `scatter_grid_<Y>.png` | Scatter plot grid of each parameter vs. response |
| `correlations_<Y>.csv` | Spearman & Pearson correlations with bootstrap 95% CIs |
| `corr_spearman_<Y>.png` | Spearman correlation bar chart |
| `corr_pearson_<Y>.png` | Pearson correlation bar chart |
| `perm_cv_<Y>.csv` | Cross-validated permutation importance (mean $R^2$ drop) |
| `pareto_perm_cv_<Y>.png` | Permutation importance bar chart |
| `xgb_gain_<Y>.csv` | XGBoost gain importance values |
| `pareto_gain_<Y>.png` | Gain importance bar chart |
| `perm_groups_<Y>.csv` | Grouped permutation importance |
| `pareto_perm_groups_<Y>.png` | Grouped importance bar chart |
| `feature_corr_<Y>.csv` | Feature-to-feature correlation matrix |
| `morris_<Y>.csv` | Morris $\mu^*$ and $\sigma$ values |
| `morris_mustar_<Y>.png` | Morris $\mu^*$ bar chart |
| `morris_sigma_<Y>.png` | Morris $\sigma$ bar chart |
| `pdp/pdp_<Y>_<param>.png` | PDP/ICE plots for top parameters |
| `shap_bar_<Y>.png` | SHAP mean \|value\| bar chart |
| `shap/shap_dep_<Y>_<param>.png` | SHAP dependence plots |
| `shap_interactions_<Y>.csv` | SHAP interaction matrix |
| `shap_interactions_<Y>.png` | SHAP interaction heatmap |
| `sobol_<Y>.csv` | Sobol $S_1$ and $S_T$ indices |
| `sobol_S1_<Y>.png` | Sobol first-order bar chart |
| `sobol_ST_<Y>.png` | Sobol total-order bar chart |
| `sobol_S2_<Y>.csv` | Sobol second-order interaction matrix |
| `sobol_S2_<Y>.png` | Sobol $S_2$ interaction heatmap |

# 8 Analysis Methods: Background and Interpretation

This section explains each sensitivity analysis method in detail. For each method we describe: (1) what it computes, (2) the mathematical background, (3) the output files, and (4) how to interpret the results.

## 8.1 Scatter Plot Grid

### 8.1.1 What It Does

For each input parameter $x_i$, a scatter plot of $x_i$ versus the response $y$ is produced. All subplots are arranged in a single grid image.

### 8.1.2 Background

This is the simplest and most direct visual method. It makes no assumptions about the relationship between parameters and response. You see the raw data.

### 8.1.3 Interpretation

- **Clear trend** (upward/downward slope, curve) — the parameter has a visible effect on the response.

- **Funnel shape** (variance changes with $x_i$) — the parameter influences not only the mean but also the variability of the response (heteroscedasticity).
- **Flat cloud** (no pattern) — the parameter has little or no influence.
- **Clusters or gaps** — may indicate discrete regimes, thresholds, or nonlinear effects.

#### 8.1.4 Output

`scatter_grid_<Y>.png`

## 8.2 Pearson and Spearman Correlation

### 8.2.1 What It Does

Computes the Pearson and Spearman rank correlation coefficient between each parameter $x_i$ and the response $y$, along with bootstrap 95% confidence intervals.

### 8.2.2 Mathematical Background

**Pearson correlation ($r$)** measures the strength of the *linear* relationship:

$$r_{x_i,y} = \frac{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)(y_k - \bar{y})}{\sqrt{\sum_{k=1}^{n}(x_{ik} - \bar{x}_i)^2 \cdot \sum_{k=1}^{n}(y_k - \bar{y})^2}} \tag{1}$$

$r = +1$ means perfect positive linear relationship; $r = -1$ means perfect negative linear relationship; $r = 0$ means no linear relationship (but there may still be a nonlinear one).

**Spearman correlation ($\rho$)** replaces values by their ranks before computing the Pearson formula. It measures the strength of any *monotonic* relationship (linear or not). If $\rho$ is large but $r$ is small, the relationship is monotonic but nonlinear (e.g., logarithmic).

**Bootstrap confidence intervals.** The toolkit resamples the data (with replacement) 1000 times by default and computes the correlation on each resample. The 2.5th and 97.5th percentiles of the resampled distribution form the 95% CI. If the CI does not include zero, the correlation is statistically significant at the 5% level.

### 8.2.3 Interpretation

- $|r|$ or $|\rho| > 0.7$ — strong relationship.
- $0.3 < |r|$ or $|\rho| \leq 0.7$ — moderate relationship.
- $|r|$ or $|\rho| \leq 0.3$ — weak or no relationship.
- **Sign** indicates direction: positive means $y$ increases as $x_i$ increases.
- If the bootstrap CI includes zero, the correlation is not statistically significant.
- Compare Pearson vs. Spearman: a large difference suggests a nonlinear (but monotonic) effect.

### 8.2.4 Output

`correlations_<Y>.csv`, `corr_spearman_<Y>.png`, `corr_pearson_<Y>.png`

## 8.3 Surrogate Model and Cross-Validation

### 8.3.1 What It Does

Fits a machine learning regression model (XGBoost or HistGradientBoosting) to approximate the mapping $\mathbf{x} \mapsto y$. The model quality is assessed via $k$-fold cross-validation $R^2$.

### 8.3.2 Background

Many sensitivity analysis methods (Sobol, PDP, SHAP, Morris as implemented here) require evaluating the response at arbitrary parameter combinations. Since only the DOE samples have actual simulation results, a *surrogate model* (also called emulator or metamodel) is trained to predict $y$ from $\mathbf{x}$.

The surrogate is an XGBoost gradient-boosted tree ensemble (400 trees, learning rate 0.05, max depth 6). If XGBoost is not installed, scikit-learn's `HistGradientBoostingRegressor` is used as a fallback.

**Cross-validation $R^2$.** The data is split into $k$ folds (default $k = 5$). For each fold, the model is trained on $k - 1$ folds and evaluated on the held-out fold. The $R^2$ score measures how much variance is explained:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{2}$$

$R^2 = 1$ means perfect prediction; $R^2 = 0$ means the model is no better than predicting the mean; $R^2 < 0$ means the model is worse than the mean.

### 8.3.3 Surrogate Quality Gate

If the mean CV $R^2$ falls below the threshold (default 0.5), the toolkit prints a warning and **skips all surrogate-dependent methods** (PDP, SHAP, Sobol, Morris) for that response. This prevents misleading results from a poor surrogate.

**What to do if the quality gate triggers:**

- Add more samples (`senstoolkit extend`).
- Check for outliers or data errors in the response column.
- The non-surrogate methods (correlation, scatter, permutation importance) still run and may provide useful insights.

### 8.3.4 Output

`summary.json` (contains `cv_r2_mean`, `cv_r2_std`, `train_r2`, `surrogate_ok`)

## 8.4 Permutation Importance (Cross-Validated)

### 8.4.1 What It Does

Measures how much the model's predictive performance ($R^2$) drops when each parameter's values are randomly shuffled.

### 8.4.2 Background

Permutation importance (Breiman, 2001) works as follows:

1. Train the model on the training fold and compute the baseline $R^2$ on the validation fold.

2. For each parameter $x_i$: randomly shuffle (permute) the values of $x_i$ in the validation set, re-predict, and measure the new $R^2$.
3. The *importance* of $x_i$ is the drop in $R^2$:

$$\text{Importance}(x_i) = R^2_{\text{baseline}} - R^2_{\text{permuted}} \tag{3}$$

4. Repeat across all CV folds and multiple permutation repeats; report the mean and standard deviation.

This is **model-agnostic**: it measures the importance of a feature to the model's predictions, regardless of the model type.

#### 8.4.3 Interpretation

- **Large positive value**: the parameter is important — shuffling it destroys predictive power.
- **Near zero**: the parameter contributes little; the model can predict almost as well without it.
- **Negative value** (rare): shuffling actually improved the score, which typically indicates noise or overfitting.
- The standard deviation across folds indicates how stable the importance estimate is.

#### 8.4.4 Output

`perm_cv_<Y>.csv`, `pareto_perm_cv_<Y>.png`

### 8.5 XGBoost Gain Importance

#### 8.5.1 What It Does

Reports the total reduction in the loss function ("gain") attributable to each parameter across all splits in all trees of the XGBoost ensemble.

#### 8.5.2 Background

Each time a tree node splits on parameter $x_i$, the split produces a gain: the improvement in the loss function. The *gain importance* of $x_i$ is the sum (or average) of these gains over all trees and all splits involving $x_i$.

This is an **internal metric of the XGBoost model**. It is fast to compute (no re-evaluation needed) but can be biased towards high-cardinality or correlated features.

#### 8.5.3 Interpretation

- Higher gain = more important for the model's predictions.
- Compare with permutation importance: if gain is high but permutation importance is low, the feature may be redundant (another correlated feature carries the same information).
- Only available when XGBoost is the surrogate model.

#### 8.5.4 Output

`xgb_gain_<Y>.csv`, `pareto_gain_<Y>.png`

## 8.6 Grouped Permutation Importance

### 8.6.1 What It Does

Identifies groups of highly correlated parameters (using hierarchical clustering at a threshold of $|\text{corr}| \geq 0.9$) and permutes entire groups together to assess their joint importance.

### 8.6.2 Background

When two parameters $x_i$ and $x_j$ are highly correlated, permuting $x_i$ alone may not cause a large $R^2$ drop because $x_j$ carries similar information. This leads to *underestimation* of their individual importances.

Grouped permutation importance solves this by:

1. Computing the absolute correlation matrix $|C|$ among all parameters.
2. Using agglomerative hierarchical clustering (complete linkage) with distance $d = 1 - |C|$ to form groups of correlated parameters.
3. Permuting all parameters in a group *simultaneously* and measuring the joint $R^2$ drop.

### 8.6.3 Interpretation

- A group with a large importance value means those correlated parameters are jointly important.
- If a group has high grouped importance but each member has low individual importance, the effect is shared among the correlated parameters.
- The `feature_corr_<Y>.csv` file lets you inspect which parameters are correlated.

### 8.6.4 Output

`perm_groups_<Y>.csv`, `pareto_perm_groups_<Y>.png`, `feature_corr_<Y>.csv`

## 8.7 Morris Elementary Effects Screening

### 8.7.1 What It Does

Computes the Morris Elementary Effects (EE) to screen parameters into three categories: negligible, linear/additive, and nonlinear/interactive.

### 8.7.2 Mathematical Background

The Morris method (Morris, 1991) works in a discretized parameter space with $p$ levels. For each parameter $x_i$, an *elementary effect* is:

$$\text{EE}_i = \frac{f(x_1, \ldots, x_i + \Delta, \ldots, x_d) - f(x_1, \ldots, x_i, \ldots, x_d)}{\Delta} \tag{4}$$

where $\Delta$ is a fixed step size. Multiple elementary effects are computed along random *trajectories* through the parameter space.

Two summary statistics are reported:

- $\mu_i^* = \frac{1}{r}\sum_{k=1}^{r} |\text{EE}_i^{(k)}|$ — the mean of the *absolute* elementary effects. Measures overall influence.

- $\sigma_i = \mathrm{std}(\mathrm{EE}_i^{(1)}, \ldots, \mathrm{EE}_i^{(r)})$ — the standard deviation of the elementary effects. Measures nonlinearity and/or interactions.

#### 8.7.3 Interpretation

- **High $\mu^*$, low $\sigma$**: the parameter has a strong, mostly linear/additive effect.
- **High $\mu^*$, high $\sigma$**: the parameter has a strong effect that is nonlinear and/or interacts with other parameters.
- **Low $\mu^*$, low $\sigma$**: the parameter has negligible influence and can potentially be fixed.
- Morris is a *screening method*: it efficiently identifies which parameters matter most, especially useful when the number of parameters is large ($d > 10$) and full Sobol analysis would be too expensive.

#### 8.7.4 Output

`morris_<Y>.csv`, `morris_mustar_<Y>.png`, `morris_sigma_<Y>.png`

**Requires:** SALib package and a sidecar parameter file.

### 8.8 Partial Dependence Plots (PDP) and Individual Conditional Expectation (ICE)

#### 8.8.1 What It Does

Shows how the predicted response changes as one parameter varies, while averaging over (PDP) or showing individual traces for (ICE) the other parameters.

#### 8.8.2 Background

The *partial dependence function* for parameter $x_i$ is:

$$\hat{f}_i(x_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{f}(x_i,\ \mathbf{x}_{-i}^{(k)}) \tag{5}$$

where $\mathbf{x}_{-i}^{(k)}$ denotes all other parameters from the $k$-th sample. In other words: for each value of $x_i$ on a grid, we predict $y$ for every sample (keeping their other parameter values), then average.

**ICE curves** show the individual predictions $\hat{f}(x_i, \mathbf{x}_{-i}^{(k)})$ without averaging, so you can see whether the effect of $x_i$ is the same for all samples or varies (indicating interactions).

#### 8.8.3 Interpretation

- **Steep PDP curve**: the parameter has a strong effect on the response.
- **Flat PDP curve**: the parameter has little effect.
- **Nonlinear PDP curve**: the effect is nonlinear (e.g., saturation, threshold).
- **ICE curves that diverge** (spread apart): the effect of $x_i$ depends on the values of other parameters — an indication of *interactions*.
- **ICE curves that are parallel**: no interactions; the effect of $x_i$ is the same regardless of other parameter values.

PDPs are generated for the top-$k$ most important parameters (by permutation importance).

### 8.8.4 Output

`pdp/pdp_<Y>_<param>.png` (one plot per top parameter)

## 8.9 SHAP (SHapley Additive exPlanations)

### 8.9.1 What It Does

Computes SHAP values for each sample and each parameter, quantifying each parameter's contribution to the prediction.

### 8.9.2 Mathematical Background

SHAP values are based on Shapley values from cooperative game theory (Shapley, 1953; Lundberg & Lee, 2017). For each sample $k$ and parameter $i$, the SHAP value $\phi_i^{(k)}$ satisfies:

$$\hat{f}(\mathbf{x}^{(k)}) = \phi_0 + \sum_{i=1}^{d} \phi_i^{(k)} \tag{6}$$

where $\phi_0$ is the average prediction. The SHAP value $\phi_i^{(k)}$ is the parameter's *marginal contribution*, averaged over all possible orderings of parameters.

`senstoolkit` uses `TreeExplainer` for tree-based models, which computes exact SHAP values in polynomial time.

**Global importance.** The mean absolute SHAP value $\overline{|\phi_i|}$ across all samples measures the parameter's global importance.

**Dependence plots.** Plotting $\phi_i^{(k)}$ vs. $x_i^{(k)}$ reveals how the parameter's effect varies across its range.

### 8.9.3 Interpretation

- **Mean $|\phi_i|$ bar chart**: ranks parameters by global importance (analogous to permutation importance but exact for the model).
- **Dependence plot with a clear trend**: the parameter has a consistent directional effect.
- **Dependence plot with vertical spread at each $x_i$ value**: interactions with other parameters.
- **Red dashed lines** at $\pm 0.5\sigma_y$: SHAP values exceeding half the response standard deviation indicate a practically significant contribution.
- SHAP values sum to the prediction, so they are *additive* and directly comparable across parameters.

### 8.9.4 Output

`shap_bar_<Y>.png`, `shap/shap_dep_<Y>_<param>.png`

**Requires:** `shap` and `xgboost` packages.

## 8.10 SHAP Interaction Values

### 8.10.1 What It Does

Computes the pairwise SHAP interaction values, producing a $d \times d$ matrix showing how pairs of parameters jointly influence the prediction.

### 8.10.2 Background

SHAP interaction values decompose the SHAP value into main effects and pairwise interactions:

$$\phi_i^{(k)} = \phi_{ii}^{(k)} + \sum_{j \neq i} \phi_{ij}^{(k)} \tag{7}$$

The diagonal $\phi_{ii}$ captures the main effect of parameter $i$; the off-diagonal $\phi_{ij}$ captures the interaction between parameters $i$ and $j$.

### 8.10.3 Interpretation

- **Large diagonal values**: the parameter has a strong main effect.
- **Large off-diagonal values**: the two parameters interact — the effect of one depends on the value of the other.
- The heatmap makes it easy to spot which pairs interact most strongly.
- Useful for identifying which parameter combinations should be explored further (e.g., in 2D PDP or targeted experiments).

### 8.10.4 Output

`shap_interactions_<Y>.csv`, `shap_interactions_<Y>.png`

**Requires:** `shap` and `xgboost` packages.

## 8.11 Sobol Sensitivity Indices

### 8.11.1 What It Does

Decomposes the total variance of the response into contributions from individual parameters (first-order, $S_1$), total effects including all interactions (total-order, $S_T$), and pairwise interactions (second-order, $S_2$).

### 8.11.2 Mathematical Background

Sobol indices (Sobol', 1993) are based on the ANOVA-like decomposition of a function:

$$f(\mathbf{x}) = f_0 + \sum_i f_i(x_i) + \sum_{i<j} f_{ij}(x_i, x_j) + \cdots \tag{8}$$

**First-order index $S_{1,i}$:**

$$S_{1,i} = \frac{\mathrm{Var}_{x_i}[\mathbb{E}_{\mathbf{x}_{\sim i}}(y \mid x_i)]}{\mathrm{Var}(y)} \tag{9}$$

This is the fraction of total variance explained by $x_i$ *alone*, not considering interactions.

**Total-order index $S_{T,i}$:**

$$S_{T,i} = 1 - \frac{\mathrm{Var}_{\mathbf{x}_{\sim i}}[\mathbb{E}_{x_i}(y \mid \mathbf{x}_{\sim i})]}{\mathrm{Var}(y)} \tag{10}$$

This is the fraction of total variance explained by $x_i$ *and all its interactions* with other parameters.

**Second-order index $S_{2,ij}$:**

$$S_{2,ij} = \frac{\mathrm{Var}_{x_i,x_j}\left[\mathbb{E}_{\mathbf{x}_{\sim ij}}(y \mid x_i, x_j)\right]}{\mathrm{Var}(y)} - S_{1,i} - S_{1,j} \tag{11}$$

This captures the variance explained by the *interaction* between $x_i$ and $x_j$ specifically.

**Estimation.** The toolkit uses the Saltelli (2002, 2010) estimator for $S_1$ and the Jansen (1999) estimator for $S_T$. If SALib is installed, it is used for both $S_1/S_T$ and $S_2$; otherwise a manual implementation provides $S_1$ and $S_T$ (with $S_2 = 0$ as a placeholder).

### 8.11.3 Interpretation

- $S_{1,i} \approx S_{T,i}$: the parameter acts mainly alone (no significant interactions).
- $S_{T,i} \gg S_{1,i}$: a large part of the parameter's influence comes through interactions with other parameters.
- $\sum_i S_{1,i} \approx 1$: the model is approximately additive (no interactions).
- $\sum_i S_{1,i} \ll 1$: significant interactions exist.
- $S_{1,i} < 0.05$: the parameter has negligible direct influence.
- $S_{T,i} < 0.05$: the parameter has negligible total influence (including interactions) and can be fixed.
- $S_{2,ij}$ identifies specific interacting pairs.

**Diagnostics.** If $\sum S_{1,i} > 1.05$ or any $S_T < S_1$, the toolkit prints a warning. This usually means the sample count is too low; increase `sobol_samples` or add more DOE points.

### 8.11.4 Output

`sobol_<Y>.csv`, `sobol_S1_<Y>.png`, `sobol_ST_<Y>.png`, `sobol_S2_<Y>.csv`, `sobol_S2_<Y>.png`

## 9 Comparing Methods: When to Use What

| **Method** | **Needs surrogate?** | **Captures nonlinearity?** | **Captures interactions?** | **Best for** |
|---|---|---|---|---|
| Scatter grid | No | Visual | Visual | Initial exploration |
| Pearson corr. | No | No | No | Linear screening |
| Spearman corr. | No | Monotonic | No | Monotonic screening |
| Perm. importance | Yes (CV) | Yes | Indirectly | Overall ranking |
| XGBoost gain | Yes | Yes | No | Quick model-internal check |
| Grouped perm. | Yes (CV) | Yes | Yes (groups) | Correlated parameters |
| Morris screening | Yes | Yes | Yes ($\sigma$) | Efficient screening ($d > 10$) |
| PDP/ICE | Yes | Yes | Yes (ICE) | Shape of effect |
| SHAP | Yes | Yes | Yes (dependence) | Per-sample attribution |
| SHAP interactions | Yes | Yes | Yes (pairwise) | Interaction identification |
| Sobol $S_1/S_T$ | Yes | Yes | Yes ($S_T - S_1$) | Variance decomposition |
| Sobol $S_2$ | Yes | Yes | Yes (pairwise) | Interaction quantification |

Table 3: Comparison of sensitivity analysis methods.

**Recommended workflow:**

1. Start with scatter plots and correlations for a quick visual overview.
2. Check the surrogate $R^2$. If it is high ($> 0.8$), all surrogate-based methods are reliable.
3. Use permutation importance to rank parameters.
4. Use PDP/ICE to understand the *shape* of each important parameter's effect.
5. Use Sobol indices for a rigorous variance decomposition.
6. Use SHAP interactions or Sobol $S_2$ to identify interacting parameter pairs.
7. Use Morris screening as a first pass when you have many parameters ($> 10$).

## 10 Design of Experiments: Background

### 10.1 Sobol Quasi-Random Sequences

A Sobol sequence is a *low-discrepancy* (quasi-random) sequence that fills the parameter space more uniformly than pseudo-random sampling. Key properties:

- **Space-filling**: Sobol sequences avoid the clumping and gaps typical of pseudo-random sampling, especially in high dimensions.
- **Deterministic**: given the same seed and dimensionality, the sequence is exactly reproducible.
- **Extensible**: new points can be added without moving existing ones (used by `extend`).

- **Low discrepancy**: the deviation from uniform coverage decreases as $O((\log n)^d/n)$, much faster than Monte Carlo's $O(1/\sqrt{n})$.

### 10.2 Linear vs. Log Scale

- **Linear**: samples are drawn uniformly between `min` and `max`. Use for parameters where equal absolute changes are equally important (e.g., temperature from 300 to 800 K).
- **Log**: samples are drawn uniformly in log-space, i.e., $x = \exp(\text{Uniform}(\ln(\text{min}), \ln(\text{max})))$. Use for parameters that span multiple orders of magnitude (e.g., diffusion coefficient from $10^{-3}$ to $10^1$).

### 10.3 Sample Size Guidelines

- Minimum: $10 \times d$ samples (where $d$ is the number of parameters).
- Recommended: 50–200 $\times\, d$ for reliable Sobol index estimation.
- The toolkit defaults to $\max(10d, 50)$ if no sample count is specified.
- Use `senstoolkit extend` to add more samples iteratively.

## 11 Troubleshooting

**`surrogate_ok: false`**
The surrogate model's CV $R^2$ is below the threshold. Surrogate-based methods (Sobol, PDP, SHAP, Morris) are skipped. Solutions: add more samples, check for data errors, or lower the threshold via `-r2-threshold`.

**Sobol $\sum S_1 > 1$** Insufficient samples for reliable Sobol estimation. Increase `-n-samples` or `sobol_samples`.

**SHAP/Morris skipped**
The required optional packages (`shap`, `SALib`) are not installed. Install with `pip install senstoolkit[full]`.

**Extend fails: "No seed found"**
The sidecar file was generated before the seed-tracking feature was added. Provide `-seed` manually with the same seed used for the original design.

**No sidecar file found**
Sobol and Morris analyses require parameter bounds from the sidecar `.params.json` file. Re-generate the DOE with `senstoolkit design`, or create the sidecar file manually.

## 12 References

- Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5–32.
- Jansen, M.J.W. (1999). Analysis of variance designs for model output. *Computer Physics Communications*, 117, 35–43.
- Lundberg, S.M. & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *NeurIPS*.
- Morris, M.D. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33(2), 161–174.

- Saltelli, A. (2002). Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145, 280–297.
- Saltelli, A. et al. (2010). Variance based sensitivity analysis of model output. *Computer Physics Communications*, 181, 259–270.
- Shapley, L.S. (1953). A value for $n$-person games. *Contributions to the Theory of Games*, 2, 307–317.
- Sobol', I.M. (1993). Sensitivity estimates for nonlinear mathematical models. *Mathematical Modelling and Computational Experiments*, 1, 407–414.