HAF friction model — program user guide

*Supplementary material. Companion reproducibility package for the article. Two independent implementations are provided; they read the same input file and produce the same outputs (identical file names and numbering), and their results coincide. The guide is split into two self-contained parts: read **Part I** if you use MATLAB/Octave, or **Part II** if you use Python.*

Part I — MATLAB / Octave program (`HAF_model.m`)

I.1. Purpose

`HAF_model.m` is a single, self-contained program that reads the traction dataset and **reproduces the entire quantitative content of the paper**: Tables 1–4 and Figures 1–3, 6–21, together with the complete appendix figure sets A1–A10. All uncertainties are obtained with the residual (BIG) bootstrap. Figures 4–5 of the paper compare bootstrap schemes and are intentionally not reproduced here, since this program implements the BIG scheme only.

I.2. Requirements

MATLAB R2016b or newer, **or** GNU Octave. No toolbox is required. The Excel export (`HAF_tables.xlsx`) additionally requires MATLAB R2019a+ (function `writecell`); it can be switched off (see §I.7) with no other effect.

I.3. Files

Two files, placed in the **same folder**: `HAF_model.m` (the program) and `SRR_frot_data.csv` (the dataset).

I.4. How to run

1. Put `HAF_model.m` and `SRR_frot_data.csv` in one folder.
2. Open MATLAB (or Octave) and set the current directory to that folder.
3. Run the program (type `HAF_model` in the Command Window, or press **Run**).

The program prints its progress, then creates two output folders (see §I.8). Nothing else is required.

I.5. Input data format

`SRR_frot_data.csv` is a semicolon-separated file with one header line and one measurement per row:

```
rugo;pression;temperature;vitesse;SRR;F
rough;1.6;80;20;-18.95;-0.0193
...
```

The six columns are: surface roughness (`rough`/`smooth`, or a number), contact pressure (GPa), temperature (°C), entrainment speed (m/s), slide-to-roll ratio SRR (%), and the measured friction coefficient F. Rows sharing the same first four values form one traction curve; the 14 curves are detected automatically.

I.6. The model

Each traction curve is described by the three-parameter HAF form

$$F(\mathrm{SRR}) = 2\,\nu\left(\frac{1}{1+\exp(-h\,\mathrm{SRR})} + a\,\mathrm{SRR} - 0.5\right),$$

with h (hysteresis), a (attrition) and ν (friction level). The dependence of each coefficient on the operating conditions is described by linear laws (Eqs. 11–13 of the paper), e.g. for h:

$$h = h_i + h_l\,\mathrm{load} + h_t\,\mathrm{temp} + h_s\,\mathrm{speed} + h_r\,\mathrm{rough},$$

and likewise for a and ν. The **complete model** substitutes these laws into the HAF form and estimates the 15 coefficients directly against all friction points by non-linear least squares.

I.7. Options

All settings are gathered in the `opt` structure at the top of the file:

- `opt.dataFile` — input file name (default `SRR_frot_data.csv`).
- `opt.nBootCurve` — per-curve bootstrap resamples (default 1000; up to 100000 for publication-quality intervals).
- `opt.nBootComplete` — bootstrap resamples of the 15-parameter complete model (default 200).
- `opt.refitMode` — `'local'` (fast; refit from the current estimate) or `'grid'` (faithful to the original scheme).
- `opt.zsig` — z-value for the significance flag in the tables (default 1.96).
- `opt.progress` — live progress line during the two bootstraps.
- `opt.makeTables`, `opt.tableDir` — write the CSV tables and their folder.
- `opt.makeXlsx`, `opt.xlsxFile` — also write an Excel workbook (R2019a+).
- `opt.bodyFigs`, `opt.appendixFigs` — produce the body and/or appendix figures.
- `opt.saveFigs`, `opt.figDir`, `opt.figFmt`, `opt.figRes` — save figures, folder, format and resolution.
- `rng(7)` — fixed random seed, ensuring exactly reproducible results.

I.8. Outputs

Two folders are written next to the program.

`HAF_tables/` — the four tables, as CSV (and, if enabled, `HAF_tables.xlsx`): Table 1, per-curve HAF parameters h, a, ν (and SSE) for each of the 14 curves; Table 2, linear laws **per surface** (rough and smooth separately); Table 3, the **generalized** linear law (roughness as a factor); Table 4, the **complete** non-linear model — the 15 coefficients with bootstrap standard deviation and significance.

`HAF_figures/` — figures saved as PNG and BMP. Body: Fig. 1 (HAF simulation), 2 (traction curves), 3 (SSE surfaces), 6 (parameter KDE), 7–8 (friction density), 9 (parameter histograms), 10 (box plots), 11–12 (per-surface coefficients and R^2), 13 (generalized coefficients), 14–15 (complete coefficients and R^2), 16 (modelled vs experimental), 17 (residuals), 18 (reconstructed curves), 19 (prediction density), 20 (predicted curves), 21 (best/mean/worst). Appendix: A1 (SSE surfaces), A2 (bivariate KDE), A3–A4 (friction density) and A10 (reconstruction), one per curve; A5 (histograms), A6 (box plots), A7 (per-surface coefficients), A8 (generalized coefficients, with R^2), A9 (complete coefficients, with R^2), one per factor.

I.9. Run time and reproducibility

Run time is dominated by the two bootstraps. With the default settings the program completes in a few minutes on a standard computer; increasing `nBootCurve` toward 100000 sharpens the intervals at a proportional cost. Because the seed is fixed (`rng(7)`), a given set of options always yields identical numbers and figures.

Part II — Python program (`haf_model.py`)

II.1. Purpose

`haf_model.py` is a single, self-contained program that reads the traction dataset and **reproduces the entire quantitative content of the paper**: Tables 1–4 and Figures 1–3, 6–21, together with the complete appendix figure sets A1–A10. All uncertainties are obtained with the residual (BIG) bootstrap. Figures 4–5 of the paper compare bootstrap schemes and are intentionally not reproduced here, since this program implements the BIG scheme only. It is a direct port of the MATLAB program and yields the same results.

II.2. Requirements

Python 3 with the packages `numpy`, `scipy`, `matplotlib`, `pandas`, `openpyxl` and `pillow`. Install them once with:

```
pip install numpy scipy matplotlib pandas openpyxl pillow
```

II.3. Files

Two files, placed in the **same folder**: `haf_model.py` (the program) and `SRR_frot_data.csv` (the dataset).

II.4. How to run

1. Put `haf_model.py` and `SRR_frot_data.csv` in one folder.
2. Open a terminal in that folder.
3. Run: `python haf_model.py`.

The program prints its progress, then creates two output folders (see §II.8). Figures are rendered off-screen (no display needed). Nothing else is required.

II.5. Input data format

`SRR_frot_data.csv` is a semicolon-separated file with one header line and one measurement per row:

```
rugo;pression;temperature;vitesse;SRR;F
rough;1.6;80;20;-18.95;-0.0193
...
```

The six columns are: surface roughness (`rough`/`smooth`, or a number), contact pressure (GPa), temperature (°C), entrainment speed (m/s), slide-to-roll ratio SRR (%), and the measured friction coefficient F. Rows sharing the same first four values form one traction curve; the 14 curves are detected automatically.

II.6. The model

Each traction curve is described by the three-parameter HAF form

$$F(\mathrm{SRR}) = 2\,\nu\left(\frac{1}{1+\exp(-h\,\mathrm{SRR})} + a\,\mathrm{SRR} - 0.5\right),$$

with h (hysteresis), a (attrition) and ν (friction level). The dependence of each coefficient on the operating conditions is described by linear laws (Eqs. 11–13 of the paper), e.g. for h:

$$h = h_i + h_l\,\mathrm{load} + h_t\,\mathrm{temp} + h_s\,\mathrm{speed} + h_r\,\mathrm{rough},$$

and likewise for a and ν. The **complete model** substitutes these laws into the HAF form and estimates the 15 coefficients directly against all friction points by non-linear least squares.

II.7. Options

All settings are gathered in the `OPT` dictionary at the top of the file (same key names as the MATLAB version):

- `dataFile` — input file name (default `SRR_frot_data.csv`).

- `nBootCurve` — per-curve bootstrap resamples (default 1000; up to 100000 for publication-quality intervals).
- `nBootComplete` — bootstrap resamples of the 15-parameter complete model (default 200).
- `refitMode` — `'local'` (fast; refit from the current estimate) or `'grid'` (faithful to the original scheme).
- `zsig` — z-value for the significance flag in the tables (default 1.96).
- `progress` — live progress line during the two bootstraps.
- `makeTables`, `tableDir` — write the CSV tables and their folder.
- `makeXlsx`, `xlsxFile` — also write an Excel workbook (via openpyxl).
- `bodyFigs`, `appendixFigs` — produce the body and/or appendix figures.
- `saveFigs`, `figDir`, `dpi` — save figures, folder and resolution.

The program is deterministic (fixed random seed), so a given set of options always yields identical numbers and figures.

II.8. Outputs

Two folders are written next to the program.

`HAF_tables/` — the four tables, as CSV (and, if enabled, `HAF_tables.xlsx`): Table 1, per-curve HAF parameters h, a, ν (and SSE) for each of the 14 curves; Table 2, linear laws **per surface** (rough and smooth separately); Table 3, the **generalized** linear law (roughness as a factor); Table 4, the **complete** non-linear model — the 15 coefficients with bootstrap standard deviation and significance.

`HAF_figures/` — figures saved as PNG and BMP. Body: Fig. 1 (HAF simulation), 2 (traction curves), 3 (SSE surfaces), 6 (parameter KDE), 7–8 (friction density), 9 (parameter histograms), 10 (box plots), 11–12 (per-surface coefficients and R^2), 13 (generalized coefficients), 14–15 (complete coefficients and R^2), 16 (modelled vs experimental), 17 (residuals), 18 (reconstructed curves), 19 (prediction density), 20 (predicted curves), 21 (best/mean/worst). Appendix: A1 (SSE surfaces), A2 (bivariate KDE), A3–A4 (friction density) and A10 (reconstruction), one per curve; A5 (histograms), A6 (box plots), A7 (per-surface coefficients), A8 (generalized coefficients, with R^2), A9 (complete coefficients, with R^2), one per factor.

II.9. Run time and reproducibility

Run time is dominated by the two bootstraps. With the default settings the program completes in a few minutes on a standard computer; increasing `nBootCurve` toward 100000 sharpens the intervals at a proportional cost. Because the seed is fixed, a given set of options always yields identical numbers and figures.