ADDING THE MLFM TO PVPMC/PVLIB

To be presented at WCPEC8 in Milan Sep 2022

Steve Ransome[1] (SRCL) and
Juergen Sutterlueti (Gantner Instruments)
[1] steve@steveransome.com : www.steveransome.com

Steve Ransome Consulting Limited

3BV.3.14
13:30 - 15:00 Tue 27th Sep 2022
WCPEC #8 Milano, Italy

INTRODUCTION

- The Loss Factors Model (LFM) quantifies independent, normalised losses from IV parameters (e.g. pr_dc, i_sc, r_sc, i_ff, v_ff, r_oc, v_oc, t_mod)
- The Mechanistic Performance Model (MPM) fits normalised, meaningful coefficients to LFM losses vs. poa_global(kW/m^2), temp_mod(C), ws(ms^{-1})

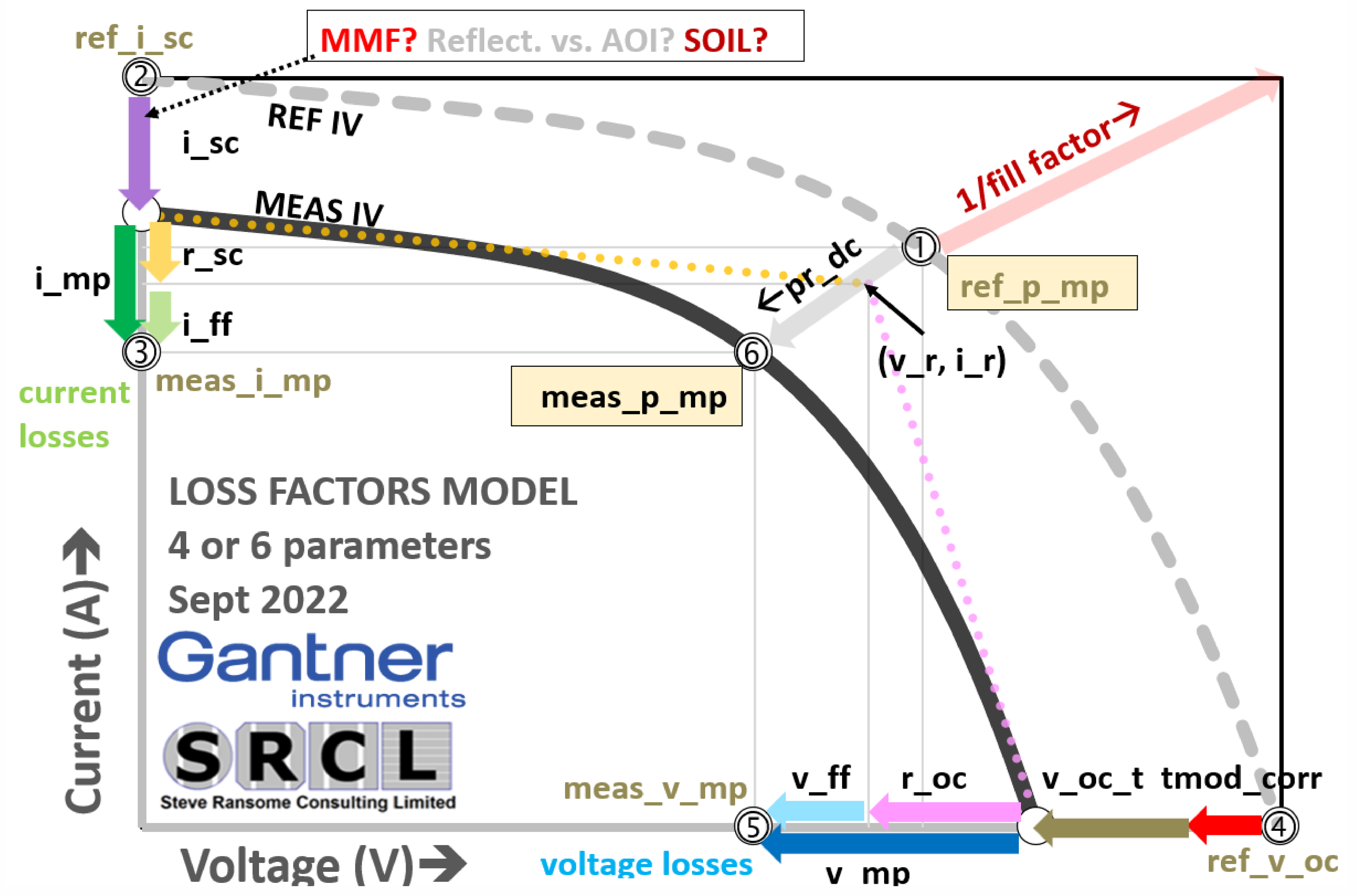
`MPM =c_1 +c_2*(temp_mod-25) +c_3*log10(poa_global) +c_4*poa_global +c_5*ws +…`

A tutorial is being added to PVLIB to :-

- Import weather and IV parameters (outdoor or indoor matrix data)
- Normalise IV parameters vs. datasheet e.g. 'norm_voc =meas_voc/stc_voc'
- Generate normalised LFM losses for all (pr_dc, i_sc, r_sc … v_oc)
- Fit robust, normalised MPM coefficients "c_n" to LFM losses
- Characterise losses, predict performance at all conditions, validate inputs, estimate any degradation rates and causes.

Loss Factors Model

4=matrix (r_sc and r_oc not known), 6=iv_curve (r_sc and r_oc known),



[A] IMPORT MEASUREMENT DATA → meas
weather : poa_global(G) temp_module(T) wind(W)
Matrix (4 pts) : (i_sc , i_mp, v_mp, v_oc) or
IV (6 pts) : (i_sc, r_sc, i_ff, v_ff, r_oc, v_oc)

[B] GET REFERENCE STC VALUES → ref
datasheet values :
ref : i_sc, i_mp, v_mp, v_oc
temp coeffs :
alpha_i_sc, beta_voc, gamma_p_mp

[I] DEFINE MATRIX POINTS → matr
matrix points for fitting include STC, LIC, HTC …
poa_global=(0.1-1.1), temp_module=(10-75C)

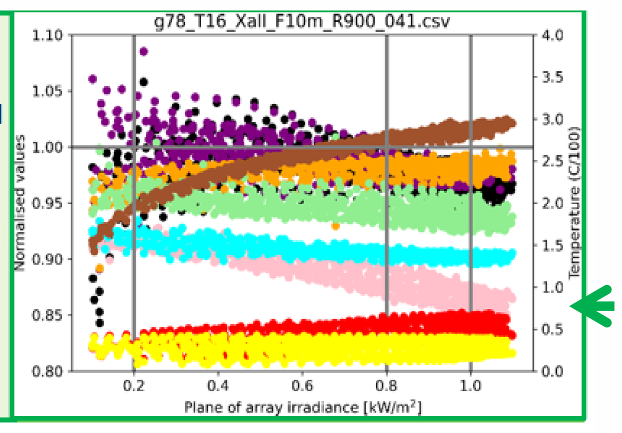
[C] CALCULATE NORMALISED LOSSES → norm
def mlfm_meas_to_norm(dmeas, ref):
e.g.
pr_dc = meas_pmp / ref_pmp / poa_global
norm_voc = meas_voc / ref_voc

[E] PERFORM SANITY CHECK ON DATA
filter by values or stdev
mark outliers, filter them out
erase (or interpolate) missing data

[F] CALC. STACK LOSSES → stack
def mlfm_norm_to_stack(dnorm, ff):
from 1/FF(top) to pr_dc(bottom)
Easier to see loss by parameter
and degradation changes with time

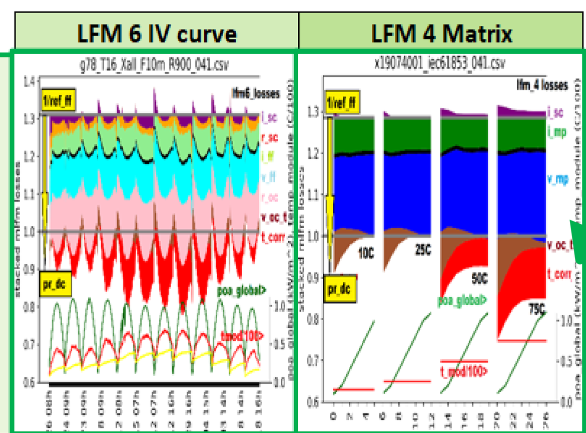
[H] MECHANISTIC FIT TO NORMALISED LOSSES
def mlfm_fit(data, var_to_fit):
fit_loss = fn(poa_global, temp_mod, ws,
c1 … cn)

[D] Normalised loss(↓) vs. poa_global (→)



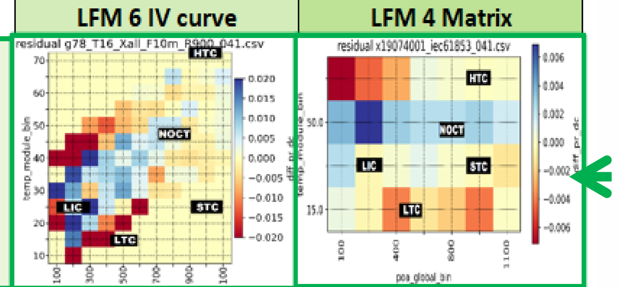
[G] Stacked loss(↓) vs. Time(→) or Meas(→)

LFM 6 IV curve | LFM 4 Matrix



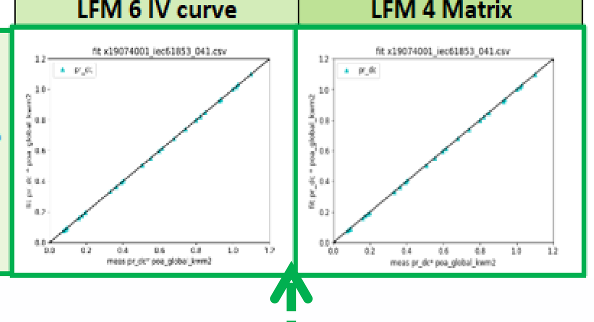
[J] residual heatmap : pr_dc meas-fit (T↑)vs (G→)

LFM 6 IV curve | LFM 4 Matrix



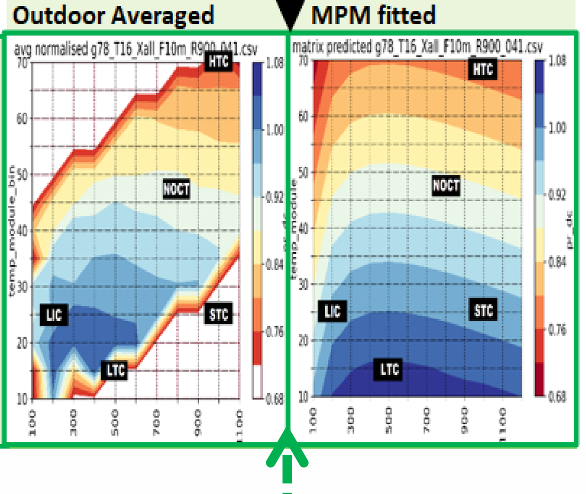
[K] Scatter Fit : norm_pmp Fit(↑) vs. Meas(→)

LFM 6 IV curve | LFM 4 Matrix



[I] Contour maps : pr_dc (colours) vs. temp_mod (↑) poa_global (→)

Outdoor Averaged | MPM fitted



Key : DATA | CALCS | GRAPHS

Flow chart for MLFM tutorial in PVLIB
SRCL/Gantner Instruments
Ver : 220919

```
def mlfm_meas_to_norm(dmeas, ref): # [C]
# pr_dc = 1/ff * product(norm_lfm losses)
# Convert measured power, current and voltage to normalized values.
```

```
# Scatter plot normalised LFM values(↑) vs. irradiance(→) # [D]
fig_scatter = plot_mlfm_scatter(meas, norm, mlfm_meas_file)
```

```
# remove values outside limits e.g. <0.5 or >1.5 # [E]
norm = norm[((norm['pr_dc'] > 0.5) & (norm['pr_dc'] < 1.5))]
# remove all mlfm values outside x~3 stdevs
for lfm in ('i_sc','r_sc','i_ff','v_ff','r_oc','v_oc'):
    norm = norm[((norm[lfm] - norm[lfm].mean()) /
                  norm[lfm].std()).abs() < stdevs
        ]
```

```
def mlfm_norm_to_stack(dnorm, fill_factor): # [F]
# pr_dc = 1/ff - sum(stack_lfm losses)
# Converts normalised values to stacked subtractive normalized losses.
```

```
#Stacked losses(↑) vs. date and time (or matrix measurement)(→) # [G]
fig_stack = plot_mlfm_stack(
    dmeas=meas, dnorm=norm, dstack=stack, # dataframes measurements
    xaxis_labels=12,                      # show #x_labels or 0=show all
    is_i_sc_self_ref=False,               # isc self referenced?
    is_v_oc_temp_module_corr=True,        # voc temperature corrected?
)
```

```
def mlfm_6(dmeas, c_1, c_2, c_3, c_4, c_5=0., c_6=0.): # [H]
# Predict normalised LFM values from data in ``dmeas``.
# Normalized LFM values are given by
# c_1 + c_2*(T_m - 25) + c_3/log10(G_POA) + c_4*G_POA + c_5*WS + …
```

```
# Sample Python code
```

```
# Measured vs. MPM Fitted vs. poa_global(→),
# temp_module(↑) # [I]
Contour_plot = plot_contourf(
    df=matr2,
    y_axis='temp_module', x_axis='poa_global',
    z_axis=mlfm_sel,
    title='matrix predicted ' + mlfm_meas_file,
    vmin=0.7, vmax=1.05,
    levels=9                # num colours
)
```

```
# plot fit(↑) vs. measured(→)
# include a 1:1 line # [K]
fit_plot = plot_fit(
    dmeas=meas,
    dnorm=norm,
    fit=mlfm_sel,
    title='fit ' + mlfm_meas_file
)
```

```
# Residual MLFM fit heatmap(colours)
# vs. poa_global(→)temp_module(↑) # [J]
heatmap_plot = plot_heatmap(
    dnorm=norm, dmeas=meas,
    fit=mlfm_sel,
    y_axis='temp_module_bin', x_axis='poa_global_bin',
    z_axis='diff_' + mlfm_sel,
    title='residual ' + mlfm_meas_file
)
```

How to contribute to PVLIB

INFORMATION :

- **Stack Overflow :** http://stackoverflow.com/questions/tagged/pvlib
- **Google groups :** https://groups.google.com/forum/#!forum/pvlib-python
- **GitHub issues :** https://github.com/pvlib/pvlib-python/issues
- **Pull requests :** https://github.com/pvlib/pvlib-python/pulls
- **Jupyter Notebook tutorials :** https://github.com/pvlib/pvlib-python/tree/master/docs/tutorials

- **Follow PEP8 :** https://www.python.org/dev/peps/pep-0008/. (Max line length 79 chars)
- **Add your project :** https://github.com/pvlib/pvlib-python/wiki/Projects-and-publications-that-use-pvlib-python.
- **Variables and Symbols :** https://pvlib-python.readthedocs.io/en/stable/user_guide/variables_style_rules.html#variables-style-rules
- **Documentation style :** https://pvlib-python.readthedocs.io/en/stable/contributing.html#documentation

Coding :

- **PVSystem** and **Location** classes provide convenience wrappers around the core pvlib functions.
- **Remove logging calls and print statements.**
- **Include documentation and Comprehensive unit tests**
- **Functions must return the desired output** for all inputs see https://github.com/pvlib/pvlib-python/issues/394

PVPMC : https://pvpmc.sandia.gov/
Document Library : https://pvpmc.sandia.gov/resources-and-events/documents/
Code of conduct : https://github.com/pvlib/pvlib-python/blob/master/CODE_OF_CONDUCT.md

SUMMARY

- The MLFM is being added to PVLIB python
- The robust modelling of the LFM and MPM have been described
- These are proving useful in industrial projects from single module OTFs to large power plants
- Information has been given on how other coders can contribute their projects

Further reading :

[EUPVSEC 35 Brussels 2018] Adaptable PV performance modelling
http://www.steveransome.com/PUBS/1809_PVSEC35_Bruss)SRCL_5CV1.28%20paper.pdf

[PVSC 46 Chicago 2019] Checking the new IEC 61853.1-4 with high quality 3rd party data to benchmark its practical relevance in energy yield prediction
http://www.steveransome.com/PUBS/1906_PVSC46_Chicago_Ransome_Presented.pdf

[PVSC 49 Philadelphia 2022] Benchmarking PV performance models with high quality IEC 61853 Matrix measurements (Bilinear interpolation, SAPM, PVGIS, MLFM and 1-diode)
http://www.steveransome.com/pubs/2206_PVSC49_philadelphia_4_presented.pdf

[PVPMC Salt Lake City 2022] IMPROVING ANALYSIS METHODS FOR IEC 61853 MATRIX MEASUREMENTS
https://pvpmc.sandia.gov/download/8494/

Acknowledgements :
Gantner Instruments
https://www.gantner-instruments.com/
CFV for sample data
https://pvpmc.sandia.gov/download/7701/

Steve Ransome Consulting Limited