pytrf

a Python toolbox for the analysis of time series and terrestrial reference frames

Paul Rebischung, Julien Barnéoud, Kevin Gobron, Maylis de La Serve

IGN Institut National de l'Information Géographique et Forestière · Géodata Paris, l'école des ingénieur·es géographes · Université Gustave Eiffel · IPGP · Université Paris Cité

General information

Motivation

- Missing functionalities in existing time series analysis software (CATS, Hector…)
- Develop modern, flexible software to succeed CATREF software for:
  - ITRF combination center
  - IGS terrestrial frame combinations
  - Galileo terrestrial reference frame maintenance

History

- 2011–2019: Progressive buildup of Python2 code in the frame of IGS reference frame activities
- 2020: Organization, complementation, optimization and modernization into Python3 package
  Creation of (private) github repository
- 2021: Extensive use for IGS repro3 SINEX combinations and station position time series analysis
- 2021–now: Collaborative development
  Transfer to individual researchers & students
- 2023–2025: Trainings for IGN Survey Department, Geoscience Australia, Natural Resources Canada
- 2026: Make github repository public (awaiting IGN's greenlight for open-source distribution)
- Future: Write complete documentation

Distribution

Locations and organizations of currently registered pytrf *users*

Architecture

date.py

Handle transformation between date formats
(MJD, YYYY-MM-DD hh:mm:ss, GPS week…)

class date:

```
>>> t = date.from_mjd(61192.65)
>>> print(t)
2026-06-01 15:36:00
>>> t.week
2421
>>> t.tsnx()
'26:152:56160'
```

erp.py

Read, write and manipulate ERP time series

class erp:

```
>>> BuA = erp.read_bua('finals2000A.data')
>>> BuA.interp(np.arange(61142.5, 61149.5))
>>> BuA.write_igs('finals2000A.erp')
```

sinex.py

Read, write and manipulate SINEX solutions

class sinex:

- "read" / "write"
- "load" / "dump" (internal pickle format)
- "unconstrain": Recover unconstrained normal eq.
- "fix_params": Fix certain parameters
- "del_params": Reduce certain parameters
- "del_sta": Reduce certain stations
- "add_mc": Add NNR, NNT and/or NNS constraints
- "neqinv": Solve normal equation
- "compare": Helmert comparison with other "sinex"
- "propagate": Propagate long-term solution to epoch
- "add_psd": Add post-seismic deformation models
- And lots of other methods…

const.py

Useful constants, including:

- conversion factors between angular units
- offsets between different time scales
- GRS80 ellipsoid parameters

io.py

Miscellaneous parsers / writers:

- DOMES number catalogue (codomes.snx)
- discontinuity lists (soln.snx)
- ANTEX (partial parsing only)
- IGS site logs (partial parsing only)

math.py

Useful math routines, including:

- some linear algebra
- coordinate conversions
- Vondrak filter
- fast Lomb-Scargle periodogram

utils.py

Miscellaneous low-level routines

snxcmb.py

SINEX combination routines

- combination of SINEX solutions from different ACs
- long-term stacking of time series of SINEX solutions
- iterative combination with automatic outlier detection/removal and, optionally, variance component estimation
- More in "SINEX analysis & combination"

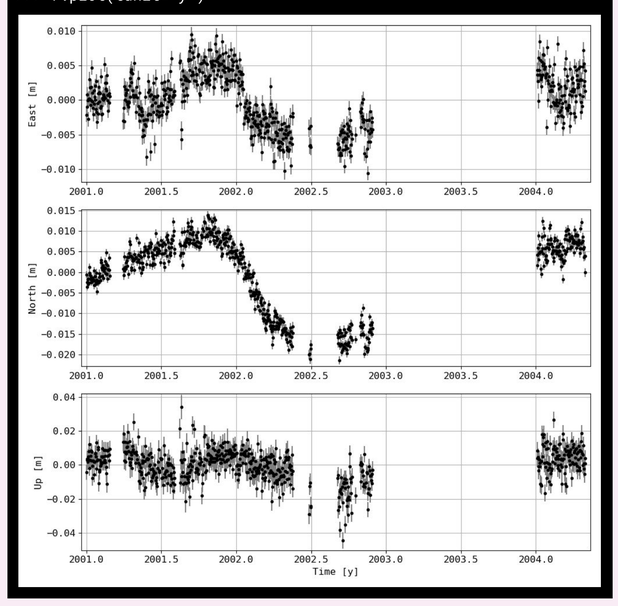
ts.py

Read, analyze and model time series

class ts:

- A simple container for (multi-dimensional) time series data
- Flexible "read" method not tied to any particular format
- Some basic utilities like slicing, "trim", "detrend", "plot"…

```
>>> r = ts.read('IGUA.tenv3', skiprows=1,
                usecols=(3, 8, 10, 12, 14, 15, 16),
                format=('t', 'x', 'y', 'z', 'sx', 'sy', 'sz'),
                tunit='d', yunit='m',
                dims=('East', 'North', 'Up'), dtrd=1)
>>> r.plot(tunit='y')
```



class model:

- "model" objects encode the trajectory and stochastic models [to be] adjusted to a time series.
- See example in "Time series analysis".

Time series analysis

Philosophy

1. Read some time series, i.e.: `>>> r = ts.read(...)`

- dates $\mathbf{t} = [t_1, t_2, \ldots, t_n]$
- values $\mathbf{y} = [y_1, y_2, \ldots, y_n]$
- formal errors $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_n]$ (optional)
- integration intervals $\mathbf{T} = [T_1, T_2, \ldots, T_n]$ (optional)

2. Specify a model: $\mathbf{y} \sim \mathcal{N}(\mathbf{f}(\mathbf{x}), \mathbf{Q}(\boldsymbol{\beta}))$ `>>> m = model(r, ...)`

- **Trajectory model** = sum of deterministic functions — e.g., linear trend + annual oscillation
- **Stochastic model** = sum of noise components — e.g., white noise + power-law noise

3. "Fit" the model, i.e., estimate: `>>> m.fit(...)`

- vector of deterministic parameters $\mathbf{x}$
- vector of stochastic parameters $\boldsymbol{\beta}$

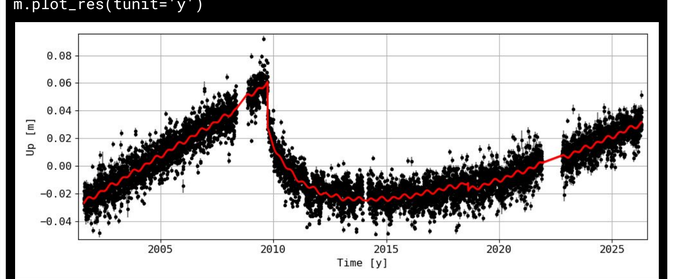
4. Compare different models using model selection criteria

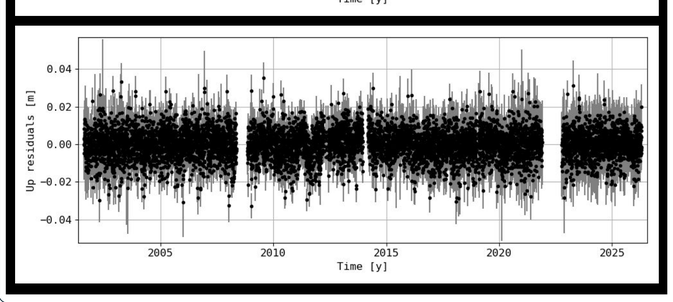
Features

- Iterative fitting with automatic outlier detection
- Two estimators for stochastic parameters:
  - Maximum likelihood (ML), like in CATS, Hector…
  - Restricted maximum likelihood (REML) → **unbiased** estimates
- Several optimization methods available:
  Nelder-Mead, BFGS, Newton-Raphson…
- Covariance matrix of stochastic parameters is provided.
- Estimates of individual noise components are provided.
  (obtained by Wiener-filtering the trajectory model residuals)
- Likelihood ratio tests for the automatic detection of:
  - offsets
  - trend changes
  - periodic signals
- Bayesian information criterion (BIC) for model comparison
- Time series simulation
- In development: GUI

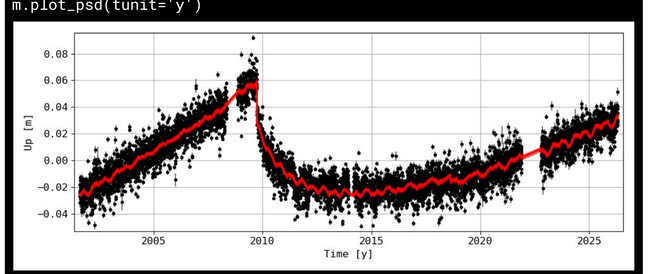
Example: Adjustment of trajectory + stochastic model

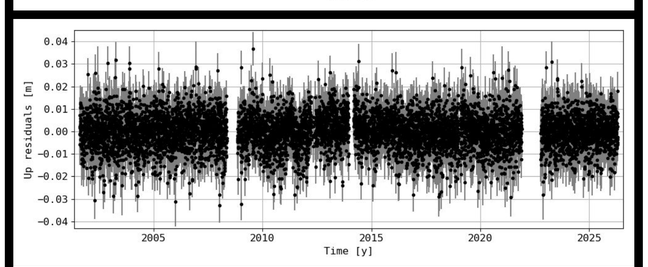
```
# Read time series
r = ts.read('ASPA_igs.xyz', usecols=(2, 4, 5, 6, 7, 8, 9, 10, 11, 12),
    format=('t', 'x', 'y', 'z', 'sx', 'sy', 'sz', 'cxy', 'cxz', 'cyz'),
    dtrd=1, rotate=True)

# Restrict time series to its 3rd (Up) component for this example
r = r[2]

# Initialize model with linear trend, annual & semi-annual sine waves,
# and variable white noise
m = model(r, deg=[0, 1], per=[365.25, 182.625], noise=['vw'])

# Add position offsets
m.add_jumps(deg=[0], t=[54740.00, 55103.74, 58349.01, 59863.84])

# Add velocity change
m.add_jumps(deg=[1], t=[58349.01])

# Add logarithmic function to account for post-seismic deformation
m.add_log(t0=55103.74)

# Iteratively fit model and reject outliers
m.fit_iter(thr_raw=5, thr_norm=5)

# Plot adjusted trajectory model and its residuals
m.plot_fit(tunit='y', show=False)
m.plot_res(tunit='y')
```

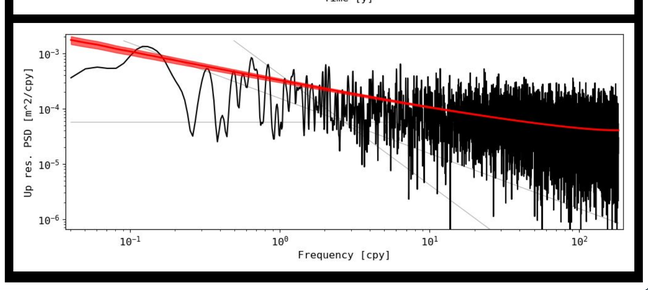




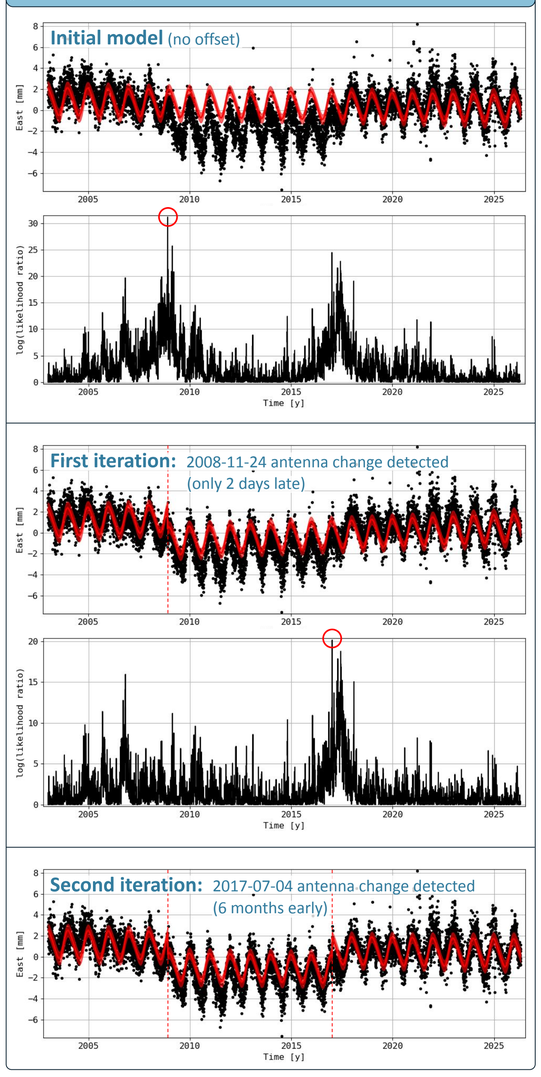

```
# Add some more periodic terms into trajectory model
for T in [351.5/k for k in range(1, 9)] + [14.76, 14.19, 13.62]:
    m.add_sine(T)

# Add power-law noise into stochastic model
m.add_pl()

# Re-fit model
m.fit()

# Plot adjusted trajectory model, its residuals, and their PSD
m.plot_fit(tunit='y', show=False)
m.plot_res(tunit='y', show=False)
m.plot_psd(tunit='y')
```







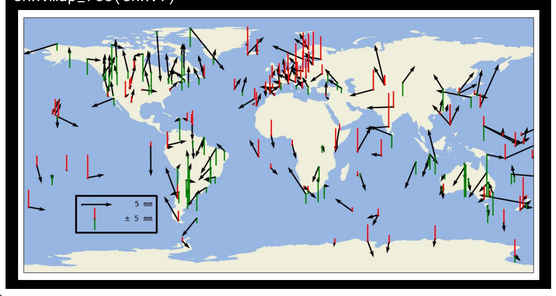
Example: Automatic offset detection



SINEX analysis & combination

Example: Helmert comparison

```
# Read IGc20 SINEX file, PSD models and discontinuity list
ref = sinex.read('IGc20.ssc')
psd = sinex.read('psd_IGc20.snx')
solns = read_solns('soln_IGc20.snx')

# Propagate IGc20 station coordinates to April 25th, 2026
ref.trim_solns('26:115:43200', solns)
ref.propagate('26:115:43200')
ref.add_psd(psd)

# Read IGS daily combined solution for April 25th, 2026
snx = sinex.read('IGS0OPSSNX_20261150000_01D_01D_SOL.SNX')

# Iterative Helmert comparison / Draw map of residuals
(T, Q) = snx.compare_iter(ref, helmerts='RST', thr_raw=5)
snx.map_res(snx.v)
```




```
Main statistics
---------------

# observations      : 618
# parameters        : 7
WRMS East           :    1.799 mm
WRMS North          :    2.187 mm
WRMS Up             :    6.110 mm

Estimated Helmert parameters
----------------------------

TX   :   -0.000 +/-   0.049 mm
TY   :    0.000 +/-   0.049 mm
TZ   :   -0.000 +/-   0.049 mm
SC   :   -0.266 +/-   0.077 ppb
RX   :   -0.000 +/-   0.002 mas
RY   :   -0.000 +/-   0.002 mas
RZ   :    0.000 +/-   0.002 mas

Station position residuals
--------------------------

             |     Raw residuals [mm]     |    Normalized residuals    |
-------------|----------------------------|----------------------------|
 code pt soln |     E        N        H    |     E        N        H    |
-------------|----------------------------|----------------------------|
 ABMF  A    4 |    0.351    0.518    3.296 |    0.171    0.285    0.466 |
 ALBH  A    6 |    0.224    3.755   -1.455 |    0.132    1.859   -0.259 |
 ALGO  A    5 |    1.545    3.767   -1.650 |    0.732    1.544   -0.226 |
[...]
```

Generic combination model

$$\mathrm{E}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \vdots\\ \boldsymbol{x}_n\end{bmatrix}\right) = \begin{bmatrix}\boldsymbol{J}_1 & (t_1-t_0)\boldsymbol{K}_1 & \boldsymbol{H}_1 & & \\ \vdots & \vdots & & \ddots & \\ \boldsymbol{J}_n & (t_n-t_0)\boldsymbol{K}_n & & & \boldsymbol{H}_n\end{bmatrix} \cdot \begin{bmatrix}\boldsymbol{x}\\ \dot{\boldsymbol{x}}\\ \boldsymbol{\theta}_1\\ \vdots\\ \boldsymbol{\theta}_n\end{bmatrix}$$

$$\mathrm{var}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \vdots\\ \boldsymbol{x}_n\end{bmatrix}\right) = \begin{bmatrix}\sigma_1^2\boldsymbol{Q}_1 & & \\ & \ddots & \\ & & \sigma_n^2\boldsymbol{Q}_n\end{bmatrix}$$

- $(\mathbf{x}_1, \mathbf{Q}_1), \ldots, (\mathbf{x}_n, \mathbf{Q}_n)$ — input solutions (station positions, ERPs, …) & their covariance matrices
- $\mathbf{x}$ — combined parameters — estimated
- $\dot{\mathbf{x}}$ — combined parameter rates (station velocities) — optionally estimated
- $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ — (subsets) of Helmert parameters between input solutions and combined solution — estimated
- $\sigma_1^2, \ldots, \sigma_n^2$ — corrective variance factors — optionally estimated
- $\mathbf{J}_1, \ldots, \mathbf{J}_n$ $[\mathbf{K}_1, \ldots, \mathbf{K}_n]$ — matrices of ones and zeros mapping parameters in input solutions to parameters [parameter rates] in combined solution
- $\mathbf{H}_1, \ldots, \mathbf{H}_n$ — matrices of Helmert parameter partial derivatives

snxcmb.py features

- Can be used for:
  - the combination of solutions from different ACs (e.g., IGS daily SINEX combinations)
  - the long-term stacking of time series of solutions (e.g., IGS cumulative solution)
  - the combination of solutions from different techniques (e.g., ITRF)
- Automatic iterative detection/removal of outliers
- Automatic iterative estimation of corrective variance factors $\sigma_1^2, \ldots, \sigma_n^2$ by VCE (optional)
- Possibility to reduce Helmert parameters
  (saves a lot of computation time in the case of a long-term stacking)
- Possibility to combine/stack GNSS satellite PCOs and VLBI radiosource coordinates
- Several possible strategies to define the frame of the combined solution:
  - minimal (a.k.a. NNR, NNS and NNT [-rate]) constraints
  - internal constraints (in development)
- In development: estimation of periodic (e.g., seasonal) station motions