Inverse Problem

$$\hat{\mathbf{x}} = \left(\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{H} + \mathbf{P}^{-1}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{y}. \quad (1)$$

Input and Output of red_tide

The red_tide package is available for download as a GitHub repository at `https://github.com/lkach/red_tide`. The standard input and output arguments for typical use are outlined here. This package is written in the MATLAB language, but translation to other programming languages is welcome and encouraged. It has also been designed to work in the compatible GNU Octave language with the necessary packages installed. Input for red_tide is flexible, with several options and default settings. The following is the full input syntax using the MATLAB language:

```
red_tide(t, y, FSR_Cell)
```

where

```
FSR_Cell =
{{n_lowNO, df_NO, n_lowO, tide_f,
  fband_centers, n_sidebands,
  df_sidebands, inertial},
 {f_spec, spec},
 {R_input, R_format, Cov_cutoff, Window}}
```

Here `t` is the time basis of the corresponding data time series `y`. The other terms in equation (1) are constructed from the components of the cell array `FSR_Cell`, which is further divided into three cells in order to clearly separate the inputs based on function: the first contains information to construct the f̲requency basis of the model, the second to construct **P** from s̲pectral information, and the third to construct the prior r̲esidual autocovariance **R**. `n_lowNO` is the number of low non-orthogonal frequencies to include, spaced by frequency steps of `df_NO`. `n_lowO` is the number of low frequencies (higher than the ones already prescribed) that are orthogonally spaced by Δf. The cell array `tide_f` lists all the tidal frequencies that are to be fit, either numerically or with shorthand string characters. `fband_centers` is formatted the same way and lists the centers of frequency bands to be modeled. Its elements correspond to those of `n_sidebands`, which lists the number of discrete frequencies of spacing `df_sidebands` on either side of each band's center. `inertial` is an optional vector that includes the latitude and number and spacing of frequencies for modeling the band of frequencies at which near-inertial oscillations are observed. Alternatively, the first cell within `FSR_Cell` may simply contain an integer `N` up to 37, which results in red_tide modeling the first to `N`'th order tidal constituents as defined by the National Ocean Service. The pair of identically sized vectors `f_spec` and `spec` are the frequency basis and power spectrum of the process to be modeled. `R_input` is a scalar or vector containing information which, when paired with the string `R_format`, determines the assumed structure of misfit noise. `Cov_cutoff` sets the maximum time lag at which misfit is assumed to be correlated, and `Window` is a windowing function for the prior misfit autocovariance in order to condition the matrix $\mathbf{R}$ for efficient inversion.

Before calculating model coefficients, each matrix in equation (1) is constructed in its own subroutines:

- The regressor matrix $\mathbf{H}$ is a collection of sines and cosines of the given frequencies and at times `t`.
- The prior model autocovariance matrix $\mathbf{P}$ is constructed by interpolating the given spectrum `spec` from the frequencies in `f_spec` to those of the model and setting that as the diagonal of $\mathbf{P}$. Model coefficients are assumed to be uncorrelated by default, therefore $\mathbf{P}$ is diagonal, though the user may bypass this step and use an arbitrary matrix for $\mathbf{P}$.
- The prior noise autocovariance matrix $\mathbf{R}$ is constructed by setting the main diagonal to the prior variance of $\mathbf{r}$ and setting the n'th upper and lower off-diagonals to the n'th lagged prior autocovariance of $\mathbf{r}$. The prior autocovariance of $\mathbf{r}$ is either given directly or calculated as the inverse Fourier transform of a given spectrum, usually with a constant spectral slope. $\mathbf{R}$ is therefore a Toeplitz matrix when there are no gaps in $\mathbf{y}$, otherwise it is symmetric.

Model coefficients are calculated by solving equation (1) indirectly by taking advantage of optimized subroutines in MATLAB, which avoids the computationally expensive, and sometimes memory-limited, explicit calculation of $\mathbf{R}^{-1}$.