Inverse Problem

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

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 frequency basis of the model, the second to construct $\mathbf{P}$ from spectral information, and the third to construct the residual covariance $\mathbf{R}$. `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, given either as numbers 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[1]. The pair of identically sized vectors `f_spec` and `spec` are the frequency basis and power spectrum of the process to be analyzed. `R_input` is a scalar or vector containing information

[1]`https://tidesandcurrents.noaa.gov/harcon.html?id=1612340`

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 residual covariance 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 model covariance 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 $\mathbf{P}$ (see below).
- The residual covariance matrix $\mathbf{R}$ is by default 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 covariance of $\mathbf{r}$. The expected covariance 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 by default constructed as a Toeplitz matrix when there are no gaps in $\mathbf{y}$; in the event that there are gaps, it is symmetric but not Toeplitz. If the user wants to use a matrix $\mathbf{R}$ that is constructed to suit their particular problem, an arbitrary one may be given (see below).

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 $\mathbf{R}^{-1}$.

Arbitrary Inputs

When analyzing multiple time series with otherwise the same input parameters, computational time can be wasted on repeatedly constructing the same matrices ($\mathbf{H}$, $\mathbf{R}$, and $\mathbf{P}$) over and over. Additionally, one might wish to bypass the construction of these matrices altogether and have direct control. To accommodate both of these scenarios, as well as to allow other useful options, red_tide allows the input of predefined matrices that bypass their internal construction via the following syntax:

```
red_tide(t, y, FSR_Cell, 'var1', val1, 'var2', val2, etc.)
```

The string inputs `'val1'`, `'val2'`, etc. may be selected out of: `'F'`, `'H'`, `'P'`, `'R'`, `'Fig'`, `'InvMethod'`, and `'InterpMethod'`, where the string is followed by the value it is to take, overriding any internal construction of that variable that red_tide would otherwise perform. The first option is the vector of frequencies to which the time series is to be fitted, and the next three are the matrices that appear in equation (1) described above.

The option `'Fig'` may be set to `'on'` in order to produce a cursory figure that includes time- and frequency-domain information about the input and output, including: data (time series), fitted time series, data periodogram, assumed model spectrum, assumed residual spectrum, and model amplitude. The option `'InvMethod'` (inversion method) may be set to either `'default'` (default) or `'Cholesky'`. The former solves R\H, while the latter defines `L=ichol(R)` and solves L\H. The default option seems faster in all cases, but the option to use Cholesky factorization remains in case it is ever deemed relevant or necessary. The option `'InterpMethod'` (interpolation method) dictates the method used to interpolate the model prior spectrum to the modeled frequencies (default `'loglinear'`, see "P_make" documentation). This has to do with constructing **P** from information in `FSR_Cell`, and has nothing to do with interpolating the data.