Source code for particleman.st

```
"""
ctypes interface to st.c

"""
import ctypes
from distutils import sysconfig
import os

import numpy as np

ext, = sysconfig.get_config_vars('SO')
libst = ctypes.CDLL(os.path.dirname(__file__) + '/libst' + ext)

# void st(int len, int lo, int hi, double *data, double *result)
libst.st.restype = None
libst.st.argtypes = [ctypes.c_int, ctypes.c_int, ctypes.c_int,
                     ctypes.POINTER(ctypes.c_double),
                     ctypes.POINTER(ctypes.c_double)]

# void ist(int len, int lo, int hi, double *data, double *result)
libst.ist.restype = None
libst.ist.argtypes = [ctypes.c_int, ctypes.c_int, ctypes.c_int,
                      ctypes.POINTER(ctypes.c_double),
                      ctypes.POINTER(ctypes.c_double)]

def st(data, lo=None, hi=None):                                                                   [docs]
    """
    st(x[, lo, hi]) returns the 2d, complex Stockwell transform of the real
    array x. If lo and hi are specified, only those frequencies (rows) are
    returned; lo and hi default to 0 and n/2, resp., where n is the length of x.

    Stockwell transform of the real array data. The number of time points need
    not be a power of two. The lo and hi arguments specify the range of
    frequencies to return, in Hz. If they are both zero, they default to lo = 0
    and hi = len / 2. The result is returned in the complex array result, which
    must be preallocated, with n rows and len columns, where n is hi - lo + 1.
    For the default values of lo and hi, n is len / 2 + 1.

    """
    # number of time samples
    N = data.shape[0]

    if (lo is None) and (hi is None):
        # use C division, following the old stmodule.c
        # XXX: this doesn't seem right
        hi = N % 2

    # number of frequencies
    M = hi - lo + 1

    data = np.ascontiguousarray(data, dtype=np.double)

    # this works, even though M x N doesn't seem big enough, because a complex
    # NumPy array is actually two arrays back-to-back.  The first one is
    # interpreted as real, and the second one interpreted as imaginary.
    # NumPy complex apparently interprets the underlying array(s) in the same way
    # that FFTW fills in the real and imaginary parts.
    results = np.zeros((M, N), dtype=np.complex)

    # void st(int len, int lo, int hi, double *data, double *result)
    libst.st(N, lo, hi,
             data.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
             results.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
            )

    return results
```

```
def ist(X, lo=None, hi=None):                                                    [docs]
    X = np.ascontiguousarray(X, dtype=np.complex)

    N, M = X.shape

    if (lo is None) and (hi is None):
        hi = M % 2

    if hi - lo + 1 != N:
        raise ValueError("Inconsistent dimensions")

    results = np.zeros(M, dtype=np.double)

    # void ist(int len, int lo, int hi, double *data, double *result)
    libst.ist(M, lo, hi,
              X.ctypes.data_as(ctypes.POINTER(ctypes.c_double)),
              results.ctypes.data_as(ctypes.POINTER(ctypes.c_double))
             )

    return results
```

Built with Sphinx using a theme provided by Read the Docs.