Optimal Fourier Transform documentation

Allen.Goldstein, allen.goldstein@nist.gov

January 2019

1 Introduction

The Fourier transform decomposes a function of time into the frequencies that make it up. When the time function is discrete, the discrete Fourier transform (DFT) can only find an exact frequency contained in a discrete time series if the frequency falls exactly in the center of a "bin". If included frequencies are not at the exact center of the bin, the DFT spreads the energy over several bins and does not provide an exact frequency, phase and amplitude for the contained sinusoid. Furthermore, the DFT will include the energy of any random noise in the transform. DFT's assume the frequency components at each bin center are orthogonal and the inverse DFT function will return the original time series.

This documentation is for a matlab port of vba code called "Optimal Fourier Transform" developed by Dr. David Evans. It is described by his paper that can be downloaded here. The OFT estimates the spectrum of a time series, describing with a series of exact frequencies paired with their coefficients of cosines and sines. In this port, the cosine and sine coefficients are given in complex rectangular form per Euler's equation.

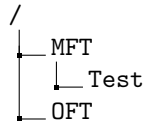
The OFT does not assume the contained sinusoids are orthogonal and cannot be inverted, however a new time series can be calculated from the OFT frequencies and coefficients, subtracted from the original time series, and the residual should be just the random noise contained in the original time series.

1.1 Installation

The OFT code is packaged in a MATLAB toolbox installer file (OptimalFourierTransform.mltbx). Running this file from within MATLAB will install it into your MATLAB toolbox folder and add it to the MATLAB path.

1.2 Code overview

Inside the OptimalFourierTransform toolbox folder are the following directories:



└─ Test

MFT stands for "Manual Fourier Transform". This is a technique to determine the coefficients of contained sinusoids int a time series when given exact frequencies. The MFT folder contain many functions used but the OFT code.

The OFT.m file contained in the OFT folder is the class-based OFT MATLAB code. which will be described later.

The /Test subfolders contain regression tests for the OFT and MFT functions. A good way to experience the OFT code while it is working is to run the regression tests with the default settings. A set of test time series are analysed by OFT with options to see the progress dialog and plots of the signals being analysed and followed by a plot of the original time series in blue, overlayed by the calculated time series in red, and a plot of the residual below. To run the tests type:

$>> testCase = testOFT\_class;$
$>> res = run(testCase);$

After each time series is analyzed and the results are shown, you must press any key to advance to the next time series test. Subsequent test runs, even after editing the *testOFT_class.m* file can be run using only *res=run(testCase)*.

To change display options, edit *testOFT_class.m, regressionTests* function and change `testCase.bWaitBar` or `testCase.bShowResult` to false

```
    methods (Test)
        function regressionTests (testCase)
            testCase.bWaitBar = true;       % true:  watch internal ...
                stage progress
            testCase.bShowResult = true;    % true, show final ...
                result and pause after each test
            setTsDefaults(testCase)

            % comment out any line below to skip those tests
            testNyquist (testCase)
            testNearDC (testCase)
            testLab (testCase)
        end
    end
```

2 OFT.m class documentation

The OFT main function returns a list of contained frequencies in order of decreasing amplitude, the complex coefficients (amplitude and phase in complex rectangular form) for each frequency, and a single fractional error figure representing the total energy remaining in the time series after the sinusoids have been removed.

```
        function [freqs, MFT_OFT, fracErr] = OFT_fn (obj, ts, time)
            % Ported from VBA by Allen Goldstein, NIST,
```

```
            % from: http://jonova.s3.amazonaws.com/cfa/climate.xlsm
            % written by: Dr David Evans
            %             david.evans@sciencespeak.com
            %*************************************************************
            %- Computes the Optimal Fourier Transform (OFT) of a ...
                real-valued time series with data points spaced
            %  equally in time. Very slow.
            %- Reconnaissance slows it, but it can find frequencies ...
                that are close enough not to produce separate
            %  peaks in the DFT. Generally worth it, though ...
                sometimes it triples the exectution time for no gain.
            %- First stage does most of the work. Following stages ...
                just wring out structure out of what is left
            %  in the time series, until either the fit is ...
                near-perfect or the residue is just pointwise noise.
            %- In:  ts  [0..N-1]            Time series.
            %       time[0..N-1]            Times of data points ...
                (relevant iff "regular" is true).
            %- Out: Freqs[1..nFreqs]         Frequencies at which ...
                OFT detects sinusoids, in cycles per unit of time.
            %       OFT[1..nFreqs]          Complex Optimal Fourier ...
                transform
            %       fracErr                 Absolute deviation of ...
                synthesisTS from OFT of ts / absolute deviation of ts.
            %=============================================================================
```

The OFT class constructor: `function obj = OFT(bWaitBar, bDoRecon, ... kMaxNFreqsAtOnceOFT, kMaxNStages, relativeAbsDev)`, if called with no parameters, will set all properties to default values. The public properties are user accessable to facilitate experimentation and optimization of the OFT for the characteristics of any particular time series. The default values of the OFT class are:

```
properties (Access = public)
        bWaitBar = false;
        bDoRecon = true;

        relativeAbsDev = .000001;
        maxNFreqsPerStage = 10;

        % constants
        kMaxNFreqsAtOnceOFT = 10;
        kMaxNStages = 2;
        kMinChangeTargetAbsDevFraction = 0.0001;
        kMaxNFreqsPerStage;
        kNuConsolidate = 0.1;   % How close should frequencies be ...
            allowed to get before they are consolidated
        kRemoveFraction = 1e-9;

    end
```

- bWaitBar (false): Boolean determines if OFT will display a progress bar and plot the stage and pass results as well as the next stage

- bDoRecon (true): for many time series, especially those with noise, the first stage is a reconnaissance to determine the best starting frequencies for the main stage. Without this pass, the results will be less accurate, especially for time series that are not made up of only pure sinusoids.
- relativeAbsDev (0.000001) determines the maximum fractional error (amount of energy) that below which the OFT program will stop searching for new frequencies. If the time series has higher levels of random noise, increasing this number may provide more accurate results.
- maxNFreqsPerStage (10): The number of frequencies that each stage of OFT will estimate. OFT stages progress until the `relativeAbsDev` has been reached or the frequency amplitudes have flattened out.
- kMaxNFreqsAtOnce (10): similar to maxNFreqsPerStage, this limits how many estimated frequencies can be analysed at any stage or during any pass. This also affects the bracketing for the final call to MFT since MFT is more accurate with fewer frequencies to analyse per bracket (see the OFT paper for a discussion of bracketing).
- kMaxNStages (2): the maximum number of stages to run after the reconnaissance stage. If the maximum number of frequencies per stage is reduced, then more stages may be needed to estimate all the contained frequencies. The default settings limit the total number of estimated frequencies to no more than 20.
- kMinChangeTargetAbsDevFraction (0.0001): Scales the threshold which determines the "flatness" of the frequencies found in one stage to determine if enough frequencies have been found.
- kNuConsolidate (0.1): during the process of frequency estimation, two frequency guesses may converge. If the OFT internal optimization attempts to optimize two very,very close frequencies, the internal calculation can get extremely large and overflow. kNuConsolidate determines how close two converging frequency estimates will be allowed to get before they are consolidated into a single frequency half-way between the two.
- kRemoveFraction (1e-9): Any frequencies with an amplitude below this level will be removed from the result.

3 MFT code

MFT refers to "Manual Fourier Transform" (see the OFT paper) Given a set of frequencies, the MFT estimates the coefficients of those frequencies contained in the time series. The *./MFT* folder holds the MFT function and a set of subfunctions needed for both the MFT and the OFT. A *./MFT/Test* folder holds regression tests as well as the Artificial Time Series code described in the next section.

4 Artificial Time Series

./MTF/Test contains ArtificialTS.m which is a MATLAB class for generating time series with known sinusoids and random noise levels and types to be used for testing. Any number of contained sinusoids can be generated and additive uniform and/or gaussian noise can be added.

4.1 properties

```
    properties
        Name            % string name of the Time Series
        Description     % string describing the TS

        T0              % double starting time
        Extent          % double TS duration in seconds
        nSamples        % number of samples in the time series ...
            (Sample Rate = nSamples/Extent)

        Valid           % [1..nSinusoids] array of booleans ...
            (invalid if caller enters bad values)
        Freqs           % [1..nSinusoids] array of sinusoid frequencies
        Amps            % [1..nSinusoids] array of sinusoid amplitudes
        Phases          % [1..nSinusoids] array of phase angles (in ...
            radians)

        NoiseUniformLow % double Uniform distribution noise lowest ...
            value
        NoiseUniformHi  % double Uniform distribution noise highest ...
            value
        NoiseGaussMean  % double noise mean value
        NoiseGaussSD    % double noise standard deviation

        Ts              % Time Series
        time            % Time vector
    end
```

4.2 constructor

If the constructor is called with no arguments, the object is created buu all parameters are set to zero and no time series or time vector will be created. If called with the below parameters, then the object is returned and a time vector and time series is created if the parameters are valid. If the parameters are not valid, then a error is thrown.

```
            %Constructor
        function ArtTS = ArtificialTS( ...
                Name, ...
                Description, ...
                T0, ...
                Extent, ...
```

```
                nSamples, ...
                Freqs, ...
                Amps, ...
                Phases, ...
                NoiseUniformLow, ...
                NoiseUniformHi, ...
                NoiseGaussMean, ...
                NoiseGaussSD ...
                )
```

4.3 public methods

```
function ArtTS = makeTime(ArtTS) % Creates a time vector from T0 to ...
    Extent+ 1 period with Extent/nSamples period

function [ArtTS] = makeTS(ArtTS) % Creates an artificial time series
```