Summary of DAS data preprocessing

by Xin Xing

April 3 2019

Input structure 'dsi': (matlab version)

- `dt = dsi.fh{8}`.
- `dat = dsi.dat{1}` (single DSI record), each column of dat is the recorded trace by one channel.

Processing Steps: (consider one trace $x_0 = \mathrm{dat}(:, j)$)

1. **Remove trend of the trace**

   Apply detrend to trace x_0 from dat,

   $$x_1 = \mathrm{detrend}(x_0) \tag{1}$$

   which removes the best straight-line fit linear of x_0 and returns the residual x_1.

2. **Decimation**

   Substep 1: Antialiasing low-pass for each trace

   Parameter:

   - Input parameter: `dt_new = 0.008`.
   - Nyquist Frequency of `dt` and `dt_new` as

     $$\begin{aligned} \mathrm{fNyquist\_new} &= 0.5 / \mathrm{dt\_new} \\ \mathrm{fNyquist\_old} &= 0.5 / \mathrm{dt} \end{aligned}$$

     and their ratio `Wn = fNyquist_new/ fNyquist_old`.
   - Butterworth lowpass IRR order `order = 3`

   Calculation:

   a. get the filter design (two short vectors) using matlab function 'butter'

   $$[B, A] = \mathrm{butter}(\mathrm{order}, \mathrm{Wn}, \text{'low'})$$

   where B and A are small vectors and this operation can be done in each processor.

   b. apply matlab function 'filtfilt' with 'B' and 'A' over x_1 as

   $$x_2 = \mathrm{filtfilt}(B, A, x_1) \tag{2}$$

   where x_2 is an vector of the same length as x_1.

Substep 2: Resample each trace

Calculate the new **relative** samping rate `R=round(dt_new/dt)`. The "relative" here means regarding the original sampling rate as 1. Then, resample x_2 as

$$x_3 = \text{resample}(x_2, 1, R), \tag{3}$$

which first fits vector x_2 using a 'smooth curve' and then samples the obtained curve at a new rate R to get x_3. The vector x_3 can be either longer or shorter than the old x_2 depending on R.

Some auxiliary info setting:

- `dsi.fh{7} = length(x3)`
- `dsi.fh{8} = dt_new`. (now `dt =dt_new`)
- `dsi.fh{9} = 0`
- `dsi.fh{10} = (length(x3)-1)*dt_new`

3. **Apply time-domain moving mean (or root-mean-square(rms)) normalization**

   Parameters:

   - Input parameter: moving window size (sec.) : `winLen_sec = 0.5`
   - # of trace points per window:

     `nPoint_per_win = 2 * floor(winLen_sec/dt/2) + 1;`
   - half # of trace points per window:

     `nPoint_halfWin = floor(nPoint_per_win / 2);`
   - midpoint of the time window:

     `nPoint_mid_win = floor(nPoint_per_win / 2) + 1;`

   Apply moving mean (or rms) to x_3 and get a vector x_4 of the same length.

   Case 1 : indices $(i - \text{nPoint\_halfWin})$ and $(i + \text{nPoint\_halfWin})$ are not out of range.

$$x_4(i) = x_3(i) / \text{mean}(\ \text{abs}(\, x_3(\, (i - \text{nPoint\_halfWin}) : (i + \text{nPoint\_halfWin})\,)\,)\ ) \tag{4}$$

   or

$$x_4(i) = x_3(i) / \text{rms}(\ x_3(\, (i - \text{nPoint\_halfWin}) : (i + \text{nPoint\_halfWin})\,)\ ) \tag{5}$$

   Case 2 : index $(i - \text{nPoint\_halfWin})$ is less than 1

$$x_4(i) = x_3(i) / \text{mean}(\ \text{abs}(x_3(1 : i))\ ) \tag{6}$$

   or

$$x_4(i) = x_3(i) / \text{rms}(\ x_3(1 : i)\ ) \tag{7}$$

Case 3 : $(i + \text{nPoint\_halfWin})$ is larger than length(x_3).

$$x_4(i) = x_3(i)\,/\,\text{mean}(\ \text{abs}(\,x_3(i\!:\ \text{length}(x_3))\,)\ \ ) \tag{8}$$

or

$$x_4(i) = x_3(i)\,/\,\text{rms}(\ x_3(i\!:\ \text{length}(x_3))\ ) \tag{9}$$

4. **Spectral whitening**

   Input Parameters:

   - F1 - a scalar; low end cut (Hz).

     Matlab value `F1 = 0.002`

   - F2 - a scalar; min pass band (Hz).

     Matlab value `F2 = 0.006`

   - F3 - a scalar; max pass band (Hz).

     Matlab value `F3 = 14.5`

   - F4 - a scalar; high stop (Hz).

     Matlab value `F4 = 15`

   - eCoeff - whitening coefficient (between 0 and 1). The closer eCoeff is to 1, the more severe the flattening

     Matlab value `eCoeff = 1`

   Other parameters:

   - `fNyquist = 0.5 / dt_new`
   - `fSampling = 2 * fNyquist`
   - `nPoint = length(x4)`
   - `nfft` = the smallest integer that is a power of 2 and also greater than `(2*nPoint-1)`
   - `df = fSampling / nfft`
   - `f_LHS = (df, 2*df, ..., fNyquist)`

   Substep 1: Construct the filter 'shapingFilt' (a small vector, can be done in each processor)

   a. Let $z = [0, 0.5, 1, 1, 0.5, 0]$ and zf$=[0, F1, F2, F3, F4, \text{fNyquist}]$.

   b. obtain 'shapingFilt_LHS' via

   ```
   shapingFilt_LHS = interp1(zf,z,f_LHS, 'linear');
   ```

   which piecewise-linearly interpolates the function F that satisfies F(zf) = z to obtain the values at f_LHS, i.e., `shapingFilt_LHS = F(f_LHS)`.

c. flip the vector `shapingFilt_LHS`, i.e., head to tail and tail to head, to obtain `shapingFilt_RHS` and concatenate the two vectors together as

```
shapingFilt = [shapingFilt_LHS, shapingFilt_RHS].
```

Substep 2: Spectral whitening process

a. Concatenate `(nfft-nPoint)` zeros at the end of x_4 to make its length to be `nfft`.

b. Apply FFT to the modified x_4 to get a `nfft`-dimensional vector

```
gatherSpec = fft(x4).
```

c. Entrywise whitening of the spectral vector `gatherSpec` as

$$\text{gatherSpec\_whitened}(i) = \frac{\text{gatherSpec}(i) + 0.001}{|\text{gatherSpec}(i)|^{\text{eCoeff}} + 0.001}. \tag{10}$$

where 0.001 is added for numerical stability to avoid the case when 0 is divided by 0.

d. Entrywise product between `gatherSpec_whitened` with `shapingFilt` to get

```
gatherSpec_filted(i) = gatherSpec_whitened(i) * shapingFilt(i)
```

e. Inverse FFT and subsampling to obtain nPoint-dimensional vector x_5.

```
gather_temp = real(ifft(gatherSpec_filted));
x5 = gather_temp(1:nPoint);
```

5. **Frequency domain cross-correlation**

Input parameters:

- `dx = 2.0`
- `direction = 'left' or 'right'` used to decide the master trace

Other paramters:

- `nPoint = length(x5)`
- `nTrace = number of traces`
- `nfft` = the smallest integer that is a power of 2 and also greater than `(2*nPoint-1)`

Substep 1: Frequency domain cross-correlation

a. Apply FFT to each trace x_5 to get a `nfft`-dimensional vector X and calculate the conjugate of the obtained vector Xc.

```
X = fft(x5)
Xc = conj(X)
```

b. Select the master trace `x_master` to be the 'X' from the first ('left') or the last ('right') trace.

c. Define variable `offset` according to the direction

```
if direction == 'left', offset = [0, dx, 2*dx, ..., (nTrace-1)*dx];
if direction == 'right', offset = [(nTrace-1)*dx, (nTrace-2)*dx, ..., dx, 0]
```

d. Calculate cross correlation vector `specXcorr` of length nfft as

`specXcorr(i) = Xc(i) * x_master(i);` (entrywise product)

Substep 2: Return to time domain

a. Apply inverse FFT to specXcorr and get real part of the obtained vector as `gatherXcorr_temp`

`gatherXcorr_temp = real(ifft(specXcorr))`

where this vector is still of length nfft

b. Reorder and subsample `gatherXcorr_temp` as

$$\text{gatherXcorr} = \begin{pmatrix} \texttt{gatherXcorr\_temp((nfft} - \texttt{nPoint} + \texttt{2)}: \texttt{nfft)} \\ \texttt{gatherXcorr\_temp(1}: \texttt{nPoint)} \end{pmatrix} \tag{11}$$

where this vector is of length (2*nPoint-1).

c. For each original trace x_0, the final processed vector x_6 is

$$x_6 = \text{gatherXcorr} \tag{12}$$

which is stored in the dsi_xcorr.dat$\{1\}(:, j) = x_6$, assuming that x_0 is the jth trace.

Auxilliary Info Assignment.

```
dt_new = dsi.fh{8};  (dsi is the result from the Step 4 -- spectral whitening)
nPoint_new = 2*nPoint - 1;
tmax = dt_new * (nPoint - 1);

% file headers
dsi_xcorr.fh = cell(1, 13)
dsi_xcorr.fh{1} = nTrace
dsi_xcorr.fh{7} = nPoint_new
dsi_xcorr.fh{12} = 1
dsi_xcorr.fh{13} = 1
dsi_xcorr.fh{3:6} = dsi.fh{3:6}
dsi_xcorr.fh{8} = dsi.fh{8} = dt_new
dsi_xcorr.fh{9} = -tmax
dsi_xcorr.fh{10} = tmax

% trace headers
```

```
dsi_xcorr.th{1} = zeros(84, nTrace)

dsi_xcorr.th{1}(1, :) = 1 : nTrace

dsi_xcorr.th{1}(12, :) = [nTraces, nTraces, ..., nTraces]  (of length nTraces)

dsi_xcorr.th{1}(13, :) = [1, 2, ..., nTrace];

dsi_xcorr.th{1}(53, :) = offset

dsi_xcorr.th{1}(62:64, :) = dsi_th{1}(62:64, :);


% data

dsi_xcorr.dat{1}(:, i) = gatherXcorr.
```