FADER Manual

Ziqi ZHANG

Contents

1 Introduction

1.1 Background and Workflow Overview

The Earth, in large portions, is covered in oceans, sediments, and glaciers. High-resolution body wave imaging in such environments often suffers from severe reverberations, that is, repeating echoes of the incoming scattered wavefield trapped in the reverberant layer, making interpretation of lithospheric layering difficult. FADER provides a systematic data-driven approach, using autocorrelation and homomorphic analysis, to solve the twin problem of detection and elimination of reverberations without a priori knowledge of the elastic structure of the reverberant layers. Figure 1 gives an overview of the workflow in FADER. The signal processing workflow starts with echo detection, then identifying and automatically tuning the reverberation parameters (using both the autocorrelation and homomorphic analysis), and finally ends with resonance removal. If the data does not present strong reverberations, as measured by an echo quality factor, the workflow exits. The workflow can also be used to scan large datasets for signal distortion due to reverberation.

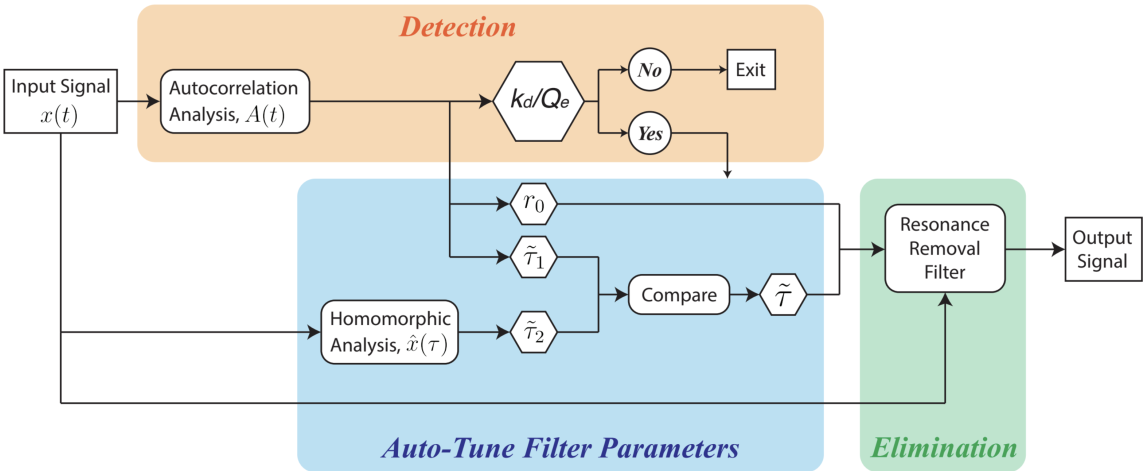

Figure 1: Signal processing workflow for FADER. Detection is obtained from an estimate of the echo number, k_d, and the consequent echo quality factor, Q_e. The response of the reverberating layer, characterized by reverberation strength, r_0, and echo delay time, $\tilde{\tau}$, is auto-tuned using autocorrelation and homomorphic analysis. The resonance removal filter is the inverse of the reverberating response and is applied only when repeating echoes (reverberations) are detected.

1.2 Contents of Package

The FADER package contains two main codes:
FADERWorkflow.m performs the detection and elimination of reverberations for a single station data;
FADERWorkflow_Array_Detection.m performs the detection of reverberations for multiple-station data.

There are four (4) directories in the package:
Functions/ contains all the MatLab functions supporting the workflow;
Data/ is the default directory for storing input data;
filteredRF/ is the default directory for storing output filtered RF (receiver function) from *FADER-Workflow.m*;
Array/ is the defulat directory for storing reverberation detection results, RF plots, autocorrelation

plots from *FADERWorkflow_Array_Detection.m*.

2 Usage: Single Station Detection and Elimination

Detection and elimination of reverberations on single station data is performed by **FADERWorkflow.m**.

2.1 Parameters Setup

This step is listed as **Step 0.** in the main code:

```
1 FADERDir = '/scratch/tolugboj_lab/Prj4_Nomelt/FADER/';
2 addpath([FADERDir 'Functions/']);
3
4 kd_thresh = 2;
5 man_determine = 1; % mannually check autocorrelation plot
6 tolerance = 0; % Echo delay time from autocorrelation and homomorphic
7
8 saveRF = 1;
9 saveDir = strcat(FADERDir,'filteredRF/');
10 savename = 'NE68';
11 smoothopt = 1; % smooth filtered RF
12
13 Hkopt = 1;
14
15 Vp = 6.4; % of crust
16 Hwin = [25 55];
17 kwin = [1.6 1.9];
18 resfac = 100;
19 pWAc = [0.6 0.3 -0.1]; % weighting factors for H-k stacking
20 wid = 0.1; % width of Gaussian windows in H-k stacking
21
22 man_PbS = 1; % manually confirm PbS arrival for H-k stacking
```

The following entries need to be properly setup:

Line 1: **FADERDir** should be directed to the location of this package.

Line 4: **kd_thresh** is the threshold, k_{thr}, for echo number, k_d, to determine the presence of reverberation. The default setting is 2. Users can set a higher k_{thr} value if detection is required for the strongest reverberations.

Line 5: **man_determine** is set to 1 if the user wants to manually inspect the observed and fitted autocorrelation curves, and set to 0 otherwise.

Line 6: **tolerance** is the maximum accepted error between the echo delay times obtained from autocorrelation and homomorphic analysis. If the difference is larger than this tolerance, the user will be asked to manually inspect the plots from each analysis to determine the echo delay time. The default setting at 0 forces the user to manually inspect at all times.

Line 8: **saveRF** is set to 1 if the user wants to save the original and filtered RFs into a MatLab structure, and set to 0 otherwise.

Line 10: **savename** is the file name for the saved RFs.

Line 11: **smoothopt** is set to 1 if the user wishes to smooth the filtered RFs, and set to 0 otherwise.

Line 12: **Hkopt** is set to 1 if the user wishes to perform H-k stacking, and set to 0 otherwise.

Line 15-17: **Vp**, **Hwin** (thickness searching window), and **kwin** (P-to-S velocity ratio searching window) for the H-k stacking.

Line 18: **resfac** (resolution factor) along each axis (H and k) for the H-k stacking.
Line 19: **pWAc** is the weighting factor for Ps, Pps, and Pss phases for the H-k stacking.
Line 20: **wid** is the width of the Gaussian windows for the H-k stacking.
Line 22: **man_PbS** is set to 1 if the user wishes to manually inspect the RF traces to determine the arrival time for the PbS phase (the Ps conversion of the reverberant layer), and set to 0 otherwise.

2.2 Load in RF Data

This step is listed as **Step 1.** in the main code. The main code provides two options for input data: (1) Synthetic RF from Telewavesim[1], and (2) Real RF from MTC (multi-taper correlation)[2].
For the Synthetic RF, we encourage the users to explore the Telewavesim and SynRFWorkflow[3] packages, both available on GitHub. For real data (also for synthetic data if you prefer to use other programs other than Telewavesim), although the MTC RF program is not available to public, the main code can be easily modified to accommodate your own input RF data:

```
1  dataopt = 2;
2
3  switch dataopt
4
5      case 1 % Telewavesim Synthetics
6
7          RFmat = load(strcat(FADERDir,'Data/Synthetics/M1_1.0Hz_syn.mat'));
8          R = RFmat.rRF;
9          t = RFmat.time; y = RFmat.garc; nY = length(y);
10         R1 = R(1,:);
11         [epiDist, rayP] = raypToEpiDist(y, 1, localBaseDir);
12
13     case 2 % Real Data from MTC RF
14
15         network = 'YP';
16         station = 'NE68';
17         epiDistRange = [30 95];
18         resolutionFactor = 1;
19         RFOUTDIR  = [FADERDir 'Data/MTCRF/'];
20         [rrfAmpArray, timeAxisHD, binAxisHD] = ...
21             loadAndPrepRF(network, station, resolutionFactor, epiDistRange, RFOUTDIR)
    ;
22         R = rrfAmpArray; t = timeAxisHD; y = binAxisHD; nY = length(y);
23         R1 = R(1,:);
24         [epiDist, rayP] = raypToEpiDist(y, 1, FADERDir);
25
26 end
```

For both synthetic and real RF, the final required input data are the following:
R (line 8 and 22): RF traces in matrix of size [number of traces] by [sample points].
t (line 9 and 22): Time vector containing relative time to the first P arrival at each sample point in seconds. Length should match number of columns in **R**.
rayP (line 11 and 24): Ray parameter vector containing ray parameter of each RF trace. Length should match number of rows in **R**.

[1] Audet, P., Thomson, C.J., Bostock, M.G., and Eulenfeld, T. (2019). Telewavesim: Python software for teleseismic body wave modeling. Journal of Open Source Software, 4(44), 1818.
[2] Olugboji, T. M., Park, J. (2016). Crustal anisotropy beneath Pacific Ocean-Islands from harmonic decomposition of receiver functions. Geochemistry, Geophysics, Geosystems, 17(3), 810–832.
[3] **SynRFWorkflow** is a MatLab package which reads in the velocity sturctures and ray parameters, and outputs the synthetic RFs in MatLab structure file format using Telewavesim.

Notes:

- In the data provided, **y** (line 9 and 22) is the epicentral distance vector. Function **raypToEpiDist** is used to convert epicentral distance to ray parameter.
- Only one trace will be used for the detection (and tuning the filter parameters) at current version; the default setting is to use the first trace (line 10 and 23).

We provide the following synthetic and real RF data in the package:

- Synthetic RF generated by Telewavesim, using models M0, M1, M2, and M3 in (Zhang and Olugboji, 2022)[4] as shown in Figure 2.
- Real RF calculated by MTC algorithm, from station YP.NE68 in Songliao Basin, Northeastern China, station XO.LT10 from the AACSE array in the Aleutian subduction zone, stations from Earthscope transportable array in the Mississippi embayment.

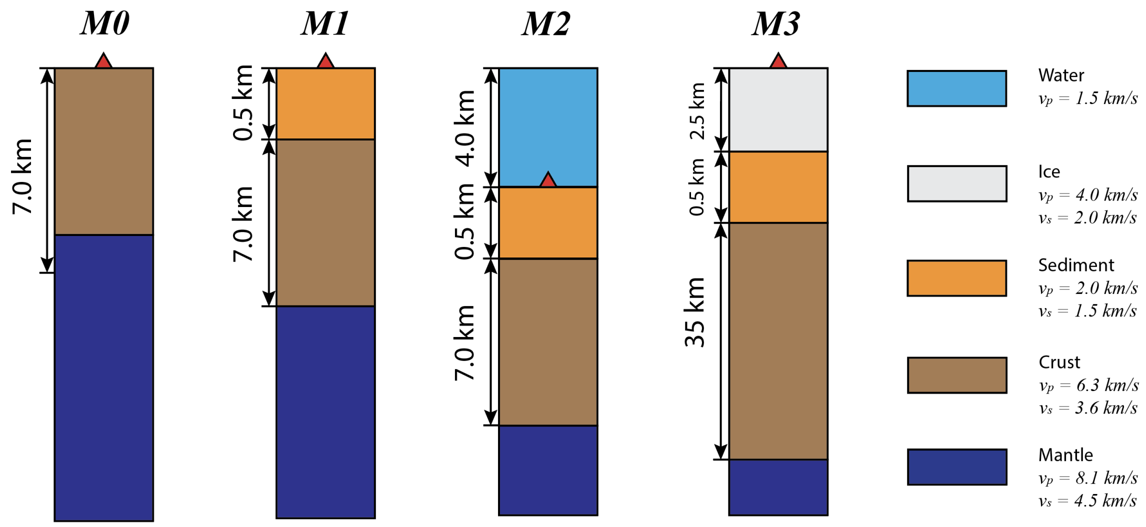

Figure 2: Velocity models used in synthetic RF. M0: no reverberation, M1: sediment reverberation, M2: water and sediment reverberations, M3: ice and sediment reverberations. Red triangles indicate station locations.

2.3 Reverberation Detection using Autocorrelation Analysis

The input RFs will be passed to autocorrelation analysis to detect the presence of reverberation, as well as tune the filter parameters. Synthetic RF from model M1 will be used to demonstrate the workflow here.
This step is listed as **Step 2. and 3.** in the main code:

```
[SedPre,tlaga,r0,tac,ac,dsin] = ...
    DeterminationTest_func(R1,t,kd_thresh,man_determine);

if SedPre == 0
    if Hkopt == 0
        fprintf('No reverberation detected. Exiting FADER.\n');
```

[4]Zhang, Z., & Olugboji, T. (2022). Lithospheric Imaging through Reverberant Layers: Sediments, Oceans, and Glaciers. Submitted to JGR Solid Earth.

```
    else
        tlag = 0;
        tPbS = 0;
        fprintf('No reverberation detected. Will run H-k stacking directly.\n');
    end
else
    fprintf('Reverberation detected. Analyzing ...\n');
```

The output **SedPre** (line 1) indicates the presence of reverberations. Note that Filter parameter r_0 (**r0** in line 1) and $\tilde{\tau}$ (**tlaga** in line 1) are also obtained in this step.

If **man_determine** is set to 0, the program will automatically determine the presence of reverberations based on the user specified threshold $k_t hr$. If **man_determine** is set to 1, the user will be asked to visually inspect the observed and fitted autocorrelation curves to determine the presence of reverberations:

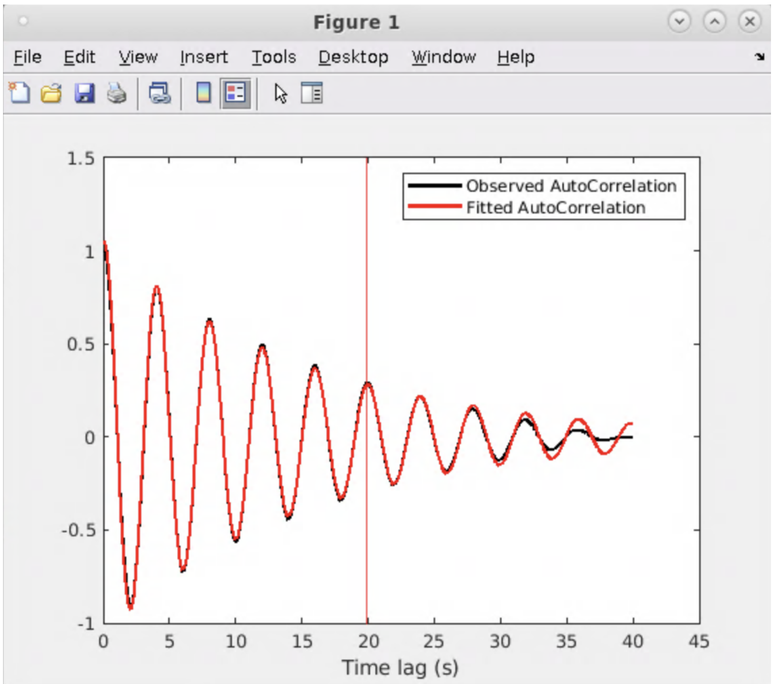

Figure 3: Observed and fitted autocorrelations curves for user inpection.

The following message will be displayed in the command window:

```
Please see Figure 1.
kd = 24.05, Echo Quality Factor: 1  (Indicates echo)
Manually determine if reverberation presents.
1 = Yes, 0 = No:
```

In this case, the user is expected to enter "1" in the command window to confirm the presence of reverberations. The following message will then be displayed:

```
Reverberation detected. Analyzing ...
```

On the contrary, if the user (or the program if **man_determine** is set to 0) determines that there is no reverberation in the data, and **Hkopt** is set to 0, the following message will be displayed:

```
No reverberation detected. Exiting FADER.
```

If the **Hkopt** is set to 1, the following message will be displayed and H-k stacking will be performed on the original input RFs:

```
No reverberation detected. Will run H-k stacking directly.
```

2.4 Confirming Echo Delay Time using Homomorphic Analysis

The input RFs will then be passed to homomorphic analysis to further confirm the echo delay time estimation. This step is listed as **Step 4.** in the main code:

```
if tlaga < 1
        twin_cstack = [0 2];
    else
        twin_cstack = [tlaga-1 tlaga+1];
    end

    [tlagc,cstackt,cstackA] = ceps_func(R1,t,twin_cstack);

    if abs(tlagc - tlaga) < tolerance * max(tlagc, tlaga)
        tlag = 0.5 * (tlagc + tlaga);
    else
        if man_determine
            tlag = tlag_check(tac,ac,dsin,tlaga,cstackt,cstackA,tlagc,tolerance);
        else
            tlag = tlaga;
        end
    end

    fprintf('tlag = %3.2f s, r0 = %3.2f.\n',tlag,r0);
```

If **man_determine** is set to 0 and the difference between **tlaga** (echo delay time estimated from autocorrelation analysis) and **tlagc** (echo delay time estimated from homomorphic analysis) is smaller than **tolerance**, the final accepted echo delay time will be tuned as the average of the two. If **man_determine** is set to 0 and the difference between **tlaga** and **tlagc** is larger than **tolerance**, then **tlaga** will be accepted as the final echo delay time.

If **man_determine** is set to 1, the following message will be displayed in the command window, and the user will be asked to visually inspect the following plots from autocorrelation and homomorphic analysis, respectively:

```
Please see Figure 2.
Two-way travel time obtained from autocorrelation
and cepstrum stack are off more than tolerance ( 0 %).
Autocorrelation: 1.99 s;
Cepstrum stack : 1.98 s.
Please manually input two-way travel time:
```

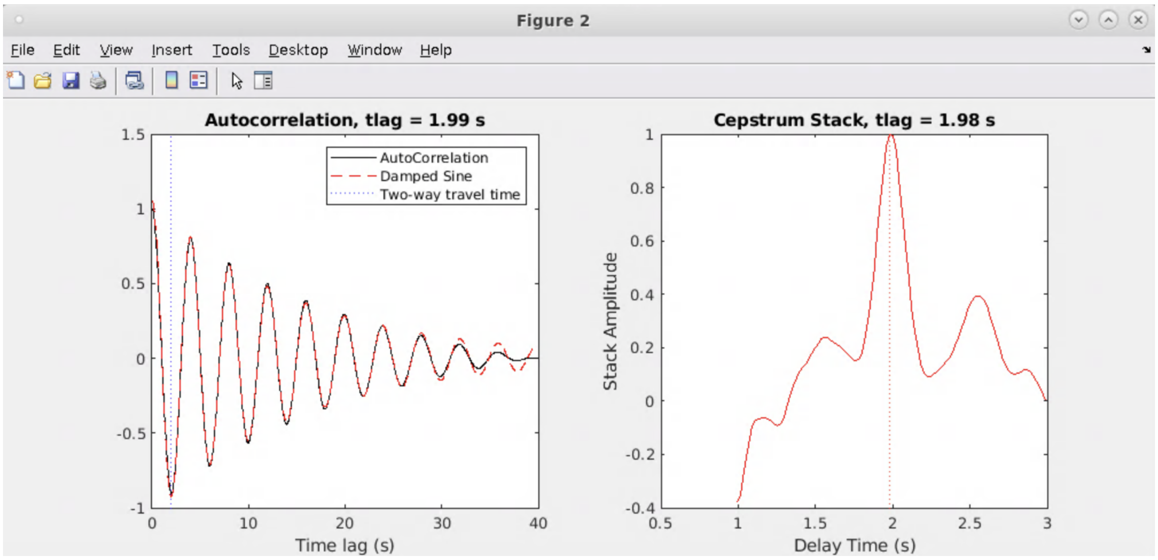

Figure 4: Observed and fitted autocorrelations curves and cepstrum stack for user inpection.

The user is expected to manually input the echo delay time in the command window, in this case, 1.99 s. The following message will be displayed in the command window to confirm the obtained filter parameters:

```
tlag = s, r0 = .
```

2.5 Filtering RFs

This step is listed as **Step 5.** in the main code:

```
fprintf('\nFiltering data ...\n');
R_flted = filterRF_FADER(R,t,tlag,r0,smoothopt);

if saveRF
    fprintf('Saving filtered RF ...\n');
    nname = strcat(saveDir,savename,'.mat');
    save(nname,'R','R_flted','t','y');
end
```

The input RFs will be filtered using the parameters tuned in the previous steps. If **saveRF** is set to 1, both the original and the filtered RFs will be saved into a MatLab structure file in **filteredRF/** folder.

2.6 Optional H-k Stacking

This step is optional as per user specifies in the parameters setup, and is listed as **Step 6.** in the main code:

```
if Hkopt

    if SedPre == 1
        tPbS = tPbS_Confirm(R, t, y, man_PbS);
        R2stack = R_flted;
```

```
    else
        R2stack = R;
    end

    [stackArray, Hrange, krange, HBest, kBest] = ...
        HkStacking(R2stack, t, rayP, Vp, Hwin, kwin, resfac, pWAc, wid, ...
        SedPre, tlag, tPbS,1);

end
```

If reverberations were detected previously and **man_PbS** is set to 1, user will be required to manually inspect the RF traces to confirm the arrival time of PbS phase. If **man_PbS** is set to 0, PbS arrival will be automatically picked.

3 Usage: Multi Station Detection

Detection of reverberations on multi station (array) data is performed by **FADERWorkflow_Array_Detection.m**.

3.1 Parameters Setup

This step is listed as **Step 0.** in the main code:

```
1 FADERDir = '/scratch/tolugboj_lab/Prj4_Nomelt/FADER/';
2 addpath([FADERDir 'Functions/']);
3
4 kd_thresh = 2;
5 man_determine = 0; % mannually check autocorrelation plot
6
7 saveDir = strcat(FADERDir, 'Array/');
```

Line 1, 4, and 5 need to be properly set up as instructed in section 2.1.

3.2 Load in RF Data

This step is listed as **Step 1.** in the main code:

```
network = 'TA';
stalist = {'S38A',...};

for ista = 1:length(stalist)

    station = char(stalist(ista));
    fprintf('Analyzing station %s\n',station);

    epiDistRange = [30 95];
    resolutionFactor = 1;
    RFOUTDIR  = [FADERDir 'Data/MTCRF/'];
    [rrfAmpArray, timeAxisHD, binAxisHD] = ...
        loadAndPrepRF(network, station, resolutionFactor, epiDistRange, RFOUTDIR);
    R = rrfAmpArray; t = timeAxisHD; y = binAxisHD; nY = length(y);

    % use the first available (non-nan) trace
```

```
17     jr = 1;
18     for ir = 1:size(R,1)
19         if sum(isnan(R(ir,:))) < 1 && sum(abs(R(ir,:)) < 1e-3)
20             R1 = R(ir,:);
21             break;
22         end
23         jr = jr + 1;
24     end
25
26     if jr > size(R,1)
27         continue;
28     end
29
30     [epiDist, rayP] = raypToEpiDist(y, 1, FADERDir);
31     hh = RFWigglePlot_any(R, t, y);
32     title([station ' 1st Available Trace: ' num2str(jr)]);
33
34     if ~exist([FADERDir 'Array/' network '/RF/'],'dir')
35         mkdir([FADERDir 'Array/' network '/RF/']);
36     end
37
38     saveas(hh,[FADERDir 'Array/' network '/RF/' station '_RF.pdf']);
```

MTC RFs at TA (transportable array) are used as an example here. Note that this part can be modified to read in your own data.

Line 9 - 14: Same as Line 17 - 21 in section 2.2, but now included in a for loop to read in RFs from each station.

Line 17 - 28: Find the first available RF trace (which contains no NAN sample points). This part is added because of the empty traces in the RF files due to bad epicentral distance coverage.

Line 31 - 38: Input RFs are saved as wiggle plots at **Array/[network]/RF/[station]_RF.pdf**.

3.3 Reverberation Detection using Autocorrelation Analysis

This step is listed as **Step 2.** in the main code:

```
[SedPre,tlaga,r0,tac,ac,dsin,Qe,kd] = ...
        DeterminationTest_func(R1,t,kd_thresh,man_determine);

f1 = figure(1);
clf;
p1 = plot(tac,ac,'k-','DisplayName','Observed AutoCorrelation','linewidth',2);
hold on;
p2 = plot(tac,dsin,'r-','DisplayName','Fitted AutoCorrelation','linewidth',2);
xlabel('Time lag (s)');
legend([p1 p2]);

title(sprintf('Station %s, kd = %4.2f, Qe = %1d',station, kd, Qe));

if ~exist([FADERDir 'Array/' network '/ac/'],'dir')
    mkdir([FADERDir 'Array/' network '/ac/']);
end

saveas(f1,[FADERDir 'Array/' network '/ac/' station '_ac.pdf']);

% write kd & Qe into text files
if ista == 1
```

```
fid = fopen([saveDir network '.txt'],'a+');
fprintf(fid,'%6s %2s %2s %4s %4s\n',"Station","Qe","kd","r0","tlag");
end

fprintf(fid,'%6s %2d %4.2f %4.2f %4.2f\n',char(station),Qe,kd,r0,tlaga);
```

This step performs the reverberation detection described in section 2.3, and saves the observed and fitted autocorrelation curves as **Array/[network]/ac/[station]_ac.pdf** for each station. It also outputs a text file **Array/[network].txt** containing the detection results, i.e., echo number k_d, echo quality factor Q_e, reverberation strength r_0, and echo delay time $\tilde{\tau}$ (from autocorrelation analysis).