Filter and Deconvolution

Jonathan M. Lees
University of North Carolina, Chapel Hill
Department of Geological Sciences
CB #3315, Mitchell Hall
Chapel Hill, NC 27599-3315
email: jonathan.lees@unc.edu
ph: (919) 962-1562

July 22, 2012

1 Signal Package

There is a package in R called signal that replicates the functionality of the signal processing toolbox in MATLAB. There are many features in the signal package that are useful and can be applied in slightly different ways than the implementation presented in RSEIS.

For example, consider the creation and application of a butterworth filter. In signal,

Figure 1:

```
library(RSEIS)
library(signal)
    bf <- butter(3, 0.1)                          # 10 Hz low-pass filter
     t <- seq(0, 1, len = 100)                     # 1 second sample
     x <- sin(2*pi*t*2.3) + 0.25*rnorm(length(t))# 2.3 Hz sinusoid+noise
     y <- filtfilt(bf, x)
     z <- filter(bf, x) # apply filter
 zz = butfilt(x, fl=0, fh=10,  deltat=1/100,  type="LP" ,  proto="BU")
     plot(t, x, type='l')
     lines(t, y, col="red")
     lines(t, z, col="blue")
   lines(t, zz, col="purple")
   legend("bottomleft", legend = c("data", "filtfilt", "filter", "butfilt"),
            pch = 1, col = c("black", "red", "blue", "purple"), bty = "n")
```

Figure 2:

```
library(signal)
    bf <- butter(2, 0.1)                          # 10 Hz low-pass filter
     t <- seq(0, 1, len = 100)                     # 1 second sample
     x <- rep(0, times=length(t))
     x[floor(length(x)/2)] = 1
     y <- filtfilt(bf, x)
     z <- filter(bf, x) # apply filter
 zz = butfilt(x, fl=0, fh=10,  deltat=1/100,  type="LP" ,  proto="BU", npoles=2)
     plot(t, x, type='l')
     lines(t, y, col="red")
     lines(t, z, col="blue")
```

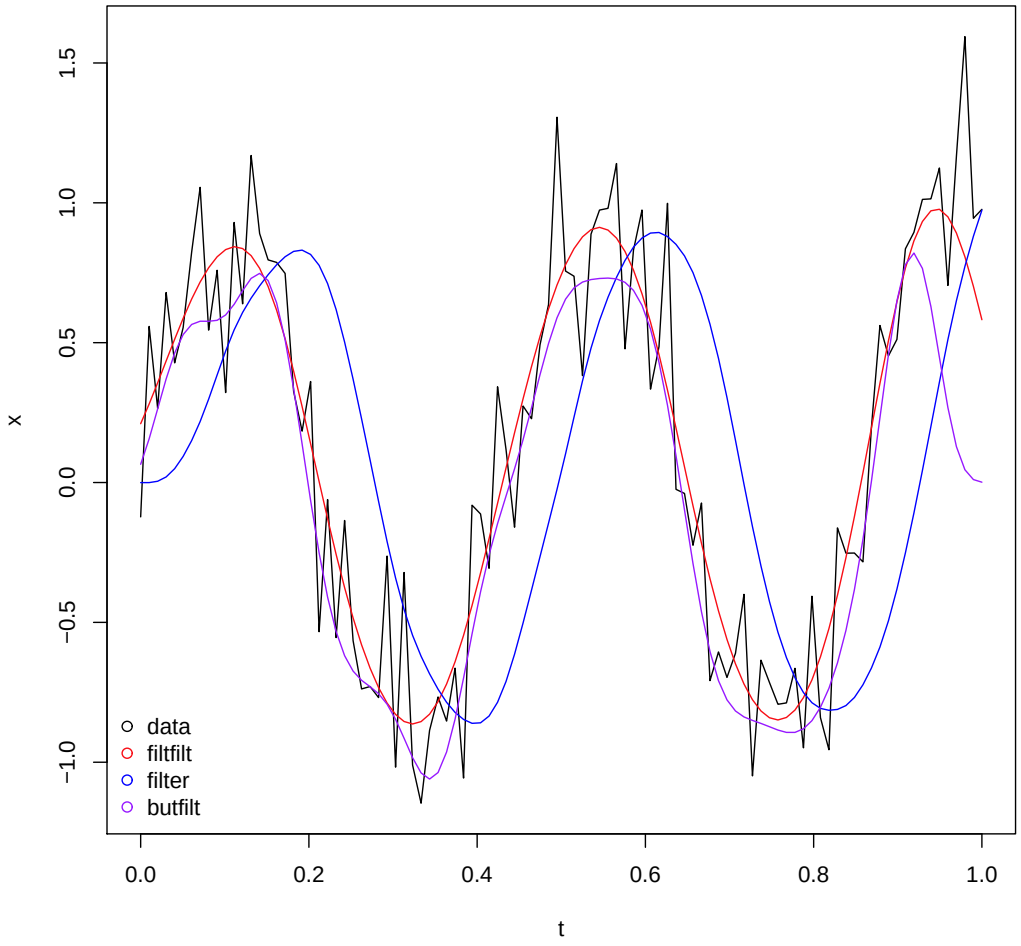

Figure 1: Butterworth filter applied with filtfilt, filter and butfilt.

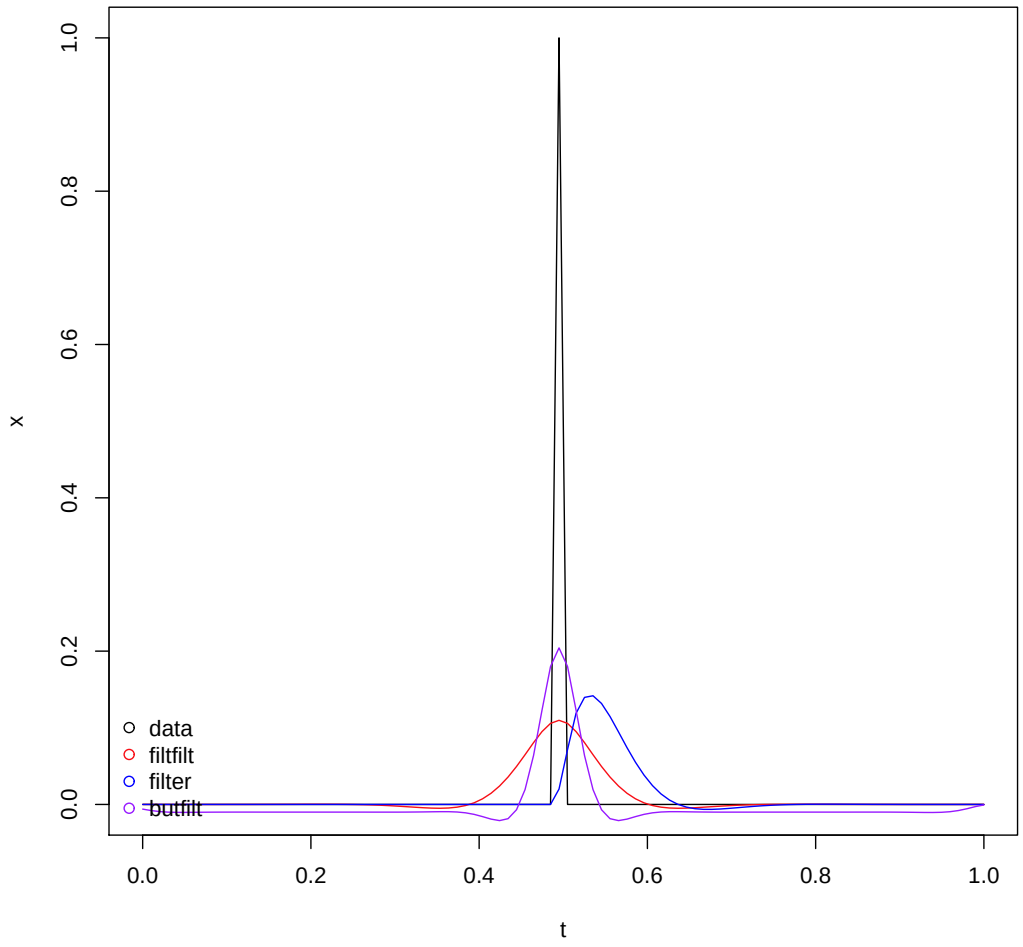

Figure 2: Butterworth filter applied to an impulse: Impulse response functions.

```
lines(t, zz, col="purple")
legend("bottomleft", legend = c("data", "filtfilt", "filter", "butfilt"),
       pch = 1, col = c("black", "red", "blue", "purple"), bty = "n")
```

Figure 3:

```
library(signal)
                    # 10 Hz low-pass filter
    t <- seq(0, 1, len = 100)                    # 1 second sample
    x <- rep(0, times=length(t))
```

```
     x[floor(length(x)/2)] = 1
         bf2 <- butter(2, 0.1)
y2 <- filtfilt(bf2, x)
         bf3 <- butter(3, 0.1)
y3 <- filtfilt(bf3, x)
         bf4 <- butter(4, 0.1)
y4 <- filtfilt(bf4, x)
     plot(t, x, type='l')
     lines(t, y2, col="red")
     lines(t, y3, col="blue")
   lines(t, y4, col="purple")
   legend("bottomleft", legend = c("data", "2-poles", "3-poles", "4-poles"),
            pch = 1, col = c("black", "red", "blue", "purple"), bty = "n")
```

Figure 3:

```
library(signal)
                    # 10 Hz low-pass filter
     t <- seq(0, 1, len = 100)                    # 1 second sample
     x <- rep(0, times=length(t))
     x[(floor(0.25*length(x))):(floor(0.75*length(x)))] = 1
         bf2 <- butter(2, 0.1)
y2 <- filtfilt(bf2, x)
         bf3 <- butter(3, 0.1)
y3 <- filtfilt(bf3, x)
         bf4 <- butter(10, 0.1)
y4 <- filtfilt(bf4, x)
     plot(range(t), range(x, y2, y3, y4) , type='n')
lines(t, x, type='l')
     lines(t, y2, col="red")
     lines(t, y3, col="blue")
   lines(t, y4, col="purple")
   legend("bottomleft", legend = c("data", "2-poles", "3-poles", "4-poles"),
            pch = 1, col = c("black", "red", "blue", "purple"), bty = "n")
```

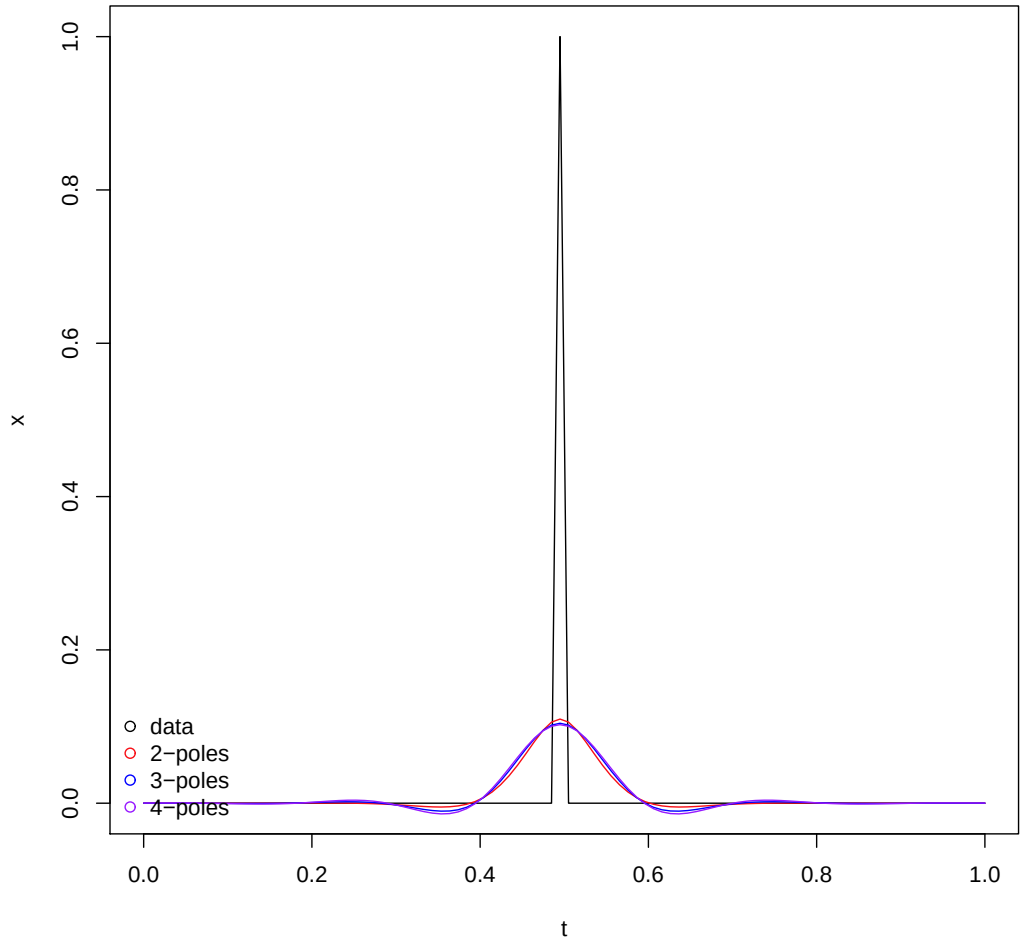

Figure 3: Butterworth filter applied to an impulse: Impulse response functions.

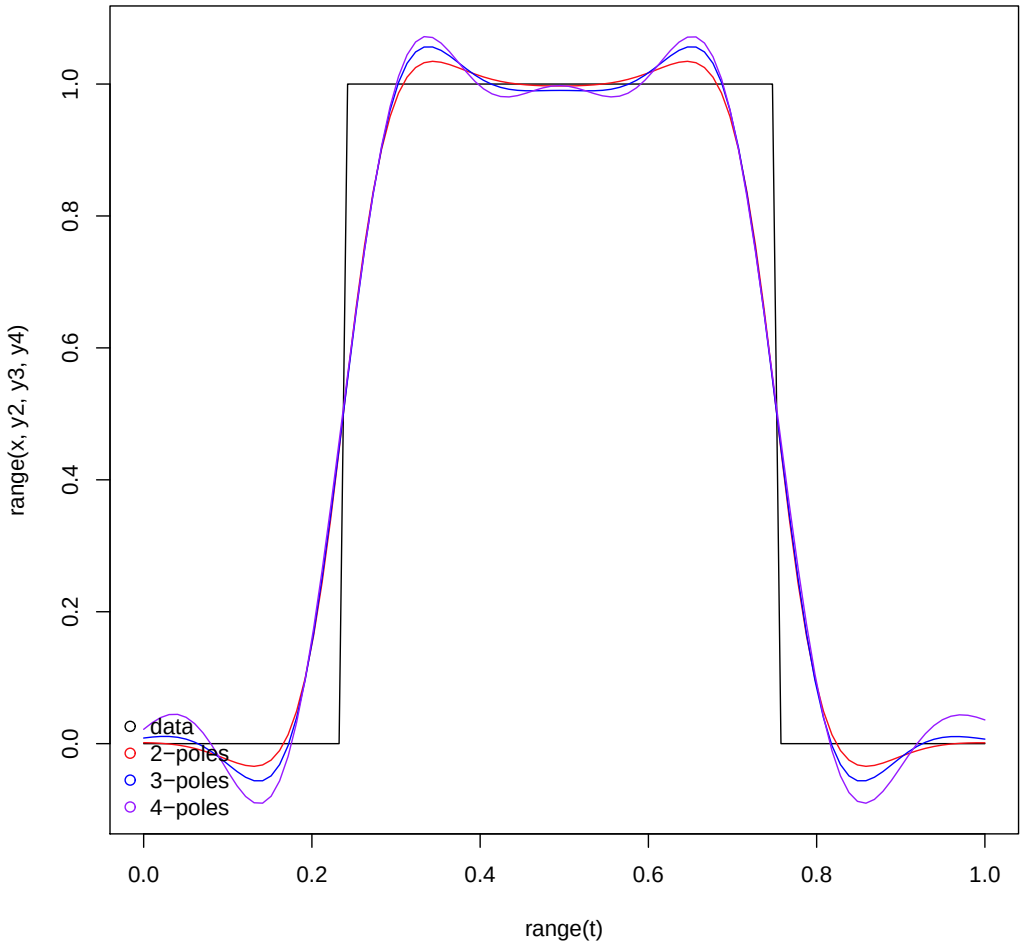

Figure 4: Butterworth filter applied to an impulse: Impulse response functions.

Deconvolution of seismic data can be accomplished by several methods. In this document I will relate steps to remove the instrutment response of a seismic recording.

2 Poles and Zeros

In package **REIS** we use the convention originally established in the SAC software system. In SAC poles and zeros are defined along with a set of scaling parameters that are applied to the traces. To correctly extract the proper units on the output care must be taken to make sure all the scaling parameters are applied correctly.

Get some data in the correct RSEIS format.

2.1 Reading the data

There are a number of seismic sensors that are somewhat standard and they are already coded into **REIS** . You can access these with the preset program:

If the instrument you are using is not listed among the preset instruments or if you know that the parameters you need do not match the ones that are preset you can read in your own parameters by following the simple example:

In the case we just create a character vector with the correct information. Note that these are chacter strings, as if they were extracted from a file. The program can read these in

Normally instrument response information will be stored in a file on disc, and can be read in and stored properly ReadSet.Instr. Here is an example reading in the response for a CMG3T instrument:

```
Rcmg3t = ReadSet.Instr('./data/CMG3T.inst.response')
```

The number of poles and zeros are decleared and a couple of constnts, such as the normalization factor and the sensitivity are provided.

To load the preset instruments into an R session the command PreSet.Instr can be used. This just sets up a list of instruments. The list can be extended by adding new instrument definitions by the user.

```
[1] "40T"      "3T"       "L28"      "LE3D20s"  "GEOSP1"
[6] "CMG3ESPC" "60T"      "TRIL120"
```

3 Deconvolution and Velocity

The seismic instrument recording the data in example data KH was manufactured by the Guralp company in the UK. The model was a CMG40T commonly used on volcano studies.

3.1 Counts to Volts

The data was recorded on a RefTek data acquisition system (DAS) in the field and signals were stroed on a flash drive. When the data is recorded the raw data is saved as "counts". The DAS manufacturers typically provide a conversion facor for converting DAS counts to volts. This is saved in the header files of the seismic data and in **REIS** removed when the data is read into memory.

Data conversion programs *JSAC.seis* and *JSEGY.seis* apply the counts-to-volts conversion internally. If there is concern that this is not being down correctly, the feature can be turned off and conversion can be down outside of the I/O.

3.2 Decon

```
data(KH)
 Kal = PreSet.Instr()
     inst = rep(0, length(KH$STNS))
     inst[KH$STNS=="9024"] = 1
     VH =    VELOCITY.SEISN(KH, sel = 1:length(KH$JSTR), inst = inst, Kal = Kal)
```

The program VELOCITY.SEISN uses the sel and inst parameters to determine whether to apply the filters and what instruments should be used. In this case only one traces was there so, it was fairly simple.

3.3 Waterlevel

The method used here to accomplish the deconvolution is called the waterlevel method. Since deconvolution involves spectral deconvolution small values in the denomenators will result in large, possibly undesirable fluctuations in the output. Normally this would be considered noise and it is best to suppress, or regularize, the division by adding a small number to the values so they reach a stable "waterlevel". The waterlevel value has a default value of $1e-08$ but this can be changed at run time. Raising the waterlevel value will result in smoothing of the resultant signal.

4 Acceleration

If it is desired to remove the isntrument response and convert to acceleration the velocity data will need to be differntiated. Currently there is no function in **REIS** for doing this automatically at this time, but differentiation is easily done by any simple differencing.

5 Displacement

If it is desired to remove the instrument response and convert to displacement, the velocity data will need to be integrated. This combined procedure can be accomplished using the DISPLACE.SEISN function. This function is executed in the same manner as VELOCITY.SEISN, accept here a filt should be applied to remove the very long period trends.

```
data(KH)
 Kal = PreSet.Instr()
     inst = rep(0, length(KH$STNS))
     inst[KH$STNS=="9024"] = 1
     DH =    DISPLACE.SEISN(KH, sel = 1:length(KH$JSTR), inst = inst, Kal = Kal)
##   swig(DH, PADDLAB=c("CENTER", "fspread", "HALF", "PREV"))

dis1 = butfilt(DH$JSTR[[1]], fl=1/60, fh=8, deltat=DH$dt[1], type="HP", proto="BU",
        npoles=2)
```

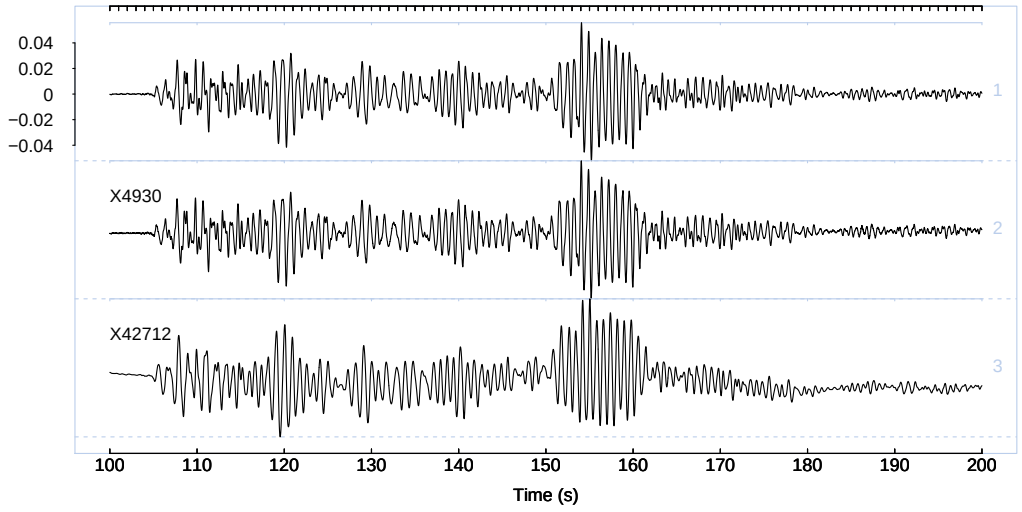

Figure 5: Deconvolved data: top: raw data in volts, center: velocity data in m/s; bottom: displacement data in meters.

```
JPOST(file="./FIGS/decon2.eps", width=10, height=6)
## PLOT.MATN(cbind(KH$JSTR[[1]], VH$JSTR[[1]], DH$JSTR[[1]]),  dt = KH$dt[1])

## PLOT.MATN(cbind(KH$JSTR[[1]], VH$JSTR[[1]], dis1),  dt = KH$dt[1])

units = c("Volts", "m/s", "m" )
PLOT.MATN(cbind(KH$JSTR[[1]], VH$JSTR[[1]], dis1),  dt = KH$dt[1],
          WIN=c(100, 200), units=units )
dev.off()
```