

Supplementary Information

Claire Longo *, Tom Burr and Kary Myers

Los Alamos National Laboratory, Los Alamos, NM 87545, USA; E-Mails: tburr@lanl.gov (T.B.); kary@lanl.gov (K.M.)

* Author to whom correspondence should be addressed; E-Mail: clongo01@unm.edu; Tel.: +1-505-500-5916; Fax: +1-505-667-4470.

Received: 24 April 2013; in revised form: 17 May 2013 / Accepted: 22 May 2013 /

Published: 18 June 2013

Abstract: Wavelet analysis is known to be a good option for change detection in many contexts. Detecting changes in solution volumes that are measured with both additive and relative error is an important aspect of safeguards for facilities that process special nuclear material. This paper qualitatively compares wavelet-based change detection to a lag-one differencing option using realistic simulated solution volume data for which the true change points are known. We then show quantitatively that Haar wavelet-based change detection is effective for finding the approximate location of each change point, and that a simple piecewise linear optimization step is effective to refine the initial wavelet-based change point estimate.

Keywords: change detection; Haar wavelet; piecewise linear

```
## R code to illustrate wavelet change detection with simulated SM data ##
# Steps:
#(1) Use wavelets to detect change point regions
#(2) Use breakpoints in the wavelet-detected change point regions to
#     get initial estimate of start index and stop index of change region.
#(3) Use optimize to refine initial estimates of start and stop index.

## First, create simulated data ##
generate.data = function(n=256, cpoints=c(100, 130, 200, 220), cvals=c(20, 30, 30),
sigma.meas.add=1, sigma.meas.rel=0.015) {
yvals <- numeric(n)
yvals[1:(cpoints[1]-1)] = cvals[1]
yvals[cpoints[1]:cpoints[2]] = yvals[cpoints[1]-1] +
cvals[2]*(cpoints[1]:cpoints[2]-cpoints[1])/(cpoints[2]-cpoints[1])
yvals[(cpoints[2]+1):(cpoints[3]-1)] <- yvals[cpoints[2]]
```

```

yvals[cpoints[3]:cpoints[4]] = yvals[cpoints[3]-1] -
cvals[3]*(cpoints[3]:cpoints[4]-cpoints[3])/(cpoints[4]-cpoints[3])
yvals[(cpoints[4]+1):n] <- yvals[cpoints[4]]
yvals = yvals*(1+sigma.meas.rel*rnorm(n=length(yvals))) +
sigma.meas.add*rnorm(n=length(yvals))
yvals}
y = generate.data()

```

Steps 1-3:

(1) High resolution wavelets to detect change point region.

```

yscale = y/(y^2*sigma.rel^2 + sigma.add^2)^.5
temp.haar = dwt(yscale,"haar",n.levels=9)
temp = up.sample(temp.haar[[4]], 2^4) # use for wavelet change detect
# experimented with various resolutions. See Figure 3.
## Check for outliers using Studentized coefficients ##
temp[is.na(temp)] = 0
sse = cov.mve(temp[temp!=0])$cov
student.C = temp/sse^.5
outliers = which(abs(student.C) > a.wavelet.threshold)
if(length(outliers)) {
outliers = (1:n)[outliers]
}
CP = outliers

```

CP has the approximate location of each change point region

(2) Use breakpoints to get initial estimate of start and stop index

of each change point region

```

library(strucchange)
rec1.temp = min(CP); shipl.temp = max(CP)
indices = (rec1.temp-k2):(rec1.temp+k1)
ytemp = yscale[indices]
temp4 = breakpoints(ytemp ~ 1, breaks=2,h=2)
rec1.temp = indices[temp4$breakpoints[1]]
rec2.temp = indices[temp4$breakpoints[2]]
# repeat for th2 (end of receipt), and also repeat for th3 (start of shipment), and th4 (start of shipment)

```

(3) Refine rec1.temp and rec2.temp using optimize or multi-scale wavelets

k = 7

k = 7 is a user choice based on experiments with modest-sized windows

around breakpoints estimate.

```
temp.indices = (rec1.temp-k):(rec1.temp+k)
```

Refine using optimize:

```
th1 = optimize(fnw, temp.indices, x = temp.indices, y=y[temp.indices],
sigma.add=sigma.add,sigma.rel=sigma.rel)$minimum
```

```

# Refine using multiscale wavelets:
temp.indices = (recl.temp-7):(recl.temp+8) # need 2^j points, so used 16 points
temp.haar =dwt(yscale[temp.indices],"haar")
ntemp = length(temp.indices)
temp.sig = (2*sigma.add^2 + yscale[2:ntemp]^2*sigma.rel^2 +
yscale[1:(ntemp-1)]^2*sigma.rel^2)^.5
temp1 = c(0,diff(yscale[temp.indices])/temp.sig)
temp2 = up.sample(temp.haar[[1]], 2^1) # use for wavelet change detect
temp3 = up.sample(temp.haar[[2]], 2^2) # use for wavelet change detect
temp4 = up.sample(temp.haar[[3]], 2^3) # use for wavelet change detect
temp2[is.na(temp2)] = 0;temp3[is.na(temp3)] = 0
temp4[is.na(temp4)] = 0
temp1a.indices = min((1:ntemp)[abs(temp1)==max(abs(temp1))])
temp2a.indices = min((1:ntemp)[abs(temp2)==max(abs(temp2))])
temp3a.indices = min((1:ntemp)[abs(temp3)==max(abs(temp3))])
temp4a.indices = min((1:ntemp)[abs(temp4)==max(abs(temp4))])
temp1a = temp.indices[temp1a.indices];temp2a =temp.indices[temp2a.indices]
temp3a = temp.indices[temp3a.indices];temp4a = temp.indices[temp4a.indices]
#th1.mswavelet = mean(temp1a,temp2a,temp3a,temp4a)
th1.mswavelet = min(temp1a,temp2a,temp3a,temp4a)
# Trial-and-error with off-line simulations to find
# effective method to combine temp1a-temp4a
# Any modern machine learning option can be considered, such as k-nearest-nbr.
# Choosing the smallest of the 4 estimates works well for first of 2 change
# points and choosing the largest of the 4 estimates works well for second of 2.
# However,this depends on the length of the search indices and the location of
# the true event.
th1.mswavelet = min(temp1a,temp2a,temp3a,temp4a)
# Remark: It is useful to evaluate temp1a-temp4a for
# measurements that have with zero measurement error.
# For example, when the true start index of the first receipt is 500
# and the search indices range from 495 to 510, temp1a-temp4a
# are 501, 501, 507, and 503, respectively, and these four values
# vary slightly as measurement error is added.
# When the true start index of the first receipt is 501 and the
# search indices have the same 495 to 510 range, temp1a-temp4a are
# 507, 503, 507, and 503 without measurement error in y.

```

Note: The “find-one-changepoint-at-a-time” version of breakpoints is easy to implement in R, highly accurate, and very fast. After finding the approximate location of a changepoint in a contiguous range of indices, tempindices, call the optimize function using:

```

th = optimize(fnw, interval=tempindices, x = tempindices, y = y[temp.indices])$minimum
with the function fnw defined as

```

```
fnw = function(th, x, y, sigma.rel=0.015, sigma.add=1) {# conditional minimum SSQ given
theta
  X = cbind(x, pmax(0, x - th))
  wts = 1/(sigma.add^2 + y^2*sigma.rel^2)
  sum(lsfrit(X, y, wt=wts)$resid^2)
} # contributed to Rnews by W. Venables
```

© 2013 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/3.0/>).