FFTWTOOlS Timing, and Simple Example (Version 0.9-2)

Karim J. Rahim

May 3, 2013

1 Timing Example

This is a simple example demonstrating comparing timing of `fftw` in the package `fftwtools` and showing how to replace the `R` function `fft` with `fftw` if you so choose. We begin with a quick demonstrating that speed difference in the default case. The performance improvement is only visible with large data sets. In this example I am using one million data points. You can test timing at any size and decide which function to use. Also the package `FFTW` package allows you to specify plans which should improve performance if multiple transforms are done of a data set of the same size.

First we look at the time required for he default `fft` routine.

```
> library("fftwtools")
> set.seed(10)
> ## we try power of 2 but we can try other values
> ## we do ffts of 2^20 points
> g <- rnorm(2^20)
> ##timing # Start the clock!
> ptm <- proc.time()
> # Loop through
> for (i in 1:100){
+     fft(g)
+ }
> # Stop the clock
> proc.time() - ptm

   user  system elapsed
 16.489   0.000  16.528

>
```

Next we look at replacing `fft` with `fftw` without any other changes.

```
> ##timing # Start the clock!
> ptm <- proc.time()
> # Loop through
> for (i in 1:100){
+     fftw(g)
+ }
> # Stop the clock
> proc.time() - ptm

   user  system elapsed
  8.116   0.000   8.136
```

Finally we look to see how much additional improvement can by had by not returning the complex conjugate which is not required for real data. This speed up is likely due to decreased memory allocation.

```
> ##timing # Start the clock!
> ptm <- proc.time()
> # Loop through
> for (i in 1:100){
+     fftw(g, HermConj=FALSE)
+ }
> # Stop the clock
> proc.time() - ptm

   user  system elapsed
  7.517   0.000   7.532
```

2 Replace R's `fft` call with `fftw`

I do recommend you do this in general, but it may be an easy way to speed up code or code in packages that call `fft` by replacing all `fft` calls with `fftw`. This may be of use when working with large data sets.

```
> ## basic option ot overwrite calls
> fft <- function(z, inverse = FALSE) {
+     fftwtools::fftw(z, inverse=inverse)
+ }
> mvfft <- function(z, inverse=FALSE) {
+     fftwtools::mvfftw(z, invese=inverse)
+ }
```

If you are interested in the additional improvement had not returning the complex conjugate in real data, you can overwrite the call in the following manner:

```
> fft <- function(z, inverse = FALSE) {
+     fftwtools::fftw(z, inverse=inverse, HermConj=FALSE)
+ }
```

The last method may break certain calls depending on if the complex conjugate, or at least the length of the original real data, is required to use an inverse `ttt`. So if you are doing the latter you should be more careful and may want to look into the other functions provided in the packages `FFTW` and `fftwtools`.

2.1 Clean up

If you replace the `R`'s call to `fft` with `fftw`, it is good practice to clean up the replacement, and return calls to `fft` and `mvfft` to the standard `R` routine when you are finished using `fftw`.

```
> rm(fft, mvfft)
```