Computing Fourier Series Numerically For Simple Functions Using R

Lewis Peaty

25 March 2016

Here is an example of how to compute and plot Fourier Series for simple functions using only the standard R library.

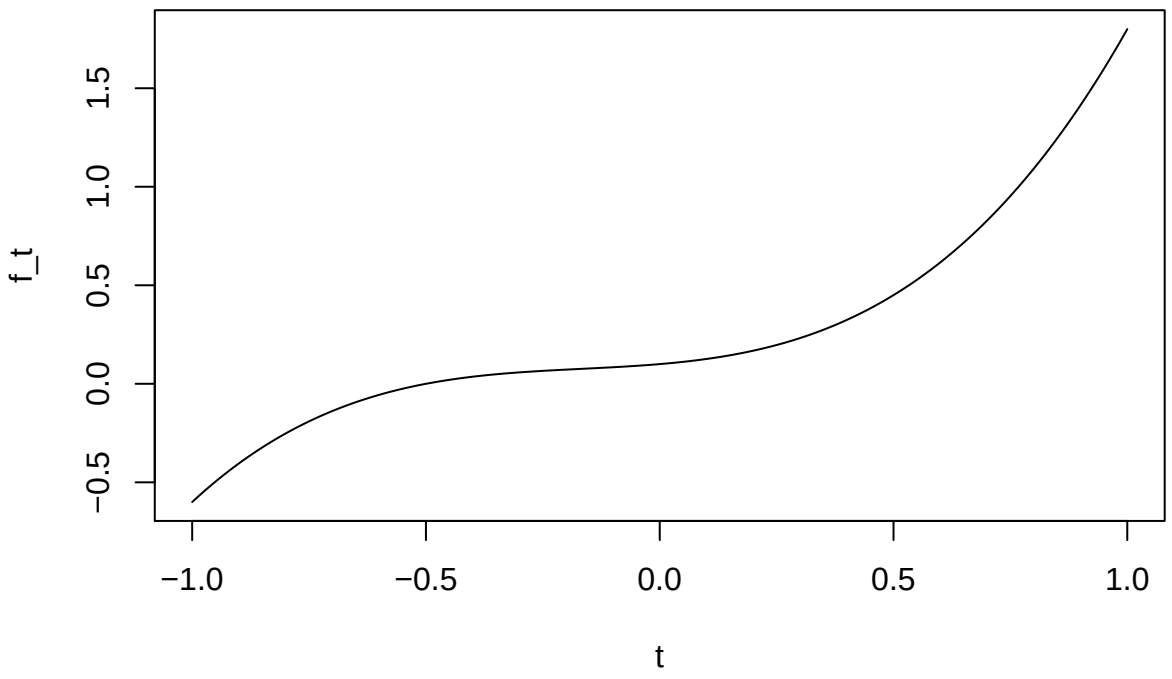
```
# try using some of these functions instead
# f<- function(x){cos(x)}
# f<- function(x){x>-0.5 & x<0.5}
# f<- function(x){x>0}

# edit these values to change the function, interval, or the number of harmonics,
f <- function(x){x^3 + 0.5 * x^2 + 0.2 * x + 0.1} # the function for which we want to
                                                  #find the Fourier Series
lower <- -1 # lower bound
upper <- 1 # upper bound
increment <- 0.01 # time increment (smaller -> slower)
num_harmonics <- 400 # number of harmonics to use- try a small value for fun!

t <- seq(from=lower,to=upper,by=increment) # vector of time values to evaluate f over
f_t <- sapply(t,f) # evaluate f over t
```

A plot of the original function:

```
plot(t,f_t,"l")
```



Calculating the Fourier Coefficients:

```
f0 <- 1 / (upper - lower)   # fundamental frequency of f
a_n <- as.numeric(0) # vector of cosine coefficients
b_n <- as.numeric(0) # vector of sine coefficients
a_n[1] <- sum(f_t) / 2 # compute a_0 by integrating f_t / 2. This looks a little odd because R
                       #does not zero-index vectors
for(n in 1:num_harmonics){ # compute coefficients for each harmonic in turn
  w <- f0 # shouldn't there be 2*pi here??
  # compute Cosine Fourier Coefficient a_n by integrating f * cos(n w x)
  c_n <- sapply(t,function(x){cos(n*w*x)})
  a_n <- append(a_n, sum(f_t * c_n))
  # compute Sine Fourier Coefficient b_n by integrating f * sin(n w x)
  s_n <- sapply(t,function(x){sin(n*w*x)})
  b_n <- append(b_n, sum(f_t * s_n))
}
```

The coefficients have been computed. They can be tabulated and displayed (only showing the first 20 here). Note that everything is offset by 1 because R starts vector indices at 1 not 0. E.g. a_n[1] is actually coefficent a_0:

```
coefs <- cbind(a_n,b_n)
if(length(coefs[,1])>20){print(coefs[1:20,])}else{print(coefs)}
```

```
##               a_n         b_n
##  [1,]  26.9675000   0.0000000
##  [2,]  50.5750895  26.4907472
##  [3,]  41.0670228  48.4817958
##  [4,]  27.0176001  62.3242302
##  [5,]  10.7658678  65.9002182
##  [6,]  -5.0538939  58.9977936
##  [7,] -17.9964353  43.3084238
##  [8,] -26.2418428  22.0534035
##  [9,] -28.8998607  -0.6792175
## [10,] -26.1263627 -20.7477678
## [11,] -19.0351413 -34.7529421
## [12,]  -9.4297907 -40.6569195
## [13,]   0.5840668 -38.1064673
## [14,]   9.0238636 -28.4053311
## [15,]  14.4056958 -14.1553815
## [16,]  15.9973228   1.3460688
## [17,]  13.9037598  14.8178967
## [18,]   8.9772510  23.6634074
## [19,]   2.5802379  26.4582117
## [20,]  -3.7407986  23.1646679
```

The Fourier Series can be plotted in the time domain to see how it compares with the original function:

```
# This function will evaluate the Fourier Series in the time domain
F <- function(coefs,f0,ts){
  w <- f0 # shouldn't there be 2*pi here??
  n_coefs <- nrow(coefs)
  accumulator <- numeric(length(ts))
  accumulator <- accumulator + coefs[1,1] # a_0
```

```
  for(n in 1:n_coefs-1){ # R vectors are not zero-indexed so this part looks a bit funny
    accumulator <- accumulator + sapply(ts,function(x){coefs[n+1,1]*cos(n*w*x)+coefs[n+1,2]*sin(n*w*x)})
  }
  accumulator
}
```

Plot of the computed Fourier Series:

```
plot(t,F(coefs,f0,t),"l")
```