fMRI fourier Curve Smoothing

Zach Wang

7/9/2020

1. Progress Summary

We are focusing on a single node to study the variance of the signal time series. In this Rmarkdown file, it is specified to use only *Node 1*. Replace node subset when calling the corresponding functions to study different nodes. Part 3 used Generalized Cross Validation metrics to find the number of basis functions that best fit the data, which also means the curve is under-smoothed. This number will be used in 5(a) for the under-smoothed case. Part 4 defined function to smooth the node, which allows user to specify the number of fourier basis function. Part 5 defined function to plot the smoothed curve. It takes three inputs–> data: the value of the smoothed curve; register: 0 if we want to plot the original data and 1 if we want every complete sinusoidal curve starts at 0; standardized: 0 if we don't want each plot to be standardized on horizontally and 1 if we want to standardize.

2. read data and attach packages

3. Generalized cross validation approach to select fourier basis models

```
fourier_selection <- function(time_subset, data_mat, node_subset, kList){
  smoothK.unwrapped = matrix(0, length(kList), kList[1])
  colnames(smoothK.unwrapped) = c('k', 'gcv', 'sse')
  for(row in 1:length(kList)) {
        basis <- create.fourier.basis(c(1,600), kList[row])
        smoothList <- smooth.basis(time_subset, data_mat[,node_subset], basis)
        smoothK.unwrapped[row, 1] = kList[row]
        smoothK.unwrapped[row, 2] = mean(smoothList$gcv)
        smoothK.unwrapped[row, 3] = smoothList$SSE
}
  par(mfrow=c(1,2))
  plot(smoothK.unwrapped[,1], smoothK.unwrapped[,2], xlab='K', ylab='GCV')
  title(main=paste("Generalized-cross-validation"))
  plot(smoothK.unwrapped[,1], smoothK.unwrapped[,3], xlab='K', ylab='SSE')
  title(main=paste("SSE"))
return(smoothK.unwrapped)
}
```

4. Defined fourier smoothing functions

To study a single brain node response, specify the node number in the *node_subset* list.

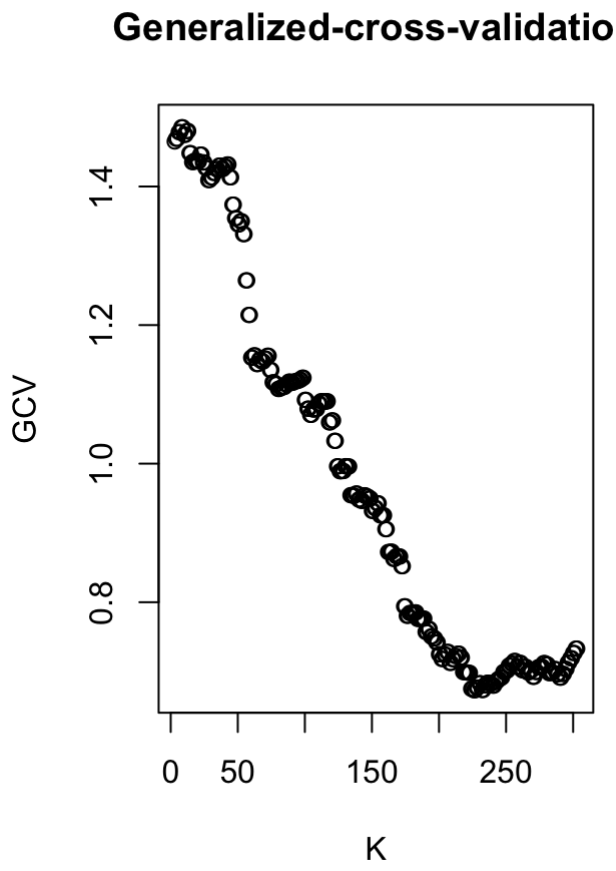
```
f_fourier_smooth <- function(time_subset, data_mat, node_subset, k){
  basis <- create.fourier.basis(c(time_subset[1],time_subset[length(time_subset)]), k)
  fd_obj <- smooth.basis(time_subset, data_mat[time_subset,node_subset], basis)
  smoothfd <- fd_obj$fd
  plot(smoothfd)
  title(main=paste("Fourier Basis Smoothing of node:", node_subset, ", Basis_number:",k
 ))
  return(fd_obj)
}
```

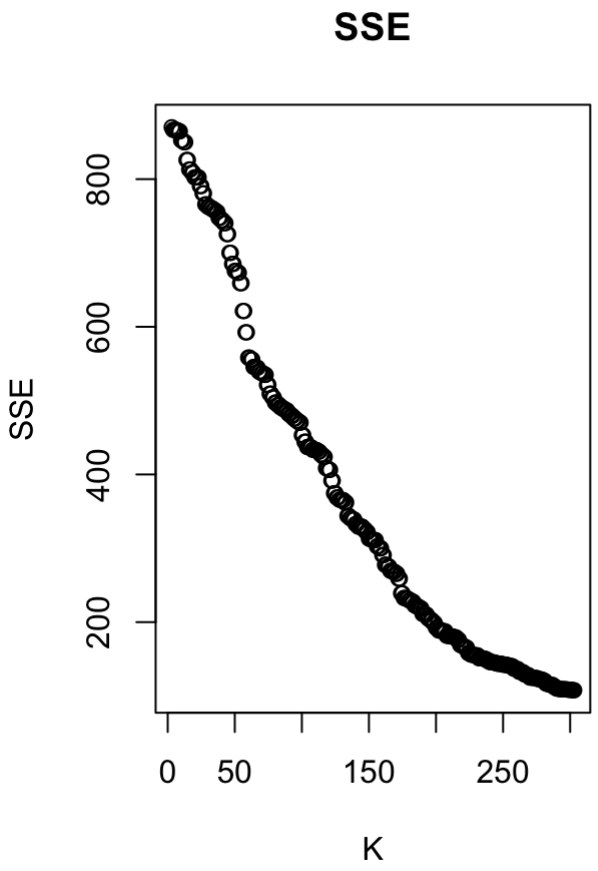
5. define the function to extract periodic cycle of a single node response

```
plot.periodicCycle = function(data, register, standardized){
  x=diff(ifelse(data>0,1,0))       #crossed 0---> -1: pos to neg,    1: neg to pos
  z_idx=(1:599)[x!=0]             #returns: location index where curve crosses X-axis
  N=idivide(length(z_idx),3)
  i=1
  result=data.frame(cycle=integer(), time=integer(), y_value=integer())
  while (i<=N){
    if (standardized==0){
      if(register==0){
        tmp=data.frame(cycle=i, time=seq(z_idx[1+(i-1)*3],z_idx[3+(i-1)*3]),
                       y_value=smoothed_curve[z_idx[1+(i-1)*3]:z_idx[3+(i-1)*3]])
      }
      else{
        tmp=data.frame(cycle=i, time=seq(1,length(seq(z_idx[1+(i-1)*3],z_idx[3+(i-1)*3
])))),
                       y_value=smoothed_curve[z_idx[1+(i-1)*3]:z_idx[3+(i-1)*3]])
      }
      result=rbind(result,tmp)
      i=i+1
    }
    else{
      if(register==0){
        tmp=data.frame(cycle=i, time=scale(seq(z_idx[1+(i-1)*3],z_idx[3+(i-1)*3])),
                       y_value=smoothed_curve[z_idx[1+(i-1)*3]:z_idx[3+(i-1)*3]])
      }
      else{
        tmp=data.frame(cycle=i, time=scale(seq(1,length(seq(z_idx[1+(i-1)*3],z_idx[3+(i-
1)*3])))),
                       y_value=smoothed_curve[z_idx[1+(i-1)*3]:z_idx[3+(i-1)*3]])
      }
      result=rbind(result,tmp)
      i=i+1
    }
  }
  ggplot(result, aes(time, y_value,group=cycle, colour=cycle)) + geom_line() + theme(leg
end.position="top")
}
```

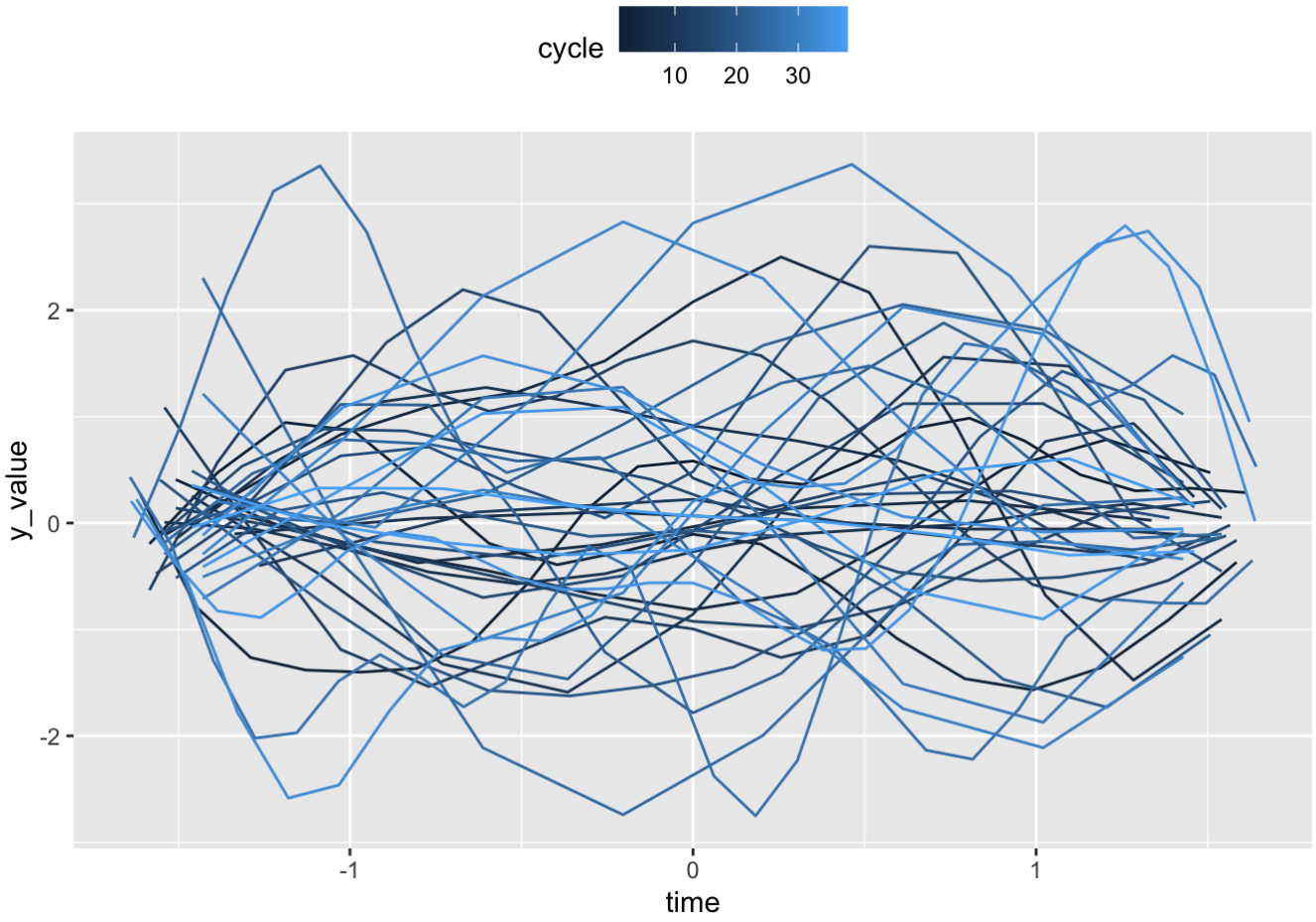
5(a). Undersmooth with basis functions (the smalled GCV score)

```
# UnderSmoothed

## register: 0 --> plot data on the original timeline
##           1 --> register every complete sinusoidal curve starting at 0;

## standardized: 0 --> data will not be standardized
##             : 1 --> scale every complete sinusoidal curve to [0,1]. Notes: if standar
dized, it is equivalent to    register at 0 and standardized

selection_result=fourier_selection(time_subset=c(1:600), data_mat, node_subset=c(1), kLi
st=c(3:303))
```



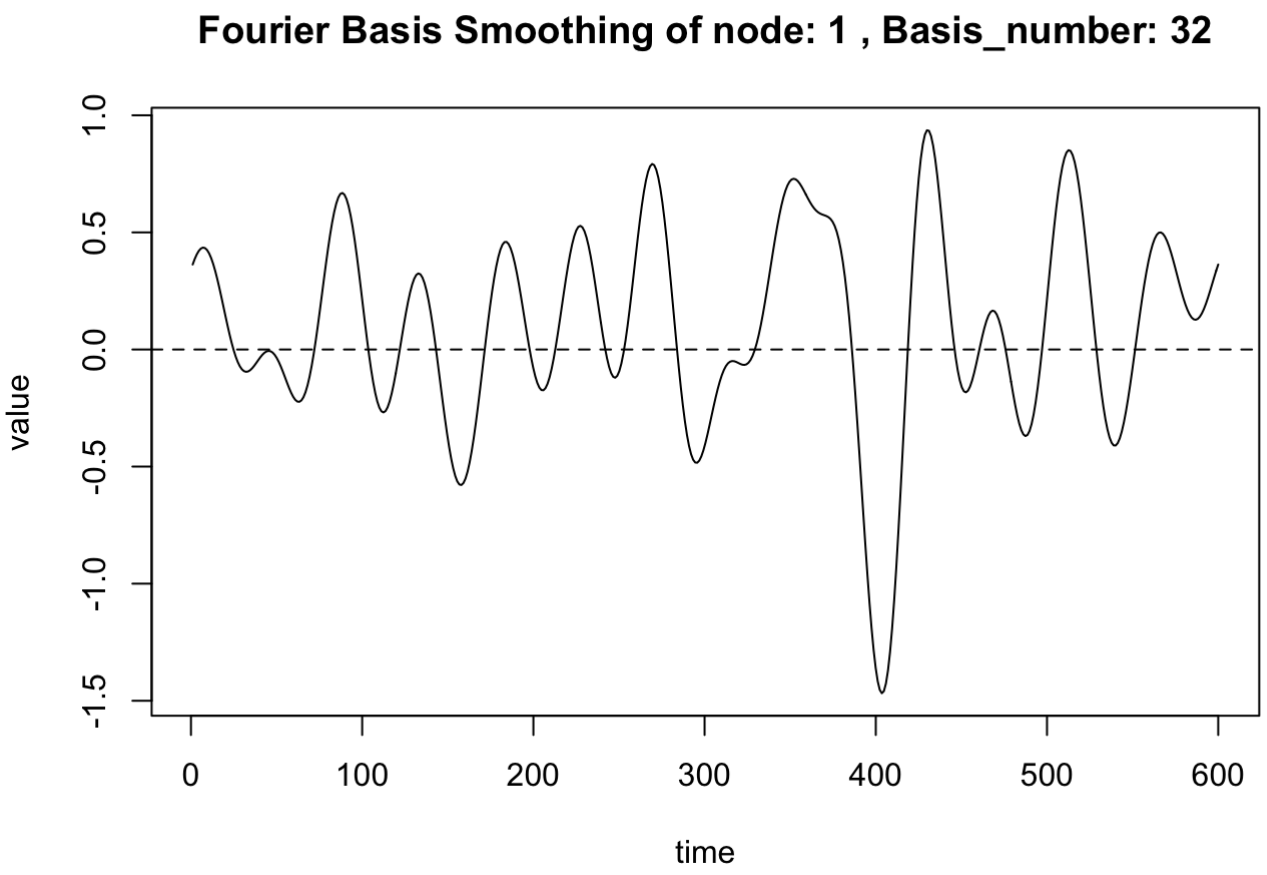



```
result_obj <- f_fourier_smooth(time_subset=c(1:600),data_mat,node_subset=c(1)
                               ,k=selection_result[which.min(selection_result[,2]),1])

smoothed_curve = eval.fd(c(1:600),result_obj$fd)
plot.periodicCycle(data=smoothed_curve, register=0, standardized=1)
plot.periodicCycle(data=smoothed_curve, register=1, standardized=1)
```

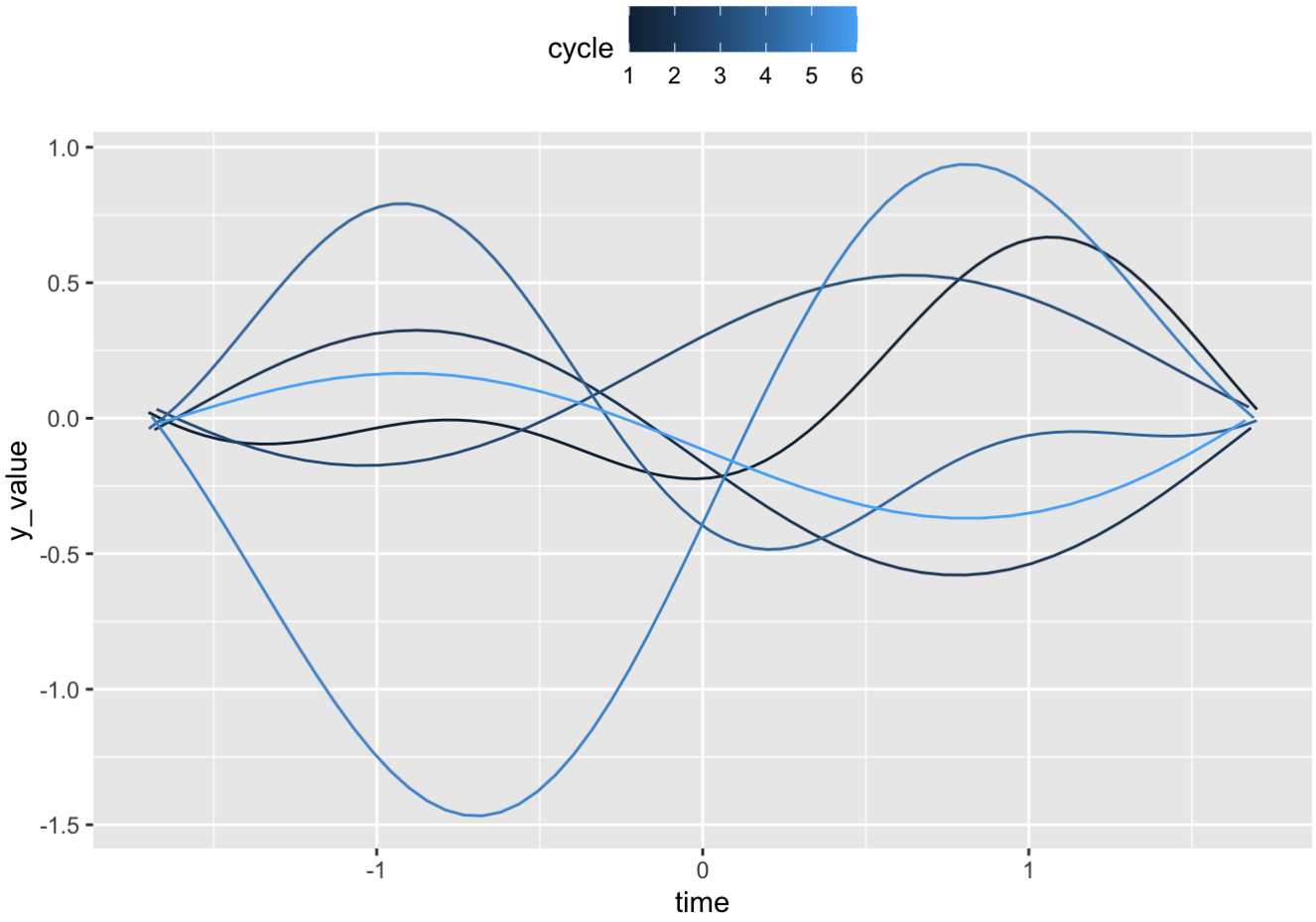


5(b). Oversmooth with basis functions

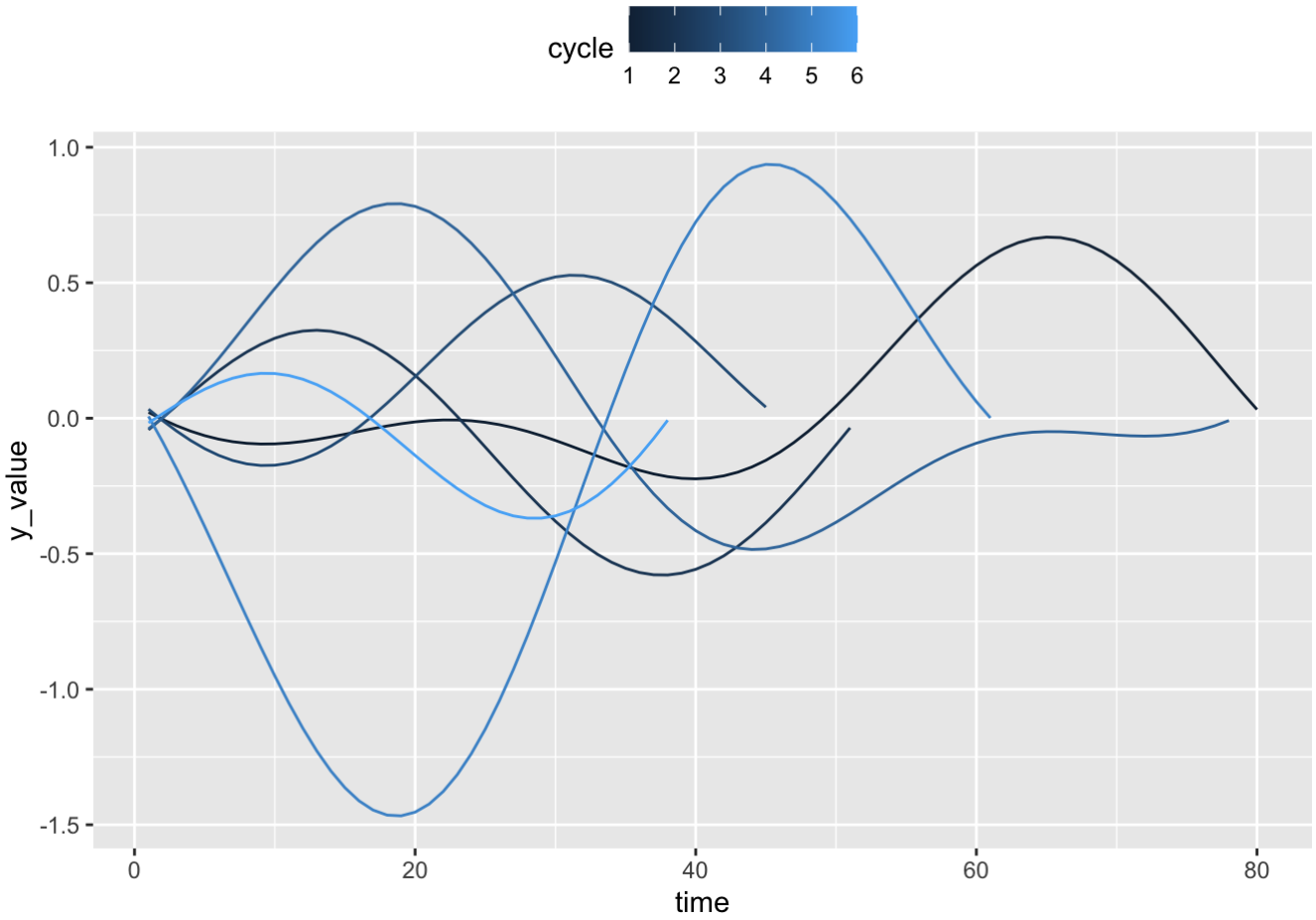
```
# OverSmoothed, k=32

## register: 0 --> plot data on the original timeline
##           1 --> register every complete sinusoidal curve starting at 0;

## standardized: 0 --> data will not be standardized
##             : 1 --> scale every complete sinusoidal curve to [0,1]. Notes: if standar
dized, it is equivalent to     register at 0 and standardized

result_obj <- f_fourier_smooth(time_subset=c(1:600), data_mat, node_subset=c(1), k=32)
```




```
smoothed_curve = eval.fd(c(1:600),result_obj$fd)
plot.periodicCycle(data=smoothed_curve, register=1, standardized=1)
```




```
plot.periodicCycle(data=smoothed_curve, register=1, standardized=0)
```


cycle
1 2 3 4 5 6
1.0
0.5
0.0
-0.5
-1.0
-1.5
y_value
0
20
40
60
80
time