Limiting one muscle:

Commands: Run selected chunk: *Cmd+Shift+Enter*. Insert chunk: *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file). #Write cost functions

```
fill_costs = function(df,m, fmax_vector) {
  F0 = matrix(fmax_vector,dim(df)[1],m,byrow=TRUE)
  # L1
  df[,(m+1)] = rowSums(df[,1:m])
  # L2
  df[,(m+2)] = (rowSums(df[,1:m]^2))^(1/2)
  # Lw1
  df[,(m+3)] = rowSums(df[,1:m]*F0)
  # Lw2
  df[,(m+4)] = (rowSums((df[,1:m]*F0)^2))^(1/2)


  return(df)

}
```

#Load data and compute cost

```
my_column_names <- c("FDP",
                     "FDS",
                     "EIP",
                     "EDC",
                     "LUM",
                     "DI",
                     "PI",
                     "L1",
                     "L2",
                     "L1W",
                     "L2W")
points_db_80 <- read.csv("finger_forcevector_0.8075310608430573_1484789302756.csv", header=F
fmax_vector <- c(123, 219, 124.8, 129.6, 23.52, 21.6, 91.74)
```

```
points_w_cost<- fill_costs(points_db_80,7,fmax_vector)
colnames(points_w_cost) <- my_column_names
```

#All points for an 80%-of-max task

```
generate_parcoord_plot_with_costs<- function(points_dataframe, fraction_of_maxforce_for_task
  points_with_costs <- fill_costs(points_dataframe, 7, fmax_vector)
  points_with_costs$TaskForce <- rep(fraction_of_maxforce_for_task, length(points_with_costs
  colnames(points_with_costs) <- my_column_names

  # parcoord(points_with_costs, var.label=TRUE, ylim=c(0,1))

  nonweighted_max_observed_cost <- max(points_with_costs$L1)
  weighted_max_observed_cost <- max(points_with_costs$L1W)

  #unweighted
  points_with_costs$L1 <- points_with_costs$L1 / nonweighted_max_observed_cost
  points_with_costs$L2 <- points_with_costs$L2 / nonweighted_max_observed_cost
  #weighted
  points_with_costs$L1W <- points_with_costs$L1W / weighted_max_observed_cost
  points_with_costs$L2W <- points_with_costs$L2W / weighted_max_observed_cost

  require(GGally)
  require(ggplot2)
  p_raw <- ggparcoord(points_with_costs, scale='globalminmax', alpha=0.025, boxplot=FALSE, m
  p <- p_raw + theme_bw() + theme(
    panel.grid.major.x = element_line(color = "black", size = 0.1),
    panel.grid.major = element_blank(),
    legend.position = "none"
)
  return(p)
}
```

```
generate_parcoord_plot_with_costs(points_dataframe = points_db_80, fraction_of_maxforce_for_
                    "FDS",
                    "EIP",
                    "EDC",
                    "LUM",
                    "DI",
                    "PI",
                    "L1",
                    "L2",
                    "L1W",
                    "L2W",
                    "TaskForce"), fmax_vector)
```

#View all points as boxplots:

```
alldata <- points_db_80[,1:7]
colnames(alldata) <- my_column_names[1:7]
boxplot(alldata, xlab="Muscle", ylab="Activation", col="lightblue", main="all 1000 solutions
lo_cost_summary <- summary(alldata)
print(lo_cost_summary)
```

#what about parcoord axes being parallel (few line crossings between muscle actiavtions)?

```
cor(points_w_cost$FDP, points_w_cost$FDS)
```

#what about many crossings between two muscles?

```
cor(points_w_cost$LUM, points_w_cost$DI)
```

```
cor(points_w_cost$EIP, points_w_cost$EDC)
```

```
library(corrplot)
corrplot(cor(points_w_cost), order = "hclust", addrect=5)
```

#Let's grab the bottom 10% of L2W cost and see how the muscle activations are distributed

```
total_points <- length(points_w_cost[,1])
remaining_points <- points_w_cost[order(points_w_cost$L2W),][1:100,]
boxplot(remaining_points[,1:7], xlab="Muscle", ylab="Activation", col="lightblue", main="Bot
lo_cost_summary <- summary(remaining_points[,1:7])
print(lo_cost_summary)
```

Limiting one muscle:

Our dataset can be used to simulate a 40% reduction in activation (due to muscle dysfunction, for example) in the two index finger muscles innervated by the radial nerve (EIP and EDC).

```
radial_nerve_damaged_points <- points_w_cost[points_w_cost$EIP < 0.6 & points_w_cost$EDC < (
len_remaining <- length(radial_nerve_damaged_points[,1])
boxplot(radial_nerve_damaged_points[,1:7], xlab="Muscle", ylab="Activation", col="lightblue"
lo_cost_summary <- summary(radial_nerve_damaged_points[,1:7])
print(lo_cost_summary)
```

#When flexor digitorum profundus has resting tonicity of 0.2:

```
hypertonic_points <- points_w_cost[points_w_cost$FDP > 0.2,]
len_remaining <- length(hypertonic_points[,1])
boxplot(hypertonic_points[,1:7], xlab="Muscle", ylab="Activation", col="lightblue", main=pas
lo_cost_summary <- summary(hypertonic_points[,1:7])
print(lo_cost_summary)
```

Manual observations on the effects upon other muscles when FDP activation is kept above 0.2:

- FDS becomes constrained between .09 and 0.16, with middle 50% of solutions in a range spanning only .02697 (between .13190 and .10493)
- EDC goes from being redundant (with bounds of 0 and 1), to being only in the upper half (0.5 to 0.88)

#Which muscle, when hypotonic, slices the FAS more—PI or DI? ##Let's limit each to 20% of maximal distal fingertip force.

```
PI_reduced <- points_w_cost[points_w_cost$PI < 0.20,]
len_remaining <- length(PI_reduced[,1])
boxplot(PI_reduced[,1:7], xlab="Muscle", ylab="Activation", col="lightblue", main=paste(len_
lo_cost_summary <- summary(PI_reduced[,1:7])
print(lo_cost_summary)
```

```
DI_reduced <- points_w_cost[points_w_cost$DI < 0.2,]
len_remaining<- length(DI_reduced[,1])
boxplot(DI_reduced[,1:7], xlab="Muscle", ylab="Activation", col="lightblue", main=paste(len_
lo_cost_summary <- summary(DI_reduced[,1:7])
print(lo_cost_summary)
```

```
DI_reduced <- points_w_cost[points_w_cost$DI < 0.2,]
len_remaining<- length(DI_reduced[,1])

library(reshape2)
pre_and_post_meltdb <- function(pre_df, post_df, constraint_str= "after constraints"){
#assemble post
post_melt<- melt(post_df)
post_melt$group <- constraint_str
#assemble pre
colnames(pre_df) <- my_column_names[1:7]
full_melt <- melt(pre_df)
full_melt$group <- "original"
#concatenate
pre_and_post <- rbind(post_melt, full_melt)

library(ggplot2)
p<- ggplot(pre_and_post, aes(x = variable, y = value, fill = group)) +
  geom_boxplot() +
  scale_fill_manual(values = c("lightblue", "grey"))
return(p)
}

pre_and_post_meltdb(points_db_80[,1:7], DI_reduced[,1:7], "DI reduced")

boxplot(DI_reduced[,1:7], xlab="Muscle", ylab="Activation", col="lightblue", main=paste(len_
```

```
lo_cost_summary <- summary(DI_reduced[,1:7])
print(lo_cost_summary)
```

#showing the wide bounds with small IQR for a muscle at higher force (not 80%)

```
high_force_points <- read.csv("finger_forcevector_25.379547626496084_1484881649920.csv")
par(mfrow=c(3,3))
lapply(1:7,
       function(x) {
        hist(high_force_points[,x], xlim=c(0,1))
        summary(high_force_points[,x])
        }
       )
```