Overview

These two Matlab files create successive DAPL codes from the original DAPL code, varying one or two parameters over each successive measurement session. The file *successive\_iso.m* is the file in which a user is able to change the pre-set parameter values if so desired. Furthermore it contains two possibilities to define a parameter that will vary in value for each successive measurement session. *successive\_iso.m*call upon the function *create\_successive\_iso.m*, which is the actual function that rewrites the parameters in the DAPL code and saves these copies.

Additional resources

Need more help?

Check the resources, and then see Ken

Main content

**Part 1: How to Use successive\_iso**

1. If you want to change the out\_folder, this will need to be changed in the create\_successive\_iso.m
2. The numbers set for pretriggersamples through wave\_amplitude are the predefined values as used by the DAPL code. These values can be changed according to the wishes of the user.
3. The string placed in varname1 contains the first parameter that will be varied among the successive measurements. The parameter mentioned in varname1 must match one of the parameters defined above. If the system receives a parameter name that is unknown to the system an error message will be send to the user. However, the files will be written whether or not the system receives an illegal parameter name, so be aware of this.
4. values1 contains the values that will be used for each successive measurement. The number of successive measurements depends on the length of values1 (and values2).
5. For example:
6. varname1 = 'ramppoints';
7. values1 = [500 1000 1500 2000];
8. In this case, the parameter ramppoints will be varied for each successive measurement. The corresponding values for ramppoints will be 500 points for the first measurement, 1000 for the second, 1500 for the third and 2000 for the fourth. Since values1 is four values long, four successive measurement DAPL codes will be written and saved.
9. If so desired, a second parameter can be defined to vary among successive measurements. varname2 is used in the same manner as varname1 and likewise values2 contains the successive values for the values2 parameter.
10. Since the number of successive measurements depends on the length of both values1 and values2, these two vectors are required to have the same length. If values1 and values2 are not of equal length, the system will give an error.
11. In case either varname1, varname2 or both are not used, the strings can be set to '\*'. Even without parameters that vary, the number of DAPL codes that are written will be of the length of values1 and values2.

**Part 2: The Workings of successive\_iso**

*successive\_iso.m* is the file in which the parameters are defined for the successive measurements. These values are then used for the *create\_successive\_iso.m* function.

%% Rewrite a DAPL code for successive measurements for isotwitch\_combined

% Koen van de Poll

% 6/5/2014

% Campbell Muscle Lab

%% Read in isotwitch\_combined and replace variables if so desired

pretriggersamples = 10000;  % 10000 is 2 sec at 5kHz

isoton            = 32000;      % Voltage to hold

isoinc            = 200;

nsamples          = 30000;  % Don't make bigger than 30000. Limit to return motor to zero before ending study

isopoints         = 2500;   % pts to hold after breakout

ramppoints        = 2500;   % number of points to take to move the motor back to zero

interramppoints   = 0;      % number of points

interfldiff       = 0;      % number of points

propgain          = 50;

Ki                = 20;

flpolarity        = -1;

flswitch          = 1;

flthreshold       = 0;

smoothing\_points  = 10;

integral\_points   = 50;

gain\_asymmetry    = 50; %positive value here helps the downward drift when the motor reverses

wave\_center       = 14811;

wave\_amplitude    = 3146;

% Non-isotwitch values

count             = 260000; % Number of sample points

inputs            = 8;      % Number of input channels

sample\_time       = 0.0002; % Duration of each sample point in seconds

% The measurement duration is the number of sample points times the

% duration of each sample point, divided by the number of input channels.

% I.e. count\*sample\_time/inputs

varname1          = '\*'; %Insert here the parameter that will vary across different measurements

                         %If no repetitive measurements use: '\*'

values1           = [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]; %The values for the above mentioned variable, length determines number of measurements

varname2          = '\*'; %Insert here the parameter that will vary across different measurements

                         %If no repetitive measurements use: '\*'

values2           = [32000 22938 22938 19661 19661 16384 16384 13107 13107 9830 9830 6554 6554 32000]; %The values for the above mentioned variable, length determines number of measurements

% Calls the function and writes out the DAPL codes

create\_successive\_iso(varname1,values1,varname2,values2,pretriggersamples,isoton,isoinc,nsamples,isopoints,ramppoints

**Part 3: The Workings of create\_successive\_iso**

1. The source\_folder is specified, where the DAPL code is read in from  
   The out\_folder is specified, where the repeated DAPL codes and the freeform.txt file are placed.

source\_folder='C:\Lab\Charles\Trabeculae\20140722\_human\';  
out\_folder='C:\Lab\Charles\Trabeculae\20140722\_human\';

1. The original DAPL code *isotwitch\_PIcont.dap* is read in as a string and placed in *rep0*. After that, a freeform\_file is created, or if it already exists it is emptied.

rep0 = fileread('isotwitch\_PIcont.dap');  
  
freeform\_file = fopen([out\_folder 'freeform\_file.txt'], 'w');  
fprintf(freeform\_file,'%s', int2str(length(values1)));

1. A loop is started with the length of the number of values in *values1*and *values2*.

for i = 1:length(values1)

1. Within the loop, if the parameter specified in *varname1*equals one of the known parameters, the predefined value is replaced with the value defined in *successive\_iso.m*

if strcmp(varname1,'pretriggersamples')  
    pretriggersamples = values1(i);  
end  
  
if strcmp(varname1,'isoton')  
    isoton = values1(i);  
end  
  
if ...

Idem for all parameters.

1. The same is executed for *varname2*.

if strcmp(varname2,'pretriggersamples')  
    pretriggersamples = values2(i);  
end  
  
if strcmp(varname2,'isoton')  
    isoton = values2(i);  
end  
  
if ...

Idem for all parameters.

1. All predefined parameter values are replaced by the values defined in *successive\_iso.m*

% Replace the pre-set parameter values in DAPL with the specified values  
rep1  = strrep(rep0,'pretriggersamples = 10000',['pretriggersamples = ' int2str(pretriggersamples)]);  
rep2  = strrep(rep1,'isoton = 0',['isoton = ' int2str(isoton)]);  
rep3  = strrep(rep2,'isoinc = 0',['isoinc = ' int2str(isoinc)]);  
rep4  = strrep(rep3,'nsamples = 30000',['nsamples = ' int2str(nsamples)]);  
rep5  = strrep(rep4,'isopoints = 2500',['isopoints = ' int2str(isopoints)]);  
rep6  = strrep(rep5,'ramppoints = 2500',['ramppoints = ' int2str(ramppoints)]);  
rep7  = strrep(rep6,'interramppoints = 0',['interramppoints = ' int2str(interramppoints)]);  
rep8  = strrep(rep7,'interfldiff = 0',['interfldiff = ' int2str(interfldiff)]);  
rep9  = strrep(rep8,'propgain = 200',['propgain = ' int2str(propgain)]);  
rep10 = strrep(rep9,'Ki = 20',['Ki = ' int2str(Ki)]);  
rep11 = strrep(rep10,'flpolarity = -1',['flpolarity = ' int2str(flpolarity)]);  
rep12 = strrep(rep11,'flswitch = 1',['flswitch = ' int2str(flswitch)]);  
rep13 = strrep(rep12,'flthreshold = 1',['flthreshold = ' int2str(flthreshold)]);  
rep14 = strrep(rep13,'smoothing\_points = 10',['smoothing\_points = ' int2str(smoothing\_points)]);  
rep15 = strrep(rep14,'integral\_points = 10',['integral\_points = ' int2str(integral\_points)]);  
rep16 = strrep(rep15,'gain\_asymmetry = 0',['gain\_asymmetry = ' int2str(gain\_asymmetry)]);  
rep17 = strrep(rep16,'wave\_center = 14811',['wave\_center = ' int2str(wave\_center)]);  
rep18 = strrep(rep17,'wave\_amplitude = 3146',['wave\_amplitude = ' int2str(wave\_amplitude)]);  
rep19 = strrep(rep18,'count 260000',['count ' int2str(count)]);  
rep20 = strrep(rep19,'pause 6510',['pause ' int2str(pause)]);

1. The rewritten DAPL code is saved as a .txt file.

% Write out the DAPL codes with chosen parameters  
file\_handle   = fopen([out\_folder 'isotwitchPItest\_' int2str(i) '.txt'],'w');  
  
fprintf(file\_handle,'%s',rep20);

1. The freeform.txt file is written containing the locations of the written DAPL files.

% Write out the freeform text code with the DAPL files  
  
freeform\_file = fopen([out\_folder 'freeform\_file.txt'], 'a');  
new\_line = sprintf('\n');   %Moves to a "next line" in a string  
tab = sprintf('\t');        %Inserts a horizontal tab in a string  
  
fprintf(freeform\_file,'%s', [new\_line int2str(inputs) tab num2str(sample\_time) tab int2str(count) tab out\_folder 'isotwitchPItest\_' int2str(i) '.txt']);

1. If an illegal parameter name is given for *varname1*or *varname2*, or if *values1*and *values2*are not of the same length, an error message is given.
2. The process within the loop (step 4 - step 9) is repeated for the next values given in *values1*and *values2*.