```
'This script is written in Spike2 language (CED, Cambridge, UK). It calculates the muscle
transfer function and creates the values for current injection into the model AGR.

'This script is based on Chuck Geier's thesis (Scott Hooper's laboratory at Ohio University)
http://crab-lab.zool.ohiou.edu/hooper/
'and was initially created by Ralph DiCaprio (Dec 2002) and modified by Dirk Bucher (Dec
2002).
'It creates a muscle tension output from a spike event channel using a te-at function. Initially
used for the p1 muscle in Panulirus interruptus

'assign:

'       -       number of sequential filters            should normally be 3.
'       -       time constant of the filters            is used for all of them
'       -       time steps                                              sets the "sampling"
interval of the output. this is automatically converted to the nearest multiple of the file time
resolution.
'       -       % of maximum y range                                maximum y range is 5V.
This sets the amplitude of the response to a single event. this is important
'                                                                                  because if
the whole channel was scaled, different files would always have different kernel amplitudes.
'  -   scaling factor              this scales the output with a factor that was derived in our
experiments and adapts the output values for use as current injection in the AGR model

'There is no threshold filter built in (see Chuck's thesis).
'----------------------------------------------------------------------------------------------------------
--------------------

'This script needs a data file with an event channel containing the times of motor neuron
spikes. In our case the GM neurons
'The output is then scaled with a scaling factor (default = 1.1282) that was determined in our
experiments

'initialize windows etc
if ViewKind() <> 0 then                                   'checks if the associated window is a
time view...
   Message("This isn't a time view!");                  'if there is no data file
        FrontView (App (3));                              'brings script to front...
        Halt;                                       'and stops it
endif

WindowVisible (3);              'sets the time view to the front

'define variables
var ok%, cur1, cur2, wholefile%;


if Cursor (2)<Cursor (1) or Cursor (1)=-1 or Cursor (2)=-1 then ' if Cursor (1) and Cursor (2)
are not present or not in the right order....
        DlgCreate("ATTENTION!");
        DlgText("Cursor (1) and (2) not set correctly.",2,1);
```

```
            DlgLabel(1, "Analyse whole file?",2,2);
                        '...choose if you want to analyse the whole file
            ok%:=DlgShow (wholefile%);

            if ok%=1 then
                                                                        'sets analysis to whole file or...
                        cur1:= 0;

                        cur2:=MaxTime ();
            else
        interact("get Cursors 1 and 2 and select time range for analysis",1023);
        cur1 := Cursor(1);
                                                            'get cursor times
        cur2 := Cursor(2);

            endif

else
            cur1 := Cursor(1);
                                                            'if present and in the right order, get cursor times

            cur2 := Cursor(2);
endif


'define more variable
var name$;
var scale;
var i%,tstep;
var nspks%,spktime[100000];
var mch%;
var mintau;
var tausteps;
var maxtau;
var x;
var ok1%,tau,spkch%,outch%,pexp%, points%, scalingfactor;

 'start dialog for kernal time input
DlgCreate ("Muscle transform Filter");
DlgChan (1, "Import channel:",2);
DlgChan (2, "New channel #:",128);
DlgString (3, "New channel name:",15);
DlgInteger (4,"# of sequential filters:",1,3);
DlgReal (5,"time constant (ms):",0.0,3000);
DlgReal (6,"Time steps [ms]:",0.0000,2000);
DlgReal (7,"% of maximum y range:",0,100);
DlgReal (8,"Scaling factor for output values",0,100);
name$:="muscle";

'dialog presets
scale:=1;
```

```
tau:=320;
pexp%:=3;
tstep:=1;

mintau:=10;
tausteps:=10;
maxtau:=3000;
scalingfactor:=1.1282; 'this is used to scale the output for use in the AGR model (as
determined in our experiments).
ok1%:=DlgShow(spkch%,outch%,name$,pexp%,tau,tstep,scale, scalingfactor);   'assign
variables



   if ok1%<>1 then        'if Cancel was clicked
FrontView (App (3));           'brings script to front...
HALT;                                 '...and stops it
endif

var sinc[100000];
pexp%:=pexp%-1;                          'converts # of filters to exponential
tstep:=tstep/1000.0;           'converts time step to seconds

var interv,tfactor%;

interv:=Binsize();                      'reads the file time resolution
tfactor%:= tstep/interv;        'gets nearest multiple of file time interval
if tfactor%=0 then                      'resolution can't exceed file time resolution!
   tstep:=interv;
else
   tstep:=tfactor%*interv;              'sets time step to nearest multiple of file time interval
endif

var endbin%;
endbin%:= cur2/tstep;                   'gets bin # of analysis window end
var startbin%;
startbin%:=cur1/tstep;                  'gets bin # of analysis window start

var tauscale;                           'scales the number of points used for the kernel array so
that
if pexp%=0 then                                 'the function just declines to zero
   tauscale:=0.007;
else
   if pexp%=1 then
      tauscale:=0.01;
   else
      tauscale:=0.012;
   endif
endif
points%:=(tauscale*tau)/tstep;
```

```
var muscleout[10000000];


for i% := 0 to points%-1 step 1 do
                    'Build muscle impulse response
   sinc[i%]:=Pow(i%*tstep,pexp%)*Exp((-i%*tstep)/(tau/1000.0));
next;

var amp,imax%;                                                              'scale this
function to maximum value of single event response
imax%:=Max(sinc[:points%-1]);
amp:=sinc[imax%];
ArrDiv(sinc[:points%-1],amp);
ArrMul(sinc[:points%-1],scale/20);

nspks%:=ChanData(spkch%,spktime[],cur1,cur2);   'get spike times

var ind%, k%;

mch%:=MemChan(1,0,tstep);                                  'create waveform buffer
channel

for k% :=0 to nspks%-1 step 1 do              'loop through spike times
   ind%:=spktime[k%]/tstep;                            'get bin # of spike (index # in array)
   ArrAdd(muscleout[ind%:ind%+(points%-1)],sinc[:points%-1]);          'add function to
array
next;

ARRmul(muscleout[],scalingfactor);            'muliplies the output by the scalingfactor

MemSetItem(mch%,0,cur1+tstep,muscleout[startbin%:endbin%-startbin%]);      'import array
into waveform channel

         MemSave(mch%,outch%, 0, 0);                   'saves buffer channel to a real channel
         ChanDelete(mch%, 0);                                  'deletes the buffer channel
         ChanShow(outch%);                                     'shows new channel
         ChanTitle$(outch%,name$);                   'names new channel
         Optimise(outch%);                                     'optimise y-axis

   message("Channel number ",outch%," contains the muscle output");

var xval[100000], xy%, j%;

for j% := 0 to points%-1 step 1 do                   'loop through kernel points
   xval[j%]:=j%*tstep;                                                    'fill array with time
values
next;
```