// Kalman filtering

var EL = new Array();

EL[0] = new Array();

EL[1] = new Array();

var VL = new Array();

VL[0] = new Array();

VL[1] = new Array();

var EL\_N = new Array();

var VL\_N = new Array();

var COVL\_N = new Array();

var N;

/\*

SecondStage

Optimize the parameter bets with the EM algorithm, and estimate the firing rate with the parameter the beta .

arguments:

spike\_time: spike train

returns

kalman\_data: firing rate

internal parameters

mu: mean rate

beta0: initial value of the parameter beta

beta: hyper-parameter beta

kalman\_data: estimated firing rate

\*/

function SecondStage(spike\_time){

var mu = spike\_time.length / (spike\_time[spike\_time.length - 1] - spike\_time[0]); // mean firing rate

var beta0 = Math.pow(mu,-3);

var beta = EMmethod(spike\_time,beta0);

var kalman\_data = KalmanFilter(spike\_time,beta);

return kalman\_data;

}

/\*

function ThirdStage(spike\_time,beta){

NGEMInitialize(spike\_time,beta);

beta = NGEMmethod();

var nongaussian\_data = NGF(beta);

var D = NGFD();// times

var dt = NGFdt();

return nongaussian\_data;

}

\*/

/\*

EMmethod

estimates a parameter beta with the EM algorithm.

arguments:

spike\_time: spike train

beta0: initial value of the parameter beta

returns

beta2: estimated parameter

internal parameters

beta1: parameter value before updating

beta2: parameter value updated

T0: auxiliary parameter for correcting the result in the case that multiple spikes were recorded at the same time

kalman: result of the Kalman filtering

\*/

function EMmethod(spike\_time,beta0){

KFinitialize(spike\_time);

var beta1 = 0;

var beta2 = beta0;

var T0;

for(var j=0;j<100;j++){

beta1 = beta2;

var kalman = KalmanFilter(spike\_time, beta1);

beta2 = 0;

T0=0;

for(var i=0; i<N-1; i++){

if(spike\_time[i+1]-spike\_time[i]>0){

beta2 += (kalman[1][i+1]+kalman[1][i]-2\*kalman[2][i]+(kalman[0][i+1]-kalman[0][i])\*(kalman[0][i+1]-kalman[0][i]))/(spike\_time[i+1]-spike\_time[i]);

}else{

T0 += 1; // correction for the case that multiple spikes were recorded at the same time, or interspike intervals are zero

}

}

beta2 = (N-T0-1)/(2\*beta2);

}

return beta2;

}

/\*

KFinitialize

sets initial values of N, EL, VL in the Kalman filtering.

arguments:

spike\_time: spike train

internal parameters:

mu: inverse of mean inter spike interval, or the mean firing rate

IEL: mu

IVL: mu^2/3

\*/

function KFinitialize(spike\_time){

N = spike\_time.length - 1;

// N = interspike interval length

var mu = 0;

for(var i=0;i<N;i++){

mu += spike\_time[i+1]-spike\_time[i];

}

mu = N/mu;

// filtering

var IEL = mu;

var IVL = (mu/3)\*(mu/3);

var A = IEL - (spike\_time[1]-spike\_time[0])\*IVL;

EL[0][0] = (A+Math.sqrt(A\*A+4\*IVL))/2;

VL[0][0] = 1/(1/IVL+1/(EL[0][0]\*EL[0][0]));

}

/\*

KalmanFilter

estimates the firing rate

arguments:

spike\_time: spike train

beta: hyper-parameter beta

returns

outdata: estimated firing rate

outdata[0]: estimated value

outdata[1]: its variance

outdata[2]: its covariance

\*/

function KalmanFilter(spike\_time,beta){

for(var i=0;i<N-1;i++){

EL[1][i]=EL[0][i];

VL[1][i]=VL[0][i]+(spike\_time[i+1]-spike\_time[i])/(2\*beta);

A=EL[1][i]-(spike\_time[i+2]-spike\_time[i+1])\*VL[1][i];

EL[0][i+1]=(A+Math.sqrt(A\*A+4\*VL[1][i]))/2;

VL[0][i+1]=1/(1/VL[1][i]+1/(EL[0][i+1]\*EL[0][i+1]));

}

EL\_N[N-1] = EL[0][N-1];

VL\_N[N-1] = VL[0][N-1];

var H = new Array();

for(var i=N-2;i>=0;i--){

H[i] = VL[0][i]/VL[1][i];

EL\_N[i]=EL[0][i]+H[i]\*(EL\_N[i+1]-EL[1][i]);

VL\_N[i]=VL[0][i]+H[i]\*H[i]\*(VL\_N[i+1]-VL[1][i]);

COVL\_N[i]=H[i]\*VL\_N[i+1];

}

var outdata = new Array();

outdata[0] = new Array();

outdata[1] = new Array();

outdata[2] = new Array();

for(var i=0;i<N;i++){

outdata[0][i]=EL\_N[i];

outdata[1][i]=VL\_N[i];

outdata[2][i]=COVL\_N[i];

}

return outdata;

}

/\*

function NGEMmethod(spike\_time, beta){

}

\*/