**GENERAL**

L2norm: computes the L2 normalized version of vec

function [nvec] = l2norm(vec)

normvecs: normalizes each column in the matrix vecs to have unit L2 norm

function [nvecs] = normvecs(vecs)

rectify: sets all values in vec < 0 to 0

function [ rvec ] = rectify(vec)

flatten: Flattens 2-d matrix into column using reshape (L 1)

function v = flatten( mat )

jittervec: jitter vector vec with uniform noise of amplitude amplit

function [jitx] = jittervec(vec,amplit)

returns only values inside the interval

function [insidevals] = isInside(values,interval)

structureIndex: computes the total spatiotemporal power in each filter of the bank tks

function [strucinds] = structureIndex(tks,sdim)

getDSIndex: get Direction Selectivity Index for a filterbank a la FErster & Priebe 2005

function [dsis] = getDSIndex(tks,SDIM)

getOptDrGrating: generates the optimal moving grating stimulus for a given filter bank, based on the first excitatory filter

function [stimOpt,stimCon] = getOptDrGrating(trf,sdim)

dsamplebstim: downsample array of 2d spatial stims along the spatial dimensions by factor dsr

function [dsti] = dsamplebstim(stim,dsr)

averageslices: for a set of 2d filters, computes the average over the time slices specified in tslices for each filter. Outputs meanslices in long-k format. (K x Nfilts matrix).

function [meanslices] = averageslices(allks,sdim,spixids,tslices)

**GLM FITTING**

Parameter structure ‘pars’ stored in ~/Data/rust/infos/StandardParametersRust.mat

create\_GLM0 \_jmm: initializes a model with N\_m NL modules based on the matrix of input filter coefficients ks (N\_k x N\_m) and the default module parameter structure in defmod. Mname specifies the model name

function [glm0] = createGLM0(ks,defmod,mname)

fitNLHI: fits both NL and history dependence

fitNLHIc: uses potential monotonicity and/or positivity constraints

nfitNLHI: wrapper around fitNLHI that first normalizes RF coefs)

fitNL\_jmm: fits the coefficients of the NL for each module given inputs which is the stimulus filtered by the internal RFs of each module and the spike bins

function [fo] = fitNL(fo,input,spkbs)

fitHI\_jmm: fits the coefficients of the PSC terms for each module given the inputs which is the stimulus filtered by the internal RFs of each module and the spike bins spkbs.

function [fo] = fitHI\_jmm(fo,input,spkbs)

fitGLMweights\_jmm: updates the weights of the different NL modules given the raw stimulus ustim as input

function [fglm] = fitGLMweights\_jmm(glm,ustim,spkbs)

fitWeights\_jmm: updates the weights of the different NL modules given a matrix fX containing the outputs of each module.

function fo = fitWeights\_jmm(fo,fX,spkbs)

normalizeRFs: normalizes the filters of all NL modules in glmod to have filter output variance 1. Pulls the extra multiplier into the model weighting factors (out of the filter coefs)

function [glmod] = normalizeRFs(glmod,ustim)

fitNLHI\_jmm: fits the weights, NL functions, and PSC terms for all modules in myglm

function [myglm] = fitNLHI\_jmm(myglm,ustim,spkbs,lltol)

getLLGLM\_jmm: calculates the LL and penalized LL of the GLM model, given the stimulus and spikes

function [nll, pnll] = getLLGLM(glmod,stim,spkbs)

getLL\_jmm: calculates the LL and penalized LL given the scalar value kx which is the sum of all terms inside the spiking non-linearity (which is assumed log(1+exp)

function [nll, pnll] = getLL(kx,glmod,spkbs)

procModul: complete processing of the stimulus by the NL module mod (internel filtering, applies NL, and convolves with PSC term)

function fgh = procModul(mod,stim)

applykfilters: computes the output of the internel kernels of each module in glmod

function outp = applykfilters(glmod,stim)

kfilterInput: applies the internal filter k to the stimulus using conv2

function fstim = kfilterInput(stim,k)

rescalemodrf: rescales the RF coefficients of the module refmod by an overall factor a (pulling this factor into the weighting term for that module)

function [refmod] = rescalemodrf(refmod,a)

nlin\_proc\_stim: processes the stimulus stim with the non-linear function specified by NL and NLx assuming tent basis representation

function proc\_s = nlin\_proc\_stim( stim, NL, NLx )

piece\_proc\_stim: computes the output of a single tent basis function (specified by center and edge vector di) applied to stimulus s

function proc\_s = piece\_proc\_stim( s, center, di )

tbrep: projects the input stimulus (already filtered by internal kernel presumably) onto the tent basis function rep of the nonlinearity. Output is thus, (NT x NBfs)

function brfstim = tbrep(stim, tentx)

plotfo1d: plots fitted model for 1d stimulus (e.g. bar stimuli used in Touryan2001)

function test = plotfo1d(fo)

plotfo1dp: plots fitted model for 1d stimulus. plots only specified modules

function test = plotfo1dp(fo,smods,mult)

plotfo1dpSparse: same as plotfo1dp, except a more paired down display

function test = plotfo1dpSparse(fo,smods,mult)

predictRate: Take an input stimulus and GLM model and predict the firing rate as a function of time

function [prate] = predictRate(glmod,ustim)

**STC CODES**

getSTCfilters: takes as input a stimulus matrix, vector of spike bins, number of time lags flen, and numbers of positive and negative eigenvectors to find npos and nneg. Returns the STA, and a matrix of STC vectors biggest positive first to smallest negative. Also returns matrix fstim of the stimulus projected into STC subspace, and evs the eigenvalues of the STC SVD.

function [sta, stcs, fstim, evs] = getSTCfilters(stim,spikebins,flen,npos,nneg)

getSTCfilters2: Same as above, but projects out the STA before computing STC

function [sta, stcs, fstim] = getSTCfilters2(stim,spikebins,flen,npos,nneg)

plot1dfilterbanks: takes as input fbanks, a cell array of filter banks (matrices of filters), and the stim dimension SDIM and plots the filters for each bank

function [blens] = plot1dfilterbanks(fbanks,SDIM)

getCLOCrf\_jmm:

function [tlrf] = getCLOCrf\_jmm(tpos,tks,SDIM,ninits,ooptions)

locIndexVar\_jmm: takes as input a filter tk, a center location for the filter, and the stimulus dimensionality, and returns a filter localization index with respect to that center location. Localization is defined as the sum of temporal variance across pixels, weighted by the squared distance of the pixel from the RF center.

function [locindex] = locIndexVar\_jmm(tk,rfcenter,SDIM)

getPowerProfs: compute the temporal variance as a function of pixel space for all filters in the matrix tks

function [powerprofs] = getPowerProfs(tks,SDIM)

alignPProfs: align power profiles of input filter kernels trfs, and return aligned, spline fitted RFs.

function [arfs, profmeans] = alignPProfs(trfs,SDIM,npoints)

findPCAs:determine & extract the first nevs principle components underlying these RFs

function [basevecs,pcofs] = findPCAs(arfs,SDIM,nevs)

getFreqAngle: gets dominant angle & spatial frequency of the RF. Returns the max spatial frequency, the frequency angle, the 2d power spectrum, and the single-fourier component filter

function [sfreq alpha sabs testx] = getFreqAngle(trf,sdim,sx,sy)

epanechekov: creates a symmetric inverted quadratic of length len?

function [efilt] = epanechikov(len)

plot2drf: Plots the RF and gives a matrix containing it back

function RF = plot2drf(k,sdim,zrange)

findLocalSubunits: find linear combinations of nestcs that optimize a locality criterion. Iteratively, projections are found wrt. a NONPARAMETRIC concentraton measure and projected out while search continues in remaining subspace.

function [subunits rflocs] = findLocalSubunits(nestcs,ncomps,SDIM)

findLocalSubunits1dVar: find linear combinations of basis vectors bvs that optimize a quadratic locality criterion centered on a set of spatial positions

function [fbanks] = findLocalSubunits1dVar(bvs,sdim,npos,nbanks,ninits)

2+1D STC Codes:

findLocalSubunits2d: computes ncomps localized kernels from nestcs input filter bank by maximizing a nonparametric locality criterion specified in powerconc2d. It has to do with looking at localization of power along different slices in pixel space, but I don’t understand the procedure..

function [subunits rflocs] = findLocalSubunits2d(nestcs,ncomps,stalen,spixids,npix,nangles)

findLocalSubunits2dVar: find linear combinations of obvs that optimize a quadratic locality criterion determined by the best-fitting d-mat given the overall spatial distribution of temporal variance.

ncomps = number of orthogonal components extracted at each gridpoint. Makes use of locIndex2dVar, getMatchedDmat, and getDistMatE11

function [locrfs,locbetas] = findLocalSubunits2dVar(obvs,ncomps,ntslices,spixids,npix,ngrid,ninits)

locIndex2dVar: localization index reminding of computation of variance -- squared deviation from rfcenter and rfshape is interpreted similar to a probability distribution

function [locindex] = locIndex2dVar(tklong,stalen,npix,spixids,distmat)

getMatchedDmat: Computes the distance matrix that best captures the net spatial distribution of temporal power in the filter bank tks.

function [ tdmat, tang, v1, v2 ] = getMatchedDmat(tks,npix,spixids,ntangles,flen)

getDistMatE11: compute distance mat given center position vector, rotation angle, and x and y scale parameters.

function [zmat] = getDistMatEll(npix,cpos,aldeg,v1,v2)