GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110−1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software−−to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the

Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine−readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine−readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

```
    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA


Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type     This is
free software, and you are welcome to redistribute it
    under certain conditions; type
The hypothetical commands parts of the General Public License.  Of course, the c
ommands you use may
be called something other than mouse-clicks or menu items--whatever suits your p
rogram.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  nomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

```
% ASYMMETRY: Temporal Asymmetry Index
%            [alpha_t] = ASYMMETRY(trf)  returns the asymmetry index of a
%            given temporal response profile. Indicies of 0 indicate no skew,
%            negative indicies indicate a right−handed skew, and positive
%            indicate a left−handed skew.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA

%THIS FILE HAS NOT YET BEEN IMPLEMENTED.
```

```
% DECOMPOSESTRF: STRF decomposition.
%                [strfi,trf,srf,sigmas,k]= DECOMPOSESTRF(strf_measured) returns
%                strfi, a m x n x k matrix where strfi(:,:,k) is the kth linear
%                strf component created by SVD and ready to be modeled. trf and
%                srf are the temporal receptive field and the spectral receptive
%                field, respectively.  K is determined by noise_estimation.m ,
%                and represents the number of singular values used to create
%                strfi.

% Graham Voysey Senior Project 2005-2006
% Boston University, College of
% Engineering, Department of Biomedical Engineering
% Natural Sounds and Neural Coding Lab
%
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

function [strfi,trf,srf,sigmas,strfi_total_number,acausal_sigmas] =decomposestrf(
strf_measured)

% step two: deompose STRF
[srf, s, trft]=svd(strf_measured);
trf=trft';
sigmas=diag(s);

[strfi_total_number,acausal_sigmas]=noise_estimation(strf_measured,sigmas); % he
eey, this
                                                                   % works now.

% step three: create K strfs in one large matrix, which may or may not be the
% Right Thing.

% create a matrix strfi whose rows x columns are the same as
% STRF_Cell, and is strfi_total_number deep.

%strfi=zeros(size(STRF_Cell,1), size(STRF_Cell,2),strfi_total_number); <--what
%was i *thinking* ?

for(k=1:strfi_total_number),
  strfu=srf(:,k)*trf(k,:);
  temp=sigmas(k)*strfu;
  strfi(:,:,k)=temp;
end
```

```
% GABOR_SRF_MODEL : Least-Squares Modeling of SRF data.
%                   [ydataFit,x,resnorm] = gabor_srf_model(xdata,ydata)

% Graham Voysey
% Original version of code provided by Gilberto Grana, endymion at bu dot edu
% Extensive assistance by Rajiv Narayan, rn at bu dot edu
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

% !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
% !NOTE! Requires MATLAB Optimization Toolbox!
% !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

% FITTER - A lazy-man's function for fitting exponentially-increasing data.
% Xdata and Ydata have to be defined from the beginning as XDATA = Time, YDATA =
% all points from t10 analysis

function [ydataFit,ydataAdj,x,resnorm,x0,tc] = gabor_srf_model(xdata,ydata)

numunits = size(ydata,1);
if ndims(ydata)==3
    numsets = size(ydata,3);
else
    numsets = 1;
end

% ydataAdj=ydata;

% Get the maximum values of each tCurve for each unit and, if applicable,
% each number of sets created
for i=1:numunits
    for j=1:numsets
        [meanYmax(i,j), meanYmaxdex(i,j)] = max(ydata(i,:,j));
        [meanYmin(i,j), meanYmindex(i,j)] = min(ydata(i,:,j));
        ydataAdj(i,meanYmaxdex(i,j):end,j) = meanYmax(i,j);

    end
end

% figure
% plot(xdata,ydataAdj)
% hold on
% plot(xdata,ydata,'.')

for i=1:numunits
    for j=1:numsets
        % x0(i,:)=[meanY_max(i)+5 meanY_max(i) 0.001]
        x0(i,:,j)=[meanYmax(i,j) meanYmax(i,j)-meanYmin(i,j) 0.001]';
%         ub(i,:,j)=[100 meanYmax(i,j)-meanYmin(i,j) 1]';
%         lb(i,:,j)=[meanYmin(i,j) 0 0]';
%        [x(i,:,j),resnorm(i,j)]=lsqcurvefit(@srf_gabor_fun,x0(i,:,j),xdata(i,:)
,ydata(i,:,j),lb(i,:,j),ub(i,:,j));
        [x(i,:,j),resnorm(i,j)]=lsqcurvefit(@srf_gabor_fun,x0(i,:,j),xdata(i,:),
ydata(i,:,j));
       ydataFit(i,:,j)=srf_gabor_fun(x(i,:,j),xdata(i,:));
    end
```



```
end

%tc=1./x(:,3,:);
% figure
% plot(xdata,ydata,'.')
% hold on
% plot(xdata,ydataFit)

% for i=1:size(ydata,1)
% %    x(i,:)-0.5*max(yfitdata(i,:))
%     t_half(i)=-tc(i)*log((x(i,1)-0.5*max(yfitdata(i,:)))/x(i,2));
% end
```

```
% GABOR_STRF : STRF Modeling with Gabor Functions.
%              GABOR_STRF() begins the process of modeling a STRF with a series
%              of gabor functions.  It will clear all current variables, then
%              ask the user for a data set to load.  The data set should be in
%              the form of Sen, et al's "oldsongs20" set, and contain a matrix
%              STRF_Cell which contains the STRF to be modeled.  Progress from
%              that point is automatic; GABOR_STRF() will output a host of
%              information about the STRF, a model, and a figure containing a
%              plot of the model and a plot of the original for comparison.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA
function [] = gabor_strf(strfdatapath,songdatapath,target_dir)

% define constants here
parameter_number_srf=5;
parameter_number_trf=5;


% some file stuff is declared here
current_dir=pwd;
path(path,current_dir);
path(path,'./computefr');                  % kind of assumes you're in /path/to/gabo
r_strf.
[p,f,e]=fileparts(strfdatapath);
outfile = fullfile(target_dir,[f,'-model']);
outfig = fullfile(target_dir,[f,'-modelfig']);
% Step one: user input to decide what STRF to load.
% each matfile contains three matricies.  we only care about STRF_Cell.  the
% other two are matricies of standard deviations
load(strfdatapath);

% SVD and noise determination.
[strfi,trf,srf,sigmas,strfi_total_number,acausal_sigmas] = decomposestrf(STRF_Ce
ll);

% begin modeling section.
% freq_ and time_ coeff have to have hardcoded constants.
 freq_coeff=zeros(parameter_number_srf,strfi_total_number); % num. param. in the
 freq. eqn is 5
 time_coeff=zeros(parameter_number_trf,strfi_total_number); % num. param. in the
 time eqn is 5

 freq_resnorm=zeros(1,strfi_total_number); % resnorms are of dimension 1
 time_resnorm=zeros(1,strfi_total_number);

 freq_model=zeros(size(srf,1),strfi_total_number); % 31 frequency channels
 time_model=zeros(size(trf,1),strfi_total_number); % 601 time channels.

% create pile of sub-models. blame matlab for the array-of-arrays data
% structure.  freq_coeff is a 5xstrfi_total_number matrix of the free
% paramters in the frequency gabor function, and freq_model is those
% parameters applied to a properly-dimensioned matrix.

for k=1:strfi_total_number,
```



```
  [freq_coeff(:,k),freq_resnorm(k),freq_model(:,k)]=sfitstrf(srf(:,k));
  [time_coeff(:,k),time_resnorm(k),time_model(:,k)]=tfitstrf(trf(k,:));
end
% begin computation of statistics.

% seperability index calculation
 seperability = seperability_index(sigmas,strfi_total_number);
 % similarity index calculation for srf and trf
 freq_si=zeros(1,strfi_total_number);
 time_si=zeros(1,strfi_total_number);
 % mean-squared error calculation for srf and trf
 freq_mse=zeros(1,strfi_total_number);
 time_mse=zeros(1,strfi_total_number);
% populate statistics matricies.
for k=1:strfi_total_number,
  freq_si(k)=sii(srf(:,k),freq_model(:,k));
  time_si(k)=sii(trf(k,:),time_model(:,k)');

  freq_mse(k)=mse(srf(:,k),freq_model(:,k));
  time_mse(k)=mse(trf(k,:),time_model(:,k)');
end

% creation of model and sampled original STRFs.  modelplot and actualplot
% reconstructed using the first strfi_total_number singular values, wholeplot
% is with every singular value.
modelplot=reassemble_strf(sigmas,freq_model,time_model,strfi_total_number,size(S
TRF_Cell));
actualplot=reassemble_actual_strf(sigmas,srf,trf,strfi_total_number,size(STRF_Ce
ll)); ...
wholeplot=reassemble_actual_strf(sigmas,srf,trf,size(sigmas,1),size(STRF_Cell));
% scale!
minre=min(min(STRF_Cell));              % min of real STRF
maxre=max(max(STRF_Cell));              % max of real STRF
minmo=min(min(modelplot));              % min of model
maxmo=max(max(modelplot));              % max of model
modelplot=(((modelplot-minmo)/(maxmo-minmo))*(maxre-minre))+minre; %scales model
 to min/max of STRF_Cell


% calculation of model/actual firing rates.  see computefr for details.
[modelfr]=computefr(songdatapath,modelplot);
[actualfr]=computefr(songdatapath,actualplot);

% save data and figures!

ver=str2num(version('-release'));       % saves in appropriate .mat format
                                        % because mathworks fails at life

if (ver>=14)
   save(outfile,'-v6','time_model','freq_model','freq_coeff','time_coeff','seperability','freq_si
','time_si','freq_mse','time_mse','modelplot','actualplot','srf','trf','sigmas','acausal_sigmas','
modelfr','actualfr');
else
   save(outfile,'time_model','freq_model','freq_coeff','time_coeff','seperability','freq_si','tim
e_si','freq_mse','time_mse','modelplot','actualplot','srf','trf','acausal_sigmas','sigmas','modelfr
','actualfr');
end
```

```
% GABOR_STRF_BATCH : Batch modeling and image generation using
% GABOR_STRF-generated data.
%

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

a=dir('./*.mat');                        % this is getting all modeled STRF
                                         % mat-files from current directory -
                                         % not the best way to do things!

b={a.name};                              % further evidence of >=matlab7.1 requir
ements.
b=b';
srfit=zeros(length(b),1);
trfit=zeros(length(b),1);
srmse=zeros(length(b),1);
trmse=zeros(length(b),1);

for i=1:length(b),
load(b{i});
srfit(i)=freq_si(1);                     % takes time, frequency SIs and MSEs
                                         % for first singular values.
trfit(i)=time_si(1);
srmse(i)=freq_mse(1);
trmse(i)=time_mse(1);
end

h=figure;
plot(srfit,'+:','linewidth',2);
xlabel('STRF Index','fontsize',22);
ylabel('Fit Value','fontsize',22);
title('Spectral Similarity Index, Sigma=1, 10 STRFs','fontsize',22);

j=figure;
plot(trfit,'+:','linewidth',2);
xlabel('STRF Index','fontsize',22);
ylabel('Fit Value','fontsize',22);
title('Temporal Similarity Index, Sigma=1, 10 STRFs','fontsize',22);

setparams;
load params;

for i=1:length(b),
load(b{i});

k=figure; imagesc(params.strfTScale, params.strfFScale,modelplot);
axis xy;
axis(params.strfAxesParams);
xlabel('Time (ms)');
ylabel('Freq (kHz)');
title(strcat(b{i},',model'));
print(k,'-dpng','-r300',strcat(b{i},'model'));

l=figure;imagesc(params.strfTScale, params.strfFScale,actualplot);
```



```
axis xy;
axis(params.strfAxesParams);
xlabel('Time (ms)');
ylabel('Freq (kHz)');
title(strcat(b{i},',actual'));
print(l,'-dpng','-r300',strcat(b{i},'actual'));
end
```

```
% GABOR_STRF_EARLAB : creation of EarLab Data Viewer-readable binary files
% from GABOR_STRF-generated data.
%

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

function [a] = gabor_strf_earlab(inputpath,outputpath,numstrfs)
a=dir(strcat(inputpath,'/*.mat'));        % all .mat files from inputpath;
                                          % these should be models only!
b={a.name};                               % further evidence of >=matlab7.1 requir
ements.
b=b';
[b,indb]=sort(b);
earlab_matrix=zeros(1617,numstrfs);       % create a bunch of 1617-element row
                                          % vectors
for i=1:numstrfs,
  load(strcat(inputpath,'/',b{i}));
  earlab_matrix(:,i)=modelfr;
end

earlab_matrix=earlab_matrix';
earlab_matrix_ud=flipud(earlab_matrix);

fid=fopen(strcat(outputpath,'/earlabmodel_asciisort.binary'),'w');
count=fwrite(fid,earlab_matrix,'float');
stat=fclose(fid);

fid2=fopen(strcat(outputpath,'/earlabmodel_iicsasort.binary'),'w');
count2=fwrite(fid,earlab_matrix_ud,'float');
stat2=fclose(fid2);

% b={ 'gn03-33-oldsongs20-model.mat';
%     'bg-38-oldsongs20-model.mat';
%     'gn03-25-oldsongs20-model.mat';
%     'bg-51-oldsongs20-model.mat';
%     'gn02-4-oldsongs20-model.mat';
%     'ae-14-oldsongs20-model.mat';
%     'gn03-29-oldsongs20-model.mat';
%     'yw10-3-oldsongs20-model.mat';
%     'bg-43-oldsongs20-model.mat';
%     'gn03-31-oldsongs20-model.mat';
% }
```

```
% GABOR_SRF_MODEL : Least-Squares Modeling of TRF data.
%                    [ydataFit,x,resnorm] = gabor_srf_model(xdata,ydata)

% Graham Voysey
% Original version of code provided by Gilberto Grana, endymion at bu dot edu
% Extensive assistance by Rajiv Narayan, rn at bu dot edu
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

% !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
% !NOTE! Requires MATLAB Optimization Toolbox!
% !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

% FITTER - A lazy-man's function for fitting exponentially-increasing data.
% Xdata and Ydata have to be defined from the beginning as XDATA = Time, YDATA =
% all points from t10 analysis

function [ydataFit,ydataAdj,x,resnorm,x0,tc] = gabor_trf_model(xdata,ydata)

numunits = size(ydata,1);
if ndims(ydata)==3
    numsets = size(ydata,3);
else
    numsets = 1;
end

% ydataAdj=ydata;

% Get the maximum values of each tCurve for each unit and, if applicable,
% each number of sets created
for i=1:numunits
    for j=1:numsets
        [meanYmax(i,j), meanYmaxdex(i,j)] = max(ydata(i,:,j));
        [meanYmin(i,j), meanYmindex(i,j)] = min(ydata(i,:,j));
        ydataAdj(i,meanYmaxdex(i,j):end,j) = meanYmax(i,j);

    end
end

% figure
% plot(xdata,ydataAdj)
% hold on
% plot(xdata,ydata,'.')

for i=1:numunits
    for j=1:numsets
        % x0(i,:)=[meanY_max(i)+5 meanY_max(i) 0.001]
        x0(i,:,j)=[meanYmax(i,j) meanYmax(i,j)-meanYmin(i,j) 0.001]';
%         ub(i,:,j)=[100 meanYmax(i,j)-meanYmin(i,j) 1]';
%         lb(i,:,j)=[meanYmin(i,j) 0 0]';
%        [x(i,:,j),resnorm(i,j)]=lsqcurvefit(@srf_gabor_fun,x0(i,:,j),xdata(i,:)
,ydata(i,:,j),lb(i,:,j),ub(i,:,j));
        [x(i,:,j),resnorm(i,j)]=lsqcurvefit(@trf_gabor_fun,x0(i,:,j),xdata(i,:),
ydata(i,:,j));
       ydataFit(i,:,j)=srf_gabor_fun(x(i,:,j),xdata(i,:));
    end
```



```
end

%tc=1./x(:,3,:);
% figure
% plot(xdata,ydata,'.')
% hold on
% plot(xdata,ydataFit)

% for i=1:size(ydata,1)
% %    x(i,:)-0.5*max(yfitdata(i,:))
%     t_half(i)=-tc(i)*log((x(i,1)-0.5*max(yfitdata(i,:)))/x(i,2));
% end
```

```
% GENERATE_HTML_DOCS : generate HTML documentation using m2html


% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA



path(path,'./docs/m2html');
path(path,'../');
m2html('mfiles',{'.','./computefr'},'htmldir','docs/html/gabor_strf','recursive','off','source','off
','graph','on','indexfile','docs');
```

```
% MSE: Mean-Squared Error.
%      E = MSE(signal1,signal2) returns the mean-squared error between the two,
%      considering signal1 to be the "original" signal and signal2 to be the
%      "model".
%
%      signal1 and signal2 must be of like dimension.
%
%      If the squared-sum of the components of signal1 is 0, no MSE can be
%      calculated.
%
%      MSE will always return a value between 0 and 1, with values closer to 0
%      indicating a closer match between signals.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA



function[mserr] = mse(original,model)

% check that input1 and input2 are of the same size
if (nargin ~=2)
  error('Not enough inputs');
end

if (size(original) ~= size(model))
  error('Input dimension mismatch');
end

% compute MSE.
num=0;
den=0;
for i=1:size(original,1)
  for j=1:size(original,2)
    num=num+((model(i,j)-original(i,j))^2);
    den=den+(original(i,j)^2);
  end
end

if (den==0)
  error('MSE uncalculatable; denominator is zero!');
end

mserr=num/den;
```

```
% NOISE_ESTIMATION: estimation of noise levels in STRFs.
%                   NUM_SIG_VALS = NOISE_ESTIMATION(STRF,sigmas) returns the max
imum
%                   number of significant values needed to recreate a STRF.
%                   The SVD of the acausal portion of the STRF is taken, and
%                   the highest value found is then used as a threshhold
%                   value to highpass filter the singular values of the
%                   overall STRF.  STRF is an experimental STRF, sigmas is
%                   its associated vector of singular values.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

function [k,acausal_sigmas] = noise_estimation(STRF,sigmas)

% primary error checking (i'll do this later)

acausal=STRF(:,1:floor(size(STRF,2)/2));          % acausal is for t<0, not
                                                  % t<=0.  no assumptions
                                                  % about STRF size are made,
                                                  % hurray.

acausal_sigmas=svd(acausal);             % decompose acausal portion, extract
                                         % singular values

% secondary error checking
if (acausal_sigmas(1)>sigmas(1))
  error('Acausal STRF more informative than Causal - This is an Ex-Zebra Finch!');
end

% begin highly ineffectual sort-and-count code block
k=0;
for j=1:size(sigmas,1),
  if (sigmas(j)>acausal_sigmas(1))
    k=k+1;                               % dear god why doesn't matlab have a
                                         % ++ operator?
  end
end
% end highly ineffectual sort-and-count code block
```

```
% PLOT_STRF_MODELS : Plots appropriately−windowed STRF models and
%                    experimental STRFs.  Saves them as 300dpi PNG files.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA

setparams;
load params;
a=dir('./*.mat'); %todo: this path should depend on user input.
b={a.name};
b=b';
for i=1:length(b),
load(b{i});

h=figure; imagesc(params.strfTScale, params.strfFScale,modelplot);
axis xy;
axis(params.strfAxesParams);
xlabel('Time (ms)');
ylabel('Freq (kHz)');
title(strcat(b{i},'−model'));
print(h,'−dpng','−r300',strcat(b{i},'−model.png'));

j=figure;imagesc(params.strfTScale, params.strfFScale,actualplot);
axis xy;
axis(params.strfAxesParams);
xlabel('Time (ms)');
ylabel('Freq (kHz)');
title(strcat(b{i},'−actual'));
print(j,'−dpng','−r300',strcat(b{i},'−actual.png'));
end
```

```
% REASSEMBLE_ACTUAL_STRF :  Recombine many trfs and srfs into an overal STRFm.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA % Graham Voysey


function [reassembled] = reassemble_actual_strf(sigmas,srf,trf,strfi_total_number,size_of_strf)

reassembled=zeros(size_of_strf);
for k=1:strfi_total_number,
  temp=sigmas(k)*srf(:,k)*trf(k,:);
  reassembled=reassembled+temp;
end
```

```
% REASSEMBLE_STRF :  Recombine many model STRFi’s into an overal STRFm.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA % Graham Voysey


function [reassembled] = reassemble_strf(sigmas,srf,trf,strfi_total_number,size_of
_strf)

reassembled=zeros(size_of_strf);
for k=1:strfi_total_number,
  temp=sigmas(k)*srf(:,k)*trf(:,k)’;     % remember to invert when calling
                                        % function if needed
  reassembled=reassembled+temp;
end
```

```
% SEPERABILITY_INDEX: Determination of seperability.
%                     [alpha] = seperability_index(sigmas,k) takes sigmas, a row
%                     vector of significant values, and k, a number representing
%                     the kth−highest−order significant value to be considered.
%                     It returns alpha, a number between 0 and 1 that represents
%                     the seperability of the STRF represented by the
%                     significant values.  Values closer to 1 indicate a
%                     well−seperable STRF.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA



function [alpha] = seperability_index(sigmas,k)

num=sigmas(1)^2 − sum(sigmas(2:k).^2);
den=sigmas(1)^2 + sum(sigmas(2:k).^2);

if (den==0)
  error('Seperability uncalculatable; denominator is zero!');
end

alpha=num/den;
```

```
% SETPARAMS : Set global parameters for images

% Graham Voysey
% Senior Project 2005-2006
% Adapted from Rajiv's computefr script

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA


%SETPARAMS.M
%Set global parameters
%Remember to run this script after updating values

function setparams()

%current default parameters

fname='params';

        params = struct (...
            'songSampRate',32000,...                    %Sampling Rate of song (hz)

            'songAxesParams',[1000,5000,0.25,8.0],...   %Axis params for plotting [ms,ms,khz,khz]
            'specFftN',128,...                          %DFT length for spectrogram (samples)
            'specWindow',128,...                        %window size for computing spectrogram (samples)
            'specNumOverlap',64,...                     %spectrogram window overlap (samples)
            'specBandwidth',[250,8000],...              %freqs of interest in the spectrogram (hz)
            'strfSampRate',1000,...                     %Sampling Rate of strf (hz)
            'strfAxesParams',[0,100,0.25,8.0],...       %Axis params for potting strf [ms,ms,khz,khz]
            'strfTScale',[-300:1:300],...               %time scale for strf (ms)
            'strfFScale',[0.375:0.25:7.875],...         %freq scale for strf (khz)
            'strfCropRange',[300:399],...               %time of interest in the strf (samples)
            'defaultSongDir','/home/rn/data/strfdata/songs',...  %default directory for songs
            'defaultStrfDir','/home/rn/data/strfdata/strfs',... %default directory for strfs
            'defaultAnalysisDir', '/home/rn/data/strfdata/analysis',... %Defaults dir for analysis files
            'firingOnset',1,...                         %start of stimulus (samples), use to introduce delay
            'firingDuration',6000, ...                  %length of firing rate vector (ms), set to max song length
            'firingRateGain',10.0, ...                  %Firing Rate Scale factor
            'numSpikeTrainsPerSong',10, ...                 %Number of spike trains to generate per song
            'spikeDistTau',10, ...                      %default time constant for spike distance (ms)
            'spikeDistT',6000,...                       %default T value (length of spike train) for spike distance (ms)
            'spikeDistTauRange','1:5:100',...           %Default range of tau values to test (ms)
            'spikeDistTRange','0:200:6000',...          %Default range of T values to test (samples)
            'sterfSignature', 'sterfv1.1'...                     %Current sterf data format version
        );


    save(fname,'params')
    s=sprintf('Saved parameters in %s',fname);
    disp(s)
```

```
% SFITSTRF:  Spectral Receptive Field Modeling
%            [x, resnorm, model] = SFITSTRF(srfi) seeds lsqcurvefit with
%            appropriate−ish values to begin minimization fitting, calls
%            lsqcurvefit to minimize the spectral receptive field to the
%            function defined in SRF_GABOR_FUN.  Returns the row vector
%            x of coefficents, values of the resnorms, and the model itself.


% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA

function [x, resnorm, model] = sfitstrf(srfi)

xdata=[500:250:8000];                   % frequency range
ydata=srfi’;                            % the transpose of the input SRF
[ymax,yind]=max(ydata);                 % beginning to pick values to
                                        % populate x0
cf=xdata(yind);                         % center frequency
halfmax=ymax/2;
tophalf=find(ydata>halfmax);
if size(tophalf)==1                     % sanity check to make sure tophalf
                                        % has more than one value
  tophalf=[2+tophalf−1,2+tophalf+1];
end

bw=xdata(tophalf(end))−xdata(tophalf(1)); % bandwidth
x0=[ymax cf bw 5 0];
%lb=−ones(5,1)*inf;
%ub=−lb;
%opts=optimset(’tolfun’,1e−4,’maxiter’,inf);
%[x,resnorm]=lsqcurvefit(@srf_gabor_fun,x0,xdata,ydata,lb,ub,opts);
[x,resnorm]=lsqcurvefit(@srf_gabor_fun,x0,xdata,ydata);
model=srf_gabor_fun(x,xdata);           % populate the model and preserve dimens
ion
```

```
% SI: Matrix Similarity Index.
%     S = SI(signal1,signal2) returns the similarity index between two matrix
%     inputs signal1 and signal2, after first "vectorizing" the inputs.

% Graham Voysey
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−1301 USA

function [similarity] = si(original,model)
% Like sii.m, this will calculate a similarity index between two items.
% However, these items are no longer vectors, and are m x n matricies instead.
% "The statistically significant samples of the STRF that exceeded a
% significance criterion of P < .002 were converted into a unidimensional vector
% from which the SI was determined" − qiu et al.

% THIS FILE HAS NOT YET BEEN IMPLEMENTED.
```

```
% SII: Vector Similarity Index.
%     S = SII(signal1, signal2): returns the similarity index between the two,
%     considering signal1 to be the "original" signal and signal2 to be the
%     model.
%
%     Calculates a similarity index between two vectors using the algorithm
%     given in "Gabor Analysis of Auditory Midbrain Receptive Fields:
%     Spectro-Temporal and Binaural Composition", Qiu et al, J. Neurophysiol,
%     2003.  Numerically equivalent to the Pearson Correlation Coefficient.
%
%     If either signal1 or signal2 have a norm-2 of 0 (eg, they are zeros-only),
%     no index can be calculated.
%
%     SII will always return a value between -1 and +1.  Values closer to abs(1)
%     indicate a closer correlation.  A value of 1 will be returned if the
%     signals are identical, and a value of -1 if they differ only by a negative
%     sign.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

function[similarity] = sii(original,model)
% error checking
if (nargin ~=2)
  error('Not enough inputs');
end

if (size(original) ~= size(model))
  error('Input dimension mismatch');
end

if (norm(original,2) == 0 || norm(model,2) ==0)
  error('similarity uncalculatable; a norm = 0!');
end
% calculation
similarity=dot(original,model)/(norm(original,2)*norm(model,2));
```

```
% SKEW: Determination of Temporal Skew.  [T] = skew(trf) returns the skewed
%        temporal axis to provide a better fit for gabor models.
%        Physiologically this accounts for differences in neuron onset/offset
%        times which skew the temporal profile and make it harder to fit it with
%        gabor functions.

% Graham Voysey
% Senior Project 2005-2006
% Boston University, College of Engineering, Department of Biomedical Engineering
% Natural Sounds and Neural Coding Lab
%
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or
% (at your option) any later version.
%
% gabor_strf is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with gabor_strf; if not, write to the Free Software
% Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

function [T] = skew(trf)
% THIS FILE HAS NOT YET BEEN IMPLEMENTED.
T=trf;
```

```
% SRF_GABOR_FUN : Gabor function for modeling of the Spectral Receptive Field.
%                 fitgabor = SRF_GABOR_FUN(x,xdata)

% Graham Voysey
% Original version of code provided by Gilberto Grana, endymion at bu dot edu
% Extensive assistance by Rajiv Narayan, rn at bu dot edu
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−13v01 USA

function FITGABOR = srf_gabor_fun(x,xdata)
FITGABOR = x(1)*exp(−((2*(xdata−x(2))/(x(3))).^2)).*cos(2*pi*x(4)*(xdata−x(2))+
x(5));
% x(1): K, strength of response
% x(2): omega_0, center frequency
% x(3): BW, bandwidth (cannot be zero)
% x(4): Omega_o, best ripple density
% x(5): P , spectral phase
```

```
% TFITSTRF:  Temporal Receptive Field Modeling
%            [x, resnorm, model] = TFITSTRF(srfi) seeds lsqcurvefit with
%            appropriate-ish values to begin minimization fitting, calls
%            lsqcurvefit to minimize the spectral receptive field to the
%            function defined in TRF_GABOR_FUN.  Returns the row vector x of
%            coefficents, values of the resnorms, and the model itself.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA
% Graham Voysey
% Senior project
% Last Modified: 01 03 06

function [x, resnorm, model] = tfitstrf(srfi)

xdata=[-300:300];                        % time range, ms
ydata=srfi;                              % the TRF
[ymax,yind]=max(ydata);                  % starting to populate x0
cf=xdata(yind);                          % "center frequency"
halfmax=ymax/2;
tophalf=find(ydata>halfmax);             % sanity check: make sure x3 isn't 0
if size(tophalf)==1
  tophalf=[tophalf-1,tophalf+1];
end
bw=xdata(tophalf(end))-xdata(tophalf(1));
x0=[ymax cf bw 5 0];
%lb=-ones(5,1)*inf;
%ub=-lb;
%opts=optimset('tolfun',1e-4,'maxiter',inf);
%[x,resnorm]=lsqcurvefit(@srf_gabor_fun,x0,xdata,ydata,lb,ub,opts);
[x,resnorm]=lsqcurvefit(@srf_gabor_fun,x0,xdata,ydata);
model=srf_gabor_fun(x,xdata);            % populate model
```

```
% TIMEWARP:  Time-Skewing Function.
%            [scaled_time]=TIMEWARP(trf) does skewing of the TRF based on qiu's handwave.

% Graham Voysey
% Senior Project 2005-2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110-1301 USA

function [scaled_time] = timewarp(trf)

%THIS FILE HAS NOT YET BEEN IMPLEMENTED.
scaled_time = trf;
```

```matlab
% TRF_GABOR_FUN : Gabor function for modeling of the Temporal Receptive Field.
%                 fitgabor = TRF_GABOR_FUN(x,xdata)

% Graham Voysey
% Original version of code provided by Gilberto Grana, endymion at bu dot edu
% Extensive assistance by Rajiv Narayan, rn at bu dot edu
% Senior Project 2005−2006

% Boston University, College of Engineering, Department of Biomedical
% Engineering
% Natural Sounds and Neural Coding Lab
% gvoysey at bu dot edu

% This file is part of gabor_strf.
%
% gabor_strf is free software; you can redistribute it and/or modify it under
% the terms of the GNU General Public License as published by the Free Software
% Foundation; either version 2 of the License, or (at your option) any later
% version.
%
% gabor_strf is distributed in the hope that it will be useful, but WITHOUT ANY
% WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
% A PARTICULAR PURPOSE.  See the GNU General Public License for more details.
%
% You should have received a copy of the GNU General Public License along with
% gabor_strf; if not, write to the Free Software Foundation, Inc., 51 Franklin
% St, Fifth Floor, Boston, MA 02110−13v01 USA

function FITGABOR = trf_gabor_fun(x,xdata)
FITGABOR = x(1)*exp(−((2*(xdata−x(2))/(x(3))).^2)).*cos(2*pi*x(4)*(xdata−x(2))+
x(5));
% x(1): Strength of temporal response
% x(2): Peak latency
% x(3): time−skewed duration of response (cannot be 0)
% x(4): temporal modulation frequency
% x(5): phase of the sinusiodal component about the peak latency.
```

```
%COMPUTEFR computes the firing rate from a strf and a song
%   FR = COMPUTEFR(songname, strfname)
%   Returns the firing rate obtaing from songname and strfname

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [fr,spec] = computefr(songname,strffull)

%load parameters
        load params;

%load songfile
        song=getsong(songname);

%crop region of interest
         strf=strffull(:, params.strfCropRange);

%compute spectrogram of song
       [tSpec,fSpec,spec]=getsongspec(song,params);

%compute firing rate
        [fr,frlength]=getfr(spec, strf, params);

% %plot song spec
%             figure;
%                       [tScale,fScale,cLims] = getplotsong (spec, tSpec, fSpec,
 params);

% %             %Display spectrogram
% %            imagesc (tScale,fScale,spec,cLims);
%             imagesc(tScale,fScale,spec);
%            axis xy; %set cartesian coord mode
%             xlabel('Time (ms)');
%             ylabel('Freq (kHz)');

% %display strf
%     figure;
%             %get plotting params
% %            [cLims]=getplotstrf(strf);

%             %plot the strf
% %            imagesc (params.strfTScale, params.strfFScale, strffull, cLims);
%             imagesc (params.strfTScale, params.strfFScale, strffull);
%             axis xy
%             axis(params.strfAxesParams);
%             xlabel('Time (ms)');
%             ylabel('Freq (kHz)');
```

```
%getfr.m
%compute firing rate from strf and spectrogram of song
% function [fr,sizefr]=getfr(spec,strf,params)

%This file is part of computefr, written by Rajiv Narayan and used with permission.

function [fr,sizefr]=getfr(spec,strf,params)

    dimStrf = size(strf);
    dimSpec = size(spec);
    fSpecRange = dimStrf(1); %number of filter banks (rows)

    inputChannels = zeros(fSpecRange,dimStrf(2)+dimSpec(2)-1);
  %===========================================

  %Perform convolution STRF * stimulus

    for i = 1:fSpecRange
        inputChannels(i,:) = conv(strf(i,:),spec(i,:)-mean(spec(i,:)));
    end

    %Generate firing rate
        stimDur = dimStrf(2)+dimSpec(2)-1;
%        fr = zeros(params.firingDuration,1);
        fr=zeros(stimDur,1);
        fr(params.firingOnset:params.firingOnset+stimDur-1) = params.firingRateGain.*sum(inputChannels)';
        fr(find(fr < 0))=0;
%         meanfr = mean(fr(params.firingOnset:params.firingOnset+stimDur-1));
%         disp (meanfr);

% sizefr=[params.firingDuration,1];
sizefr=stimDur;
```

```
%GETPLOTSONG.M
%Returns spectrogram params that can be used with imagesc
%[tScale,fScale,cLims,axisParams] = GETPLOTSONG(spec,tspec,fspec)
% tscale        timescale
% fscale        freq scale
% clims         scale factor for imagesc
% spec          spec data
% tspec         time scale
% fspec          freq scale
% params        Structure containing global variables

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [tScale,fScale,cLims]=getplotsong(spec,tspec,fspec,params)

                tonset      = params.firingOnset; %start of stimulus (samples)
        maxamp = max(max(spec));
        cLims = [maxamp-75 maxamp];
        tScale=tonset+tspec*1000; %ms
        fScale=fspec/1000;  %khz
```

```
%GETPLOTSTRF.M
%Returns color scale range for strf that can be plotted using imagesc
%[cLims] = GETPLOTSTRF(STRF)
% clims         scale factor for imagesc
% STRF      STRF data

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [cLims]=getplotstrf(strf)

    %compute range to scale the strf for display using imagesc
    maxForward = max(max(strf));
    minForward = min(min(strf));
    absForward = max(abs(maxForward),abs(minForward));
    cLims=[-absForward absForward];
```

```
%GETSONG.M - Read song file
%[SONG] = getsong(songfile)
%Returns the songfile as a vector(SONG),
% returns [] if file not found

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [SONG,SONGLEN] = getsong(songfile)

if (nargin<1)
    s=sprintf('insufficient arguments to %s',mfilename('fullpath'))
    error(s)
end

DISPERR=1; %display error message

if (exist(songfile,'file'))

        SONG=load(songfile);
        SONGLEN=length(SONG);

else %we couldnt find the file
        SONG=[];

        if (DISPERR)
            s=sprintf('File %s not found',songfile);
            disp(s)
        end
end
```

```
%getsongspec.m
%[tSpec,fSpec,spec]=getsongspec(stim,params)
%tspec  time vector
%fSpec  freq vector
%spec   spectrogram
%stim   song vector
%params Structure containing global parameters (see Setparams.m)
%Compute spectrogram of stimulus

%Changes
%Fixed Freqency scaling variable, Changed 7/24/2003, RN
%Lower limit of frequency included while cropping the strf, 3/28/2006, RN

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [tSpec,fSpec,spec]=getsongspec(stim,params)
    SAMP_RATE=params.songSampRate;
    window = params.specWindow;
    nfft = params.specFftN;
    noverlap = params.specNumOverlap;
    [b,fSpecOrig,tSpecOrig] = specgram(stim,nfft,SAMP_RATE,window,noverlap);
    b=b+(b==0)*eps; %to avoid dbz , 7/1/2003 RN
    bdB = 20*log10(abs(b));

    %keep frequency channels between 250 Hz to  8 KHz
    fSpecInd = find((fSpecOrig<=params.specBandwidth(2))&(fSpecOrig>=params.spec
Bandwidth(1)));
    fMinInd = min(fSpecInd);
    fMaxInd = max(fSpecInd);
    fSpec = fSpecOrig(fMinInd:fMaxInd);
    specOrig = (bdB(fMinInd:fMaxInd,:));

    %perform upsampling by 2
    tSpec = interp(tSpecOrig,2);

    spec = zeros(length(fSpec),length(tSpec));

    for fInd = 1:length(fSpec)
        spec(fInd,:) = interp(specOrig(fInd,:),2);
    end
```

```
%GETSTRF.M - Read strf file
%[STRF] = getstrf(strffile)
%Returns the strffile as a vector(SONG),
% returns [] if file not found

%This file is part of computefr, written by Rajiv Narayan and used with permissi
on.

function [STRF] = getstrf(strffile)

if (nargin<1)
    s=sprintf('insufficient arguments to %s',mfilename('fullpath'))
    error(s)
end

DISPERR=1; %display error message

if (exist(strffile,'file'))

        STRF=load(strffile,'ascii');

        %Crop STRF from 0 to 100 ms
%       STRF = STRF(:,200:299);

else %we couldnt find the file
        STRF=[];

        if (DISPERR)
            s=sprintf('File %s not found',strffile);
            disp(s)
        end
end
```

```
%GETSTRFPARAMS
% computes STRF parameters
% y = getstrfparams(x,params)
% y - structure containing the strf params:
%       y.cf - peak frequency (kHz)
%       y.tpeak - time to peak (ms)
%       y.ei - Excitation / Inhibition Ratio
%       y.maxval - Excitatory peak value
%       y.maxcoord - indices of maxval
%       y.minval - Inhibitory peak value
%       y.mincoord - indices of minval
%       y.absmaxval - Absolute max value
%       y.absmaxcoord - indices of absmaxval

%This file is part of computefr, written by Rajiv Narayan and used with permission.

function y = getstrfparams(x,params)

%find max across columns
[maxcol, maxind] = max(x);
%find maximum
[y.maxval, maxt] = max(maxcol);
y.maxcoord = [maxind(maxt), maxt];

%find min value
[mincol, minind] = min(x);
[y.minval, mint] = min(mincol);
y.mincoord = [minind(mint), mint];

%find absolute maximum
[y.absmaxval, i] = max(abs([y.maxval, y.minval]));

switch (i)
    case 1 %then take the slice through maxval
        y.absmaxcoord = [maxind(maxt), maxt];
        y.tpeak = params.strfTScale(maxt); %time to peak in ms
        y.cf = params.strfFScale(maxind(maxt)); %peak frequency (kHz)

    case 2 %then take the slice through minval
        y.absmaxcoord = [minind(mint), mint];
        y.tpeak = params.strfTScale(mint); %time to peak in ms
        y.cf = params.strfFScale(minind(mint)); %peak frequency (kHz)

    otherwise
        disp (mfilename);
        disp('Unable to compute tpeak , cf');
end

%compute E/I Ratio
y.ei = abs(y.maxval) / abs(y.minval);
```

```matlab
%SETPARAMS.M
%Set global parameters
%Remember to run this script after updating values

%This file is part of computefr, written by Rajiv Narayan and used with permission.


function setparams()

%current default parameters

fname='params';

        params = struct (...
            'songSampRate',32000,...                    %Sampling Rate of song (hz)
            'songAxesParams',[1000,5000,0.25,8.0],...   %Axis params for plotting [ms,ms,khz,khz]
            'specFftN',128,...                          %DFT length for spectrogram (samples)
            'specWindow',128,...                        %window size for computing spectrogram (samples)
            'specNumOverlap',64,...                     %spectrogram window overlap (samples)
            'specBandwidth',[250,8000],...              %freqs of interest in the spectrogram (hz)
            'strfSampRate',1000,...                     %Sampling Rate of strf (hz)
            'strfAxesParams',[0,100,0.25,8.0],...       %Axis params for potting strf [ms,ms,khz,khz]
            'strfTScale',[-300:1:300],...               %time scale for strf (ms)
            'strfFScale',[0.375:0.25:7.875],...         %freq scale for strf (khz)
            'strfCropRange',[300:399],...               %time of interest in the strf (samples)
            'defaultSongDir','/home/rn/data/strfdata/songs',...  %default directory for songs
            'defaultStrfDir','/home/rn/data/strfdata/strfs',... %default directory for strfs
            'defaultAnalysisDir', '/home/rn/data/strfdata/analysis',... %Defaults dir for analysis files
            'firingOnset',1,...                         %start of stimulus (samples), use to introduce delay
            'firingDuration',6000, ...                  %length of firing rate vector (ms), set to max song length
            'firingRateGain',10.0, ...                  %Firing Rate Scale factor
            'numSpikeTrainsPerSong',10, ...                  %Number of spike trains to generate per song
            'spikeDistTau',10, ...                      %default time constant for spike distance (ms)
            'spikeDistT',6000,...                       %default T value (length of spike train) for spike distance (ms)
            'spikeDistTauRange','1:5:100',...           %Default range of tau values to test (ms)
            'spikeDistTRange','0:200:6000',...          %Default range of T values to test (samples)
            'sterfSignature', 'sterfv1.1'...                %Current sterf data format version
        );



     save(fname,'params')
     s=sprintf('Saved parameters in %s',fname);
     disp(s)
```