The TFSAP Source

B. Boashash [1]
S. Ouelha [2]

December 29, 2016

[1] owner of the toolbox
[2] maintainer and "technical support"

1 The Source Tree

Below is an outline and description of the directory structure of the TFSAP source:

`tfsa6.1/`

`Makefile.ubuntu`	The top-level Makefile for Ubuntu systems
`Makefile.msvc`	The top-level Makefile for Microsoft Visual C systems
`doc`	Source files to generate this documentation
`bin/`	The default directory created when installing the compiled TFSAP package
`data/`	Contains data files for the TFSAP demo
`func_m/`	Contains help m-files for the TFSAP functions
`func_src/`	Contains the C source for the TFSAP functions
`func_support/`	Contains C source for any support functions used by the TFSAP C source functions
`func_wrap/`	The C source wrapper/gateway files for the TFSAP functions to allow them to be compiled into MEX files
`gui_m/`	The M-files for TFSAP GUI and support functions
`include/`	C header files for the C source TFSAP and support functions

Additionally there are second-level Makefiles (`Makefile.ubuntu` and `Makefile.msvc`) in the three source directories (`func_src`, `func_support` and `func_wrap`).

2 The Makefiles

The top-level Makefiles are located in the base directory of the TFSAP source tree. The file `Makefile.ubuntu` should be used when compiling and installing the TFSAP package on a Ubuntu system using GCC. The file `Makefile.msvc` should be used when compiling and installing the TFSAP package on a Windows system using Microsoft Visual C. The top-level Makefiles may be logically divided into four sections.

The first section consists of parameters that the user should change before compiling. The options to be edited here are where MATLAB is installed, where TFSAP should be installed and whether TFSAP should be compiled with optimization and/or debugging information.

The second section allows the user to add names of source files that have been added to the source tree. The filenames that may be added in this section are the source filename of a new TFSAP function and the source filenames of new support source file used by a TFSAP function. Adding source code for a new TFSAP function will be described in more detail below.

The third section should not be edited. It contains appropriate compiler flags and options appropriate to the system it is written for. The fourth section contains compilation targets and rules for the functions provided by TFSAP. This section should only be edited if a new function is added to TFSAP. In this case a new target and rule should be added to compile the source for the new function.

The secondary Makefiles in the three C source sub-directories provide implicit rules for compiling C source files into object files. These should not need to be edited.

3 Creating a new TFSAP function

Adding a new function to TFSAP requires the coding of source files to perform the function and editing of the TFSAP GUI files to provide access to the new function. The process to be followed to add an arbitrary function cannot be outlined as it would require an exhausting

amount of generalisation. Presented here instead, is an example of adding the B-distribution to the available bilinear analysis methods. Only segments of the code are shown here. Please refer to the full source for a thorough examination of the method. As the B-distribution is a Quadratic class kernel, a new set of source files do not need to be created as the code can be added to the already existing Quadratic class kernel source files. The steps to take are:

- Edit the file `quadknl.c` which is the source for the Quadratic class kernel generation functions. This is found in the `func_src` directory. At the end of the file add the implementation for the B-distribution kernel generation function:

```
int b_kernel(double **kernel, unsigned window_length, int full, double sigma1)
{

    ...

}
```

- Edit the file `quadknl.h` which is the source for the Quadratic class kernel generation function headers. This is found in the `include` directory. Add a definition for the B-distribution type at the beginning of the file:

```
#define B 8
```

  At the end of the file add the declaration of the B-distribution kernel generation function:

```
int b_kernel(double **kernel, unsigned window_length, int full, double sigma1);
```

- Edit the file `quadknl.m` which is the MATLAB help file for the Quadratic class kernel generation functions. This is found in the `func_m` directory. Add the information for the use of the B-distribution kernel.

- Edit the file `quadknl_m.c` which is the wrapper/gateway file for compiling `quadknl.c` into a MEX file. This is found in the `func_wrap` directory. Add the appropriate checks to ensure that the `b_kernel` function is being called with the correct arguments:

```
else if (!strcmp(p, ``b'') || !strcmp(p, ``B''))
    kernel_type = B;

...

case(B):
    if(nrhs > 4) {
        tfsaErr("quadknl", "Too many input arguments");
        winNTcheck(nlhs, plhs);
        return;
    }
    break;

...
```

```
case( B ):
    if(nrhs < 4)  {
        tfsaErr("quadknl", "B distribution requires a smoothing parameter (sigma)");
        winNTcheck(nlhs, plhs);
        return;
    }
    if (!GoodScalar( prhs[3])) {
        tfsaErr("quadknl", "Smoothing parameter sigma must be a scalar");
        winNTcheck(nlhs, plhs);
        return;
    }

...

case(B):
    b_kernel(G_real, window_length, full_kernel, d_param);
    break;
```

- Next it is necessary to edit `quadtfd_m.c` which is the wrapper/gateway file for compiling `quadtfd.c` into a MEX file. This is found in the `func_wrap` directory. This is necessary as the `quadtfd` function must now support using the B-distribution kernel. As before we need to add the appropriate checks to ensure that the `quadtfd` function is being called with the correct arguments:

```
else if(!strcmp(p, "b") || !strcmp(p, "B"))
    kernel_type = B;

...

case(B):
    if (nrhs < 5)  {
        tfsaErr("quadtfd", "Not enough input arguments");
        winNTcheck(nlhs, plhs);
        return;
    }
    if (nrhs > 6) {
        tfsaErr("quadtfd", "Too many input arguments");
        winNTcheck(nlhs, plhs);
        return;
    }

...

case B:
    b_kernel(G_real, window_length, 0, d_param);
```

- Edit the file `quadtfd.m` which is the MATLAB help file for the Quadratic class functions. This is found in the `func_m` directory. Add the information for the use of `quadtfd` with the B-distribution kernel.

- Next it is necessary to edit `uif_btfd.m` which is the code that manages the callbacks for the Bilinear Analysis screen. This is found in the `gui_m` directory. We need to add code to make the B-distribution appear as a kernel option together with the smoothing parameter option:

```
EDIT_BPAR = 25;
TEXT_BPAR = 26;

...

B = 10;

...

if (dist == B)
    bparam = str2num(get(ud(EDIT_BPAR), 'String'));
    if isempty(bparam)
        tfsa_err('Distribution parameter value is invalid');
        return;
    end
end

...

elseif dist == B
    output1 = quadtfd(input1, wl, tr, 'b', bparam, fftl);

...

% Turn on parameter sigma for BDIST - turn it off
% for all other dists
if (dist == B)
    set( ud(EDIT_BPAR), 'Visible', 'on');
    set( ud(TEXT_BPAR), 'Visible', 'on');
    set( ud(TEXT_BPAR), 'String','Parameter sigma')
else
    set( ud(EDIT_BPAR), 'Visible', 'off');
    set( ud(TEXT_BPAR), 'Visible', 'off');
end
```

We also need to add the code to generate the smoothing parameter controls:

```
edit_bpar = uicontrol(  fignum, ...
   'Style', 'edit', ...
   'String', B_PARAM_DEF, ...
   'Units', 'normalized', ...
   'ForegroundColor', FG1, ...
   'Backgroundcolor', BG3, ...
   'Position', OPT10, ...
   'Visible', 'off');
```

```
ud(EDIT_BPAR) = edit_bpar;

text_bpar = uicontrol( fignum, ...
   'Style', 'text', ...
   'String', '', ...
   'Units', 'normalized', ...
   'ForegroundColor', FG1, ...
   'Backgroundcolor', BG1, ...
   'Position', OPT_LABEL10, ...
   'HorizontalAlignment', 'Left', ...
   'Visible', 'off');
ud(TEXT_BPAR) = text_bpar;
```

- Next it is necessary to add the default value for the smoothing parameter input control that will appear in the TFSAP GUI. Edit the file `uideflts.m` which is found in the `gui_m` directory and add:

```
B_PARAM_DEF = '0.01';
```

- The help for the B-distribution needs to be added. Firstly, the file `tfsamenu.m` which is found in the `gui_m` directory should be edited. We need to add the B-distribution option to the Bilinear Analysis sub-menu which appears under the TFSAP Help menu.

```
uimenu(h1,'Label','B-Distribution', ...
    'Callback','helpdata(''ToolBlinrBdist'',gcf);');
```

- The following should be added to the file `helpdata.m` which is found in the `gui_m` directory.

```
%=====================================================Quadratic's class

    elseif strcmp(code(10:14),'Bdist')
        ttlStr='TFSAP Help';
        hlpStr1= ...
    ['                                                          '

        ...

     '                                                          '];
     tfsahelp('initialize',figNum,ttlStr,hlpStr1,hlpStr2);
```

  Help should also be added in the same file to the section discussing the Quadratic class, which is an option under the Bilinear Analysis sub-menu.

4 Adding a new TFSAP function to the source tree

The implementation of a new TFSAP function should be contained within at least four source files. The new function (we'll take `psd` as an example) should be contained within a source file named `psd.c`. The header file for this function should be `psd.h`. The MATLAB wrapper/gateway file for the function (used in compiling the code into a MEX file) should be called `psd_m.c`. The help for this function should be contained in `psd.m`.

Any supporting source files for this function can be named as the user wishes as long as the header files have a similar name (e.g. `support.c` and `support.h`). Any demonstration data files required should be created (e.g. `psd_data`). Any new m-files required for the TFSAP GUI should be written and necessary editing of the GUI m-files should be performed.

After coding the source, the following lists where the code should be placed and what changes should be made to the top-level Makefiles.

- Add the new function source `psd.c` file to the `func_src` directory.
- Add the new function header file `psd.h` to the `include` directory.
- Add the new function wrapper source file `psd_m.c` to the `func_wrap` directory.
- Add the new function m-file `psd.m` to the `func_m` directory.
- Add the new support source file `support.c` to the `func_support` directory.
- Add the new support source header file `support.h` to the `include` directory.
- Add the new data file `psd_data` to the `data` directory.
- Add the new function source file `psd.c` to the `func_src` variable in the top-level Makefiles. The function header, wrapper/gateway and m-file filenames necessary for the Makefiles will be derived from this source name.
- Add the new support source file `support.c` to the `func_support` variable in the top-level Makefiles. The support source header filename necessary for the Makefiles will be derived from this source name.
- Add a new target and rule for the function. This consists of the function's name, the dependency objects needed to compile it and a default compile line. e.g.

  ```
  psd.$(MEXEXT) :  psd.o support.o
                $(LD) $(LFLAGS) $(CFLAGS) $^ -o $@ $(LIBS)
  ```

  (Note that a TAB character rather than multiple spaces is required at the beginning of the second line of the rule.)
- Compile, install and test the new distribution.

5 Compiling

To compile and install the TFSAP package perform the following steps (we use a GCC Ubuntu system as an example):

- Edit the first section of the top-level `Makefile.ubuntu` file to specify the base directory of the MATLAB distribution.
- Edit the first section of the top-level `Makefile.ubuntu` to specify the install directory for the TFSAP package.
- Edit the first section of the top-level `Makefile.ubuntu` to select or deselect compiling the source with debugging information.

- Edit the first section of the top-level `Makefile.ubuntu` to select or deselect compiling the source with optimization.
- Execute `make -f Makefile.ubuntu all` to compile the source.
- Execute `make -f Makefile.ubuntu install` to install the TFSAP mex files, m-files and data files to the specified installation directory.

Executing `make -f Makefile.ubuntu clean` will remove all object files, mex files and backup files from the source tree. This is useful before doing a new compilation when any changes have been made to the source tree or to the Makefile.

5.1 Re-installation on UNIX

To perform a re-installation, the directory created during the previous installation will need to be removed. If, for example, the installation directory is `bin`, this can be done by:

- Changing the permissions of the files in the `bin` directory:

  ```
  chmod u+w bin/*
  ```

- Removing the directory:

  ```
  rm -r bin
  ```

It has happened on occasion that the directory is unable to be removed because a hidden file (usually something like `.nfsEF01`) exists in the directory. You will find if this file is removed, it is replaced by another file and it thus impossible to remove the directory. These files are related to the Network File System used when running MATLAB and TFSAP. We have found that this problem can usually be fixed by stopping any MATLAB processes you have running. Following this, remove the hidden file and then the directory as specified above.

You will now be able to re-install using: `make -f Makefile.ubuntu install`