TAPADM MANUAL

Preface

TAPADM means 'Three-dimensional Articulographic Position and Align Determination with MATLAB'. It is a compilation of *MATLAB*-Scripts for basic *AG500*-data handling, especially for calculation of sensor positions from measured amplitudes. So far, *TAPADM* is a mid tier software component, it does neither cover raw data acquisition and demodulation, nor high level things like correction of head movements. Still, there are plans to release Phil Hooles scripts for further processing, but that will be a different library.
Together with the *AG500* hardware, several programs for data acquisition and processing are distributed by *CARSTENS MEDIZINELEKTRONIK COMPANY* .[1] If you are looking for handy and well-tried evaluation software, which does not requires any programming skills, you should use the Carstens software.
Please read 'changelog.txt' for latest information!

Legal notice

System requirements

TAPADM was developed with *MATLAB* 6.5, it is a port of *TAPADCON.EXE*, which was written in C++ at the *IPSK* in Munich. *TAPADM* is organized like a *MATLAB*-Toolbox itself and some features may require other Toolboxes. However, the basic functionality should be available without any toolboxes. You will find more specific information's, within the documentation of the single scripts.
TAPADM was developed under Windows XP Pro, but it should run on any platform you can get *MATLAB* for. In case you encounter any platform specific problems, please report them to me.

Installation

Just extract the Zip-Archive to wherever you like and add this location to your *MATLAB* path. Do not put the 'private' subfolder on your path.
On the *MATLAB* command window, enter '`helpwin tapad`', to see if the path setting is correct.

---

[1] http://www.articulograph.de/
[2] Email: Anderas.Zierdt@phonetik.uni-muenchen.de
[3] See the file gpl.txt for the GNU General Public License

Usage

The central function of the Toolbox is '*tapad.m*', with it you will perform the position calculation task. This is where you'll want to start, after you've read the 'changelog.txt'. As with all *TAPADM* functions you can get a n explanation of what the function does by entering '`helpwin tapad`', or '`help tapad`'. Since *TAPADM* is subject to frequent changes, this manual can only cover the general concept.

The TAPAD Functon

Tapad is the core function of the TAPADM Toolbox, which performs position calculation for a set of EMA-Recordings ('trials'), each stored in a individual data file.

```
tapad(BasePath, AmpPath, ResPath, TrialIdx, ChanIdx, TapadOptions,
StartValues, UserFunction);
```

Mandatory Arguments

BasePath is somethig like '*C:\MyStuf\EMA*', i.e. the base for all data. To split the path in two parts complicates handling at first glance, but if you want to compute different versions of data, it simplifies things. Think of BasePath as name of your recording and than use ResPath to address different folders like 'results4levenberg', 'results4filtered', 'results1stTry',…
If you use a standardized directory structure, BasePath may also help you to run scripts for different data sets, or on different computers.

AmpPath and **ResPath** are defining the relative path for input and output data, e.g. `AmpPath = 'Data\Amp'; ResPath = 'Data\Pos'`;

TrialIdx defines the trials (files) to be processed. Input data is expected from one directory, with a 4-digits file pattern, like '0001.mat'. Perform the data pre-processing prior to tapad.m to generate these input data from the AG500 '.amp' files

ChanIdx names the channels to be computed, e.g. 1:12. If there is already an existing TAPADM result file and TAPADM is used to calculate position data just for some channels, the function will keep existing position data which is not affected by this run. Thus, position calculating can be performed in a sequential order: channel by channel.

Options

-d use amplitude derivatives to weight errors
-f flip time, i. e. process data onwards from the last point
-h don't use history (i.e. last result) as start point (significantly increases computation time!)
-l use Levenberg-Marquardt instead of Newton method
-r recursive mode, where individual start values for every sample are taken from former calculated result files. The path to this files is expected as 7th function-argument
-s user supplied initial start value for the first sample in the trial. Start values are expected as a 7th function-argument. (see below)
-a automatically starts the position calculation at the best suited sample in the trial and continues in both directions from that point on. (Disables -r and -f)

Extra Arguments

StartValues is a manifold argument:
In recursive mode (-r), StartValues defines the path where files with individual start positions can be found. (similar to AmpPath)
StartValues in conjunction with -s defines the start position to be used for first sample in trial (arranged 5 x channels). Accessed by channel number, not position in ChanIdx list!!!
UserFunction: If supplied, tapad.m will call this function for every trial with the following arguments:

```
msg = UserFunction(trial, NumOfTrials, ChanIdx, Result, Residuals,
Iterations)
```

If the function returns a character string, tapad will display it during the calculation of the next trial.

Data organization

AG500-Amp and Pos data files

The *AG500*-Software saves data in a binary format with the extension *amp* for demodulated signal amplitudes and *pos* for computed positions.[4] Both formats can be read by '*loaddata.m*'.
TAPADM does not write files in any of these formats, but uses *MATLAB* binary files with the extension 'mat' to store all data.

General (IPSK) data format for Mat-files

To simplify data handling, TAPADM uses a data structure, which is used at the IPSK for several projects. The main idea is, that each file has one data set and a couple of additional info structures.
The following code example will illustrate the structure of a data file and give you a hint of data exploration:

```
>> clear all
>> load 'D:\Matlab\TAPADM\data\pos\0001.mat'
>> whos
  Name             Size                    Bytes  Class

  comment          1x1981                   3962  char array
  data           500x7x12                 168000  single array
  descriptor       0x0                         0  char array
  dimension        1x1                       372  struct array
  private          1x1                       124  struct array
  samplerate       1x1                         8  double array
  unit             0x0                         0  char array
```

[4] When we speak of positions, we usually mean both spatial position and orientation (5 coordinates)

The organization of the data here is 500x7x12, i.e. 500 samples with 7 variables (x, y, z, phi, theta, exitflag, 'rms-value') and 12 channels. (It is a result-file.)

A suggestion for directory layout

Let's assume, we have recorded lots of data during a half-day EMA session, collected from our subject Bill A. Richards and this is part of the FOO-Project at our place. Let's further assume, all EMA data is stored on a Windows-PC on Partition 'E:' than a suitable base –path could be *E:\2005\foobar* and a directory structure could than look like this:

E:\2005\foobar\calibrationr\channel1…
E:\2005\foobar\calibrationr\channel12
E:\2005\foobar\raw (containing the 'amp'-files of all trials)
E:\2005\foobar\amps (containing the amplitude mat-files of all trials)
E:\2005\foobar\pos (containing the calculated positions for all trials)
E:\2005\foobar\LevNoStart (alternative positions)
E:\2005\foobar\down8\amps (amplitude files downsampled by factor 8)
E:\2005\foobar\down8\pos (positions for the downsampled data)
E:\2005\foobar\down8\LevNoStart (alternative positions, downsampled data)
E:\2005\foobar

How to…

Prepare new AG500 measurement data

During measurement, the *AG500* creates a set of files with the extension 'amp'. You will have to convert them into *MATLAB* binary files prior to anything else. Since the original data is not really altered during the conversation, it is not essential to keep the amp-files. Yet, considering the amp-files as the 'experimental observation', I would recommend backing them up.
You will than use *'prepdata.m'* to convert the data, as in the following example: [5]

```
>> prepdata('D:\Foo3Data', 'ag500', 'amps', 1, 'FIR2030');
```

Notice that a moderate low-pass filter is applied here during the conversation (assuming the amp-files are backed up) to eliminate HF-Ripple with a cutoff frequency of 20 Hz. The idea is, that this will not affect the trajectories of the sensors, but simplifies position calculation.
It is after all, a change and I would suggest, you try both filtered and unfiltered data, to get a feeling about the effects. When in doubt, do not filter amplitudes, filter position data in the spatial domain.
A main reason for using '*prepdata.m*' with filter is downsampling. It gives you the opportunity to generate an image of your newly acquired data. An image with reduced temporal resolution (e.g. 25 Hz), that will be perfect to probe data procession within reasonable time. If you think you have found the optimal setting for '*tapad.m*' and maybe have sorted out some trials, you can start processing the real data. (A Friday afternoon is good time for that…)
The following code example shows how to downsample your data to a 25 Hz rate:

```
>> prepdata('D:\Foo3Data', 'ag500', 'ds\amps', 1, 'FIR0512', 8);
```

Even though this Toolbox provides no low-pass filter which allows downsampling by more than factor 8 (equals a 25 Hz rate), you might decide to reduce the data even more to obtain a fast glance on the result, but this will inevitably ruin the reliability of your findings with respect to sensor movements.

[5] (We assume 'D:\Foo3Data\ag500\' holds '0001.amp', '0002.amp', and so on.)