# Automatic Report Producing from Results of Optimization Tasks in Batch Signal Processing

Slavy G. Mihov[1]

*Abstract* – **In this paper is presented a program solution (created in Matlab) designed to execute similar computations over a set of input data files, store results from every single processing (text, graphics, execution time, etc.) and produce a perspicuous report (*.doc, *.pdf) with the obtained results included. This tool performs batch signal processing and removes the necessity of human interaction until all processing is over and the report is automatically generated, which tends to be very helpful for long time intervals of calculations.**

*Keywords* – **generate automatic report, batch processing.**

## I. Introduction

Optimization tasks (particularly in signal processing) are a common problem with massive theory built for the purpose. Many times signal processing is complicated, time consuming and computationally expensive, especially in performing monotonous operations over packs of numerous signal files - batch processing.

In previous works, concerning algorithm development and evaluation for processing biological signals and estimating the results, a common use case was experimenting with various parameters of the processing algorithm to seek optimal results in some meaning. In [1] is investigated a wavelet denoising procedure for the purposes of hearing-aid signal processing. Particular database of recordings of human speech captured in various noisy environments (in several signal-to-noise ratios) is subjected to processing with a denoising algorithm. The wavelet denoising algorithm used, has numerous of parameters which adjust its performance, thus seeking the optimal algorithm profile turns in an optimization task with numerous independent arguments. An evaluation metric is used to estimate the quality of processing every single database recording. In this investigation, plausible results are obtained not until processing a large enough set of test recordings, which could have never been done (easily) without using a tool for batch signal processing and automatic report generation, generalizing the obtained results.

In the same manner [2] develops an algorithm for distinguishing atypical from typical heart beats in ECG signal for the purposes of automatic feature recognition. The evaluated algorithm uses an optimal linear transformation of the available leads in a multichannel ECG signal to derive a new lead with enhanced atypical beats. The pursue of this optimality is again an optimization task searching for a single set of coefficients for the linear transformation that produce the best result (according the optimization criterion). Evaluating the algorithm and proving its functionality [2, 3] is again subjected to processing large databases of different format ECG recordings. This could have also been a tedious task without the use of a tool for batch processing and automatically accumulating the results in the form of a report for the benefit of the human to estimate the processing.

The tool used for automatic report generation has improved more and more with every successive project. It gradually evolved from a couple of script lines, becoming more and more complex, to turn to a sophisticated and a irreplaceable in algorithm development and evaluation tool.

## II. Tool for Batch Processing

For the purposes of investigation and algorithm development [1, 3] is created an environment for processing and visualizing signals in the form of input data. This development tool is supposed to greatly facilitate and automate the minor, and insignificant for the processing operations.

The primary goal of the batch processing tool is to facilitate gaining and storing in a convenient form the results from processing numerous test datasets (ECG, human speech, etc. recordings). As a result from execution, in the working directory are created groups of files containing information for the processing. For instance, for each test recording is created a text log file (*.log) containing statistic for: computation time (begin, end, continuance), optimization criterion used, optimal algorithm parameter set obtained and so on. The input signals and output results are stored in the form of diagrams in vector graphic format (*.eps, *.emf), convenient to use in documentation and consecutive processing. Amongst the valuable features of the tool for batch processing is the ability to automatically produce perspicuous report (*.pdf, *.doc) including data for all processing, parameters of computations and results in the form of signal graphics, text statistics and tables. This feature tends to be very convenient and saves much time in browsing the suitable data recordings in large databases.

### *A. Structure of the Tool*

The tool is developed as a Matlab script program, which generally tends to be platform independent, executed by Matlab Common Runtime (MCR). However, due to the fact that some parts of its functionality is based on MS Windows specific program technologies (COM/OLE), the tool occurs to be platform dependent.

[1]Slavy G. Mihov is with the Faculty of Electronics, Technical University – Sofia, blv. Kliment Ohridski 8, 1000 Sofia, Bulgaria, e-mail: smihov@tu-sofia.bg

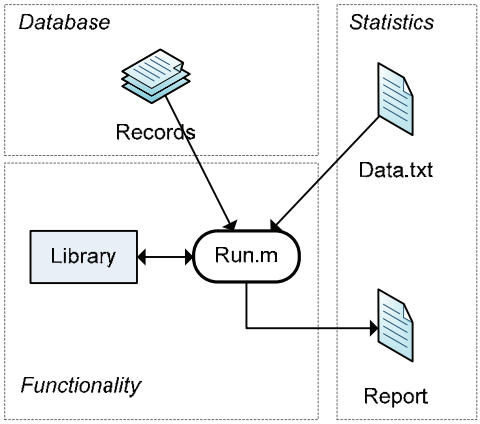


Fig. 1. Tool Structure

The program for batch processing (Run.m) consists of separate functional modules formed in a couple of script files (*.m). Figure 1 shows the general structure of the development tool and the connections between its five major parts:

**Run.m** is the core module of the tool for batch signal processing, formed as a sequential procedure consisting several processing steps. Its structure can be seen on Fig. 2.

**Library** is a set of library functions (read/write text/data files, functions for signal pre-processing, etc.), which functionality is used by the main program.

**Records** denotes the set of input data files (*.tmp) pending for processing. In [2, 3] these are multichannel (4, 8, 12) ECG recordings and in [1, 4] these are human speech recordings (clean and contaminated with noise in several SNRs).

**Data.txt** is a text file which contains specifications for the batch processing. It is used by the main program (Run.m) to define parameters such as: list of input data file names, paths, optimization criteria, constraints, optimization intervals, etc.

**Report** is the final report in the form of a MS Word document (*.doc) and a Portable Document Format (*.pdf). This report is automatically generated after all batch processing is completed and contains results in graphical, text and table form for convenience of the human reader.

Table 1 summarizes all files involved in the batch processing and automatic report generation. It lists the files in the work directory by extension, either used as an input or created during the execution of the optimization task.

TABLE I
FILES USED IN BATCH PROCESSING

| File | Purpose | Description |
|---|---|---|
| *.tmp | input data | Miscellaneous data |
| *.log | execution log | Text log |
| *.eps / *.emf | output diagram | Graphical results |
| *.doc | output report | MS Word document |
| *.pdf | output | Portable Document |

After the parameters for signal processing and optimization have been set in the text file Data.txt, the batch computations can be started by running subroutine Run.m. The last one organizes all calculations in batch mode, thus eliminates the necessity of human interaction and control until finalizing all operations which can be quite time consuming. As final result,

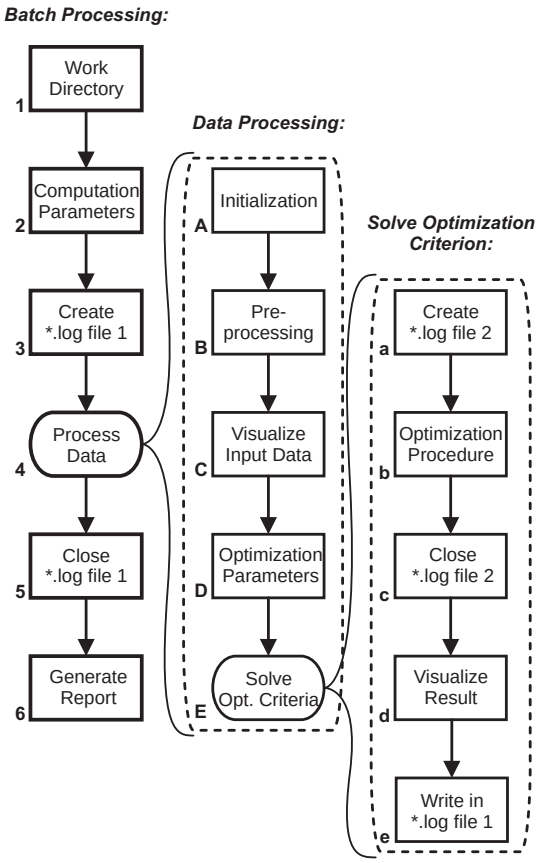


Fig. 2. Batch Execution Flow

in the working directory are created numerous files holding text and graphical information for the input data (signals) and the obtained results (Report).

### *B. Tools Specifics*

The tool for batch processing and automatic report generation (Run.m) does not need any Graphical User Interface (GUI) to operate, provided that all necessary files (Records, Data.txt) are present. It can be run in a console, deployed to a remote machine to use its computation resources and easily be scheduled for execution in time. However, setting the parameters of the batch processing task (Data.txt) can be more convenient if this is done automatically in a GUI, which is already being developed for the optimization task in [3].

The essential work in organizing the batch processing is done by the subroutine Run.m, which main stages are shown on Fig. 2 as a nested sequence of operations. The process flow is relatively straightforward consisting the following steps:

**1. Create working directory** (.\Temp\) or clear already existing one to prepare it for storing the temporary work files and processing results. In case an old directory already exists, this suggests that batch processing has been done before. Its contents is being archived (*.rar) with the current date/time in order to preserve these previous results for potential use later.

```
%% Create Temp directory
if( ~exist('.\Temp', 'dir') )
    mkdir('.\Temp');
else
```

```
    disp('Backup existing results...');
    command = sprintf('rar a "Temp (%s).rar" Temp',
datestr(now, 'dd mmmm yyyy, HH.MM.SS'));
    [~, result] = system(command);
    disp(result);
    delete('.\Temp\*.eps');
    delete('.\Temp\*.emf');
    delete('.\Temp\*.log');
end
```

**2. Load parameters for batch computations**, from external text file Data.txt – file names, intervals for optimization, criteria for optimization, procedures for pre-processing (like subroutines for removing zero line shift, tremor artifacts and powerline interference in ECG) and so on. The text file Data.txt is being parsed and all input parameters are placed in the vector variable *param*.

```
%% Read record file names and intervals
fileName = '.\Data.txt';
fid = fopen(fileName, 'r');
textscan(fid, '%s %s %s %s %s', 1); % bypass line 1
param = textscan(fid, '%s %d %d %d %d');
fclose(fid);
```

**3. Create master *.log file** to store in text form the results of the performed computations; also starts a global timer to track execution time.

**4. Successive iterative processing** of every single input data file from the list in Data.txt.

**5. Close master *.log file.**

**6. Generate automatic Report** for all signal processing (Report.doc, Report.pdf) – document containing results and statistics in text, table and graphical for the performed algorithm optimization, obtained set of optimal parameters, input signals and output data.

The iterative processing of the succession of input recordings (step 4) consists of the following major stages:

**A. Initialization** of constants, loading input signal data from Records (*.tmp) and parameters of the optimization task.

```
%% Read data file
    fileName = strcat(char(param{1}(index)));
    [~, name] = fileparts(fileName);
    fid = fopen( fileName, 'r' );
    data = fread( fid, 'int16' );
    fclose(fid);

    %% Set typ and atyp intervals
    typ1 = param{2}(index);
    typ2 = param{3}(index);
    atyp1 = param{4}(index);
    atyp2 = param{5}(index);

    %% Initialize constants
    t = 0: 1/Q: N/Q; % Scale in sampling period, [s]

    %% Signal channels
    channels = reshape(data, N, numCh);
```

**B. Pre-processing** input data before starting the main optimization task. For ECG records [3, 5] these can be any sophisticated algorithms for eliminating zero line drift, tremor artifacts, powerline interference, etc. formed as program procedures in a Library for convenient use. Similarly, when processing human speech records, these can be any adequate algorithms for filtering, denoising, resampling and so on.

**C. Visualizing input data** and storing the diagrams of the signals in vector graphic format (*.eps, *.emf).

```
%% Plot signal leads
    f1 = figure('visible','off');
    plot(   t(axe1), S1(axe1), 'k', ...
            t(axe1), S2(axe1), 'g', ...
            t(axe1), S3(axe1), 'r' );

    print( f4, '-depsc2', '-tiff', '-r2400',
sprintf('.\\Temp\\Record-%sd.eps', name) );

    print( f4, '-dmeta', '-r2400',
sprintf('.\\Temp\\Record-%sd.emf', name) );
```

**D. Define optimization parameters** for the input signals and prepare constrained task for minimization – set intervals, criteria and constraints.

**E. Perform optimization iteratively** for every single optimization criterion of the processing algorithm, create logs for execution and store plots in vector graphic formats (*.eps, *.emf) in the working directory.

```
%% Optimization
    switch criterion
        case 1
          obj = @(x)Objective1(x, LTyp, LAtyp);
        case 2
          obj = @(x)Objective2(x, LTyp, LAtyp);
        case 3
          obj = @(x)Objective3(x, LTyp, LAtyp);
        otherwise
          disp('Unexpected criterion!');
    end

    options = optimset('Display', 'iter',
'Algorithm', 'active-set', 'MaxFunEvals', 1600);
    [x, fval] = fmincon(obj, [0 0 0 0 0 0 0 0], [],
[], [], [], [], [], @Constraint1, options);
```

## III. Subroutine for Generating Report

Probably the most interesting part of the tool for batch signal processing and storing results for the performed optimizations is the subroutine for automatic report generation. It is formed as a separate subroutine, which can be invoked independently and it produces the report based on whatever input text and graphic files are present in the directory given. It consists of 4 sequential steps and below is shown its listing code.

The first step is to **Create a MS Word document** and add some text, graphics and formatting using VBA (Visual Basic for Applications) script commands invoked directly from Matlab environment [6]. Matlab's support for VBA is utilized by the command *actxserver()*, which creates a local OLE Automation server, and returns a handle to the default interface of a COM server with the programmatic identifier of MS Word.

The next steps are trivial and again use VBA commands to **Save**, **Print** (in *.pdf format) and **Close** the so created report document.

```
%% Generate report from *.eps/emf results.
function GenerateReport(directory, extension)

  filter = ['*' extension];

  % write to Word document
  Doc = actxserver('Word.Application');
  MS = invoke(Doc.Documents,'Add');

  set(Doc.Selection.Font, 'Name','Arial','Size',20);
  set(Doc.Selection.ParagraphFormat, 'Alignment',1);
  invoke(Doc.Selection, 'TypeText', 'Auto Report');
```

```
  invoke(Doc.Selection, 'TypeParagraph');

  files = dir(fullfile(directory, filter));
  for f = 1:length(files)
    fname = files(f).name;
    [~, name] = fileparts(fname);

    %Put figure in document
    invoke(Doc.Selection, 'TypeParagraph');
    graphicFile = [directory, sep, name, extension];
    invoke(Doc.Selection.InlineShapes, 'AddPicture',
graphicFile);
  end

  %Save word file
  word_name = 'Report (GUI).doc';
  full_name = [directory sep word_name];
  invoke(MS, 'SaveAs', full_name);

  % print report
  set(Doc, 'ActivePrinter', 'Adobe PDF');
  invoke(MS, 'PrintOut');

  % close document
  invoke(MS, 'Close');
  invoke(Doc, 'Quit');
  delete(Doc);

end
```

## IV. Practical Examples

Using the presented tool for batch signal processing are accomplished several experiments in former projects, involving evaluating an algorithm of interest with a large set of input data files, estimating its performance and optimizing parameters of computation. For example, in the development of the algorithm for optimal linear transformation of multiple ECG leads [2, 3] is done automatic processing of 250 12-channel ECG recordings from database CSE (Common Standards for quantitative Electrocardiography) [7], task large enough to be out of scope of manual processing (thus no parallel between automatic and manual execution, in means of accelerating computation time, can be made).

On Fig. 3 is shown a sample page from a report (consisting of over 50 pages) automatically generated after processing the set of ECG recordings.

## V. Additional Remarks

The substantial reduction of the time needed for manual processing, achieved by harnessing such specialized procedures as the one presented here, opens potentials for development and utilization of automatic tools for algorithms test, optimization and development. Such a technique is quite successfully adopted in the investigation the practical capabilities of wavelet transform analysis in speech signal denoising [1] and in the development of a novel algorithm for enhancing atypical heart beats in multichannel ECG signals [2, 3].

## VI. Conclusion

In this paper is presented a program solution (created in Matlab) designed to execute similar computations over a set of input data files, store results from every single processing (text, graphics, execution time, etc.) and produce a perspicuous report (*.doc, *.pdf) with the obtained results included. This tool performs batch signal processing and removes the necessity of human interaction until all processing is over and the report is automatically generated, which tends to be very helpful for long time intervals of calculations. The method has already been used in two different projects, particularly for solving optimization tasks.

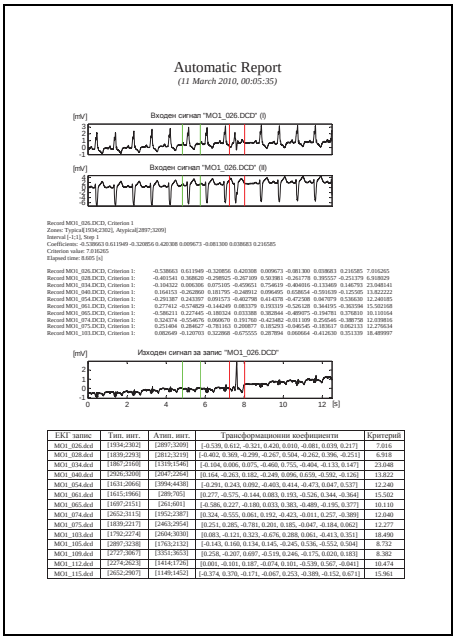


Fig. 3. Example Report Layout

## References


[1] S. Mihov, R. Ivanov, A. Popov (2009). *Denoising Speech Signals by Wavelet Transform*. The 18th International Scientific and Applied Science Conference ELECTRONICS ET-2009, B. 1, ISSN 1313-1842, Sozopol, Bulgaria, June 14-17, pp. 69-72.

[2] S. Mihov, Ch. Levkov, G. Mihov (2009). *Algorithm For Optimal Linear Transformation of 4 Holter Leads for Emphasizing Difference Between Typical and Atypical QRS Complexes*. XLIV International Conference ICEST-2009, B. 1, Veliko Tarnovo, Bulgaria, June 25-27, pp. 403-406, 2009.

[3] Ch. Levkov, S. Mihov, G. Mihov (2009). *Algorithm for Optimal Linear Transformation of 12 Standard Leads for Emphasizing Difference between Typical and Atypical QRS Complexes*. The 18th International Conference ELECTRONICS ET-2009, B. 1, ISSN 1313-1842, Sozopol, Bulgaria, June 14-17, pp. 20-23.

[4] S. Mihov, D. Doychev, R. Ivanov (2009). *Practical Investigation of Specific Types of Noise Signals for the Purpose of Suppression in Hearing-Aid Devices.* XLIV International Conference ICEST-2009, B. 1, Veliko Tarnovo, Bulgaria, June 25-27, pp. 399-402, 2009.

[5] Ch. Levkov, S. Mihov. *Multilead signal preprocessing by linear transformation to derive an ECG lead where the atypical beats are enhanced: Matlab implementation.* Proceedings of TU-Sofia, Vol. 58, Book 2, pp. 24-30, 2008.

[6] http://www.mathworks.com/matlabcentral/

[7] Journal of the American College of Cardiology, Volume 10, p.1313-1321, 1987