LTI INFORMATIQUE + GÉNIE
SOFTWARE + ENGINEERING

Report: PATS software 2.0

Authors:
Mathieu Olivier, ing. jr, Ph.D.
Éric Bisson, M.Sc.
Yann Rosan, ing. jr, M.Ing
Maxime Brousseau, B.Ing

Project Manager:
Louis Tanguay, B.Sc.A.

Recipient:
Cristina Tollefsen, Ph.D.
Defence R&D Canada - Atlantic
9, Grove St.
Dartmouth, Nova Scotia
Canada B2Y 3Z7

Contract:
W7707 – PATS
155782/A
March 2015

DRDC-RDDC-2015-C288

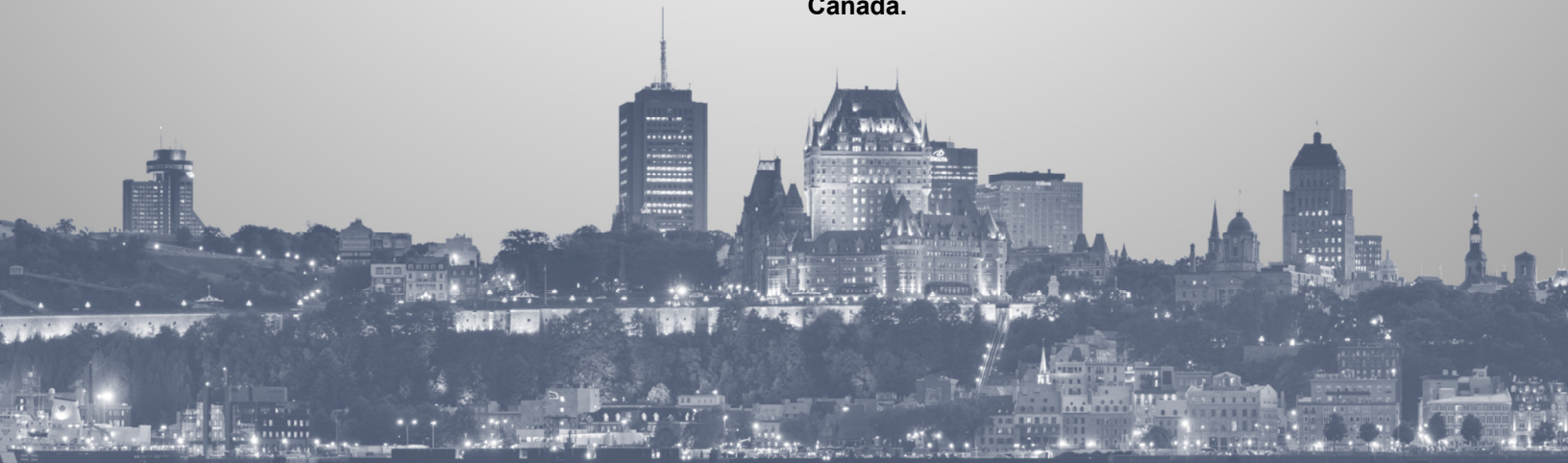


Table of contents

1 Mandate

The mandate of this project is to perform a major revision to the MATLAB-written PATS Software in order to get full benefits of MATLAB's inherent features. A major restructuration of the program is required. The specific tasks required by the Request For Proposal (RFP) are:

1. Restructure program variables:
    a. Replace hard-coded variable names with appropriate vectors, matrices, and structures in order to maximize the use of the indexing capabilities inherent in Matlab and reduce the number of variables passed on each function call.
    b. Change object data types to structure data types where appropriate.
    c. Remove "orphaned" variables that are no longer needed.
2. Improve program flow:
    a. Change all instances of hard-coded, operating system-dependent paths in order to make the code platform-independent.
    b. Re-use code in loops or functions, as appropriate, resulting in consistent data manipulation, using the capability provided by Task 1a.
    c. Move the figure generation code out of the main program and into functions.
    d. Improve automatic figure generation capability (details to be provided by Scientific Authority; may include, e.g., rearranging sub-figures, forcing uniform axis limits across figures, adjusting font sizes, correcting titles and axis labels).
3. Improve sonar performance metrics:
    a. Correct the existing implementation of the area coverage metric.
    b. Add calculation of other sonar performance metrics as determined by the Scientific Authority.

The scope of this document is to report the modifications made to the code and to propose further actions that would benefit the PATS software.

2 Approach

Firstly, the PATS software was split in four parts to allow many people to work on different files at the same time. The Subversion (SVN) client TortoiseSVN is used to track code modifications and to ensure consistent and rigorous collaborative work. Modifications to the program are validated after each significant change to ensure the program computes the same results. A verification routine that compares the results of the program against pre-defined reference results as well as with reference results obtained from an external source (for the AMOC scenario) has been integrated to *PATSBench.m*, a test bench script that launches and monitors PATS execution. The MATLAB Profiler and Comparison Tool were used extensively through the whole process to monitor the software improvements. MATLAB 2014a 64-bit was used on several computers with Microsoft Windows 7 and 8.1 operating systems.

In addition to performing the work required in the RFP, optimization and cleaning is performed throughout the code when needed in order to improve performance and readability.

3 Code modifications

The listed changes to the software were subdivided into two categories according to their impact level on the software.

3.1 Program variables restructuration

3.1.1 Program subdivision

PATS has been subdivided in four parts run consecutively from the *PATS.m* main script:

- *PATS_Configure.m*
- *PATS_AcousticData.m*
- *PATS_PostProcessing.m*
- *PATS_Display.m* .

These scripts are located in the folder named *Matlab/MainProgramScripts*. This subdivision makes the program easier to read and better organized. By doing so, it makes the main program better organized.

3.1.2 Global variables removal

Global variables (*sys*, *rdep*, *SigmaSE_BM* and *Tx_Rx*) have been removed except the *PATS_CONFIG* global structure that was created to store general information about PATS version, the studied case, and other program specific folder paths. The other variables were completely removed or replaced by structure fields. *SigmaSE* (formerly *SigmaSE_BM*) is now a field of the *Detector* structure and can be defined by the user in the file *ConfigDET.DAT*. *Tx_Rx* has been added to the *Scenario* structure. The *sys* variable already existed as the *SystemControl* field of the *Environment* structure and it has simply been replaced by the latter. For consistency, following these changes, functions with redundant input arguments were removed.

3.1.3 Classes conversion to structures

The few classes (object-oriented programming features) have been removed and replaced by structures. Most classes contained no method except for the constructor. Therefore, the transition to structures is straightforward. Functions have been created to instantiate the structures with the possibility to set field values from the functions input arguments. These functions replace the previous class constructors and they are located in the *Matlab/Structure_Definition_Functions* folder. Methods (other than the constructor) found in some classes have been transferred in the program scripts or in dedicated functions. For example, the code corresponding to methods *get.Nx* and *get.Ny* from *classGridXY* is now located in *PATS_Configure.m*. Unused methods have simply been discarded.

3.1.4 Input files reader and configuration files

Previously, the configuration file (*Config.dat*) was opened, and read line by line in order to fill out the variables instantiated from the classes. This was a potential source of error as writing the parameters in the wrong order would cause the code to produce wrong results or simply to crash. The configuration file was kept open until the end of the configuration section of PATS. In the new implementation, the configuration file is opened, read, and copied into a variable named *Configuration*. The file is then closed before doing anything else. The data are read regardless of the location of the information in the original file. The parameters are then read through a new function (*ReadSectionData.m*) that scans an entire section of *Configuration* (passed as parameter) and fills out the specified structure with the appropriate data. This is done for every section. This new procedure is less error-prone and reduces the number of disk accesses.

Other functions were implemented to read the several configuration *.dat* files (*ReadConfigAcousticModel.m, ReadConfigDetector.m, ReadConfigEnv.m, ReadConfigScenario.m,* and *ReadConfigTargetScattering.m*). The *ReadConfigEnv.m* function needs an associated sub-function (*ReadSeaData.m*). *FindAttribute.m* is a sub-function, used by configuration file reading functions, which needs as parameters the array that contains the data to scan, the section in which to search, the attribute to find, and the configuration file name (used for display in case of error). It returns the value of the parameter, as a string or a numerical value (integer, float or matrix) and its position (line) in the configuration file.

Some parameters were added to configuration files to give the user more control. By version 2.0, the *'Tracker M'*, *'Tracker N'* and *'Grid Spacing km'* parameters must be provided as user inputs.

3.1.5 Variable renaming and structure regrouping

The probability of detection variables were put in a structure named *DetectionProbability* with *data* and *avg* as field. The full data set is now returned to the main workspace for its accessibility via the *ComputeMonteCarloCPD.m* function. The other important changes in variable names are presented in Table 1 below.

Table 1: Important changes in variable names

Old variable name	New variable name
Aco	AcousticProducts
DetBarrier	Barrier
Det	Detector
Diagnostic	DiagnosticAids
Env	Environment
Mod	AcousticModel
PMet	PerformanceMetrics
P1Rx	RxPlatform(1)

P2Rx	RxPlatform(2)
P1Tx	TxPlaform(1)
P2Tx	TxPlaform(1)
PTgt	TgtPlatform
Scene	Scenario
Sys1	SonarSystem(1)
Sys2	SonarSystem(2)
Tgt	TargetScattering

3.2 Program flow improvement

3.2.1 *PATSBench* script

A script called *PATSBench.m* is introduced to run PATS, to monitor the execution time, to open a profiler, and to compare the code results with reference results. This is useful to track mistakes and performance improvements as modifications are made.

3.2.2 System dependency and interaction with the file system

The operating system dependency has been eliminated by replacing hard-coded file separators in path strings with the function *filesep* and by the use of functions that are not platform-specific.

Almost all *cd* instances have been removed in the code. Folders containing functions (Matlab folder and subfolders) have been added to the MATLAB path instead. From now on, the code "remains" in the directory from which it is called. The remaining *cd* instances are those related to the execution of the acoustic software (a .bat file is called) so it appeared to be safer to leave them there.

All relative paths such as *"…/Simulation/…"* have been replaced by *"[PATS_CONFIG.caseRoot filesep 'Simulation' …]"*. The code can thus be run from any directory without problem.

3.2.3 Figure generation improvements

All figure generation code in *PATS_PostProcessing.m* has been isolated in the separate script file *PlotSystemConfiguration.m*. This code produces a single figure called '*System Configurations*'. This script should eventually be integrated to *PATS_Display.m* for consistency.

Major modifications were made to *PATS_Display.m* and its related functions *PlotTgtRxRanges.m*, *PlotTgtRxAngles.m*, *PlotInterpolatedSE.m*, *PlotCGRLvsRange.m*, *PlotCGTLvsRange.m*, *PlotCPDAlongTrack.m*, *PlotAveragedCPD.m*, *PlotBarrierPBP.m*, and *PlotCoverageDiagram.m*. These modifications include the removal of multiple conditional control statements ('switch' and 'if') and more uniformity in the figure generation process. A cell array of figure handles (*hFig*) has been used to keep the track of all generated figures. Also, the

screenSize variable was added to store the primary display size and it is used to center the generated figures on the user screen.

3.2.4 Algorithmic related modifications

Loops (*for* loops) have been implemented using the global indexing of system related variables and structures. This allowed duplicated code to be removed in *PATS_AcousticData.m*, *PATS_PostProcessing.m,* and *PATS_Display.m*.

The function ComputeCoverageGridData.m has been optimized. Instances of the *scatteredInterpolant* function have been replaced by *interp2* by introducing proper grid mapping. This modification results in much faster grid generation while preserving the quality of the numerical interpolations.

The hard-coded Boolean switching variables used for conditional control found inside the Monte Carlo functions were moved to *PATS_Configure.m* as user settings which are then passed as input arguments to these functions. This is a temporary convenience and this feature should be removed once the right algorithm chosen.

3.3 PATS operation mode

Configuration files (*config.dat*, *configCW.dat, configFM.dat, etc.*) provide inputs for the PATS software. By some of these inputs, the user can decide if PATS must use data generated from external acoustic simulations, run the acoustic software, or use already computed coverage grids. Obviously, it is not optimal to perform all these operations every time the program runs. For example, it is not necessary to read acoustic data if one chooses to use already computed coverage grids. The parameters that control the operation mode are found in the PATS *AcousticProducts* and *PerformanceMetrics* structures. The specific operation modes are described below.

Mode A

AcousticProducts.ComputeNewData(iSystem) = 1
AcousticProducts.ComputeUsingExtData(iSystem) = 0
PerformanceMetrics.ComputeMCGrid = 1

In this mode, the acoustic software is run. Therefore, PATS won't use external data and it will compute new coverage grids.

Mode B

AcousticProducts.ComputeNewData(iSystem) = 0
AcousticProducts.ComputeUsingExtData(iSystem) = 1
PerformanceMetrics.ComputeMCGrid = 1

In this mode, PATS reads already computed acoustic simulation results, saves them to .mat format, and computes the corresponding coverage grids. The acoustic software should not be called.

Mode C

AcousticProducts.ComputeNewData(iSystem) = 0
AcousticProducts.ComputeUsingExtData(iSystem) = 0
PerformanceMetrics.ComputeMCGrid = 1

In this mode, PATS reads already computed acoustic simulation results that had been already converted to the .mat format and computes the corresponding coverage grids. The acoustic software should not be called.

Mode D

AcousticProducts.ComputeNewData(iSystem) = 0
AcousticProducts.ComputeUsingExtData(iSystem) = 0
PerformanceMetrics.ComputeMCGrid = 0

In this mode, PATS uses already computed coverage grids.

In the occurrence that an invalid combination of parameters is passed to the code, an error message will be thrown showing a table containing the appropriate combinations.

The logic of the code has been reviewed and corrected so that PATS no longer reads and converts raw acoustic data when not needed. Mode A has not been tested as the authors of this report did not have access to the acoustic software. Modes B and C work as expected. Mode D works, but the code still need external data (raw or not) to be read to run smoothly. To avoid this, appropriate variables should be saved along with coverage grid in files *MCGrid.mat*. This will require further investigation.

Binary files are now written to and read from the *Results* folder. To avoid files from System 1 being overwritten by those of System 2, system-specific sub-folders (*System1* and *System2*) have been introduced. Moreover, the file format has been changed from .bin to .mat.

3.4 Other modifications

Some variables were renamed in order to improve readability and consistency. For example, the performance metrics variables have been renamed. The renaming has been performed on an "as encountered" basis. Some renaming remains to be done in sub-functions.

Tl_Grid_CG and *rvb_Grid_CG* variables are now saved along with the *MCGrid* variable in *MCGrid.mat* in their respective system folders.

Modifications were made to *SetSystemFile.m* function to remove unused parameters and to make it compatible with other functions (e.g. string instead of cell in some cases).

MoveSimulationFiles.m is used to copy or move simulation files in *'.. /System1/out_dat'* and *'.. /System2/out_dat'* folders.

A header (comments) was added to some functions for documentation purposes. This documentation can be displayed by invoking the *help* function (e.g. *help PATSBench*).

On-screen display has been reduced in order to improve execution speed.

The function *ReadTrackFile.m* has been fixed and vectorized. The heading computation has been corrected.

Tracks have been converted from arrays to structures with informative field names.

The code reading text files that was using 'for' loop or 'while' statements was replaced with a single call to the *textscan* function.

Instances of *fprintf* within loops have been vectorized.

The initial configuration file selection dialog box was restored with the same functionality.

Conversion from string to cell removed. Other functions have been modified if necessary (some *cellmat* deleted, as well as some *char*).

The display function *disp* was replaceb by *fprintf* to improve performance.

GHB algorithm in *MonteCarloCPDC.m* function was removed. PATS no longer executes both CDST and GHB algorithms, which results in a faster execution time.

An effort has been made to remove syntaxes useless in the MATLAB environment.

Unused or replaced functions were put in a folder named *'Unused functions'* located in the *Matlab* functions folder.

Hard-coded switching variables inside Monte Carlo functions are now user-defined and passed as input arguments. This is a temporary convenience and this feature should be removed in the future.

The *patsSave* function that replaces *save* is introduced. This allows the save options to be set globally by modifying the code at a single place. In the current code, the "-v6" option is used which makes the code faster while producing larger files.

4 Remarks on PATS execution

4.1 Running PATS

The PATS code can be run by invoking the PATS.m script from MATLAB. This script adds the PATS root folder as well as the "<PATS root folder>/Matlab" folder and subfolders to the MATLAB path. As such, all MATLAB files (scripts and functions in this case) within these folders are accessible to the program without the need to change directory.

4.2 Architecture-related notes

On some specific architecture, it was observed that the program runs much faster when MATLAB is executed on a single core. Indeed, the current program does not rely on parallel computing so specifying to the operating system which core to use seems to increase the computation speed in some cases. To specify on which core the MATLAB is executed on MS Windows, open a task manager and right-click on MATALB.exe in the "Processes tab" ("details" tab in Windows 8.1). Then, select the menu "Set affinity" and select a single core from the proposed list.

4.3 Scenario selection

The scenario can be specified directly in PATS.m by uncommenting the desired scenario or by selecting the desired Config.dat file through the file selector. The file selector pops up only when the scenario is not declared in PATS.m.

4.4 Changes to configuration files

The PATS software restructuring made in version 2.0 requires some changes to the input configuration files. These changes have to be made manually by the user as follows:

- The entry *'Grid Spacing km = δ'* should be added to the *ENVIRONMENT* section of *Config.dat*, where δ is the grid spacing numerical value in km (for example, *'Grid Spacing km = 0.5'*).

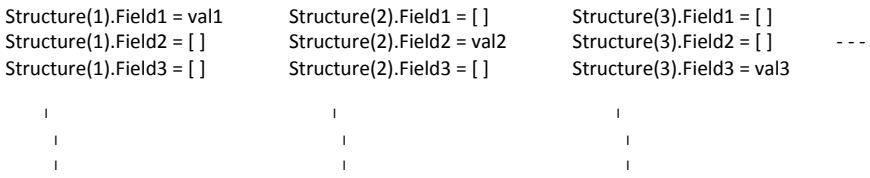
- The following entries should be added at the end of *ConfigDET.DAT*:
  - SE Variance = σ
  - Tracker M = μ
  - Tracker N = ν

  where σ, μ and ν are numerical values.

5 Proposed future work

5.1 General PATS reviewing

Verify PATS does everything properly. For instance, investigate structure arrays with the encountered ill-constructed form shown below.



Also, review the structuring of data. As an example, the *MCGrid* variable is a cell containing a 2-dimensional array of structures, structures for which fields are all empty for the majority of the structure elements constituting the array.

5.2 Code cleaning

The renaming of variables, as well as the removal of useless syntaxes should be completed for consistency over the whole program. Further work on logical aspects and readability also has to be carried out since only some of the functions were thoroughly revised.

5.3 Error handling

Error handling should be a major concern of any serious software. In the actual program state, a crash is likely to leave files opened. Moreover, improvement of error management by adding additional error messages would be beneficial for the user.

5.4 Other tasks and recommendations

All reverberation components still need to be written in bin/mat files and returned as output arguments by *Read_StoreCGModelData.m*.

Move the figure generated in *PATS_PostProcessing.m* into *PATS_Display.m*.

Keeping track of objects such as figures and file identifiers by their systematic assignment is recommended. A cell array of handles has been used to keep track of all generated figures. For consistency, the same assignment rule should be used for any new figure added. Concerning the file identifiers in PATS, the command *'fclose all'* is used at the end of the program execution to ensure that all opened files have been closed. While this actually closes all opened files, it is a better practice to systematically close each opened file when they are no longer accessed by the

program. This aspect remains to be verified in the program as some file identifiers may not be closed properly.