**Marquis 2018 – “read-me” for MATLAB code**

The files contained in this folder are used for computations in the article [Marquis, Andrew D., et al. "Practical identifiability and uncertainty quantification of a pulsatile cardiovascular model." Mathematical biosciences 304 (2018): 9-24.]. All computations are performed in MATLAB. Note that much of this code is dependent upon files in other sub-directories. We recommend you download the entire directory “Marquis2018\_MATLAB\_code” to prevent file dependency issues. Files that are in red text are the highest level scripts one should run through the console. Files that listed under “Uses” are functions needed for a given script to run properly and not throw an error.

**Data**

* **[DATAfig.m]** – this function calls “data\_process1.m, data\_process2.m, data\_process3.m” and plots the data nicely (Figure 1 in the actual manuscript)

**Uses**

* + **[DataBW1.mat, DataBW2.mat, DataBW3.mat]** – data files containing the raw hemodynamic waveforms for each individual rat (LV pressure and volume + max’s and min’s from each cardiac cycle). These files are loaded in “data\_process1.m, data\_process2.m, data\_process3.m” respectively. Note that these waveforms contain blood withdrawals that are not used in our study.
  + **[data\_process1.m, data\_process2.m, data\_process3.m]** – functions that load their corresponding “.mat” file. The input variable “tshift” dictates how much the data is shifted in time (tshift =0 means the data will not be shifted, tshift = 0.11 means the data will be shifted 0.11 seconds to the left, tshift = -0.02 means the data will be shifted 0.02 seconds to the right). This function extracts only a baseline section of data from the longer waveform in the “.mat” file and returns the baseline segment of data in the struct variable “data”

**Base Cardiovascular Code**

* **[runfile.m]** – this script is a generic runfile that solves the model - it is currently set up to solve the model using optimized parameter values. Results from running this file are stored in the data file “RUNFILE\_RECENT.mat”
* **[runfilePLOT.m]** – This script plots the results from “runfile.m” using the data file “RUNFILE\_RECENT.mat.” Plots include relative data shifting, model predicted pressures in non-LV compartments, Pressure-Volume loop, and UQ results, model simulated “Wiggers Diagram,” and a plot containing both frequentist and Bayesian UQ results. Note the Bayesian uncertainty intervals are computed in the “DRAM” subdirectory, and the Frequentist UQ results require a sensitivity matrix computed using optimized parameter values (computed in the “SensitivityAnalysis” subdirectory). The computation of the frequentist UQ intervals is done in directly in this script.

**Uses**

* + **[load\_global.m]** – This function takes the struct variable “data” as an input and uses that information to compute nominal parameter values, initial conditions, and optimization bounds. Log-scaled parameters are returned as “x0,” initial conditions are “Init,” and lower and upper parameter optimization bounds are “low” and “hi” respectively. Additionally, this function declares the global variables “DIFF\_INC” and “ODE\_TOL” which are the local sensitivity step size and ODE solver tolerance respectively. Valve resistances “Rav” and “Rmv” are also declared as global variables
  + **[modelBasic.m]** – This function defines the RHS of our model and is solved by MATLAB’s ode15s routine. Note that this RHS is defined to be solved over 1 cardiac cycle; when this function is called it is contained within a larger loop so that initial conditions are reset for heart beat (these pulsatile compartment models are prone to being stiff and solutions can often become numerically unstable when solved over large simulation times). The function takes inputs of the time domain to solve the model “t”, the state variables “y” (volumes of each compartment), parameters “par”, a reference time for beginning of systole for a given cardiac cycle “ts”, and the length of the cardiac cycle “T”. The function returns the RHS of the model – i.e. the time derivative of volume in each compartment “xdot”.
  + **[ElastanceBasic.m]** – This function defines the LV piecewise elastance function that drives the pulsatile behavior of our model. Note that this function is set up to return a scalar quantity of elastance for a given time point “t” as it is called within “modelBasic.m”. It takes inputs of the time “t”, length of the cardiac cycle “T”, and model parameters “Tsf”, “Trf”, “Emin”, and “EMax”.

**Sensitivity Analysis (Sub Directory “SensitivityAnalysis”)**

This sub-directory contains files to compute sensitivity matrices, and construct parameter subsets by structured correlation analysis.

* **[DriverBasic\_sens.m]** – this script calculates local sensitivities by calling “senseq.m” and saves the results into a .mat file
* **[plotSensResults.m] -** this script loads the results from “DriverBasic\_sens.m” and plots the ranked sensitivities, and time-varying sensitivites

**Uses**

* + **[senseq.m]** – this function calculates local sensitivities using a forward difference approximation. It takes the parameter vector “par” and data struct “data” as inputs and returns the sensitivity matrix “sens” and model residuals as outputs. Note that this code was originally written by C.T. Kelley. We have altered C.T. Kelley’s original code to call our objective cost function “CVmodel.” For more information and most recent versions of C.T. Kelley’s code(s) see the following link: <http://www.siam.org/books/kelley/fr18/matlabcode.php>
  + **[CVmodel.m]** (in parent directory) – This is our objective cost function that takes the “data” struct and “par” parameter vector as an input. The function solves the model and returns the model residuals “rout”, the least squares cost “J”, the sum of squares “SS”, the model variance “S2”, the number of data points used in the residual “N”, and the portion of the solutions to LV pressure and volume used to calculate “rout”.

**Subset selection (Sub Directory “SensitivityAnalysis”)**

* **[struc\_corr.m]** – This script loads “.mat” files that contain a sensitivity matrix. The variable “INDMAP” is a vector of integers representing the order of ranked parameter sensitivities (this is determined in DriverBasic\_sens.m and copied and pasted into this file). Running the script will display parameters that exhibit pairwise correlation above a user defined threshold in the console window. The user can then remove parameters from the subset by deleting the corresponding integer in “INDMAP.” We recommend copying and pasting “INDMAP” to the line below and providing a comment describing what parameter was removed and why.

**Least Squares Optimization (Sub Directory “Optimization”)**

This sub-directory contains files used to optimize parameters using the Levenberg-Marquardt optimizer

* **[OPT.m]** – is a script that optimizes many distinct parameter subsets. The cell-array variable “INDICES” contains vectors of parameter indices denoting different subsets. We found the most success using subsets were constructed by using “struc\_corr.m.” The purpose of this script is to screen a list of putative subsets in “INDICES.” We considered a subset to be “good” if optimized parameters did not hit their optimization bounds (crude way to assess identifiability), the qualitative model fit to data looked good (checked by using runfile.m), and the model solutions in compartments that did not have data make physiological sense.

**Uses**

* + **[newlsq\_v2.m]** – this function is our Levenberg-Marquardt optimizer (written by C.T. Kelley). See C.T. Kelley’s website [https://ctk.math.ncsu.edu/matlab\_darts.html#iffco] for detailed documentation on this optimizer (he also likely has a more recent iteration of this code available). We have altered C.T. Kelley’s original code to take parameter bounds. For more information and most recent versions of C.T. Kelley’s code(s) see the following link: <http://www.siam.org/books/kelley/fr18/matlabcode.php>
  + **[opt\_wrap.m]** – this function is the objective cost called by “newlsq\_v2.m” (written by C.T. Kelley). Note that we have altered this function to call “CVmodel\_wrap.m”
  + **[CVmodel\_wrap.m]** – is a ‘wrapper’ that merges parameters we want to optimize with the parameters that are fixed at their nominal value. It calls “CVmodel.m” to solve the model and returns the model residuals “rout” and least square cost “J”
  + **[opt6.mat, opt14.mat]** – These .mat files are two of the optimization results for “good” subsets we tested. “opt6.mat” is the identifiable subset we report in the manuscript, and “opt14.mat” is an alternative identifiable subset we could have used (mentioned in the discussion section).
* **[shift\_opt.m, shift\_op2.m, shift\_opt3.m]** – these scripts are used to find the optimal shift of the data for their corresponding rat. This ultimately serves to circumvent estimating model initial conditions. The resultant .mat files made by running these scripts are stored in the sub-directory “shift.” These scripts have the same file dependences as “OPT.m”
* **[shift\_fig.m]** – this script load the .mat files in the directory “shift” and plots the least squares cost and gradient for the data shifts. (Figure 5 in the manuscript). This script has the same file dependences as “OPT.m”

**DRAM (Sub Directory “DRAM”)**

This sub-directory contains files used to optimize parameters with the DRAM MCMC estimator and construct Bayesian UQ intervals. The sub-directory “mcmcstat” contains source code written by Harrio (can be downloaded at <http://helios.fmi.fi/~lainema/dram/> - this link has a great tutorial on how to implement Harrio’s code) that is essential to run DRAM.

* **[dram\_runfile.m]** – this is the script that runs DRAM. The bulk of this code is setting up structs that Harrio’s code uses. “mcmcrun.m” is the function that constructs the parameter chains (stored in the subdirectory “mcmcstat”). We made an altered version of this function called “mcmcrun\_displayprogress” that will display to the console what iteration the DRAM simulation is currently on. Some of our DRAM simulations can take more than 24 hours, so it is convenient to see how far along the chains are while it is running.

**Uses**

* + **[SS\_wrapped.m]** – this function is a wrapper similar to CVmodel\_wrap.m, however the difference is that it returns the sum of squares “SS” from CVmodel. Haario’s code uses the “SS” to estimate the model variance parameter chains.
* **[dram\_results.m, badDRAM.m]** – both scripts use the output of “mcmcrun” to plot the DRAM diagnostics (chain plots, parameter densities, and pairwise densities). “dram\_results.m” is set up to plot the results from our identifiable subset of parameters. “badDRAM.m” is set up to plot the results form an unidentifiable subset of parameters that is shown in the Appendix of the manuscript. It calls the data file “DRAM\_unident.mat”
* **[UP\_press.m, UP\_vol.m]** – The scripts “UP\_press.m” and “UP\_vol.m” use the output from DRAM to compute Bayesian UQ intervals (by use of Haario’s function “mcmcpred”) for the “pressure” and “volume” data respectively. “UP\_press.m” and “UP\_vol.m” also plot the UQ results using “mcmcpredplot”. We made an altered version called “mcmcpredplot\_custom” so we can manually set the figure number MATLAB generates.

**Uses**

* + **[presspred.m, volpred.m]** - functions called by “mcmcpred” to evaluate the model solution when computing Bayesian UQ intervals.
* **[struct\_reorder.m]** – For some reason (most likely a transposition error) the struct mcmcpred returns is not organized properly for “mcmcpredplot,” even though it contains the correctly computed uncertainty bounds. This script takes the struct made by mcmcpred and reorganizes the information into a format “mcmcmpredplot” will accept. There is more than likely a slicker way to fix this, as I suspect there is matrix transposition error on my part somewhere in “UP\_press.m” and “UP\_vol.m.” I cannot for the life of me fix it this error, let alone find it.