November 2015

SoluCheck Documentation

Version 2.3

Rao, Alexander H
GEORGIA INSTITUTE OF TECHNOLOGY

Contents

Introduction

Overview

Thank you for getting SoluCheck for MATLAB. With it, testing and optimizing your code should become much simpler, intuitive, and even fun! This manual is designed in the following format:

Part 1 deals with the basic functionality of SoluCheck. Included in this section are how to download and install SoluCheck into MATLAB, how to uninstall SoluCheck from MATLAB, opening SoluCheck, a labeled SoluCheck UI, and an example run through. This section is meant for those not familiar with SoluCheck or MATLAB.

Part 2 is much more in depth than part 1. In part 2, we'll examine what kind of outputs will be shown to you, and what these outputs mean. We'll also examine how to define different types of arguments, how to use the test results, and some of the advanced options that can be used with SoluCheck. This section should be easy to read for those comfortable with MATLAB syntax and the SoluCheck program.

Part 3 is for experienced MATLAB programmers looking to get even more power out of SoluCheck. In part 3, we'll discuss testing against user-generated cases, advanced options, and advanced step options for arguments. This section will only be useful to those experienced with MATLAB programming and knowledgeable about most of SoluCheck.

Part 4 serves largely as an index for all users. In this section, you'll find our FAQ, troubleshooting tips, minimum technology requirements, Developer Information, Acknowledgements, and contact information.

Happy Testing!

The SoluCheck Team

Part I: Basic Functionality

Overview:

Part I covers basic functionality of SoluCheck as well as installation and seeking help. This is meant to be utilized by users who have relatively little experience with either MATLAB or SoluCheck, or those who are simply looking for an easy refresher. Specifically, Part I will cover:

- Installation: How to install and completely uninstall SoluCheck from your computer, and how to use the SoluCheck MSI for company-wide distribution.
- Opening SoluCheck for the first time, and setting initial preferences.
- A labeled SoluCheck UI
- A detailed example SoluCheck Test!

Installation:

The base SoluCheck installer, called SoluCheck.mlappinstall, is a packaged application, ready to be installed directly into MATLAB. For the average user, the only step necessary is to open the file and follow on-screen instructions. For companywide distributions, SoluCheck MSI offers the tools necessary to install SoluCheck on all MATLAB enabled and network connected computers. For OS reasons, the MSI is only available on Windows Operating Systems.

To uninstall SoluCheck, simply right click the SoluCheck icon in the MATLAB Applications Pane, then select "uninstall." Alternatively, open SoluCheck, go to SoluCheck > Settings > Uninstall, then click "Yes" on the resulting prompt. Please be advised that this will ***irreversibly uninstall SoluCheck. If you wish to simply restart SoluCheck, simply restart MATLAB.***

Installation Problems? If SoluCheck won't install, check to make sure that your machine meets the SoluCheck minimum requirements. Otherwise, ensure that MATLAB is properly installed, and try restarting your machine. If this still does not work, email SoluCheck@gmail.com for more assistance.

Opening SoluCheck for the First Time:

Great job! You've made it to the moment you've been waiting for – testing! Before we begin, however, we need to ensure a few checkpoints, which are:

1) Make sure that your machine and OS meet the Minimum Requirements for SoluCheck (which can be found in Section 4 of this manual).
2) Ensure that you are running MATLAB v 2014a or later for full access to SoluCheck features; running SoluCheck on earlier versions is not supported fully (See Section 4 for more information)
3) If you would like to, add SoluCheck to your favorites by clicking the dropdown button on the end of your Apps tab and clicking the outlined Star next to SoluCheck.

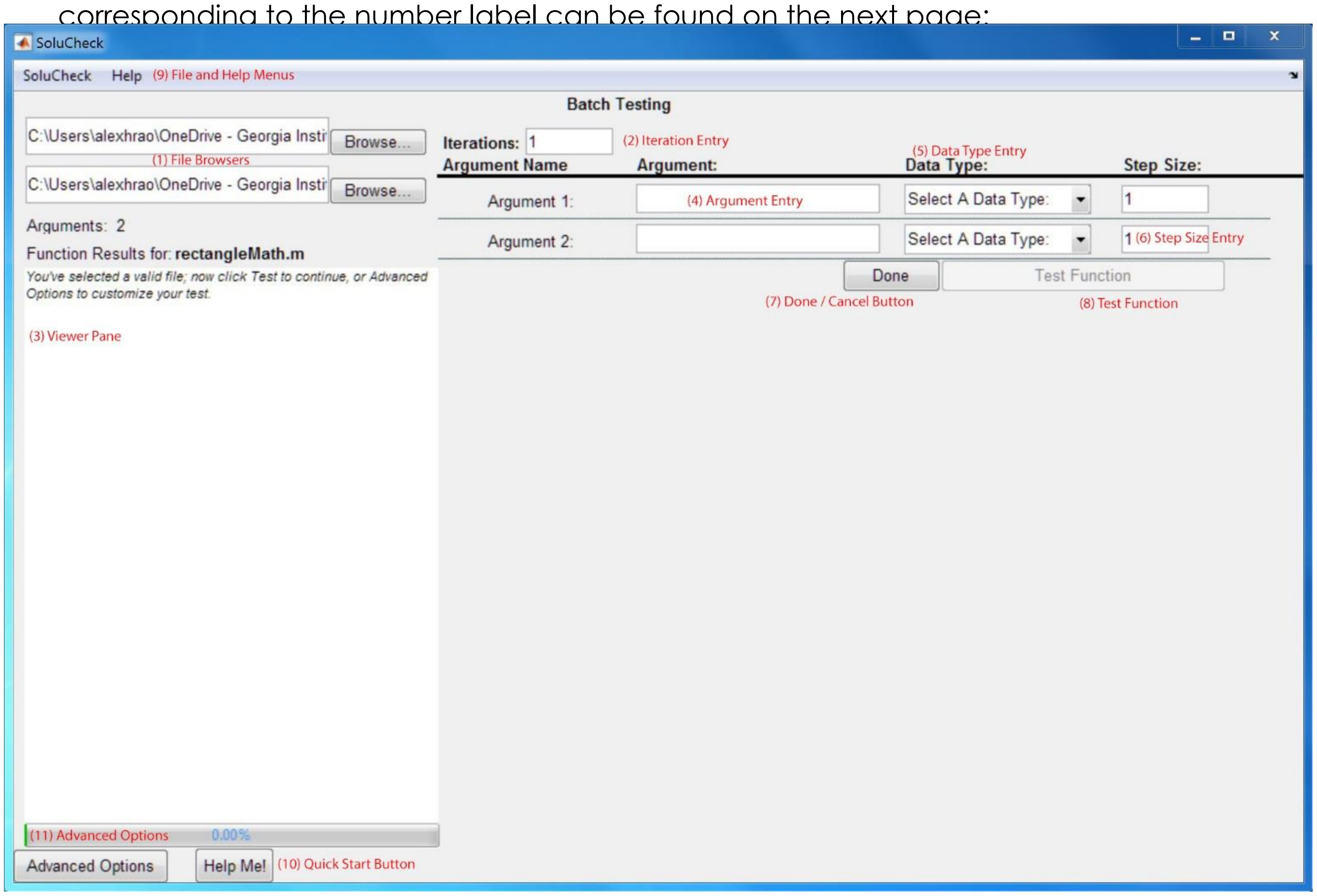
Congratulations! You have successfully set up SoluCheck, and we are now ready to open the App for the first time! First, click the button in your Apps Tab called "SoluCheck", and a window should open up; for the first run, this can take time as necessary files are found and processed. As such, ***please do NOT close MATLAB before you see the SoluCheck window!*** Failure to follow this can result in corrupted SoluCheck program files and will render the application useless.

Now that it has opened, you can set your preferences by navigating to the "SoluCheck" menu item, then going to "Settings". Here, you can mute SoluCheck or uninstall it later, if need be.

Now that we've opened SoluCheck, let's look at the UI:

SoluCheck: A Labeled User Interface

Below you will find a labeled SoluCheck UI (without Advanced Options). A Description corresponding to the number label can be found on the next page:



C:\Users\alexhrao\OneDrive - Georgia Insti

C:\Users\alexhrao\OneDrive - Georgia Insti

Iterations: 1

Arguments: 2

Function Results for: **rectangleMath.m**

You've selected a valid file; now click Test to continue, or Advanced Options to customize your test.

Argument 1: Select A Data Type: 1

Argument 2: Select A Data Type: 1

0.00%

1) File Browsers: These two fields display the path that your function and the solution function lie on, respectively. To set these fields, use the "Browse…" Buttons, and then follow the on-screen instructions.
2) Iteration Entry: This field is where you specify how many iterations you would like to test for; it can be any positive, whole number, but be advised that greater iterations will invariably take much longer!

3) Viewer Pane: This is how SoluCheck reports back to you. Here will be the results of your last test, as well as any errors, alerts, or notices that SoluCheck generated in the process. When in doubt, check here first. If your file has no description, then the viewer simply tells you what to do next; otherwise the viewer will show this description to the best of it's ability.
4) Argument Entry: Here is where you enter your argument, as a string. You should enter your argument EXACTLY as you'd like it to appear, with the exception of strings; for strings, do NOT enter the single quotation marks, unless you would like your string to contain these quotation marks!
5) Data Type Entry: This is where you select the data type of the argument you just entered. If you select Predefined Variable or Formulaic Entry, a new window will be opened up, and you will be asked to make your selection. Please note: A predefined variable is seen as constant and will NOT be stepped; ever.
6) Step Size Entry: This is where you will tell SoluCheck exactly how to step your entry; for example, a default value of 1 will mean that SoluCheck will increase the argument by one every time SoluCheck is run. Type N/A, 0, or leave this field blank if you would prefer for SoluCheck to NOT step this argument!
7) Done / Cancel Button: While SoluCheck is idle, this button will display as 'Done'; this is your indication that SoluCheck has nothing going on in the background, and that it is safe for you to close SoluCheck. If this displays as 'Cancel' instead, however, it means that SoluCheck is hard at work, and to click this button would cause SoluCheck to close without completing the task at hand.
8) Test Function: Use this to begin iteratively testing your code; this will only be available once all necessary data has been given to SoluCheck.
9) File and Help Menus: Use these menus to quit or mute SoluCheck, set settings and preferences, and get help about SoluCheck.
10) Quick Start Button: Use this button to get basic help starting SoluCheck, which includes FAQ.
11) Advanced Options: Use this to set advanced settings; 'Per Argument' settings apply to the current file; 'Per Run' settings apply to the current instance of SoluCheck, regardless of the file chosen.

SoluCheck: Our First Run:

Now that we know what the UI looks like, let's do a sample run through. For this example, we'll be using the file 'rectangleMath.m' as our code file, and 'rectangleMath_soln.p' as our solution file; both of these files are in the same folder.

1) First, click the Browse… Button for the code file.
2) In the resulting dialogue, navigate to rectangleMath.m and select it.
3) SoluCheck analyzes our code, and guesses that the solution file is in the same directory; in this case, SoluCheck is correct, but if it made a mistake, you can always correct it using the Solution File Browse button.
4) Now, after SoluCheck has finished loading, there should be empty text boxes ready to be filled with argument data. For our example, as shown above, we have two inputs for rectangleMath.m, and as such, two text boxes to fill.
5) First, we fill in the first argument text box. I will give SoluCheck a number 1 as the argument.
6) Then I tab over to the drop down list of data types, and select 'Number'.
7) Finally, I tab over once more and give a step size of 0.1
8) Now, I do the same process for the second row of arguments, but enter in different values.
9) Now that I am ready to test, I fill in the number of iterations for this run to use. I opt for 1,000 iterations this time.
10) Now, I click test function. My viewing pane changes to blue, and tells me to wait while SoluCheck is working.
11) Once it has finished, I will see the results in the viewing pane.

That's all there is to it! Later in this manual, we will go over how to use advanced options to augment SoluCheck's powerful testing facilities, making testing more precise and time-effective.

Part II: Using SoluCheck

Overview:

In part I, we learned about the most basic functionality of SoluCheck, and how to install and uninstall it. In Part II, we will delve much more deeply into the nature of SoluCheck. Specifically, Part II goes over these topics:

- What SoluCheck outputs to the command line.
- All of the options that come with the drop down data type menu.
- Some basic Advanced Options.

SoluCheck: Outputs

For each run you complete (regardless of the outcome), SoluCheck always reports back to two places: the viewing pane and the command line. We have already covered what the viewing pane will tell you, so now we shall look more closely at what the command line outputs of SoluCheck really mean.

There are a total of five (5) outputs that SoluCheck publishes for each run, and they are:

1) cAnswers: a cell array of your code's outputs.
2) cArguments: a cell array of the arguments last used.
3) cSolutions: a cell array of the solution code's outputs.
4) vecCodeTime: a vector that has all the run times for your code.
5) vecSolnTime: a vector that has all the run times for the solution code.

Using these outputs, hopefully your task of debugging will be made much easier. For example, say your function fails on the third iteration; then, you can call your function from the command line, using the 'cArguments' as your arguments, and then debug the function. This way, you know exactly where the *first* time your function failed, every time.

Additionally, you can use vecCodeTime and vecSolnTime to estimate the total amount of time taken to complete the number of iterations requested; furthermore, using extrapolation, you can guess at the amount of time an even larger number of iterations would take.

SoluCheck: Data Types

SoluCheck comes loaded with exactly seven (7) data types, and one (1) selector type, which are (in order):

1) Predefined Variable…
    a. If you select this option, a selection screen will appear, and you must click or navigate to your desired variable, and strike the return key to select the desired variable. If you wish to edit this later, you can simply click on the greyed out text box to open this window again. Please note that you MUST close this window before doing anything else!
2) String
    a. If you select this option, be advised that your single quotes are NOT needed, and indeed, if you put them in there, they will be counted as part of your string. Simply write down the string as you would like to see it displayed.
3) Number
    a. If you select this option, please note that this is NOT the same as making an array! *Your input MUST be a single number*.
4) Array
    a. If you select this option, just format your entry EXACTLY like it would appear on the command prompt, with brackets and all. This applies to any of the types of arrays, whether they are strings, logicals, or numbers.
5) Cell Array
    a. If you select this option, same as above, format your entry EXACTLY as it would appear on the command prompt, with curly braces and all.
6) Logical
    a. If this is selected, enter any type of logical entry, withere it is a true false or a logical array, entered as ether true false or 1 0.
7) Formulaic…
    a. If this is selected, a textbox will appear, and you will enter your code EXACTLY as it would be entered into a command prompt. You may enter ANY code here, but neither SoluCheck or MATLAB can guarantee that it

will work. You can assume that any currently-defined variables in the base workspace will be available to you, but you may NOT depend on any variables defined by your code. Additionally, to use the current iteration as a parameter, use the variable name ‘intIterationNumber’ in your code; additionally, formulaic entries will NOT be stepped.

You must select a data type for EACH argument to begin testing!

Advanced Options: Basic Functionality

Up until this point, we have been doing fairly simple procedures with SoluCheck; loading predefined argument types and testing iteratively and somewhat continuously. However, we are lacking two major ideals of SoluCheck: Analysis and Precision. First, we will go over SoluCheck's analysis features.

SoluCheck has rich analysis features that will help you better understand your code. To explore these options, first open Advanced Options by either clicking the button or using the drop down SoluCheck menu. Once there, click the Timing Button and the Profiling button.

The timing feature will produce a plot with two lines on two axes; the x axis is the iteration number, and the y axis is the time, in seconds. The blue collection of data points is your code's time to run, and the orange collection of data points is the solution code's time to run. This can be used for optimization purposes, or simply an efficient way to measure how much time passes during each iteration.

The profiling feature will produce 'heat maps' of your code. The profiler will open, and, if your function took sufficiently long, will display your function at the top. If you click on your function name, you will then see your own function's information, with a synopsis of your code and a 'heat-mapped' version of your code; the redder the highlighting on your code, the more time it took for the line to complete. Alternatively, you can look at number of times a line was called, or function for that matter. In the right hands, the profiler is an extremely valuable tool.

Now that we have covered basic analysis processes of SoluCheck, let's talk about SoluCheck precision. Say, for example, that you wanted SoluCheck to loop through a set range – for one of your arguments. While you could use the formulaic entry options, this doesn't always make sense. Instead, you can go to Advanced Options, then select the Max Min options. Here, give a minimum value and a maximum value; if ANY value of the entry goes outside of this range(as long as it can conceivably be converted to a double), that specific value will be replaced by the other most extreme value. For example, if you loop it from 0 to 5, and the value would become 6, the value now becomes 0. Note that this is inclusive on both sides; a minimum and maximum of one would ONLY allow one.

We will cover more of the advanced options in the next part.

Part III: Advanced Options

Overview

Up until this point, our requests of SoluCheck have been fairly simple; now, however, we will show the full functionality of SoluCheck. Of note: All add-ons should harmoniously work with each other, and authority is gracefully handled; exempt values always have highest priority, followed by maximum and minimum values, followed by array sizing.

Specifically, we will explore these capabilities:

- Argument specific options, such as ranges, file viewing, databases, and array sizing.
- Run specific options, such as notifications, loading variable files, and testing extended file types, as well as the logger.

Advanced Options: Argument Specific Options

Here you will find brief synopses of all the argument specific options:

1) Set Max/Min Values: Use this to, as previously described, tell SoluCheck that the argument must remain inclusively between two values, as given in the resulting dialogue box.
2) Set Exempt Values: Use this to tell SoluCheck that, if the value of the argument ever becomes one of the given exempt values, then replace this value with one that is supplied as a 'stand-in'.
3) Step Array Size: Use this to force SoluCheck to increase your array's size each iteration by some integer. Should this integer be negative, the array will shrink until one of the array values is 1. If positive, the resulting array will be increased, and each new value will be a supplied value.
4) View Source File: Use this to view the actual source code, and, if needed, edit it in MATLAB.
5) Variable I/O: Use this to override SoluCheck's built-in arguments in and arguments out tester, and define your own number of arguments. This can lead to unhandled errors!
6) Load Database: Use this to tell SoluCheck that you are loading in a database of arguments; the first column is your actual arguments, and the second column is your data types. Any value that could normally be used can be used here.

These are all argument specific; you MUST select a file before these will be useful!

Advanced Options: Run Specific Options

Here you will find brief synopses of all run specific options:

1) Profiler: Use this to, as described above, profile and analyze your code.
2) Timing: Again, as stated earlier, this can be used to compare your code's efficiency with that of the solution file's.
3) View Details: Use this to open the logger viewer, which keeps track of everything that happened since the last clearing. To save this text, go to SoluCheck>Save Details.
4) Load Variables: Use this to load .mat files for testing; these will NOT be loaded into the base workspace!
5) Notifications: Use this to receive an email once SoluCheck has finished testing; especially useful for long simulations with many parameters.
6) Image Testing: Use this to tell SoluCheck that you are testing images and that it will need to look for image files on each run. If the images differ, the test will be considered a failure.
7) Plot Testing: Use this to tell SoluCheck that your code generates figures and plots. It will not tell you if your code passed or failed, but will return a percent difference between the code plots and solution plots.
8) File Testing: Use this to tell SoluCheck that it should look for Text files and test them. If the text files differ, the test is considered a failure.

These are all run-specific; you only need to select them when you launch SoluCheck.

Part IV: Troubleshooting, Technology, and FAQ.

Overview:

In this section, since we have already covered the full functionality of SoluCheck, we will now explore what to do when things do not work out.

Specifically, in this part, you will find:

- Troubleshooting advice
- Minimum Technology Requirements
- FAQ
- Contact Information

Troubleshooting Advice:

If you run into problems, the first step is checking to make sure that you have entered all information correctly; SoluCheck is very good about giving back specific problems it has, so this should not normally be a problem. If you are sure everything is as it should be, ensure that your function doesn't actually error out at this point in time. Finally, if all else fails, send an email to SoluCheck@gmail.com, and we'll take a look at it. Be as descriptive as possible in your email, and make the subject say "SoluCheck Error: <your name>".

Minimum System Requirements:

Microsoft Windows:

- Windows 7 or higher, with all available service packs
- MATLAB v 2013A or higher
- 4 GB or more of RAM
- 256 MB HD space for SoluCheck
- MS Excel 2007 or higher (for some features)

Macintosh OS X

- Mac OS X 10.7.1 (Lion) or higher
- MATLAB v 2013A or higher
- 8 GB or more of RAM
- 256 MB HD space for SoluCheck
- MS Excel 2011 or higher (for some features)
- **Be advised that not all features are available on Macintosh Systems!

Linux OS

- Linux Kernel version 4.1 or later
- 4 GB or more of RAM
- 256 MB HD space for SoluCheck
- **Be advised that not all features are available on Linux Systems!

FAQ

1) How do I give complex step sizes?
    a. You must utilized Formulaic Entry… options.
2) Why can't I enter a step size?
    a. If you've selected a specialized data type, then it will be impossible for SoluCheck to step the value.
3) How do I report bugs?
    a. Send us an email at SoluCheck@gmail.com!

Developer Information:

SoluCheck was developed by Alexander H Rao, a freshman at the Georgia Institute of Technology. SoluCheck development was greatly helped by a Ms. Rebecca Cottrill, and a Mr. Hyder Hasnain, as well as a Professor Melinda McDaniel. All three of these people were instrumental to the development and distribution of SoluCheck, and I could not be more grateful to them!