Controlling HEC-RAS using MATLAB

Arturo S. Leon^{a,*}, Christopher Goodell^b

^a*School of Civil and Construction Engineering, Oregon State University, 101 Kearney Hall, Corvallis, OR 97331-3212, USA*

^b*WEST Consultants, Inc., 10300 SW Greenburg Rd, Portland, OR 97223, USA*

Abstract

The U.S. Army Corps of Engineers' Hydrologic Engineering Center's River Analysis System (HEC-RAS) is a widely used software application for performing one-dimensional and two-dimensional steady and unsteady flow river hydraulics calculations, sediment transport-mobile bed modeling, and water quality analysis. User's of HEC-RAS have often unique applications including the coupling with other software to perform system analysis such as optimization of flooding structures and multi-objective reservoir operation under uncertainty. One state-of-the-art environment for integrating software is MATLAB, which integrates computation, visualization, and programming in an easy-to-use environment. This paper presents a set of MATLAB scripts to write input files, read output files, make plots, execute parallel computations, and perform fully-automated functions of HEC-RAS. Examples of procedures are presented throughout the paper and they are illustrated using a river-reservoir network that involves ten inline structures (e.g., dams) with operation of gates at each of these dams.

Keywords: HEC-RAS, HECRASController, MATLAB, Model integration,

*Corresponding author

Email address: `arturo.leon@oregonstate.edu` (Arturo S. Leon)

1. Introduction

HEC-RAS is a widely used software application that can perform one and two-dimensional hydraulic calculations for a full network of natural and constructed channels, overbank/floodplain areas, levee protected areas; etc (Hydrologic Engineering Center 2016a, Hydrologic Engineering Center 2016b). HEC-RAS has four main modules: (1) steady flow water surface profiles, which is intended for calculating water surface profiles for steady gradually varied flow; (2) unsteady flow simulation, which can simulate one-dimensional, two-dimensional and combined one/two-dimensional unsteady flow through a full network of open channels, floodplains, and alluvial fans; (3) sediment transport computations, which is intended for the simulation of one-dimensional sediment transport/movable boundary calculations resulting from scour and deposition over moderate to long time periods; and (4) water quality analysis; which is intended to allow the user to perform river-ine water quality analyses (Hydrologic Engineering Center 2016a, Hydrologic Engineering Center 2016b). Standard applications of this model include flood wave routing and flood inundation studies.

The user's of HEC-RAS have often unique applications that may include the coupling with other software to perform system analysis such as flood risk analysis, optimization of flooding structures under uncertainty and multi-objective reservoir operation under uncertainty. A system analysis requires the use of a programming platform or environment for integrating multiple software and/or open source codes. One state-of-the-art programming

platform is MATLAB, which is a high-performance language for technical
25 computing that integrates computation, visualization, and programming in
an easy-to-use environment ([Mathworks 2015](#)).

This paper presents a set of MATLAB scripts to write input files, read
output files, and perform fully-automated functions of HEC-RAS. To the
authors' knowledge this is the first time that MATLAB is used for fully
30 controlling the input and output of HEC-RAS. The scripts described in the
paper include parallel computing (simultaneous computations of HEC-RAS),
modifying input files, accessing output files and coupling with an optimiza-
tion software. It is worth mentioning that although this paper makes use of
the USACE HECRASController described in [Goodell \(2014\)](#), the main focus
35 of this paper is on programming procedures for controlling the input and
output of HEC-RAS without relying on available functions of the aforemen-
tioned HECRASController. The reasons for the latter is that the functions
available on the HECRASController are limited and very often the user's
may want to perform tasks for which there is no function in the controller.
40 For an in-depth discussion of all functions available in the USACE HECRAS-
Controller, the reader is referred to [Goodell \(2014\)](#). This paper is divided as
follows. First, the USACE HECRASController is briefly described. Second,
MATLAB scripts for various tasks are presented throughout the paper and
they are illustrated using a river-reservoir network that involves ten inline
45 structures (e.g., dams) with operation of gates at each of these dams. Finally,
the key points of the paper are summarized in the conclusion.

2. HEC-RAS Controller

This section presents a very brief discussion on the USACE HECRAS-Controller. We will only focus on few functions of the HECRASController
50 that are very useful. For an in-depth discussion of all functions available in the USACE HECRASController, the reader is referred to [Goodell \(2014\)](#). The reader is referred to Script 1 for this discussion.

- The script line containing *actxserver* will create a new, invisible copy of HEC-RAS. The text *RAS500.HECRASCONTROLLER* is used for
55 HEC-RAS version 5.0 and *RAS41.HECRASCONTROLLER* for version 4.1.
- The function *Project_Open(ras_file)* will open a RAS file, where *ras_file* is a string.
- The function *Compute_HideComputationWindow* hides the HEC-
60 RAS computation window, which is useful when performing parallel computations or serial batch computations. The function *Compute_ShowComputationWindow* shows the computation window for each HEC-RAS computation.
- The function *Compute_CurrentPlan* runs HEC-RAS for current plan.
- The function *Project_Save* saves the HEC-RAS project
- The function *OutputDSS_GetStageFlow* is intended for extracting
65 water surface stage and flow discharge at selected cross-sections.

Script 1. Basic functions of USACE HEC-RAS Controller

```
1 function run_hec_ras_unsteady(ras_file)
2 %Written by Arturo Leon (artuleon@gmail.com), Dec 26, 2015
70 3 %h=actxserver('RAS41.HECRASCONTROLLER');
4 h=actxserver('RAS500.HECRASCONTROLLER');
5 %The above command depends on the version of HEC-RAS. I am
6 %using version 5.0. This key can be found in windows registry
7 h.Project_Open(ras_file); %Open ras file
75 8 %h.GetRASVersion; %To print version of HEC-RAS
9 h.Compute_HideComputationWindow; %To hide Computation window
10 %h.Compute_ShowComputationWindow; %To show computation window
11 %h.CurrentPlanFile; %Indicates current HEC-RAS plan file and path
12 %h.Plan_SetCurrent; %Changes current plan to supplied plan name
80 13 h.Compute_CurrentPlan(0,0); %Runs HEC-RAS for current plan
14 %[z1,z2,z3,z4,z5,z6,z7,z8,z9] = ...
15 %h.OutputDSS_GetStageFlow(River_ID{k},Reach_ID{k},Node_ID{k}, ...
16 % 0,0,0,0,z9); % This function is for extracting water surface stage
17 % and flow discharge at selected cross-sections
85 18 h.Project_Save; %Saves the project
19 delete(h); %Deletes the handle h
```

3. Reading and Writing HEC-RAS Input Files

As described in Goodell (2014), the most common HEC-RAS input text
90 files are:

1. Geometry file: *.g###
2. Steady flow file: *.f###
3. Unsteady flow file: *.u###

There are other input files such as Plan file (*.p##), Project file (*.prj), and
95 others. The manipulation of the input files are very similar so due to space
limitations we will show four examples for the *geometry* and *unsteady flow*
input files.

The first example will find and printout the title name of the geometry
file. The reader is referred to Script 2 for this example. In this script, the
100 filename will have the extension *.g##. Script 2 reads the file line by line.
Whenever the script finds the character "=", it will split the text in two (left
and right of "="). Then the script checks if the left of the text is the same
as the string "Geom Title". If it is, it will extract and printout the right of
the text, which will be the title name of the geometry file.

Script 2. Script to obtain the title name of the geometry file

```
105
1  function GetGeometryTitle(filename)
2  %Written by Arturo Leon (artuleon@gmail.com), Dec 26, 2015
3  fid = fopen (filename, 'r'); %Open file for reading
4  while ~feof(fid)
110 5      strTextLine = fgetl(fid);
6      string_temp = regexp(strTextLine, '=', 'split');
7      strGeometryTitle = string_temp(:,1);
8      %Search geometry text file for the key "Geom Title"
9      if strcmp(strGeometryTitle, 'Geom Title');
115 10          str_Geometry_obtained = string_temp(:,2);
11          fprintf('The Title of the geometry file is'), ...
12          str_Geometry_obtained
13      end
14  end
```

```
120 15 fclose (fid); %Close the text file
```

The second example updates water stage elevations and flow discharges at multiple cross-sections that will be used as the initial conditions for an unsteady flow simulation. This example corresponds to an optimization of reservoir operation in a ten-reservoir system. The reader is referred to Script 3 for this example. In this script, the filename to use will have the extension *.u##.

Script 3 first reads data of current tailwater and forebay elevation for the ten dams. Then the script reads the current inflows and outflows of the ten reservoirs, which are stored for later use. Following, the unsteady input file needs to be updated with the corresponding initial water stages and flow discharges. To perform this task, first the original unsteady file is copied to a temporal file (“24X*SNEW_temp.u01*”). Then using as baseline the temporal unsteady file, the original unsteady file (“24X*SNEW.u01*”) is rewritten with the current water stages and flow discharges. To start to replace the initial conditions, it is necessary to find the key variable “Use Restart= 0” in the temporal file that is being read. Once this string is found, it should be printed out in the file and then the initial flows and stages are also printed out in the original file being rewritten. The initial flow should contain the string “Initial Flow Loc=” at the most left part of the text line. This string should be followed by the river, reach, station and the initial flow discharge at this river station. The initial water stage should contain the string “Initial RRR Elev=” at the most left part of the text line. This string should be followed by the river, reach, station and the initial water stage at this river station. After this data is written in the original file, we should

continue reading the temporal file without writing anything until the string “Boundary Location=” is found. The latter will avoid errors in the input file due to data size incompatibility between the old (temporal file) and the new (being rewritten) input files.

Script 3. Script to update water stage elevations and flow discharges at multiple cross-sections that will be used as the initial conditions for an unsteady flow simulation

```

150 1 %Initial conditions: Update initial water stages and outflows
2 file_current_TW = [home_dir '\InitCond\CurrentTW.txt'];
3 Data_curr_TW = dlmread(file_current_TW);
4 file_current_FB = [home_dir '\InitCond\CurrentFB.txt'];
155 5 Data_curr_FB = dlmread(file_current_FB);
6 file_current_inflow = [home_dir '\InitCond\CurrentInflows.txt'];
7 Data_curr_inflows = dlmread(file_current_inflow);
8 file_current_outflow = [home_dir '\InitCond\CurrentOutflows.txt'];
9 Data_curr_outflows = dlmread(file_current_outflow);
160 10 for j=1:Number_dams;
11     k = Order_Conv(j);
12     m = 2*j;
13     XS_stage(m-1) = Data_curr_FB(k);
14     XS_stage(m) = Data_curr_TW(k);
165 15     XS_flow(m-1) = 1000*Data_curr_inflows(k);%to convert to cfs
16     XS_flow(m) = 1000*Data_curr_outflows(k);
17 end
18 filenameinput = [home_dir '\RAS_folders\24XS-Col\24XSNEW_temp.u01'];
19 filenameoutput = [home_dir '\RAS_folders\24XS-Col\24XSNEW.u01'];
170 20 copyfile(filenameoutput,filenameinput);
21 %filenameinput %Input file is the temporal file
22 %filenameoutput %Output file is the initial file

```



```

23 fid = fopen (filenameinput, 'rt'); %Open file for reading
24 fout = fopen (filenameoutput, 'wt'); %Open file for writing
175 25 while ~feof(fid)
26     strTextLine = fgetl(fid); %To read one additional line
27     if strfind(strTextLine, 'Use Restart= 0');
28         fprintf(fout, '%s\n', strTextLine);
29         for j=1:Number_dams; %Initial_flows
180 30             m = 2*j;
31             str1 = 'Initial Flow Loc=';
32             str2 = num2str(XS_flow(m-1));
33             strTextLine2 = strcat(str1, XS_IC(m-1), str2);
34             fprintf(fout, '%s\n', strTextLine2{1});
185 35             str2 = num2str(XS_flow(m));
36             strTextLine2 = strcat(str1, XS_IC(m), str2);
37             fprintf(fout, '%s\n', strTextLine2{1});
38         end
39         for j=1:Number_dams; %Initial water stages
190 40             m = 2*j;
41             str1 = 'Initial RRR Elev=';
42             str2 = num2str(XS_stage(m-1));
43             strTextLine2 = strcat(str1, XS_IC(m-1), str2);
44             fprintf(fout, '%s\n', strTextLine2{1});
195 45             str2 = num2str(XS_stage(m));
46             strTextLine2 = strcat(str1, XS_IC(m), str2);
47             fprintf(fout, '%s\n', strTextLine2{1});
48         end
49     else
200 50         fprintf(fout, '%s\n', strTextLine);
51     end
52 end

```

205

```

53 fclose (fid); %Close the text file
54 fclose (fout); %Close the text file

```

The third example updates the inflow hydrographs (two) and pre-scheduled gate outflows at ten inline structures (i.e, dams). The pre-scheduled outflows are generated by an optimization routine (Genetic Algorithm) with a pre-specified population.

210 In a genetic algorithm, a population of candidate solutions (called individuals) to an optimization problem is evolved toward better solutions. The evolution usually starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in 215 the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (recombined and randomly mutated) to form a new generation. The new generation of candidate solutions is then used 220 in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. For more details about the genetic algorithm and its application to reservoir operation the reader is referred to Wardlaw and Sharif (1999), Leon and 225 Kanashiro (2010), Leon et al. (2014), Lerma et al. (2015), Yang et al. (2015), and Chen et al. (2016). Due to space limitations, a code of a genetic algorithm is not presented herein. However, there are various codes available in the literature (e.g., <https://www.idealsoftware.com/opensource/>

genetic-algorithm.html and <http://gaul.sourceforge.net/>).

230 The update of dam outflows is done at each generation for each population of the optimization. It is worth mentioning that in an optimization-simulation framework, the initial conditions (second example) need to be updated only at the beginning of the optimization. The reader is referred to Scripts 4 and 5 for the third example. In this example, the file to update will
235 have the extension *.u##.

This example has two parts. In the first part, the data (inflow hydrographs and outflows) to be written in the unsteady file is prepared. The HEC-RAS unsteady input file (*.u##) is formatted in such a way that the data points for inflow hydrographs and outflows at inline structures can
240 have a maximum of 10 data points per line. Script 4 prepares the lines of data to be written in the unsteady file. This script first calculates the number of lines of data for the inflow hydrographs (“*NLines_inflow*”) and the outflows (“*NLines_Out*”) based on the number of data points. For instance, if the simulation period is 4 days with gate outflows specified every
245 hour, the number of data points for the outflows would be 97 that includes the initial condition ($t = 0$). In this case, the number of lines of data for the outflows (“*NLines_Out*”) would be 10. Next, the script calculates the number of data points for the last line of the inflow hydrographs (“*last_line_inf*”) and the outflows (“*last_line_dam*”). The reason
250 for the latter is because not always the number of data points of the last line is 10. Then, the data for inflow hydrographs is split in two, one for all the lines except the last one (“*Inflow_array_main*”) and the last line (“*Inflow_array_last_line*”). Likewise, the data for the dam outflows are

split into (“*Outflow_array_main*”) and (“*Outflow_array_last_line*”).

255 Once the data has been prepared, Script 4 will call Script 5 for each population of the optimization. To start to replace the data, it is necessary to find key variables in the temporal unsteady file that is being read. These variables are “Flow Hydrograph=” for the inflow hydrographs and “Rule Table=” for the dam outflows. In a similar way to the second example, after
260 the data has been written in the original file, we should continue reading the temporal file without writing anything until the strings “DSS Path=” and “Rule Operation=” have been found for the inflow hydrographs and dam outflows, respectively. The latter will avoid errors in the input file due to data size incompatibility between the old (temporal file) and the new (being
265 rewritten) input files.

Script 4. Script to prepare the data lines for the inflow hydrographs and gate outflows to be written in the unsteady file of HEC-RAS

```
1 NLines_inflow = fix((Inflow_points-0.1)/10) + 1;
2 last_line_inf = Inflow_points-10*(NLines_inflow-1); %last line infl
3 NLines_Out = fix((Outflow_points-0.1)/10) + 1;
270 4 last_line_dam = Outflow_points-10*(NLines_Out-1); %last line outfl
5 pos_data = 10*(NLines_inflow-1);
6 for j=1:Number_Inflow_hydrog;
7     arrayflow1 = Data_Inflow_hydrog(1:pos_data, j:j)';
8     Inflow_array_main(:, :, j) = reshape(arrayflow1, 10, [])';
275 9     arrayflow2 = Data_Inflow_hydrog(pos_data+1:Inflow_points, j:j);
10     Inflow_array_last_line(1, :, j) = arrayflow2'; %Transpose
11 end
12
```

```

13 %Adding initial outflows
280 14 Data_outflow_current = Data_curr_outflows';
15 for i=1:Pop_Opt;
16     temp_int1 = 10*(i-1)+1;
17     temp_int2 = 10*(i);
18     temp_array1 = Data_optimization_flows(1:Outflow_points-1, ...
285 19     temp_int1:temp_int2);
20     temp_array2 = [Data_outflow_current; temp_array1];
21     temp_array2 = 1000*temp_array2; %to convert to cfs
22     pos_data = 10*(NLines_Out-1);
23     for j = 1:Number_dams;
290 24         k = Order_Conv(j);
25         temp_array3 = temp_array2(1:pos_data,k:k)';
26         Outflow_array_main(:, :, j) = reshape(temp_array3, 10, [])';
27         temp_array4 = temp_array2(pos_data+1:Outflow_points,k:k);
28         Outflow_array_last_line(1, :, j) = temp_array4'; %Transpose
295 29     end
30     ChangeInlineStruct_Data(ras_inp{i}, ras_out{i}, NLines_Out, ...
31     Outflow_array_main, Outflow_array_last_line, NLines_inflow, ...
32     Inflow_array_main, Inflow_array_last_line);
33 end
300

```

Script 5. Script to update pre-scheduled gate outflows and hydrographs at dams

```

1 function ChangeInlineStruct_Data(filenameinput, filenameoutput, ...
2     NLines_Out, Outflow_array_main, Outflow_array_last_line, ...
3     NLines_inflow, Inflow_array_main, Inflow_array_last_line)
305 4 %Written by Arturo Leon (artuleon@gmail.com), Dec 26, 2015
5 %This is to update outflows and hydrographs at dams
6 fid = fopen (filenameinput, 'rt'); %Open file for reading

```

```

7  fout = fopen (filenameoutput, 'wt'); %Open file for writing
8  i = 0; %initialize pointer for dams
310 9  m = 0; %initialize pointer for inflow hydrographs
10  while ~feof(fid)
11      strTextLine = fgetl(fid); %To read one additional line
12      if strfind(strTextLine, 'Flow Hydrograph=');
13          m = m+1;
315 14          fprintf(fout, '%s\n', strTextLine);
15          for k = 1:NLines_inflow-1; %All lines except last one
16              %convert to real + char + reshape + transpose
17              array1 = Inflow_array_main(k, :, m);
18              Blstring=reshape(sprintf('%8.0f', array1), 8, []);
320 19              Blstring=[cellstr(Blstring)]';
20              Blstring = strjoin(Blstring, '');
21              fprintf(fout, '%s\n', Blstring);
22          end
23          array2 = Inflow_array_last_line(1, :, m);
325 24          Blstring = reshape(sprintf('%8.0f', array2), 8, []);
25          Blstring=[cellstr(Blstring)]';
26          Blstring = strjoin(Blstring, '');
27          fprintf(fout, '%s\n', Blstring);
28      elseif strfind(strTextLine, 'Rule Table=');
330 29          i = i+1;
30          fprintf(fout, '%s\n', strTextLine);
31          for k = 1:NLines_Out;
32              strTextLine = fgetl(fid);
33              fprintf(fout, '%s\n', strTextLine);
335 34          end
35          for k = 1:NLines_Out-1; %All lines except last one
36              array3 = Outflow_array_main(k, :, i);

```

```

37         Alstring=reshape(sprintf('%8.0f',array3),8,[]);
38         Alstring=[cellstr(Alstring)]';
340 39         Alstring = strjoin(Alstring, '');
40         fprintf(fout, '%s\n', Alstring);
41     end
42     array4 = Outflow_array_last_line(1,:,i);
43     Alstring = reshape(sprintf('%8.0f',array4), 8, []);
345 44     Alstring=[cellstr(Alstring)]';
45     Alstring = strjoin(Alstring, '');
46     fprintf(fout, '%s\n', Alstring);
47 else
48     fprintf(fout, '%s\n', strTextLine);
350 49 end
50 end
51 fclose (fid); %Close the text file
52 fclose (fout); %Close the text file

```

355 4. Extracting output variables, plotting and parallel computing

This section presents examples to extract water surface stages and flow discharges, plotting and to perform parallel HEC-RAS computations. There are functions available in the HECRASController for extracting water surface stages and flow discharges, and their plotting, so these tasks are straight
 360 forward. The reader is referred to Script 6 for extracting water surface stages and flow discharges and Script 7 for plotting water surface stage and flow discharge at a given river station. The HECRASController function “*OutputDSS_GetStageFlow*” allows to extract stage and flow hydrographs at a given river station. The HECRASController subroutine “*PlotStageFlow*”

365 allows to plot the stage and flow hydrograph at a given river station. An example of this plot is shown in Figure 1.

Script 6. Script to extract water surface stage and flow discharge

```

1  h=actxserver('RAS500.HECRASCONTROLLER');
2  h.Project_Open(ras_file); %Open ras file
370 3  h.Compute_CurrentPlan(0,0); %Runs HEC-RAS for current plan
4  z9 = 'error message';
5  for k=1:Num_XS_Outp;
6      i = int8(k/2+0.1); %This will give 2 ones, 2 2s, etc
7      [z1,z2,z3,z4,z5,z6,z7,z8,z9] = ...
375 8  h.OutputDSS_GetStageFlow(River_ID{k},Reach_ID{k},Node_ID{k},0,0,0,0,z9);
9      if mod(k,2) == 0 %If mod=0, then it is even. (1=odd)
10         FB_stage_Output(:,Number_dams*(j-1)+i) = z7;
11         FB_flow_Output(:,Number_dams*(j-1)+i) = z8;
12     elseif mod(k,2) == 1
380 13         TW_stage_Output(:,Number_dams*(j-1)+i) = z7;
14         TW_flow_Output(:,Number_dams*(j-1)+i) = z8;
15     else
16         error('mod(k,2) .ne. 0 ,1. Check run_hecras_unsteady')
17     end
385 18 end

```

Script 7. Script to plot water surface stage and flow discharge

```

1  h=actxserver('RAS500.HECRASCONTROLLER');
2  h.Project_Open(ras_file{Pop2}); %Open ras file
390 3  h.PlotStageFlow(River_ID{XS2},Reach_ID{XS2},Node_ID{XS2});

```

Figure 1. Example of a plot produced using the HECRASController subroutine “PlotStageFlow”

Following we present the script to perform parallel HEC-RAS computations. The reader is referred to Script 8 for this example. The script first creates a special job on a pool of workers, and connects the MATLAB client
395 to the parallel pool. This is done using the “*parpool*” function of MATLAB. Then, the scripts defines an array of handles ($h\{j\}$) with a number equal to the number of HEC-RAS computations. Next, the *parfor* function of MATLAB is used for the parallel computations. Finally, the system *taskkill* command is used to close all open HEC-RAS projects. A snapshot of simul-
400 taneous HEC-RAS computations is shown in Figure 2.

Script 8. Script to perform parallel HEC-RAS computations

```

1 %Run HEC-RAS Model for all the populations of the optimization
2 prompt = 'Will you use parallel computing (y/n)? ';
3 paral_comput = input(prompt,'s')
405 4 if paral_comput == 'y'
5     delete(gcp('nocreate'));%To avoid interactive session error
6     parpool('local',4); %Parallel computation
7 end
8 for j=1:Pop_Opt;
410 9     StrID_2 = num2str(j);
10     h{j} = StrID_2;
11 end
12 parfor j=1:Pop_Opt %Population_Optim;
13     h{j}=actxserver('RAS500.HECRASCONTROLLER');
415 14     h{j}.Project_Open(ras_file{j}); %Open ras file
15     h{j}.Compute_HideComputationWindow; %Hide Comput. Window
16     %h{j}.Compute_ShowComputationWindow; %Show Comput. Window
17     h{j}.Compute_CurrentPlan(0,0); %Run current plan
18     h{j}.Project_Save; %Saves the project
420 19     delete(h{j}) %Deletes the handle h{j}
20 end
21 %To kill hec-ras from the background
22 !taskkill /im ras.exe

```

425 5. Conclusions

This paper presents a set of MATLAB scripts to write input files, read output files, and perform fully-automated functions of HEC-RAS. Examples of various programming procedures are presented throughout the paper and they are illustrated using a river-reservoir network that involves ten inline

Figure 2. A snapshot of simultaneous HEC-RAS computations

430 structures (e.g., dams) with operation of gates at each of these dams. The procedures includes reading and writing HEC-RAS input files, extracting output variables, plotting and parallel computing. In addition, this paper presents a brief introduction to the USACE HECRASController.

Acknowledgments

435 The first author gratefully acknowledges the financial support of the Bonneville Power Administration of the U.S. Department of Energy under award number TIP#258.

References

- Chen, D., Leon, A.S., Gibson, N.L., Hosseini, P., 2016. Dimension reduction
440 of decision variables for multireservoir operation: A spectral optimization model. *Water Resources Research* 52, 36–51.
- Goodell, C., 2014. *Breaking the HEC-RAS Code*. First ed., h2ls, Portland, Oregon.

- Hydrologic Engineering Center, 2016a. HEC-RAS, River Analysis System,
445 Hydraulic Reference Manual. Version 5.0. U.S. Army Corps of Engineers,
Davis, California.
- Hydrologic Engineering Center, 2016b. HEC-RAS River Analysis System
User's Manual - Version 5.0. U.S. Army Corps of Engineers, Davis, Cali-
fornia.
- 450 Leon, A.S., Kanashiro, E., 2010. A new coupled optimization-hydraulic
routing model for real-time operation of highly complex regulated river
systems, in: Watershed Management Conference: Innovations in Water-
shed Management Under Land Use and Climate Change, ASCE-EWRI,
Madison, Wisconsin, USA. pp. 213–224.
- 455 Leon, A.S., Kanashiro, E., Valverde, R., Sridhar, V., 2014. Dynamic frame-
work for intelligent control of river flooding: Case study. Journal of Water
Resources Planning and Management 140, 258–268. doi:[10.1061/\(ASCE\)
WR.1943-5452.0000260](https://doi.org/10.1061/(ASCE)WR.1943-5452.0000260).
- Lerma, N., Paredes-Arquiola, J., Andreu, J., Solera, A., Sechi, G.M., 2015.
460 Assessment of evolutionary algorithms for optimal operating rules design
in real water resource systems. Environmental Modelling & Software 69,
425 – 436.
- Mathworks, 2015. MATLAB version 8.6.0.267246 (R2015b). The Mathworks,
Inc. Natick, Massachusetts.
- 465 Wardlaw, R., Sharif, M., 1999. Evaluation of genetic algorithms for opti-

mal reservoir system operation. *Journal of Water Resources Planning and Management* 125, 25–33.

Yang, T., Gao, X., Sellars, S.L., Sorooshian, S., 2015. Improving the multi-objective evolutionary optimization algorithm for hydropower reservoir operations in the california oroville–thermalito complex. *Environmental Modelling & Software* 69, 262 – 279.