A MATLAB Framework for Forecasting Optimal Flow Releases in a Multi-storage System for Flood Control

Arturo S. Leon[a,*], Yun Tang[b], Li Qin[c], Duan Chen[d]

[a] *Department of Civil and Environmental Engineering, Florida International University, 10555 W. Flagler Street Miami, FL 33174, USA*
[b] *Department of Civil and Environmental Engineering, University of California at Davis, 2001 Ghausi Hall, 1 Shields Avenue, Davis, CA 95616, USA*
[c] *Department of Information Science and Engineering, Ningbo University, Ningbo 315211, China*
[d] *Changjiang River Scientific Research Institute, Wuhan 430010, China*

Abstract

This paper presents a MATLAB framework for forecasting optimal flow releases in a multi-storage system for flood control. This framework combines four widely-used models intended for (1) performing hydrologic analysis for a watershed and for level-pool routing in the storage systems, (2) simulating river inundation, (3) solving the optimization problem of determining hourly optimal flow releases in a multi-storage system, and (4) data management and plotting. The integration of all software is performed in MATLAB, which is a state-of-the-art and an easy-to-use environment for integrating computation, visualization, and programming. This paper focuses on (1) presenting the MATLAB scripts for interfacing the aforementioned four software, (2) describing the rationale for the objective function of the optimization, and (3) demonstrating the practical use of the MATLAB framework by applying it to the operation of a hypothetical eight-pond system in the Cypress Creek watershed in Houston, Texas. The results of the aforementioned application show that this framework could help in mitigating flooding.

Keywords: Flood control, Genetic Algorithms, HEC-DSSVue, HEC-HMS, HEC-RAS, MATLAB, Watershed

1. Introduction

According to the National Oceanic and Atmospheric Administration (NOAA 2016), inland flooding produces more damage annually than any other weather event in the United States. For the period between 1976 and 2006, the damage produced by inland flooding averaged $6.9 billion per year (NOAA 2016).

*Corresponding author
Email address: arleon@fiu.edu (Arturo S. Leon)

According to the Earth Observatory of NASA (NASA 2017), global warming will change the major climate patterns and in general, as precipitation patterns change, storms, floods, and droughts will be more severe. Furthermore, the changes in land use associated with increasing trends in urban development will increase the peak discharge and frequency of floods (USGS 2003).

Recently, flood mitigation within the context of a watershed, where the entire watershed becomes the objective of management, is receiving increasing attention (Kusler 2004, Flotemersch et al. 2016). Within a watershed, often, a network of storage systems (e.g. detention ponds, wetlands, reservoirs) is available for flood control, however, they are rarely managed in a coordinated manner. The lack of coordination of storage systems control may limit their effectiveness for flood mitigation. Furthermore, the remote operation of water release in storage systems is not common and it is often limited to reservoirs. In an ideal flood mitigation system, all storage systems of the watershed would be operated in a coordinated manner according to precipitation forecasts to achieve certain objectives such as minimizing inundation levels at certain sections of the river. Furthermore, the storage systems would be partially or totally emptied ahead of (e.g., a few hours or a couple of days before) a heavy rainfall that would produce flooding. The storage made available by the early release would provide extra water storage during the heavy rainfall, thus mitigating floods.

Detention ponds and reservoirs are often constructed for mitigating floods as a main objective and these systems could be operated aggressively during flooding conditions for mitigating inundation. The operation of wetlands for flood control, however, could have ecological implications. For instance, many wetlands retain some water year-round supporting long-lived aquatic animals such as fish, as well as wetland amphibians and invertebrates (Leon et al. 2018). However, there are also many wetlands that naturally have a short hydroperiod, so their function is not necessarily decreased by partial draining (Leon et al. 2018). As pointed out by Leon et al. (2018), draining from wetlands involves some risks that can be minimized by not draining the wetlands fully, and by draining only when the certainty of storm events is very high. In any case, if wetlands would be used for flood control, they would be wetlands that are naturally flooded for only short periods of the year (a short hydroperiod) and support primarily facultative wetland animals and plant species that benefit from or can tolerate occasional flooded conditions but do not require them most of the year (Snodgrass et al. 2000, Ehrenfeld 2004, Tarr et al. 2005, Leon et al. 2018).

The target audience of this work is anyone interested in near real-time flood control. This could include engineers responsible for flood management and short-term flood control. This work is also intended for researchers and students interested in open-source techniques for flood mitigation. The focus of this manuscript is on flood control and thus the storage used for flood mitigation would exclude wetlands that support long-lived aquatic animals. To facilitate the coordinated operation of a multi-storage system for flood mitigation, the integration of hydrology and hydraulic models within an optimization framework would be required. Users often have unique applications that may require

modifying the framework for their particular needs. To achieve the latter, it is desirable to integrate the aforementioned models within a state-of-the-art and an easy-to-use environment that is suitable for computation, visualization, and programming. One of such platforms is MATLAB, which is a high-performance language for technical computing that integrates computation, visualization, and programming in an easy-to-use environment (Mathworks 2015). This paper presents a MATLAB framework for forecasting optimal flow releases in a multi-storage system for flood control. This paper is organized as follows: (1) the sub-component models and the rationale for the objective function are briefly described; (2) MATLAB scripts for interfacing the aforementioned sub-component models are presented; (3) a brief overview of a case study is presented to illustrate the application of the MATLAB framework. Finally, the key results are summarized in the conclusion.

2. Considerations for the optimal and coordinated operation of storage systems for flood mitigation

It is clear that each watershed prone to flooding may have specific challenges and different topographical conditions, however in general, most of these watersheds have inline storage systems (e.g., reservoirs) and off-line storage systems (e.g., detention ponds, wetlands). To make possible the optimal and coordinated operation of storage systems for flood control, it would be necessary to add or retrofit gates/siphons for water release of the storage systems so they can discharge most of the water stored in a pre-determined time. This task would include implementing the necessary hardware and software to make possible the remote operation of gates/siphons in wetlands, detention ponds and reservoirs. Remote operation of gates/siphons are necessary because very often wetlands, detention ponds and reservoirs are located in remote areas far away from urban areas. The remote operation could be achieved using the classical Supervisory control and data acquisition (SCADA) system or using latest and cheaper technologies such as those presented in Leon and Verma (2019). It is noted that adding conventional drainage pipes (e.g., sloped pipes) would require substantial earthwork as they would need to be installed in the earthen berm. Conversely, installing siphons require minimal earthwork as only anchoring of the siphon pipes over the berms would be necessary.

It is noted that a siphon flow (uphill flow followed by a downhill flow) or a pure downhill flow are gravity-driven flows which require a hydraulic gradient between the water surface level in the storage system and the discharge point (e.g., natural or artificial drainage ditch). In many instances, even near flat land can be engineered to create a storage system and achieve a hydraulic gradient of at least 0.6 or 0.9 m, which would still give significant flow velocities exceeding 1 m/s. A storage system could be engineered to be a single pond or an interconnected system of multiple ponds (Leon et al. 2018). If the area for the pond construction is consistently flat, the earthwork of a single pond would likely be the most economical option. However, in most cases, the terrain is not horizontally leveled and the earthwork would be significantly reduced if a

system of interconnected ponds with different bottom levels is used (Leon et al. 2018).

3. Model Description

As mentioned earlier, the coordinated operation of storage systems for flood control requires a computational framework that integrates hydrologic and hydraulic models within an optimization framework. There is a vast array of hydrologic, hydraulic and optimization models available in the literature. Because one of the most popular and freely available models for simulating river inundation is the U.S. Army Corps of Engineers' Hydrologic Engineering Center's River Analysis System (HEC-RAS) [Hydrologic Engineering Center 2016a, Hydrologic Engineering Center 2016b], this model was selected as the hydraulic model of the proposed framework. The hydrologic model was selected based on three criteria (1) compatibility with the aforementioned hydraulic model; (2) freely available; and (3) widely used in practice. Based on these considerations, the U.S. Army Corps of Engineers' Hydrologic Modeling System (HEC-HMS) [Hydrologic Engineering Center 2017] was used as the hydrologic model. For determining the optimal schedules of flow releases in a multi-storage system, it is needed an optimization solver intended for large-scale constrained optimization problems. A widely used solver in water resources engineering (e.g., Yang et al. 2015, and Chen et al. 2016) that meets the above criteria is the Genetic Algorithm (GA). Because the proposed framework was implemented in MATLAB, and due to the availability of the Genetic Algorithm Toolbox within MATLAB, this toolbox is used herein. To minimize scripting for data exchange between HEC-HMS and HEC-RAS (e.g., manipulation of HEC-DSS files), the U.S. Army Corps of Engineers' HEC-DSSVue model, which has several built-in utilities for manipulating data in a HEC-DSS database, was used. The HEC-HMS, HEC-RAS, MATLAB GA Toolbox, and HEC-DSSVue model are briefly described below.

3.1. Hydrologic Modeling System (HEC-HMS)

HEC-HMS is intended to simulate the complete hydrologic processes of dendritic watershed systems and includes hydrologic analysis procedures such as event infiltration, unit hydrographs, and hydrologic routing (Hydrologic Engineering Center 2017). HEC-HMS is often used for studies of flow forecasting, future urbanization impact, reservoir spillway design, flood damage reduction, floodplain regulation, and systems operation (Hydrologic Engineering Center 2017).

3.2. Hydrologic Engineering Center's River Analysis System (HEC-RAS)

A highly adopted software in inundation modeling is the U.S. Army Corps of Engineers' Hydrologic Engineering Center's River Analysis System (HEC-RAS). HEC-RAS 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 in steady gradually varied flow conditions; (2) unsteady flow simulation, which can simulate one-dimensional, two-dimensional and combined one/two-dimensional unsteady flow through an open channel network; (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 for performing riverine water quality analyses (Hydrologic Engineering Center 2016a, Hydrologic Engineering Center 2016b). Standard applications of this model include flood wave routing and flood inundation studies. In the present work, inundation is simulated using the unsteady one-dimensional (1D) HEC-RAS model, which solves the full 1D Saint-Venant equations using the four-point implicit finite difference scheme (Hydrologic Engineering Center 2016a)

3.3. The MATLAB Genetic Algorithm (GA) Toolbox

The optimization model uses the MATLAB Genetic Algorithm (GA) Toolbox (Chipperfield and Fleming 1995). The GA solves constrained and unconstrained optimization problems based on a natural selection process that mimics biological evolution (Chipperfield and Fleming 1995). The GA repeatedly changes a population of individual solutions. At each generation, the GA randomly selects individuals from the current population and uses them as parents to produce children for the next generation. After several generations, the population is expected to evolve toward an optimal solution. The GA is recommended to solve problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, nondifferentiable, stochastic, or highly non-linear (Chipperfield and Fleming 1995). For more details about the genetic algorithm and its application to water resources the reader is referred to Wardlaw and Sharif (1999), Leon and Kanashiro (2010), Leon et al. (2014), Lerma et al. (2015), Yang et al. (2015), and Chen et al. (2016).

3.4. HEC-DSSVue

HEC-DSSVue is a Java-based visual utilities software intended to plot, tabulate, and manipulate data in a HEC-DSS database file (Hydrologic Engineering Center 2009). HEC-DSSVue incorporates over sixty mathematical functions and provides several utility functions to enter data sets into a HEC-DSS database, rename data set names, copy data sets to other HEC-DSS database files, and delete data sets (Hydrologic Engineering Center 2009). More importantly for the present work, HEC-DSSVue incorporates the "Jython" standard scripting language, which allows to execute a sequence of steps in a text format from a "batch" process (Hydrologic Engineering Center 2009).

3.5. Objective Function

A single objective function f is considered in the optimization, which consists that during flood conditions, the flow must be conveyed slightly below the specified maximum water level at the cross-sections of interest (e.g., maximum water levels without producing flooding in urban areas). This objective is written as

$$f = C_u \sum_{i=1}^{\mathrm{U}} \sum_{j=1}^{\mathrm{V}} \delta_u (e_{ij} - E_i)^2 + C_l \sum_{i=1}^{\mathrm{U}} \sum_{j=1}^{\mathrm{V}} \delta_l (e_{ij} - {E_i}^*)^2 + C_w \sum_{i=1}^{\mathrm{W}} \delta_w (y_w - y_e)^2 \quad (1)$$

where the first term on the right side of Eq. (1) indicates the penalty function when the water surface level at the cross-section of interest exceeds the specified maximum water level constraint. The second term on the right side of Eq. (1) indicates the penalty function when the water surface level at the cross-section of interest is far below the specified maximum water level constraint. It is noted that there is no contradiction between the first and second term on the right side of Eq. (1). To drain a watershed as fast as possible without producing river inundation, the water level in the cross-section of interest should be near the specified maximum water level constraint, not exceeding this level and not far below. The third term on the right side of Eq. (1) indicates the penalty function when the water depth at a managed storage pond (y_w) is below the ecological water depth (y_e) specified for the storage pond. The ecological water depth can be defined as the minimum water depth in the storage pond to minimize the implications on the ecological functions of the storage pond. In Eq. (1), U, V and W are the number of cross sections of interest at which the water level constraint is checked, the number of decision variables during flood conditions, and the number of managed storage ponds, respectively. Also, e_{ij} is water surface elevation at cross-section of interest i for decision variable j (e.g., flow release at hour j), E_i is the specified maximum water level constraint at cross-section of interest i, and E_i^* is the water level located at some distance below E_i. The region between E^* and E can be seen as a buffer zone where the objective function is optimal and constant. In the present work E^* was set to be 0.9m (3ft) below E. Also, in Eq. (1), C_u, C_l and C_w are penalty constants imposed for violation of constraints when water level at the cross-section of interest exceeds E, water level at the cross-section of interest is below E^*, and water depth at a managed storage pond is below the ecological water depth specified for the storage pond, respectively. Finally, in Eq. (1), δ_u, δ_l and δ_w are given by

$$\delta_{u,v,w} = \begin{cases} 1 & \text{if respective constraint } (u,\ v \text{ or } w) \text{ is violated} \\ 0 & \text{if respective constraint } (u,\ v \text{ or } w) \text{ is satisfied} \end{cases}$$

4. The integrated model

The flow chart of the integrated model for forecasting optimal flow releases in a multi-storage system for flood control is presented in Fig. 1. In short, first, the schedule of outflows of the managed storage ponds, which are generated by the optimization model, are used by HEC-HMS to update the water levels in these ponds. Then, the outflows from HEC-HMS, which could be unmanaged flows (sub-basins without managed storage) or managed flows (sub-basins with managed storage), enter the streams in HEC-RAS model as lateral flows. Then, the HEC-RAS model simulates inundation at the watershed scale. For each simulation of inundation, the objective function is calculated using the water levels in the cross-sections of interest in HEC-RAS and the water depths in the managed storage ponds (see Eq. 1). The objective values are sent back to the optimization model and the process is repeated until the optimization stop criteria is satisfied. The version of the models used herein are: HEC-HMS 4.2.1, HEC-RAS 5.0.5, MATLAB R2018b and HEC-DSSVue 2.0.1. The scripts for running the aforementioned models and the links between these models are briefly described below.

4.1. Executing HEC-HMS from MATLAB

Listing 1 presents the main script for executing HEC-HMS from MATLAB. *ExecHMS* in Listing 1 specifies the location of the file that will be called by MATLAB (*Running_HMS.txt*) for executing HEC-HMS. The content of the file *Running_HMS.txt* is presented in Listing 2. *Run_HMS* in Listing 1 changes the entire line that starts with *OpenProject* in Listing 2 to the specified name given in *Temp* in Listing 1. The latter is necessary when running multiple instances of HEC-HMS such as when using GA optimization. Finally *dos*(*ExecHMS*) executes the HMS Model. If the run is successful, the status in Listing 1 will be 0, otherwise the status will be 1.

Listing 1. Script to execute HEC-HMS from MATLAB

```
ExecHMS = ['hec-hms.cmd -s ' home_dir '\utilitaries\Running_HMS.txt']
path_general = pwd; %This returns the working directory.
cd(path_general);
StrID = num2str(RAS_simul_ID);
InpDSS=[path_general '\RAS_HMS_folders\' General_Name_of_Project ...
    StrID '\' Folder_HMS ];
Temp = [ 'OpenProject ("' Name_of_HMS_project '", "' InpDSS '")'];
Run_HMS(InputRUN_HMS, OutputRUN_HMS, Temp);
cd(path_HMS_executable);  %Path of HMS executable
[status,cmdout] = dos(ExecHMS );
%Status = 0(successful run), 1(aborted). If aborted, don't run RAS
```

Listing 2. Content of file *Running_HMS.txt*

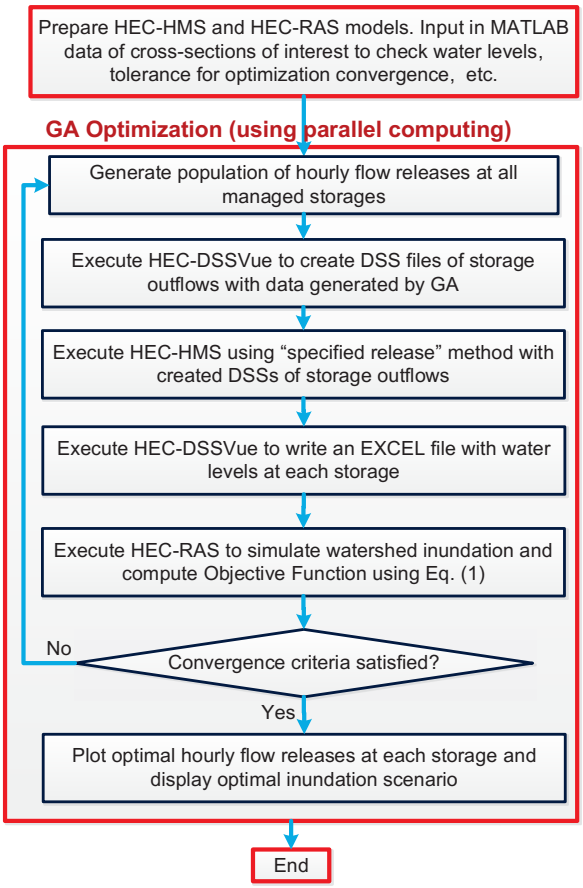

Figure 1. Flow chart of the integrated model for forecasting optimal flow releases in a multi-storage system for flood control

```
from hms.model.JythonHms import *
OpenProject ("CypressHMS", "C:\DSSTest_Wetland_Opt_11_20_2018
\RAS_HMS_folders\Cypress1\CypressHMS")
Compute ("Run 1")
Exit(1)
```

4.2. Executing HEC-RAS from MATLAB

To automate the HEC-RAS calculations and to perform parallel computations (simultaneous computations of HEC-RAS) in MATLAB, the scripts in Leon and Goodell (2016) are used and are not repeated here due to space limitations. The only difference is that the current work uses the last version of HEC-RAS (5.0.5) instead of 5.0 used in Leon and Goodell (2016). When using the version 5.0.5 of HEC-RAS, the windows registry key should be changed to actxserver('RAS505.HECRASCONTROLLER').

4.3. Executing HEC-DSSVue from MATLAB

Listing 3 presents the main script for executing HEC-DSSVue from MATLAB. The content of $ChangeDSS\_files$ in Listing 3 is presented in Listing 4. $ChangeDSS\_files$ is used to change the location and name of the storage outflows, and the schedule of outflows for each storage in $WriteDss.py$ (Listing 5). $TextTemp$ in Listing 3 indicates the new location and name of the storage outflow while as $FlowsTemp$ extracts the outflow data of the GA optimization ($x(RAS\_simul\_ID, posit\_temp3 : posit\_temp4)$ for each population (RAS_simul_ID) and each storage ($posit\_temp3 : posit\_temp4$). As observed in Listing 4, to replace the data of $TextTemp$ it is necessary to find the key variable starting with $myDss = HecDss.open$ while as for $FlowsTemp$, it is necessary to find the key variable starting with $flows$. The script in Listing 5 was adapted from the $WriteDss.py$ Jython script in Hydrologic Engineering Center (2009). Finally $system(exeTemp)$ in Listing 3 executes HEC-DSSVue with the information provided in $WriteDss.py$. If the run is successful, the status in Listing 3 will be 0, otherwise the status will be 1.

Listing 3. WriteDss.py: Script to execute HEC-DSSVue from MATLAB

```
cd(path_general);
InpWrite=[path_general '\utilitaries\WriteDss_temp_doNOTdelete.py'];
OutputWriteDSS = [path_general '\utilitaries\WriteDss.py'];
StrID = num2str(RAS_simul_ID);
InputDSS = [path_general '\RAS_HMS_folders\'  ...
    General_Name_of_Project StrID '\' Folder_HMS '\'  ...
    NameDSSFile_for_WetlandOutflows_with_active_control{kk} '.dss'];
TextTemp = [ '    myDss = HecDss.open("' InputDSS '")'];
posit_temp3 = Inflow_points_LATERAL_FROM_WETLANDS*(kk-1)+1;
posit_temp4 = Inflow_points_LATERAL_FROM_WETLANDS*kk;
flow_data = [0 x(RAS_simul_ID,posit_temp3:posit_temp4)];
%zero flow at time zero (gates are closed initially)
```

```
allOneString = sprintf('%.1f,' , flow_data);
allOneString = allOneString(1:end-1); % strip final comma
FlowsTemp = [ '    flows = [' allOneString ']'];
%FlowsTemp = '    flows = [10,100,50,76.6,67.5,97.0,93.8,65.1,73.5]
ChangeDSS_files(InpWrite, OutputWriteDSS, TextTemp, FlowsTemp);
cd(path_DSSVue_executable);
exeTemp = ['HEC-DSSVue.exe' path_general '\utilitaries\WriteDss.py'];
[status,cmdout] = system(exeTemp);
```

Listing 4. ChangeDSS_files.m: Replaces the data of *TextTemp* and *FlowsTemp*

```
function ChangeDSS_files(filenameinput, filenameoutput, TextTemp, ...
    FlowsTemp);
fid = fopen (filenameinput, 'rt'); %Open file for reading
if fid == -1
  error('Author:Function:OpenFile', 'Cannot open file: %s', ...
        filenameinput);
end
fout = fopen (filenameoutput, 'wt'); %Open file for writing
if fout == -1
  error('Author:Function:OpenFile', 'Cannot open file: %s', ...
    filenameoutput);
end
while ~feof(fid);
        strTextLine = fgetl(fid); %To read one additional line
        if strfind(strTextLine,'myDss = HecDss.open')
            fprintf(fout,'%s\n', TextTemp);
        elseif strfind(strTextLine,'flows = [');
            fprintf(fout,'%s\n',FlowsTemp);
        else
            fprintf(fout,'%s\n',strTextLine);
        end
end
fclose (fid); %Close the text file
fclose (fout); %Close the text file
```

Listing 5. WriteDss.py: Writes a DSS file (Adapted from WriteDss.py Jython script in Hydrologic Engineering Center 2009)

```
# name=WriteDss
# displayinmenu=true
# displaytouser=true
# displayinselector=true
from hec.script import *
from hec.heclib.dss import *
from hec.heclib.util import *
from hec.io import *
import java

try :
  try :
    myDss = HecDss.open("C:\CypressHMS\WetlandOutflow9.dss")
```

```
    tsc = TimeSeriesContainer()
    tsc.fullName = "/BASIN/LOC/FLOW//1HOUR/OBS/"
    start = HecTime("01Jan2018", "0000")
    tsc.interval = 60
    flows = [0.0,29.3,493.0,597.8,1112.2,27.0,453.5,495.5,492.7]
    times = []
    for value in flows :
      times.append(start.value())
      start.add(tsc.interval)
    tsc.times = times
    tsc.values = flows
    tsc.numberValues = len(flows)
    tsc.units = "CFS"
    tsc.type = "PER-AVER"
    myDss.put(tsc)

  except Exception, e :
    MessageBox.showError(' '.join(e.args), "Python Error")
  except java.lang.Exception, e :
    MessageBox.showError(e.getMessage(), "Error")
finally :
  myDss.close()
```

4.4. Executing Genetic algorithm optimization in MATLAB

Listing 6 presents the script for calling the MATLAB GA Optimization Toolbox. Because the computations are done in parallel ($'UseVectorized', true$), it is necessary to allocate the number of processors available for the optimization ($NumProcessors\_for\_parallel\_computing$). In Listing 6, *gaoptions* indicates the options used for the GA optimization including population size ($PopulationSize$), maximum number of generations ($MaxGenerations$), stall generations ($MaxStallGenerations$) and the function tolerance ($FunctionTolerance$). $MaxGenerations$, $MaxStallGenerations$ and $FunctionTolerance$ are all related to the stop criteria. The GA will stop if the maximum number of iterations or generations is attained. Also, the GA will stop if the average relative change in the best fitness function value over Stall generations ($MaxStallGenerations$) is less than or equal to Function tolerance ($FunctionTolerance$). In Listing 6, x is the best solution (decision variables), $fval$ is the value of the fitness function, $exitflag$ indicates the reason of why GA stopped, and LB and UB indicates the set of lower and upper bounds on the decision variables, respectively.

Listing 6. Script to execute GA Optimization in MATLAB

```
if Parallel_computing == 1
    delete(gcp('nocreate'));%To avoid interactive session error
    %Parallel computation
    parpool('local',NumProcessors_for_parallel_computing);
end

gaoptions = optimoptions('ga', 'UseVectorized',true,  ...
    'PopulationSize', Pop_Opt,'MaxGenerations', MaxGen_GA, ...
```

```
    'MaxStallGenerations',MaxStall, 'FunctionTolerance',10e-3);
gaoptions = optimoptions(gaoptions, 'PlotFcn', ...
{@gaplotbestf, @gaplotstopping, @gaplotbestindiv},'Display','iter');
%InitialPopulationMatrix is for assigning user-defined initial popul.
gaoptions.InitialPopulationMatrix = initial_population_GA_final;
[x fval, exitflag,output] = ga(@Fitness_vectorized,nvar, ...
    [],[],[],[],LB,UB,[],gaoptions);
```

4.5. Miscellaneous scripts

Besides the aforementioned scripts, few more important scripts that are necessary for assembling the integrated framework are briefly described in this section.

4.5.1. Checking water level in a storage at a given time

For estimating the third term on the right hand side of Eq. (1), it is necessary to know the time series of the water level in each storage. In particular, to verify if the storage is about to dry, it is necessary to know the storage water level right before the hydrograph starts to deliver water to the storage (e.g., lowest storage water level). A storage with zero or negative depth would violate the physical water depth and would lead to the abortion of the HEC-HMS program. To find the water level in a storage at a given time, HEC-DSSVue is used for writing the time series of storage water levels to an EXCEL file, from which are read by MATLAB. Lines 1 to 3 in Listing **??** write the Excel file through the Jython script *WetlandsDSSSaveExcel.py*, which content is shown in Listing 8. *Wet_data* (Line 9) in Listing **??** reads the Excel file, while as *WetElev_before_hydrog* access the desired data in the Excel file.

Listing 7. Script for finding storage water level at a given time

```
InputDSS = ['HEC-DSSVue.exe ' path_general '\RAS_HMS_folders\'  ...
    StrID '\' Folder_HMS  '\WetlandsDSSSaveExcel.py'];
[status,cmdout] = system(InputDSS);
%Read data of Wetland Characteristics
filenameinput= [path_general '\RAS_HMS_folders\' General_Project ...
    StrID '\' Folder_HMS '\wet_balance.xls'];
sheet = 'sheet1';
%Wet_data reads "numeric" data, string_wetlands reads "text" data
Wet_data= xlsread(filenameinput,sheet);
for jj = 1:NumberWetlands_managed
    %initial elevation starts from row 1
    WetElev_before_hydrog = ...
        Wet_data(Time_at_which_hydrograph_starts(jj), 2 + jj);
end
```

Listing 8. WetlandsDSSSaveExcel.py: Saves DSS data to an Excel file (Adapted from FolsomSaveExcel.py Jython script in Hydrologic Engineering Center 2009)

```
from hec.script import *
from hec.heclib.dss import *
from hec.dataTable import *
import java

#  Open the file and get the data
try:
  dssFile = HecDss.open("C:\Cypress1\CypressHMS\Run_1.dss")
  elev = dssFile.get("//WL-410/ELEVATION/01JAN2018/1HOUR/RUN:RUN 1/")
  inflow = dssFile.get("//WL-410/
    FLOW-COMBINE/01JAN2018/1HOUR/RUN:RUN 1/")
  stor = dssFile.get("//WL-410/
    STORAGE/01JAN2018/1HOUR/RUN:RUN 1/")
  totalOutflow = dssFile.get("//WL-410/
    FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
  releaseMain = dssFile.get("//WL-410-RELEASE/
    FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
  spillOvertopping = dssFile.get("//WL-410-SPILL-1/
    FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
except java.lang.Exception, e :
  #  Take care of any missing data or errors
   MessageBox.showError(e.getMessage(), "Error reading data")

#  Add Data
datasets = java.util.Vector()
datasets.add(elev)
datasets.add(inflow)
datasets.add(stor)
datasets.add(totalOutflow)
datasets.add(releaseMain)
datasets.add(spillOvertopping)

#  For this code, jython sees a List before a Vector
#list = java.awt.List()
list = []
list.append(datasets)

table = HecDataTableToExcel.newTable()

#  If you want to run Excel with a specific name and not a temp name:
# table.runExcel(list, "C:\Cypress1\CypressHMS\wet_balance.xls")
#  Or, if you would just rather create an Excel file, but not run it:
table.createExcelFile(list, "C:\Cypress1\CypressHMS\wet_balance.xls")
```

4.5.2. Plotting of optimal flows and other flow variables for each storage pond

Listing 9 presents the script for plotting the optimal schedule of outflows at each storage pond along other flow variables. The plot is done through HEC-DSSVue using the Jython script $Plot\_optim\_flow\_storage\_wetlands.py$ shown in Listing 10. In Listing 9, $Write\_Python\_for\_Plot\_Opt\_Wetl\_variables$ is used for changing the variables for each storage such as storage name ($WetlandName$), the period for the plot ($WritingTimingFromHMS$) and the location and name for saving the *jpg* figure. An example of the plot generated is shown in Figure 8. Due to space limitations, "Set the inflow and outflow colors" and "Set the

label text" in Listing 10 are included only for one of the flow variables ("Water Surface Elevation"). For the other variables, the user needs to make a copy and change the name of the variable.

Listing 9. Script for plotting optimal schedule of outflows at each storage along other flow variables

```
Input_temp = [path_general '\utilitaries\
    Plot_optim_flow_storage_wetlands_doNOTdelete.py'];
Output_temp = [path_general '\utilitaries\
    Plot_optim_flow_storage_wetlands.py'];

Text_Open_File = [path_general '\RAS_HMS_folders\'
    General_Name_of_Project StrID '\' ...
    Folder_HMS  '\Run_1.dss'];
Text_Open_File = [ 'dssFile = HecDss.open("' Text_Open_File '")'];

Text_Save_FileJPEG = [ 'plot.saveToJpeg("' path_general
    '\Optimal_results\'WetlandName{kk} '", 100)'];

ElevTemp = [ 'elev = dssFile.get("//'WetlandName{kk} ...
     '/ELEVATION' WritingTimingFromHMS '")'];

Total_outflow=['TotalOutflow = dssFile.get("//'WetlandName{kk} ...
    '/FLOW' WritingTimingFromHMS '")'];

InflowTemp = [ 'inflow = dssFile.get("//'WetlandName{kk} ...
    '/FLOW-COMBINE' WritingTimingFromHMS '")'];

StorTemp = [ 'stora = dssFile.get("//'WetlandName{kk} ...
    '/STORAGE' WritingTimingFromHMS '")'];

Opti_ReleaseTemp=['OptimRelease = dssFile.get("//'WetlandName{kk} ...
    '-RELEASE/FLOW' WritingTimingFromHMS '")'];

SpillFlowTemp=[ 'Spill = dssFile.get("//'WetlandName{kk} ...
     '-SPILL-1/FLOW' WritingTimingFromHMS '")'];

Name_Wetland_plot = [ 'plot.setPlotTitleText("Wetland ' ...
    WetlandName{kk} '")'];

%Write JPEG files
cd(path_general)
Write_Python_for_Plot_Opt_Wetl_variables(Input_temp, Output_temp,
    ElevTemp, Total_outflow, InflowTemp, StorTemp, Opti_ReleaseTemp,
    SpillFlowTemp, Name_Wetland_plot, Text_Open_File,
    Text_Save_FileJPEG);
cd(path_DSSVue_executable);
string_temp = ['HEC-DSSVue.exe ' path_general '\utilitaries\
    Plot_optim_flow_storage_wetlands.py'];
[status,cmdout] = system(string_temp);
```

Listing 10. *Plot_optim_flow_storage_wetlands.py*: Script for plotting the optimal schedule of outflows at each storage along other flow variables (Adapted from various Jython scripts in Hydrologic Engineering Center 2009)

```
from hec.script import *
from hec.script.Constants import TRUE, FALSE
from hec.heclib.dss import *
import java

#  Open the file and get the data
try:
    dssFile = HecDss.open("C:\DSSTest_Wetland_Opt_11_20_2018\
    RAS_HMS_folders \Cypress1\CypressHMS\Run_1.dss")
    elev = dssFile.get("//WL-420/
        ELEVATION/01JAN2018/1HOUR/RUN:RUN 1/")
    stora = dssFile.get("//WL-420/
        STORAGE/01JAN2018/1HOUR/RUN:RUN 1/")
    inflow = dssFile.get("//WL-420/
        FLOW-COMBINE/01JAN2018/1HOUR/RUN:RUN 1/")
    TotalOutflow = dssFile.get("//WL-420/
        FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
    OptimRelease = dssFile.get("//WL-420-RELEASE/
        FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
    Spill = dssFile.get("//WL-420-SPILL-1/
        FLOW/01JAN2018/1HOUR/RUN:RUN 1/")
except java.lang.Exception, e :
#  Take care of any missing data or errors
MessageBox.showError(e.getMessage(), "Error reading data")

#  Initialize the plot and set viewport size in precent
plot = Plot.newPlot("Wetland plots")
layout = Plot.newPlotLayout()
topView = layout.addViewport(10.)
middleView = layout.addViewport(10.)
bottomView = layout.addViewport(70.)

#  Add Data in specific viewports
topView.addCurve("Y1", elev)
middleView.addCurve("Y2", stora)
bottomView.addCurve("Y1", inflow)
bottomView.addCurve("Y1", TotalOutflow)
bottomView.addCurve("Y1", OptimRelease)
bottomView.addCurve("Y1", Spill)

panel = plot.getPlotpanel()
prop = panel.getProperties()
prop.setViewportSpaceSize(0)

#  Break our first rule - actually this creates the plot to change
plot.configurePlotLayout(layout)

panel = plot.getPlotpanel()
prop = panel.getProperties()
prop.setViewportSpaceSize(0)

#  Invert the precip and make pretty
# view0 = plot.getViewport(0)
# yaxis = view0.getAxis("Y1")
```

```
# yaxis.setReversed(FALSE)
# precipCurve = plot.getCurve(precip)

# precipCurve.setFillColor("blue")
# precipCurve.setFillType("Above")
# precipCurve.setLineVisible(FALSE)

#  Set the inflow and outflow colors
elevCurve = plot.getCurve(elev)
elevCurve.setLineColor("darkcyan")
elevCurve.setLineWidth(3.)

# Set the label text
label = plot.getLegendLabel(elev)
label.setText( "Water Surface Elevation")
label.setFont("Arial Black")
label.setFontSize(24)

#  Set the plot title
plot.setPlotTitleText("Wetland WL-420")
tit = plot.getPlotTitle()
tit.setFont("Arial Black")
tit.setFontSize(18)
plot.setPlotTitleVisible(TRUE)

plot.setSize(1500,1200)
plot.showPlot()

#  Now that it is complete, save to a .jpg and close it
plot.saveToJpeg("C:\DSSTest_Wetland_Opt_11_20_2018\
    Optimal_results\WL-420", 100)
plot.close()
```

5. Brief overview of a case study: Application of framework to an eight pond system in the Cypress Creek watershed, Houston, TX

It is noted that the purpose of this paper is not to describe a case study but instead to present a series of MATLAB scripts for forecasting hourly optimal flow releases in a multi-storage system for flood mitigation. The optimization period considered in this case study is 11 days, which means that 264 optimal flows need to be determined for each managed storage pond. Solely with the objective of illustrating the application of the MATLAB framework, this section presents a brief overview of a case study using the operation of a hypothetical eight pond system in the Cypress Creek watershed, which is located in Houston, Texas. For an in-dept discussion of this case study, the reader is referred to Tang et al. (2019a). The total area of the Cypress Creek watershed is $8.33 \text{x} 10^8$ m^2. The Cypress Creek watershed is located in northern Houston, within the Harris County Flood Control District. Figure 2 depicts the geographical location of this watershed. Cypress Creek watershed has a drainage area of about 267 sq. miles and it experiences about two to three flooding events per year on average (HGAC, 2016). The Cypress Creek watershed experienced devastating floods

during Hurricane Harvey in August 2017. The major stream in the watershed, Cypress Creek, originates from northwest of the watershed, takes a north-south course first, and then changes course to a west-east direction. There are several tributaries along the way, of which the largest tributary is named Little Cypress Creek. The upper half of the Cypress Creek watershed is mostly agricultural area, and the downstream of the watershed is mainly urban area. The upper half of the watershed was historically covered by wetlands and rice farms, and as a result there are a multitude of existing levees which can be easily repaired to restore the function of wetlands.

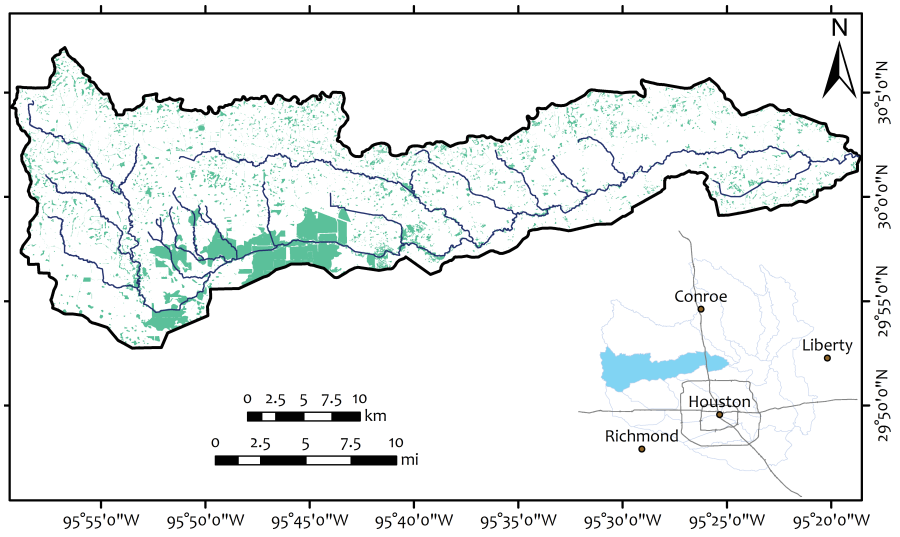

Figure 2. Geographical location of Cypress Creek watershed, TX

The hydrologic model of the Cypress Creek watershed was created in HEC-HMS. The details of the model construction, calibration and validation are discussed in Tang et al. (2019b). The HEC-HMS model of the Cypress Creek watershed was divided into 23 sub-basins, as shown in Fig. 3. As shown in this figure, the watershed was divided into three portions, upstream, midstream and downstream with total areas of $2.55\text{x}10^8$, $2.88\text{x}10^8$, and $2.90\text{x}10^8$ m^2, respectively. To help in flood mitigation, a hypothetical eight storage pond system (W300, W310, W330, W380, W390, W400, W410, and W420) with a total area ranging from 0.7 to 6.9% of the total watershed area is placed in the midstream portion of the watershed, which are displayed as yellow clouds in Fig. 3. The reason to implement storage ponds in midstream is that most of the natural wetlands and abandoned rice farms are located within this region. For user-defined water releases in HEC-HMS, the "Outflow Structures" method is selected and then for the "Release" option select *yes*. Then, specify the name of the "Gage Release" time-series data. Finally, the filename and pathname of the DSS file containing the gage release data is specified. The optimization model will up-

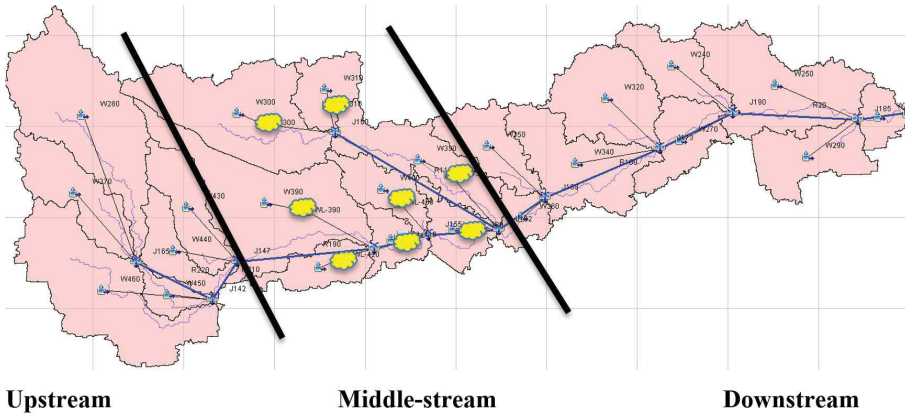

Figure 3. Screenshot of the HEC-HMS model for Cypress Creek watershed, TX along with schematics of eight hypothetical storage ponds in midstream, displayed as yellow clouds

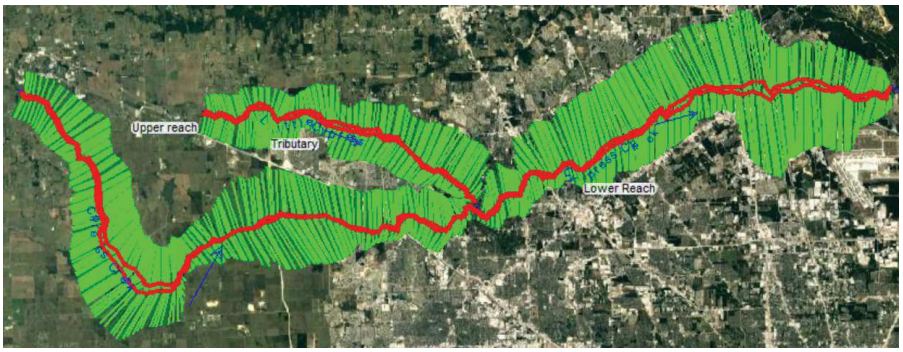

Figure 4. Screenshot of the geometry plan view of the HEC-RAS model for the Cypress Creek watershed, TX

date the gage release data for each generation of the optimization until the stop criteria is satisfied. The schedules of storage outflows of the last generation of the optimization are the optimal results. For simulating the overflows over the pond berms, the user should define the number of "Spillways" in the "Reservoir" (e.g., storage pond) module of HEC-HMS. One can simply choose one spillway and define the elevation, length and discharge coefficient for the "broad-crested spillway", which would represent a storage pond berm. The flows overtopping the spillway are labeled as "spill flow" in the MATLAB framework.

The HEC-RAS model of the major streams of Cypress Creek watershed was built using the HEC-GeoRAS tool within ArcGIS. The plan and profile views of the constructed HEC-RAS model are presented in Figs. 4 and 5, respectively. The HEC-RAS model is used to simulate inundation in the watershed. The inflow data for the HEC-RAS model is provided by the HEC-HMS model. The

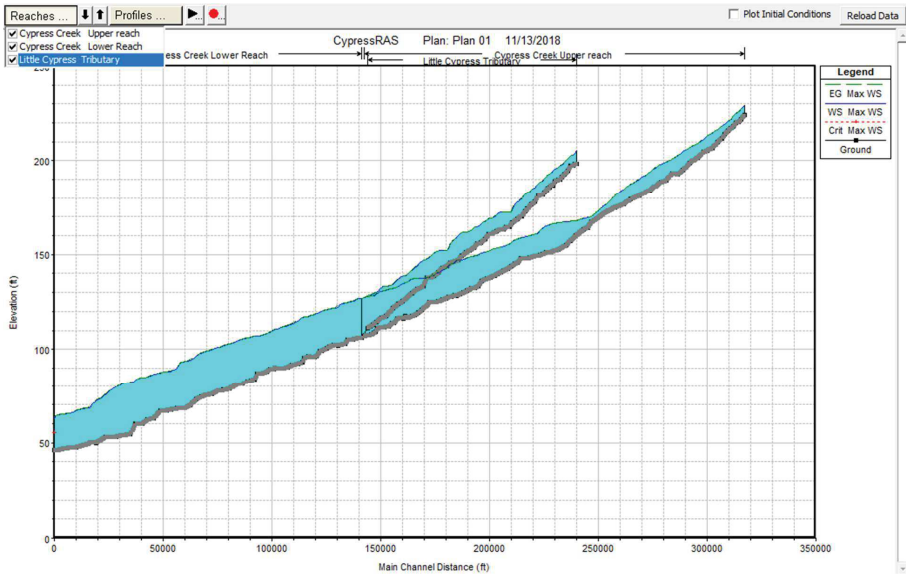

Figure 5. Screenshot of a profile view of the HEC-RAS model for the Cypress Creek watershed, TX

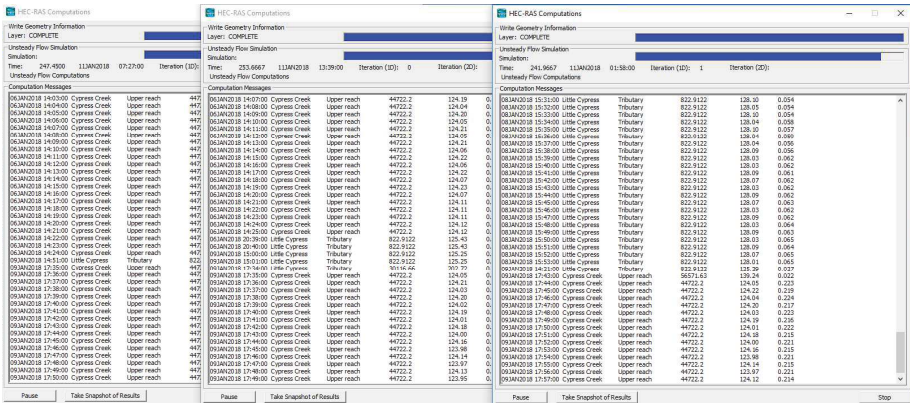

Figure 6. Screenshot of typical parallel computations when using HEC-RAS

total outflows from each managed storage pond (optimization derived outflows and spill flows) along the unmanaged flows enter the main streams in HEC-RAS as lateral flows. Thus, in this case study, only eight lateral flows change in HEC-RAS at every generation during the optimization. To speed up the computations, all HEC-RAS simulations are performed in parallel. A screenshot of typical HEC-RAS parallel computations is shown in Fig. 6. Herein we have used 18 available processors in the 8th Generation Intel Core i7-8700 (18 parallel computations), however for better display, Fig. 6 shows only three parallel simulations.

As shown in Fig. 1, once the HEC-HMS and HEC-RAS models are constructed and validated, the GA optimization generates the schedule of outflows for the managed storage ponds (eight in this case). Then, these flows are used by HEC-HMS to update the water levels in the eight storage ponds. Next, the outflows from HEC-HMS (unmanaged and managed) enter the streams in HEC-RAS as lateral flows to simulate inundation at the watershed scale. Next, the optimization model calculates the objective function according to Eq. (1) to determine the new population of schedules of flow releases at the eight ponds. This procedure is repeated until the optimization stopping criteria is satisfied. Tests using a population of 72 and a tolerance of 0.01 cfs required about 40-50 generations to converge. The screenshot of a typical convergence process of the optimization is shown in Fig. 7. After the optimization stop criteria is satisfied, the plots for the optimal schedule of outflows at all storage ponds are automatically generated. Each plot includes the time trace of the water surface elevation, storage, total inflow, spill flow and total outflow. A typical graph produced for one managed storage pond is shown in Fig. 8. As shown in Fig. 8, the optimization tends to release part of the water from the storage pond ahead of the inflow hydrograph to provide extra water storage during a heavy storm event. In the present exercise, the minimum ecological water depth was set to 0.5 ft and the model tried to keep the water level in the storage pond above this value. In an actual application, the model can be run every few hours to update the optimal schedule of outflows according to the new precipitation forecasts and updated water levels in the streams and storage ponds, if available. As mentioned earlier, in this case study 18 processors in the 8th Generation Intel Core i7-8700 was used, which requires between 1.5 to 2 hours to complete the optimization.

Fig. 9 shows downstream inundation area with and without dynamic storage management for eight storage ponds with a total combined area ranging from 0.7 to 6.9% of the total watershed area. The results in Fig. 9 indicate that when the combined storage pond area is below around 1.4% of the total watershed area, there is no visible impact from dynamic storage management. The storage ponds fill up quickly at the beginning of the rainfall event and there is no much room for dynamic storage management. When the percentage of combined storage pond area is above 1.4% of the total watershed area, dynamic storage management can significantly decrease the downstream inundation area. For example, a percentage of combined pond area of 2.1% with dynamic storage management achieves almost the same inundation as a percentage of combined pond area of

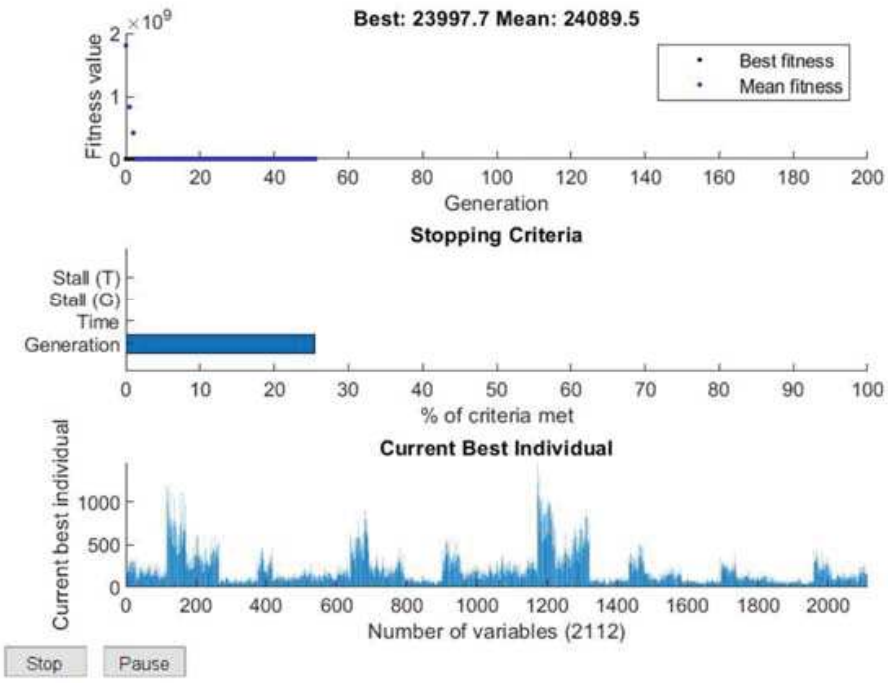

Figure 7. Screenshot of a typical convergence process for optimal schedule of storage outflows

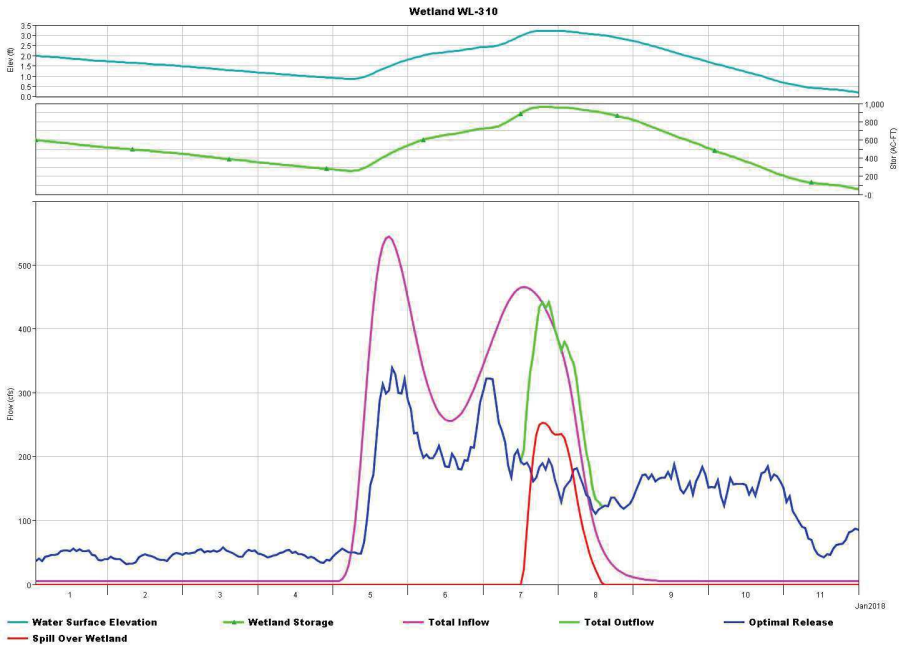

Figure 8. Typical graph produced for one managed wetland using the plotting scripts

3.5% without dynamic storage management. The latter indicates a reduction in 40% of the total combined area of storage ponds by using dynamic storage management. The results also show that inundation can be eliminated when the percentage of combined pond area is 4.8% with dynamic storage management and 6.2% without dynamic storage management.

Fig. 10 shows the time trace of inundation depth at the Kenchester Park in the Cypress Creek watershed with and without dynamic storage management for a percentage of combined pond area of 3.5%. The results in Fig. 10 show that the maximum inundation depth and inundation period decreased about 35% with storage management. In practice, land available for flood control is often limited and the results above show that dynamic storage management could play an important role in flood mitigation.

6. Conclusions

This paper presents a MATLAB framework for forecasting optimal flow releases in a multi-storage system for flood control. This framework combines four models namely, HEC-HMS, HEC-RAS, the MATLAB Genetic Algorithm (GA) Toolbox, and HEC-DSSVue. This paper focuses on presenting a set of MATLAB scripts for interfacing the aforementioned four software. The scripts are illustrated using the operation of a hypothetical eight pond system in the

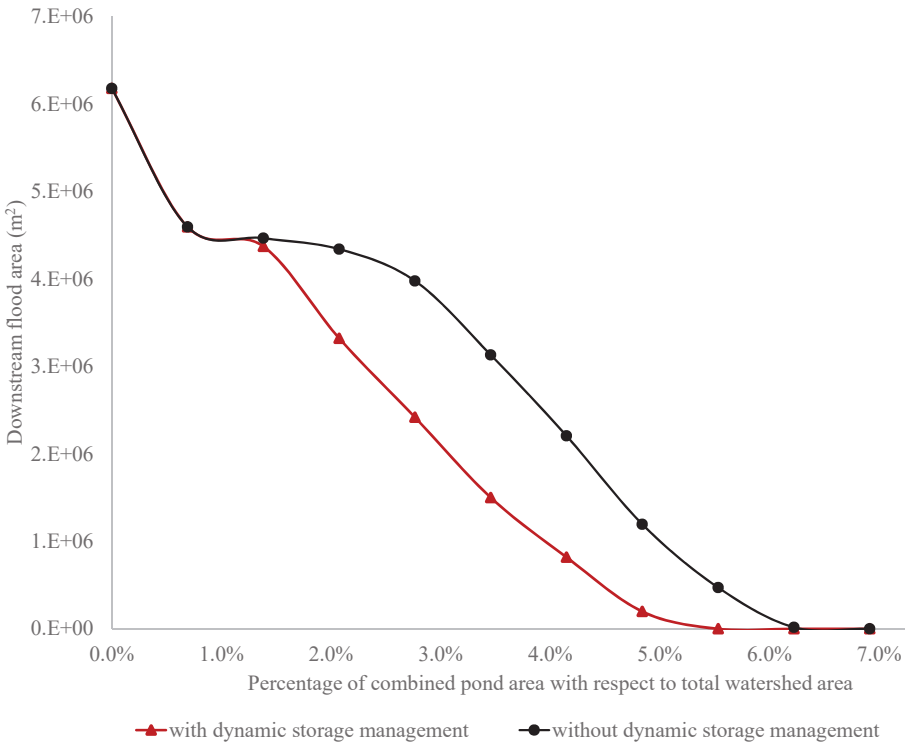

Figure 9. Inundation area with and without dynamic storage management for different percentages of combined pond area

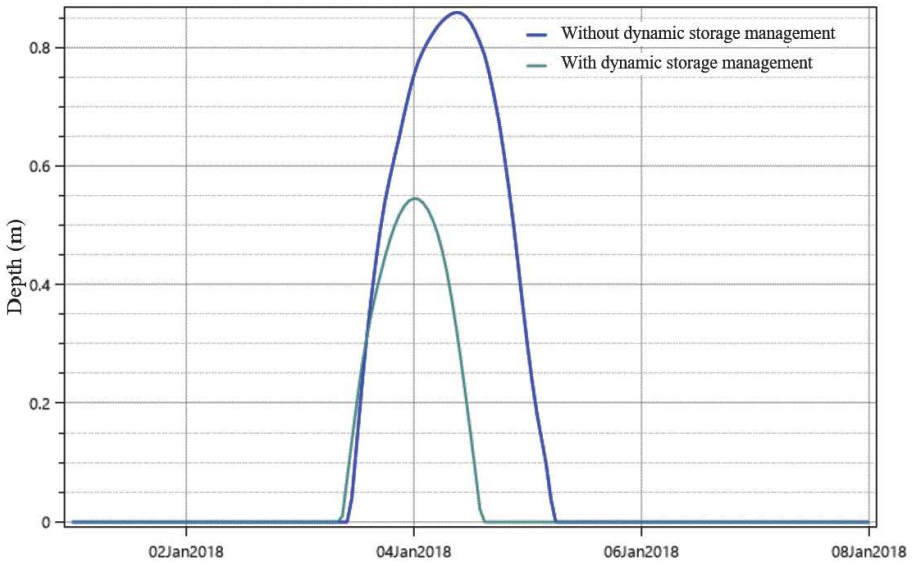

Figure 10. Time trace of inundation depth at the Kenchester Park in the Cypress Creek watershed with and without dynamic storage management for a percentage of combined pond area of 3.5%

Cypress Creek in Houston, Texas. The results of the case study indicate that dynamic storage management can help to mitigate floods. For instance, in the present case study, the total combined area of shallow storage ponds (maximum pond height 0.9 m) required for producing a given flood mitigation can be reduced in up to 40% by dynamic storage management. It is clear that these results are case dependent and cannot be generalized.

Acknowledgments

The first author was partially supported by the National Science Foundation through NSF/ENG/CBET under Award No. 1805417 and through NSF/DBI/BIO under Award No. 1820778.

References

Chen, D., Leon, A.S., Gibson, N.L., Hosseini, P., 2016. Dimension reduction of decision variables for multireservoir operation: A spectral optimization model. Water Resources Research 52, 36–51.

Chipperfield, A.J., Fleming, P.J., 1995. The matlab genetic algorithm toolbox, in: IEE Colloquium on Applied Control Techniques Using MATLAB, pp. 10/1–10/4. doi:10.1049/ic:19950061.

Ehrenfeld, J.G., 2004. The expression of multiple functions in urban forested wetlands. Wetlands 24, 719–733.

Flotemersch, J.E., Leibowitz, S.G., Hill, R.A., Stoddard, J.L., Thoms, M.C., Tharme, R.E., 2016. A watershed integrity definition and assessment approach to support strategic management of watersheds. River Research and Applications 32, 1654–1671. URL: http://dx.doi.org/10.1002/rra.2978, doi:10.1002/rra.2978. rRA-15-0132.R1.

Hydrologic Engineering Center, 2009. HEC-DSSVue User's Manual - Version 2.0. U.S. Army Corps of Engineers, Davis, California.

Hydrologic Engineering Center, 2016a. HEC-RAS, River Analysis System, 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, California.

Hydrologic Engineering Center, 2017. Hydrologic Modeling System (HEC-HMS) User's Manual - Version 4.2.1. U.S. Army Corps of Engineers, Davis, California.

Kusler, J., 2004. Multi-objective wetland restoration in watershed contexts. Technical Report. Association of State Wetland Managers. Berne, NY. URL: https://www.aswm.org/pdf_lib/restoration.pdf.

Leon, A.S., Goodell, C., 2016. Controlling hec-ras using matlab. Environmental Modelling and Software 84, 339 – 348. URL: http://www.sciencedirect.com/science/article/pii/S1364815216302730, doi:https://doi.org/10.1016/j.envsoft.2016.06.026.

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 Watershed Management Under Land Use and Climate Change, ASCE-EWRI, Madison, Wisconsin, USA. pp. 213–224.

Leon, A.S., Kanashiro, E., Valverde, R., Sridhar, V., 2014. Dynamic framework 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.

Leon, A.S., Tang, Y., Chen, D., Yolcu, A., Glennie, C., Pennings, S.C., 2018. Dynamic management of water storage for flood control in a wetland system: A case study in texas. Water 10. URL: http://www.mdpi.com/2073-4441/10/3/325, doi:10.3390/w10030325.

Leon, A.S., Verma, V., 2019. Towards Smart and Green Flood Control: Remote and Optimal Operation of Control Structures in a Network of Storage Systems for Mitigating Floods. pp. 177–189. URL: https://ascelibrary.org/doi/abs/10.1061/9780784482339.019, doi:10.1061/9780784482339.019, arXiv:https://ascelibrary.org/doi/pdf/10.1061/9780784482339.019.

Lerma, N., Paredes-Arquiola, J., Andreu, J., Solera, A., Sechi, G.M., 2015. Assessment of evolutionary algorithms for optimal operating rules design in real water resource systems. Environmental Modelling and Software 69, 425 – 436.

Mathworks, 2015. MATLAB version 8.6.0.267246 (R2015b). The Mathworks, Inc. Natick, Massachusetts.

NASA, 2017. Earth Observatory: How Will Global Warming Change Earth? URL: https://earthobservatory.nasa.gov/Features/GlobalWarming/page6.php.

NOAA, 2016. U.S. Climate Resilience Toolkit: Inland Flooding. URL: https://toolkit.climate.gov/topics/coastal-flood-risk/inland-flooding.

Snodgrass, J.W., Komoroski, M.J., Bryan, A.L., Burger, J., 2000. Relationships among isolated wetland size, hydroperiod, and amphibian species richness: Implications for wetland regulations. Conservation Biology 14, 414–419. URL: http://dx.doi.org/10.1046/j.1523-1739.2000.99161.x, doi:10.1046/j.1523-1739.2000.99161.x.

Tang, Y., Leon, A.S., Kavvas, M.L., 2019a. Impact of dynamic management of wetland storage on watershed-scale flood control. Water Resources Management Under review.

Tang, Y., Leon, A.S., Kavvas, M.L., 2019b. Impact of the size and location of wetlands on watershed-scale flood control. Water Resources Management Under review.

Tarr, T.L., Baber, M.J., Babbitt, K.J., 2005. Macroinvertebrate community structure across a wetland hydroperiod gradient in southern New Hampshire, USA. Wetlands Ecology and Management 13, 321–334. URL: http://dx.doi.org/10.1007/s11273-004-7525-6, doi:10.1007/s11273-004-7525-6.

USGS, 2003. USGS Fact Sheet FS-076-03:Effects of Urban Development on Floods. URL: https://pubs.usgs.gov/fs/fs07603/pdf/fs07603.pdf.

Wardlaw, R., Sharif, M., 1999. Evaluation of genetic algorithms for optimal 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 and Software 69, 262 – 279.