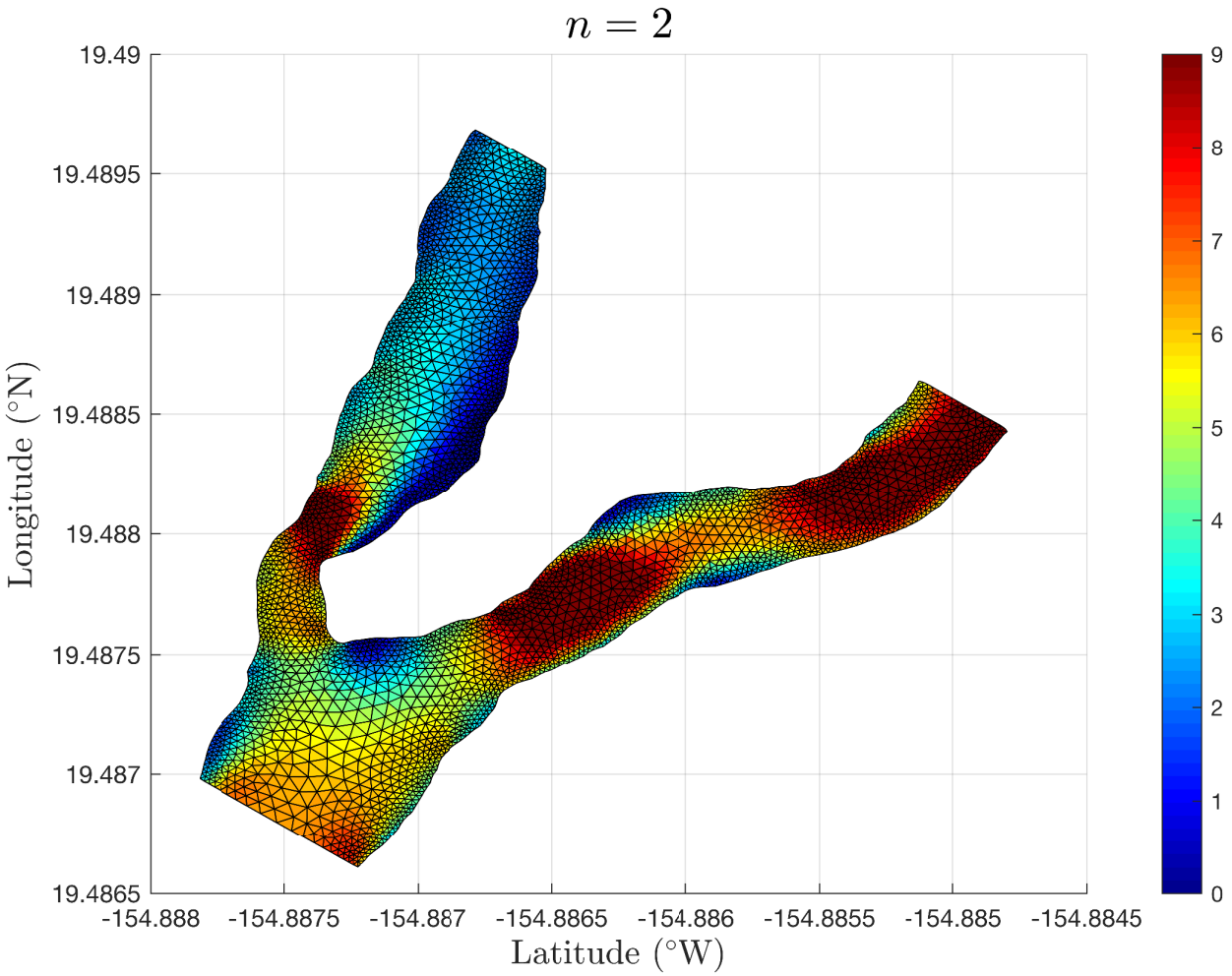

A discontinuous Galerkin finite element model for fast channelized lava flows v1.0

Colton J. Conroy

Lamont-Doherty Earth Observatory
Columbia University in the City of New York, NY

Roy M. Huffington Department of Earth Sciences
Southern Methodist University, Dallas, TX

August 2020

1 Introduction

DG Lava Flow 2D is a computer model that simulates lava flow dynamics in complex channel systems using conservation principles of mass, momentum, and energy. See the accompanying manuscript for details concerning the physics, math, and numerical methods used by the model.

2 Installation and requirements

The pre-processing, processing, and post-processing computer code used by DG Lava Flow 2D as well as all of the data used in the accompanying manuscript can be obtained at,

`https://github.com/coltonjconroy/DG_2d_lava_flows`.

A Fortran compiler is necessary to compile and run the model. We recommend Intel's Fortran compiler *ifort*, which allows for optimization of the code and can be obtained for free if the user is an educator or student at,

`https://software.intel.com/content/www/us/en/develop/tools/compilers/fortran-compilers.html`,

or if the user is an open-source contributor at,

`https://software.intel.com/content/www/us/en/develop/tools/parallel-studio-xe/choose-download/open-source-contributor.html`.

If none of these designations apply, then one can obtain the GNU Fortran compiler (*gfortan*) at,

`https://gcc.gnu.org/wiki/GFortranBinaries`.

If the user is not familiar with the Fortran programming language we recommend that they study the tutorials at,

`https://www.fortran.com/the-fortran-company-homepage/fortran-tutorials/`.

The pre-processing and visualization routines for DG Lava Flow 2D are written in Matlab, which unfortunately is not free (many research institutions have Matlab licenses available to their researchers and/or students). A free 30 day trial can be obtained at,

`https://www.mathworks.com/campaigns/products/trials.highResolutionDisplay.html?prodcode=ML`,

or if the user wishes to purchase Matlab it can be obtained at,

`https://www.mathworks.com/pricing-licensing.html?prodcode=ML&intendeduse=home`.

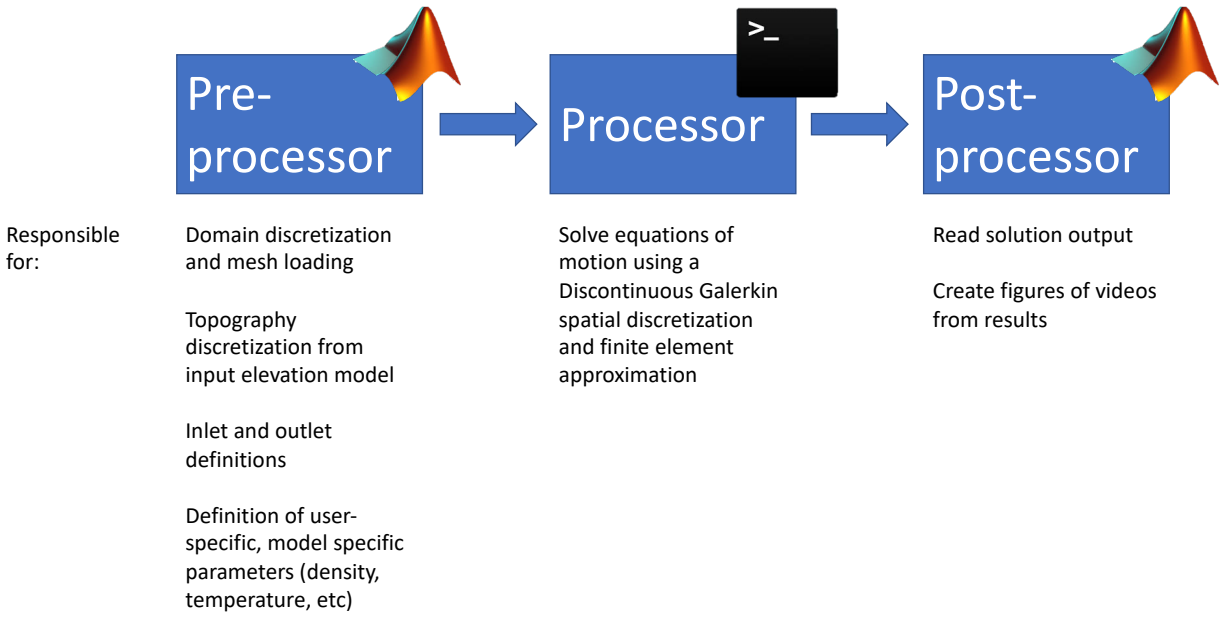

Figure 1: Tasks of the pre-processor, processor, and post-processor (credit: Einat Lev).

3 Software Content

When the user downloads the DG Lava Flow 2D package from github, there will be three different folders containing the source code titled pre-processor, processor, and post-processor (see Figure 1 for an overview of the tasks each module performs). Each folder should contain the following routines,

Pre-processor:

- write_input_parameter.m - the script the user can use to set the input parameters for a model simulation. It can be noted that the user must execute write_input_parameter.m each time they want to vary a parameter for a given model simulation.
- write_input_file.m - subroutine in write_input_parameter.m that writes the parameter input file.
- depth_int_preprocess.m - main code for pre-processing routine that creates data input files pertaining to the domain discretization (or mesh), numerical discretization, and initial conditions. Typically, the user only needs to run this routine once to set-up all of the input files for a particular application.
- DG_meshData.m - subroutine that creates element, edge, and node connectivity info including boundary edge tagging.
- plot_mesh_normals.m - subroutine that allows the user to view the element edge normal vectors to check for inconsistencies.

- create_xyz_scatter_hawaii.m - subroutine that calculates slope forcing using the centerline elevation of the channel (in this case it is specifically set-up for the y-split channel modeled in the manuscript).
- inlet_speed_from_video.m - subroutine that interpolates lava flow inlet speeds (provided by the user) to the inlet edges of the model.
- hawaii_dem_process.m - subroutine that calculates slope forcing from a digital elevation map.
- adjust_topo.m - subroutine that allows the user to perturb the topography of the digital elevation map.
- dg_basis_and_matricies.m - subroutine that evaluates the basis functions at the numerical integration points and sets up the matricies for the discontinuous Galerkin spatial discretization. Calls the following subroutines:
  - DG_discretize2D.m
    - triangle_quad.m
    - triangle_basis.m
    - gauss_jacobi_quad.m
- write_dg_input_v2.m - subroutine that creates the input file containing the basis functions evaluated at the integration points and DG matricies used in the spatial discretization.
- write_mesh_input.m - subroutine that creates the input file pertaining to domain discretization data, which includes the element, edge, and node connectivity as well as boundary data.
- write_initial_condition.m - subroutine that creates the input file that contains the initial conditions for the lava flow.

Processor:

- DI_lava_main.F90 - main processor.
- read_dg_input.F90 - subroutine that reads in the basis functions evaluated at the integration points as well as the DG matricies used in the spatial discretization.
- read_mesh_input.F90 - subroutine that reads in the domain descritization data.
- read_input_parameters.F90 - subroutine that reads in the user input parameters.
- rk_coefficients.F90 - subroutine that determines the coefficients for the explicit Runge-Kutta time stepper.
- read_initial_condition.F90 - subroutine that allocates and initializes the main flow variables as well as reads in the lava flow initial conditions.

- rhs_dg_thermo.F90 - subroutine that performs all of the element, flux, and boundary calculations of the DG spatial discretization.
- video.F90 - subroutine that writes output files at a given frequency which can be used to create a video of the simulation using the post-processing subroutine.
- global_output.F90 - subroutine that writes global output of the flow field variables when the simulation terminates. This could be due simply to the simulation ending or because the numerical solution begins to behave erratically.

Post-processor:

- lava_DI_post.m - main post-processing routine. Interpolates the discontinuous solution to the triangular mesh nodes for visualization via a weighted average. The weights are defined as the ratio of the area of an individual element to the sum of the areas of the elements contributing to an individual node.
- read_dof.m - subroutine that reads in the degrees of freedom (i.e., the basis function coefficients) of the DG approximation of the lava field at time t.
- calc_error.m - subroutine that calculates model errors.
- plot_lava_field.m - subroutine that plots model results.
- video.m - subroutine that creates a video of the model simulation.

4 Input Parameters

Model input parameters controlled by the user (and defined in the routine write_input_parameter.m) include,

- wall_bc_type - wall boundary condition type (0 == no slip, 1 == no normal flow).
- rho - lava density (kg/m^3).
- hint - steady lava flow thickness (m).
- n - viscosity power law exponent. ($n > 1$ == shear thickening, $n < 1$ == shear thinning, and $n = 1$ == Newtonian fluid)
- A, B, and C - viscosity model coefficient for temperature dependence, see the manuscript for details.
- tau_yield - yield strength of the lava (Pa).
- T_int - inlet temperature (degree K).
- T_wall - wall tempearture (degree K).
- T_crust - ground temperature (degree K).

- T_air - air temperature (degree K).
- em - emissivity constant.
- kt - thermal conductivity of basal boundary.
- vid - parameter that turns video output on or off (0 == no output, 1 == output).
- vid_frame_rate - output frequency for video output.
- limiter - parameter that turns the slope limiter on or off (0 == no limiting, 1 == limiting). The slope limiter is only used for polynomials of degree p where $p > 0$ and only if specified by the user. It can be noted that for relatively "smooth" flows no slope limiter is needed. This is an active area of research.

5 Compiling and executing the model

DG Lava Flow 2D consists of three distinct computer modules. We recommend that the user utilize all three computer modules in their investigation so that the processor functions properly, however, it is not strictly necessary for the user to utilize the pre-processor or post-processor supplied with DG Lava Flow 2D. The user can use their own programs to visualize output and create input files as long as they are of the format of the files dg_input.txt, mesh_input.txt, and DI_lava_ic.txt (see the pre-processor source code for complete formatting details). We recommend that the first time user compile and run DG Lava Flow 2D using the input files supplied with the source code. (The supplied input files correspond to those used in the Kilauea simulations in the accompanying manuscript.) Therefore, the first time user can skip to Section 5.2 which covers how to compile and execute the main processor.

5.1 Pre-processor

There are three distinct purposes of the pre-processor, which are,

1. *Domain discretization* - The processor solves the system of equations described in the accompanying manuscript equation (15) over triangular elements. The user must supply the domain triangulation and the format is as follows,

   Mesh name
   number of elements, number of nodes
   node number, x-coordinate, y-coordinate, elevation
   element number, number of neighboring elements, [neighboring element numbers]

   See hawaii_site8_velboundary_max8_min1.14 for an example. If the user does not have access to a triangular mesh generator, there is one freely available at,

   `https://github.com/coltonjconroy/ADMESH`,

which was used to create the triangular mesh in the accompanying manuscript.

The pre-processor uses the mesh information to calculate the element, edge, and node connectivity as well as calculates the area of each element, the normal vector of each edge, and the Jacobian of transformation that transforms the integrals in equation (15) from physical space to a master element space. (The processor evaluates the integrals in equation (15) over a master element (see the reference Kubatko et al. 2006 in the accompanying manuscript for details). This allows the DG spatial matricies to be pre-computed.)

2. *Discontinuous Galerkin spatial discretization and finite element approximation*- The processor performs the integral calculations of equation (15) using numerical integration. Because the basis coefficients of a given element only change with respect to time, we are able to pre-compute the integrals of the basis functions in equation (15) over a master element. The pre-processor performs these calculations in the routine dg_basis_and_matricies.m. The values are stored in the matricies $\mathbb{A}$ (the area integral), $\mathbb{B}$ (the boundary integral), and $\mathbb{C}$ (the source integral). It also creates the vectors $\boldsymbol{\phi}$ (basis evaluated at area integral points), $\boldsymbol{\phi}_b$ (basis evaluated at boundary integral points), and $\boldsymbol{\phi}_c$ (basis evaluated at source integration points). To obtain the numerical solution evaluated at the source integration points, say, for the momentum, the user would simply perform the matrix multiplication $(\boldsymbol{\phi}_c)(\mathbf{hu})$. To fully evaluate the integrals in equation (15), the processor matrix multiplies the matrix $\mathbb{A}$ with the flux function evaluated at the area integration points and then adds this to the matrix $\mathbb{B}$ multiplied with the flux function evaluated at the edge integration points and the matrix $\mathbb{S}$ multiplied with the source term evaluated at the area integration points.

3. *Topography discretization* - Calculates the slope of a digital elevation map supplied by the user "file_name.xyz" and interpolates these values to the mesh nodes of the domain discretization. The processor uses a nodal basis function, ψ, to interpolate the nodal slope values to the numerical integration points that are used to evaluate the integrals in equation (15).

   It should be noted that in the pre-processor routine there are a number of points that need to be specified as input from the mesh, these include, an (x,y) list of coordinates that correspond to line segments that delineate inlet and outlet portions of the channel (there can be multiple inlets and outlets). In the Kilauea example, we have,

   inlet_pt1.Position(1) = x-coordinate of the first point of the first inlet line segment,

   inlet_pt1.Position(2) = y-coordinate of the first point of the first inlet line segment,

   inlet_pt2.Position(1) = x-coordinate of the second point of the first inlet line segment,

   inlet_pt2.Position(2) = y-coordinate of the second point of the first inlet line segment,

L_outlet_pt1.Position(1) = x-coordinate of the first point of the left outlet line segment,

L_outlet_pt1.Position(2) = y-coordinate of the first point of the left outlet line segment,

L_outlet_pt2.Position(1) = x-coordinate of the second point of the left outlet line segment,

L_outlet_pt2.Position(2) = y-coordinate of the second point of the left outlet line segment,

R_outlet_pt1.Position(1) = x-coordinate of the first point of the right outlet line segment,

R_outlet_pt1.Position(2) = y-coordinate of the first point of the right outlet line segment,

R_outlet_pt2.Position(1) = x-coordinate of the second point of the right outlet line segment,

R_outlet_pt2.Position(2) = y-coordinate of the second point of the right outlet line segment,

The user can obtain the coordinates of these points manually by plotting up their mesh using the trimesh function in Matlab and then by clicking on an inlet/outlet point and right clicking to select "extract data point." Matlab will prompt the user to name the point they are extracting, which for the Kilauea example corresponds to inlet_pt1 etc. Matlab then stores the x and y values of the coordinate in .Position(1) and .Position(2). Once all of the relevant points have been extracted to the workspace, they should be saved in "file_name.mat" for later use. In the Kilauea example, this corresponds to the file hawaii_site8_fine_bpts.mat.

If the user wishes to use the centerline slope for the entire width of the channel or if they wish to perturb different portions of the channel's DEM then the user must also specify a list of points that delineate the different branches of the channel and save these points in the "file_name.mat" file used for the inlet/outlet points. This allows the user to build in their own logical constructs in terms of which channel branch DEM they would like to perturb etc. See the source code provided in, create_xyz_scatter_hawaii.m and adjust_topo.m, for an example.

To run the pre-processor, the user must open the main pre-processing routine depth_int_preprocess.m and fill in the user specified variables,

set_up = 'mesh' or 'pts'
mesh_type = 'new' or 'old'
edge_normal_fig = 'on' or 'off'
new_dem = 'on' or 'off'
modify_topo = 'on' or 'off'
new_dem_adjust = 'on' or 'off'
p = degree of polynomial approximation of the DG basis
hint = steady thickness value (m)
hvel = speed used to calculate initial condition (m/s)
mux = x-viscosity used to calculate initial condition (Pa $\cdot$ secs)
muy = y-viscosity used to calculate initial condition (Pa $\cdot$ secs)
rho = lava density (kg $\cdot$ m^{-3})
vel_int = average inlet speed (m $\cdot$ s^{-1})
v_amp = factor to perturb inlet velocity
save_mesh_data = 'on' or 'off' (this should always be "on" because the data file it saves is required by the post-processor).

The user must also specify the mesh file name in line 50 of the code (the mesh file should be in the pre-processor folder), which is currently set to,

hawaii_site8_velboundary_max8_min1.14.

The user executes the pre-processor by running *depth_int_preprocess* via the Matlab command line or by clicking the "run" icon in the editor window with the depth_int_preprocess.m file open in the Matlab editor window. The Pre-processor will output three files named dg_input.txt, mesh_input.txt, and DI_lava_ic.txt. Move all three of these files from the pre-processor folder to the processor folder. Finally, to complete the pre-processing step, execute *write_input_parameter* via the Matlab command line or by clicking the "run" icon in the Matlab editor window (with the write_input_parameter.m file open in the editor window). It can be noted that write_input_parameter.m is stored in the processor folder so that the user does not have to move the input file from the pre-processor folder to the processor folder each time they change a model parameter.

5.2 Processor

To compile the processor, the user must open a terminal and change directories to the folder that contains the processor source code (the input files and parameter file should be in this folder as well). The user must then enter in the prompt (if using the recommended intel compiler),

ifort -c DI_lava_main.F90

This will create the modules global.mod and precisions.mod. After compiling the main program the user must then enter the prompt,

ifort *.F90 -o lava_flow.out

which compiles and links all of the subroutines to the main program. Finally, the user executes the model by entering in the prompt,

./lava_flow.out

Upon execution, the program will ask the user to enter the total simulation time as well as the time step they wish to use (both specified in seconds). The program will then execute and display the cpu time to execute the model when the simulation has terminated. The corresponding output files will be written in the same folder that contains the executable. We recommend moving model output to separate folders (along with *all* the input files) after each simulation is complete. (The user only needs to compile the processor one time and then they can simply change the input files for different investigations).

5.3 Post-processor

The post-processor reads in the degrees of freedom (i.e., the basis coefficients of the DG approximation of thickness, momentum, heat content and etc.) and then plots the solution over the lava channel domain and/or creates a video of the lava flow dynamics. To run the post-processor and view the simulation results, the user needs to ensure that the output file "output.txt" is in the post-processing folder as well as the video files, which are labelled in the following fashion; file 0 == initial lava thickness, 1 == initial x-component of momentum, 2 == initial y-component of momentum, and 3 == initial lava heat content. Subsequent files will be numbered according to the user specified output frequency. For instance, if the user specified frequency is every 200 time steps, then the files will be numbered as 200, 201, 202, 203, 400, 401, 402, 403, and etc. Once all of the output files are moved to the post-processor folder, then the user must open lava_DI_post.m and set-up the user options in terms of what type of output they would like to visualize,

figs = 'on' or 'off'
errors = 'on' or 'off'
vid = 'on' or 'off'
v_field = 'on' or 'off'
vid_type = 'velocity' or 'thickness' or 'temperature'
vid_frame_rate = number displayed as output from terminal window
vid_file_name = 'video_file_name'
load mesh_info_file_name
nframes = number of frames for video displayed in terminal window
dt = time step used in simulation
Tmin = min temperature value displayed in figure
Tmax = max temperature value displayed in figure
Hmin = min thickness value displayed in figure
Hmax = max thickness value displayed in figure
Mmin = min viscosity value displayed in figure

Mmax = max viscosity value displayed in figure
Vmin = min speed value displayed in figure
Vmax = max speed value displayed in figure

To execute the post-processor, the user can either enter *lava_DI_post* into the Matlab prompt or click the "run" button in the editor window with the lava_DI_post.m file open. The user can then save the output by clicking the save button in the figure window. See the Matlab documentation for further details.