LISFLOOD-FP User Manual

Ben Phillips

November 6, 2016

Contents

1 Introduction **1**
1.1 Software/hardware requirements 1
1.2 Development and operation of LISFLOOD-FP 1
1.3 Manual contents 2

2 Setting up LISFLOOD-FP **3**
2.1 Generating a DEM (Digital Elevation Model) 3
2.2 Creating the boundary conditions (bci file) 6
2.3 Creating the boundary conditions (bdy files) 8
2.4 Creating the parameter files 13
2.5 Running the model 15

3 Processing the output **15**
3.1 Calculating resultant velocity 15
3.2 Calculating flood hazard 16

4 References **16**

1 Introduction

This manual provides some background information, detailed guidance on how to set up LISFLOOD-FP (particularly in relation to the methodology provided by McMillan et al. [2011]), and provides the codes necessary to do so. This document assumes that the reader is reasonably familiar with ARC GIS. Any questions email b.t.phillips@liverpool.ac.uk

1.1 Software/hardware requirements

- ARC GIS 10
- MATLAB
- Microsoft Excel
- Python - best to install a distribution (i.e. Anaconda)
- MATLAB
- Ideally a high performance computer if not running on livljobs. This will vary depending on the resolution and scale of your domain.

1.2 Development and operation of LISFLOOD-FP

LISFLOOD-FP is a two-dimensional raster based inundation model using a storage cell approach. It was originally developed by Bates and De Roo [2000] for fluvial applications only, before being extended to perform coastal inundation modelling [Bates et al., 2005]. LISFLOOD-FP was developed to try to minimize the amount of physics in these models to reduce computational cost. It is capable of producing good estimates of flood inundation extent, as shown in Bates et al. [2005]. The governing equations are also now used in commercial software for consultancy and insurance industries (e.g. JFLOW - JBA Ltd.).

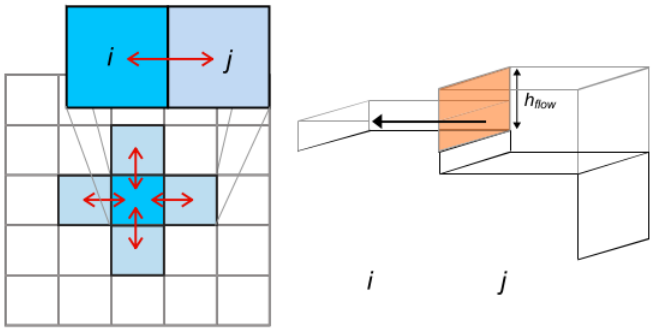

Figure 1: A graphical representation of floodplain cells in LISFLOOD-FP

LISFLOOD-FP essentially divides the floodplain up into cells (Figure 1.2). Here each cell within the grid is a storage area for which the mass balance is updated at each time step according to the fluxes of water into and out of each cell. Such models solve a continuity equation relating flow into a cell and its change in volume:

$$\frac{\Delta h}{\Delta t} = \frac{\Delta Q}{\Delta x \Delta y} \tag{1}$$

and a flux equation for each direction where flow between cells is calculated according to Mannings law (only the x direction is given here, Q_y is solved analogously):

$$Q_x^{i,j} = \frac{h_{flow}^{\frac{5}{3}}}{n} \left(\frac{h^{i-1j} - h^{ij}}{\Delta x} \right)^{0.5} \Delta y \tag{2}$$

where $h^{i,j}$ is the water free surface height [L] at the node (i, j), Δx and Δy are the cell dimensions [L], t is the time [T], n is the Mannings friction coefficient $[L^{\frac{1}{3}}T]$, and Q_x and Q_y describe the volumetric flow rates between floodplain cells $[L^3\ T^{-1}]$. The flow depth, h_{flow}, represents the depth through which water can flow between two cells, and is defined as the difference between the highest water free surface in the two cells and the highest bed elevation.

Please refer to Bates et al. [2010] for more detailed description of the governing physics and adaptive time stepping.

1.3 Manual contents

This document describes the basic set-up method of LISFLOOD-FP, as applied to coastal or combined coastal/fluvial inundation. This document provides MATLAB (or in some places Python) scripts and functions for the following, all of which are explained and presented further in this document.

Model set-up:

- Converting the domain to a start depth.
- Creating boundary conditions based on extreme water levels derived from Environment Agency methodology, in combination with river discharge (if applicable) [McMillan et al., 2011].
- Creating boundary condition points for extreme water levels derived from Environment Agency methodology [McMillan et al., 2011]

Further processing:

- Extraction of flood area and volume from the maximum water depth file
- Conversion of Vx and Vy velocity files to a resultant velocity (Vr)
- Conversion of resultant velocity and water depth to flood hazard using standard Environment Agency methodology Surendran et al. [2008].
- Calculation of economic cost using Penning-Roswell depth-damage curves.

Extreme water level boundary conditions are based on data and methodology provided by McMillan et al. [2011], although this manual does aim to provide the user with knowledge of how to implement extreme water levels based on various methodologies.

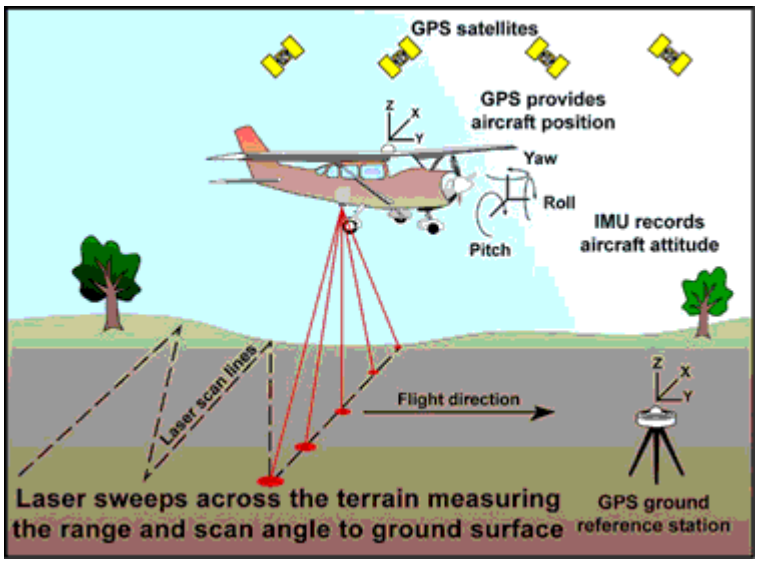

Figure 2: A schematic demonstrating airbourne LiDAR data collection.

2 Setting up LISFLOOD-FP

This section takes the user through how to set up a LISFLOOD-FP simulation for a coastal flooding application. There are additional files the model can use, but without the following it won't run The model requires the following to be able to run:

- A Digital Elevation Model for the area you want to model. See section 2.1
- A bdy (body) file. This contains the extreme water levels (and river discharge if applicable) that you want to put into the model.
- A bci file. This tells the model where and how to apply the extreme water levels (and river discharge if applicable).
- A start depth file. This tells the model which areas are already flooded (i.e. the Sea!) at the beginning of the simulation
- A parameter file. This tells the model which file does each of the above, as well as sets various other options.

2.1 Generating a DEM (Digital Elevation Model)

The DEM needs to encompass the area we want to model. The beauty of LISFLOOD-FP is that it can model very large areas at coarse resolution (e.g. from Holyhead in North Wales, to the Scottish border at 50 m resolution), or small scale at finer resolution (e.g. Fleetwood at 5 m). You will need to decide on the resolution based on the context of your research question, and the time you have available as resolution significantly impacts the computation time (the amount of time the model takes to run). Imagine you have a 100 m^2 grid and it is split into 100 10 m^2 grid cells. If you half the resolution from 10 m to 5 m instead of having 100 cells you now have 20 x 20 = 400 cells. So the model has to do four times as many calculations, because there is four times as mamy cells. Bear this in mind that for one simulation of the Fleetwood example used in this study, at 5 m resolution is approx. 24 hours.

To generate a DEM for the model, we use airbourne Light Detection and Ranging (LiDAR) data,as shown in Figure 2.1. This manual does not detail its method in detail, but beware that its standard error on the collected elevation data is approximately $\pm 0.15m$. Airbourne LiDAR surveys are often of poor quality, perhaps where the survey was taken during high spring tides or where the data has been inappropriately formatted as an integer value (essentially where the value is rounded to the nearest metre).

There are two types of DEMs, either of which potentially could be used with LISFLOOD-FP. The first is Digital Terrain Model (DTM) which is a survey representative of all topographic features, including buildings, cars and vegetation. These are probably more useful for high-resolution studies of flood hazard where the aim is to capture the gaps between buildings and stop unrealistic lateral propagation of floodwater which may not

occur, say on a densely built terraced street. They are certainly less useful for studies of economic damage, as this would require putting the heights of the building bases into the model - a very lengthy process.

The second is a Digital Surface Model (DSM) where algorithms are applied to remove buildings, cars, vegetation and other topographic features, to illustrate the elevation of the Earth's surface. This is also known as a 'bare earth' model. Assuming all you want to model is bare earth, this is perfectly appropriate. However, in most examples, and certainly here we want to add sea defences. This manual briefly explains how to add sea defences to a DSM using the best method available. DSMs are probably more appropriate for LISFLOOD-FP, as DTMs often cause water to become unrealistically trapped, which causes time step issues and eventually the model will crash. DSMs are much better for damage of economic analysis, although potentially underestimate hazard.

Data is available from the following links. Ensure you select 2 m resolution of either DSM or DTM, and choose **ASCII format**:

- England: Insert link here
- Wales: Insert link here

Once you have downloaded the data and extracted it to your desired folder:
If you are using ARC GIS

1. Open up ARC Catalog and create a file geodatabase
2. In this file geodatabase, create a **32 bit float** raster dataset. It is imperative you select 32 bit float otherwise your DEM will only be rounded to the nearest metre. A huge inaccuracy.
3. Load the LiDAR ASCIIs into this raster dataset, ensuring you keep 2 m resolution and EPSG:27700 British National Grid
4. Open ARC GIS, add the raster dataset you created and the LiDAR representing the areas you downloaded should be displayed.
5. When clipping the model domain, bear in mind the size of the area as this will impact on computation time. A good technique if you are not using river forcing is to clip the model to just below the high water mark (perhaps 3 cells of your desired resolution). This will significantly reduce computation time with no impact on results as the domain does not morphologically evolve during LISFLOOD-FP. If you are using river forcing, you will need to make the domain bigger, as the river water will not realistically drain during low tide, as it will be shut off at the model boundary and will laterally propagate. Using a polygon shapefile which covers the area you want to model, in the toolbox navigate to Spatial Analyst > Extraction > Extract by mask. Enter the files as necessary to crop your raster to your desired area.

If you struggle with ARC GIS or your computer is slow, do this bit in QGIS using the following

1. Having opened QGIS, Raster > Miscellaneous > Build virtual raster. Add the LiDAR files, or the folder if they are the only files in there. Click ok, making sure that the spatial reference is set to EPSG:27700 British National Grid
2. Open GDAL, navigate to the directory which your virtual raster is in. E.g. cd /users/benp/documents/fleetwood/lidar
3. Then enter gdal_translate -of AAIGrid input_filename.vrt output_filename.asc This builds a mosaicked raster of the LiDAR files.
4. Raster > Extraction > Clipper will let you clip the merged file to your desired area. Bear in mind the size of the area as this will impact on computation time. A good technique if you are not using river forcing is to clip the model to just below the high water mark (perhaps 3 cells of your desired resolution). This will significantly reduce computation time with no impact on results as the domain does not morphologically evolve during LISFLOOD-FP. If you are using river forcing, you will need to make the domain bigger, as the river water will not realistically drain during low tide, as it will be shut off at the model boundary and will laterally propagate.
5. Save this as a separate raster.

If you did the above in QGIS, move over to ARC now.

So you should now have the area you want to model at 2 m resolution. However, as discussed above this is likely to lead to significant computation times for even moderate domain sizes, due to a greater number of grid cells and smaller time steps. So we coarsen the resolution to account for this. However, as we coarsen the resolution the crest heights of the sea defences become smoothed (or indeed in a bare earth model may not be represented at all).

The best solution is heavily dependent on data availability of your particular study area. This is to use a shapefile which contains the location and crest heights of the defences already. Such shapefiles may be available from the Channel Coastal Observatory. Often these do not have the crest heights. Should this be the case, using ARC GIS:

1. Construct points along the line every 2 m.
2. Use Spatial Analyst > Extraction > Extract values to points. This will give you points which contain the crest heights at every 2 m along the line.
3. Now you have a shapefile containing defence crest heights (whether it is a point or polyline based shapefile), use ARC GIS toolbox Conversion > Point/polyline (whichever applicable) to Raster. **Enter your desired resolution**.
4. Now you need to resample your LiDAR data to your desired resolution. Using the toolbox, Data Management > Raster > Raster Processing > Resample. Keep x and y resolutions the same.
5. Once this is complete, using toolbox Data Management > Raster > Raster dataset > Mosaic to new raster. Enter both this raster of your defence crest points and your original raster. Number of bands keep as 1, 32 bit float raster, spatial reference British National Grid. Enter your desired resolution.

If shapefiles containing data do not exist, other methods include tracing the sea defence crest at 2 m resolution, and constructing points along the line you create. Alternatively download OS MasterMap topographic shapefiles, and try to isolate the lines which represent the sea defences. Once you have done this, same technique as above. Construct points, convert to raster and mosaic. The latter method is better, as tracing the defences it is very easy to miss spots and this allows floodwater to unrealistically outflank flood defences.

You should now have a raster at your desired resolution with accurate defence crest heights. Using the toolbox, Conversion > Raster to ASCII. You must save it as a **float** and a .asc file extension. For the purposes of this manual, this file will be referred to as **domain.asc**. The example of the Fleetwood domain published in Prime et al. [2015] is visible in Figure 3.

ARC ASCII files in this context contain georeferenced elevation data. It is essentially a massive text file with an elevation value for each grid cell. The georeferencing header is shown below.

```
ncols          1600
nrows         1600
xllcorner     318996.42958111
yllcorner     135014.42006843
cellsize      10
NODATA_value  -9999
```

where ncols and nrows are the number of columns and rows, xllcorner and yllcorner contain the x and y cartesian coordinates (in m) of the lower left hand corner of the file. Cellsize is the grid resolution, and NODATA_value specifies the null value

Now we need to generate a start depth file to tell the model which cells are already flooded at the beginning of the simulation. All we need to do is turn any positive values to zero, and turn any negative values (i.e. the bathymetry) to represent a depth (i.e. to positive numbers). This is easily done with the simple Python function below.

```
def start_depth(infile,output):
    """Reads in ARC ASCII LISFLOOD-FP domain and generates and ASCII file of start-depth"""
    data = np.loadtxt(infile,skiprows=6)
    line1 = linecache.getline(infile,1)
    line2 = linecache.getline(infile,2)
```

Figure 3: Fleetwood DSM at 5 m resolution

```
6       line3 = linecache.getline(infile,3)
7       line4 = linecache.getline(infile,4)
8       line5 = linecache.getline(infile,5)
9       line6 = linecache.getline(infile,6)
10      header = line1 + line2 + line3 + line4 + line5 + line6 #Concatenate header
11      data = data*-1
12      data[data < 0] = 0
13      data[data == -0] = 0
14      data[data == 9999] = -9999
15      np.savetxt(output,data,header=header,fmt='1.2%f')
16      return
17
18  #For example
19  start_depth('rhyl_10m.asc','rhyl_10m_start_depth.asc')
```

2.2 Creating the boundary conditions (bci file)

Recall from the introduction to this section that the BCI file tells the model where and how to read the applied extreme water levels (which we will generate in the next step).

In the provided folder entitled Coastal DesignExtreme Sea Levels, open up CFB_Extreme_Sea_Levels.shp in ARC GIS, **in the same document as your domain**. This shapefile contains extreme water levels for the UK at 2 km intervals for various return periods, from 1:1 yr to 1:10,000 yr events. These values are for still water levels only, i.e. they do not contain waves. For full details about the methodology of how these values are produced, the reader is referred to McMillan et al. [2011].

1. The first step here is to construct the points at which we want the model to fill with water. So create a line two to three cells inside the model boundary, and construct points where the resolution of the domain is the distance between each one. So for a 10 m grid you should have points every 10 metres. This is very important otherwise the model will think the resolution is fine and will become computationally prohibitive.

2. So you should have two point shapefiles, which are referred to here as "big dots" (the 2 km interval points based on the provided McMillan shapefile) and "little dots" (the points of your model boundary).

3. Using Data Management > Fields > Add XY Coordinate, ensure both shapefiles have fields with their X & Y coordinates. Save the attribute tables of both files, ensuring you only have the "big dots" which are inside your model domain.

The big dots file should look like this:

```
  292148.5524    380491.6208 563
294019.0096 380699.9378 564
295860.6982 381461.4452 565
297651.5921 382336.864  566
299369.0979 383307.0723 567
301138.8781 384179.7678 568
303075.9238 384615.6586 569
304903.9752 385395.8434 570
306782.5925 386071.9054 571
308720.0082 386509.1308 572
310651.5239 386949.4636 573
```

And the little dots file should look like this (bear in mind there will probably be a lot more points:

```
  316235.45928955   388691.81707764 1
316226.38549805 388687.61389160 2
316217.31170654 388683.41052246 3
316208.23809814 388679.20727539 4
316199.16430664 388675.00408936 5
316190.09051514 388670.80072021 6
316181.01690674 388666.59747314 7
316171.94311523 388662.39428711 8
316162.86950684 388658.19091797 9
316153.79571533 388653.98767090 10
316144.72192383 388649.78448486 11
316135.64831543 388645.58111572 12
316126.57452393 388641.37792969 13
316117.50067139 388637.17468262 14
316108.42712402 388632.97131348 15
316099.35327148 388628.76812744 16
316090.27947998 388624.56488037 17
316081.20587158 388620.36151123 18
316072.13208008 388616.15832520 19
```

The boundary condition file is laid out as follows in a tab delimited (i.e. a tab seperates each value) file: Column

1. Boundary identifier: N/E/S/W/P. North/south/east/west refers to the model boundary where perhaps a uniform river discharge or tidal level is acting. P refers to point discharge, which is what we use with McMillan et al. [2011] boundary conditions.

2. Start of boundary segment in map co-ordinates (i.e. British National Grid)

3. End of boundary segment in map-coordinates (i.e. British National Grid)

4. Boundary condition type: Refer to Bates et al. [2013] for the full list, the only ones we will be working with is HVAR (Time varying water level) and QVAR (time varying discharge). We use HVAR for McMillan extreme water levels and QVAR for river discharge.

5. Boundary condition identifier. This needs to match what we put later on in the body file, however for the McMillan extreme water levels this is automated in the code.

The Python function shown below creates the bci file. All you need is the big dots and little dots files, and the starting number of the boundary condition points. You will need to add these manually (see line 25 of the code for how to execute this function). Open up the file manually and add an extra line should you wish to add river forcing.

```
def bci(littledots,bigdots,bcifile,startbci):
    """Assigns each grid cell the nearest McMillan et al. (2011) node"""
    print('Assigns each grid cell the nearest McMillan et al. (2011) node')
    LD = np.loadtxt(littledots,delimiter=' ')
    L = len(LD) #Gets the length of the file
```

```
    BD = np.loadtxt(bigdots,delimiter='\t')
    print('Performing query')
    distance, index = spatial.KDTree(BD).query(LD)
    bci = open(bcifile,'w')
    j = 0

    for i in range(1,L):
        k = index[j]
        LD[i,2] = startbci + k #Assigns the correct bci number
        j = j+1
        x = LD[i,0] #Gets x,y,z from LD array
        y = LD[i,1]
        z = LD[i,2] #BCI number
        bci.write('%s\t%f\t%f\t%s\t%s%3.0f\r\n' % ('P',x,y,'HVAR','bc',z))
        print('P',x,y,'HVAR','bc',z)
    bci.close()
    return

#For example
bci('littledots_10m.asc','big_dots.asc','output_name.bci',563)
```

Your bci file will look like this (again bear in mind there will be a lot more points, and ignore the line spacing):

```
  P      296830.593689   381779.391113   HVAR    bc565

P   296821.571289   381775.078918   HVAR    bc565

P   296812.548889   381770.766724   HVAR    bc565

P   296803.526306   381766.454529   HVAR    bc564

P   296794.503906   381762.142273   HVAR    bc564

P   296785.481506   381757.830078   HVAR    bc564

P   296776.458923   381753.517883   HVAR    bc564

P   293190.966125   380465.884521   HVAR    bc563

P   293181.028076   380464.772522   HVAR    bc563

P   293171.090088   380463.660706   HVAR    bc563

P   293161.152100   380462.548706   HVAR    bc563

P   293151.214111   380461.436707   HVAR    bc563

P   293141.276123   380460.324890   HVAR    bc563

P   293131.338074   380459.212891   HVAR    bc563

P   293121.400085   380458.100891   HVAR    bc563

P   293111.462097   380456.989075   HVAR    bc563

P   293101.524109   380455.877075   HVAR    bc563

P   293091.586121   380454.765076   HVAR    bc563

P   293081.648071   380453.653076   HVAR    bc562

P   293071.710083   380452.541077   HVAR    bc562

P   293061.772278   380451.429321   HVAR    bc562

P   293051.834290   380450.317322   HVAR    bc562
```

2.3 Creating the boundary conditions (bdy files)

The bdy files tell the model what extreme water levels to apply. This manual focuses on setting up boundary conditions based on extreme water levels derived from Environment Agency methodology. Refer to McMillan et al. [2011] for more detail on the method. The key points to bear in mind are, are:

- Extreme water levels are a function of the astronomical tide and meteorological effects (i.e. storm surge), there is no wave action. They could also be referred to as still water levels.
- They should be considered accurate to 1 d.p.
- Annual exceedance probability: Annual exceedence probabilities (AEP) describe the likelihood of being exceeded in any given year. AEPs can also be expressed as chance. For instance, an AEP of one per cent has a chance of being exceeded of one in 100 in any given year. In coastal design this is often termed return period.

Opening up the attributes of the "big dot" shapefile in ARC GIS, you will see the extreme water levels based on the various return periods for each of the chainage points. Select the chainage points in your domain, and export as a table. Make sure it is formatted with the chainage identifier in the left hand most column, with the extreme water levels across the various return periods to the right of it. Ensure there is no headers. Name the file return-periods.csv. You will see in the Coastal DesignExtreme Sea Levels folder a spreadsheet called Design Surge Shapes. Open the Locations tab and decide which is the most suitable donor site. Let's take Llandudno as an example as the manual uses it to create bdyfiles for North Wales, from Abersarn to the Point of Ayr. Move over to the Donor Surge Shapes tab on the bottom, and find Llandudno (number 28).

In the table there are 99 hr design storm surge shapes for 40 locations across the UK. This gives us a methodology for calculating time-varying water elevations representative of storms of various return periods The values represent ratios of the surge residual (the maximum difference between the astronomical tide and the total extreme water level). 0 represents no surge, and the water level is a function of the astronomical tide alone. 1 represents that the surge residual is at a maximum, 0.5 that the surge residual is at half its maximum etc. Therefore the surge component varies depending on your underlying tide and the selected EWL. Copy the surge ratios for Llandudno or your chosen donor site into a new spreadsheet.

So we've got the surge shape, and we can select the extreme water levels from the exported "big dot" shapefile, but we need the underlying astronomical tide to create the time-varying water levels. To do this, use the National Tidal and Sea Level Facility website to find the highest predicted tide (astronomical only) for your selected donor site. Make a note of the date. Using POLTIPS provided by NOC Marine Products and Data Team, generate a 15 minute interval tidal record for your donor site a few days either side of the highest predicted astronomical tide and export as a text file.

This is where it gets slightly tricky:

1. Load the text file with the tides into Microsoft Excel along with the exported donor surge profile.
2. Drag the surge profile so that its central 1 value lines up with the highest astronomical tide.
3. Select the 99 hr of tidal data adjacent to the surge curve and export. Name the file tidal-curve.asc
4. Save the surge curve into a seperate file. Name the file surge-curve.csv

Let's think about some of the uncertainty of this method. The overwhelming one that stands out is that the chances of the highest astronomical tide coinciding with the maximum surge is very slim. Indeed the chance of this occuring decreases with tidal range, and peak surge tends to occur on the rising tide Horsburgh and Wilson [2007]. McMillan et al. [2011] strongly recommends performing sensitivity analysis of this tide-surge interaction. This manual notes the method used by Prime et al. [2015] where the surge was offset two hours either side of high tide to see if this made an impact. Again, this manual suggests you interpret this in the context of your own research question and the methodology you are applying.

Another rather obvious one is that this method excludes waves and morphological evolution through the storm. A better approach is detailed by **?**, where XBeach-G and joint probability boundary conditions are used. However, this is beyond the scope of what manual is aimed for.

Now we need to think about the sea level rise scenarios. This manual uses an example of 0 to 2 m in 0.1 m increments. Again, this depends on the context of your research question, and this manual does not aim to recommend appropriate values. Enter the sea level increments you wish to use (e.g. 0,0.1,0.2) in metres in one column in Excel and save as "sea-level-rise.csv".

Now we have all the files necessary to create the bdyfiles. Figures 4 to 7 visualise the method in order to improve understanding.

But before we create the bdyfiles for the storm, we need to get the domain filled up to the start of the storm bdy file. So let's assume that the first water level in our storm bdyfile is 1 m ODN. We need to ensure that

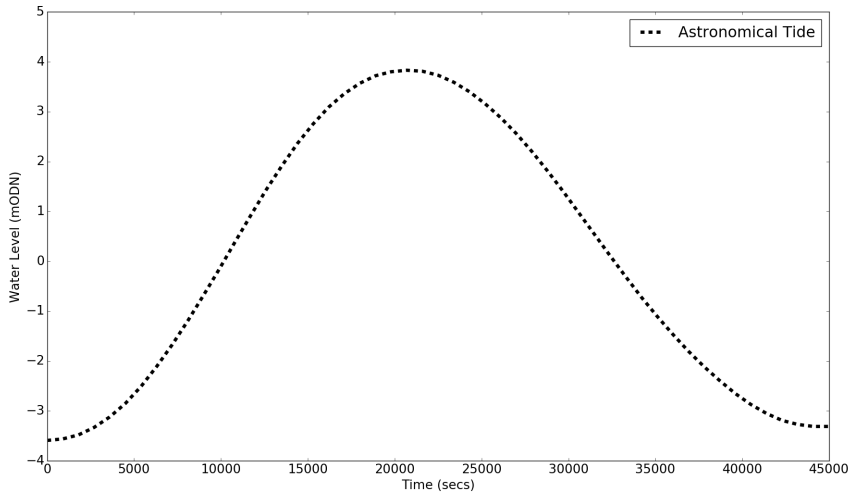

Figure 4: The astronomical tide (Stage 1)

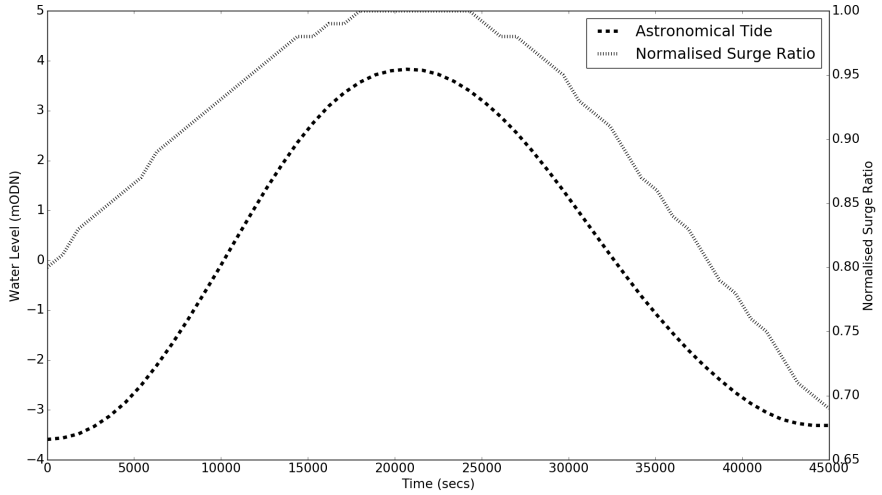

Figure 5: The astronomical tide + normalised surge ratio (Stage 2)

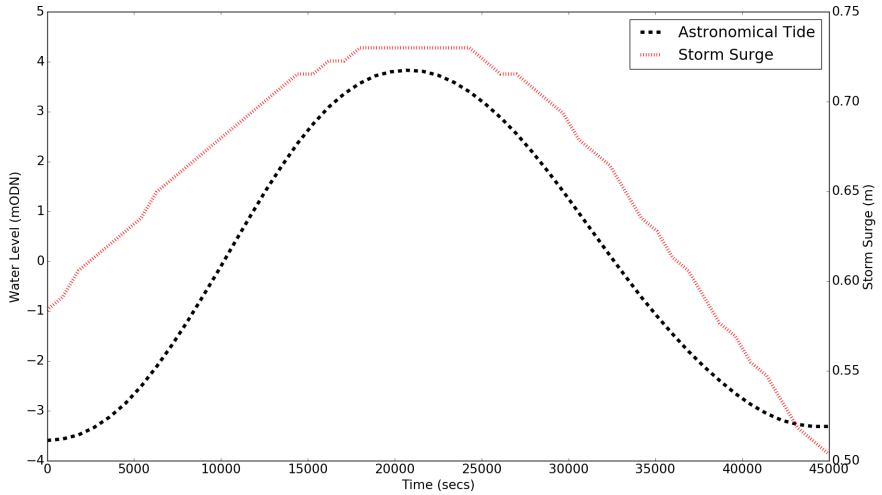

Figure 6: The storm surge (Stage 3)

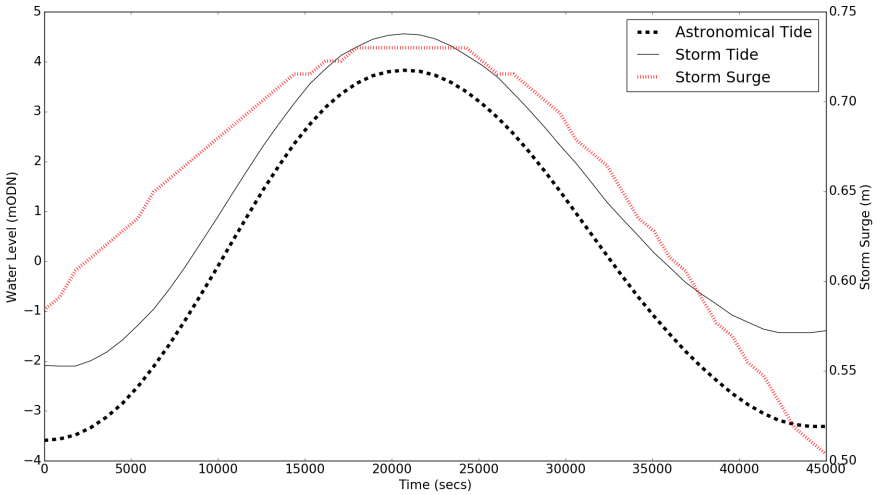

Figure 7: The storm tide (Stage 4)

the model acts on a domain where areas lower than this are flooded. So all we do is slowly fill the domain by incrementing up the water level. The code beneath does this in 100 steps. So using our example of 1 m ODN, it would fill 1 cm every 15 minutes.

The start depth for the start depth simulation is simply the file we created using the start_depth Python function, which simply tells us where the sea if full of water. We need a start depth each time the first value in the bdyfile changes. As there is zero surge at the beginning of the simulation, we only need a new start depth for each sea level increment (or river discharge scenario). Shown below is a Python script to generate start bdyfiles. This will need amending to the appropriate filenames. Note the code below is designed for extreme water level forcing only.

```
Tide = np.loadtxt('Tide_Curve_Llandudno.csv')
SeaLevelRise = np.loadtxt('SLR_NW.csv')
Return = np.loadtxt('return_periods.csv',delimiter=',')
time = np.linspace(0,89100,100)
m = 1;

for k in range(0,20):
        a = 'dee_start' + str(m) + '.bdy'
        bdy = open(a,'w')
        bdy.write('%s\t%1.0f\r\n' % ('BDY file Dee No',m))
        for i in range(0,11):
            EWL = Tide[0]
            SLR = SeaLevelRise[k]
            Start = EWL + SLR
            START = np.linspace(0,Start,100)
            Result = np.column_stack([START,time])
            bc = Return[i,0]
            bdy = open(a,'a')
            bdy.write(('%s%3.0f\r\n') % ('bc',bc))
            bdy.write(('%3s\t%4s\r\n') % ('100','seconds'))
            for line in Result:
                line.tofile(bdy,sep='\t',format='%2.2f')
                bdy.write('\r\n')
        m = m+1
```

Now we have the start bdyfiles, we can create the storm bdyfiles. This will create 16 return periods over 21 sea level rise scenarios. $16 \times 21 = 336$ bdyfiles. When using the code below you will need to amend the filenames of the tide and surge curves, the return periods table, and the sea level rise file. Note the code writes out a useful log file, where the parameters of each bdyfile is recorded separately.

```
#Creates bdy files based on McMillan scenarios – 21 sea level and 16 return periods
import numpy as np

```

```
Tide = np.loadtxt('Tide_Curve_Llandudno.csv')
Surge = np.loadtxt('Surge_Curve_Llandudno.csv',delimiter=',')
Return = np.loadtxt('return_periods.csv',delimiter=',')
SeaLevelRise = np.loadtxt('SLR_NW.csv')
time = np.linspace(0,356400,397) #99 hr simulation at 15 mins temporal resolution
m = 1;

logfile = 'lisflood_bdy_log.txt'
log = open(logfile,'w')
log.write('%s\t%s\t%s\r\n' % ('EWL','SLR','BDY No'))
log.close()

for k in range(0,21):
    for j in range(1,17):
        a = 'dee' + str(m) + '.bdy'
        bdy = open(a,'w')
        bdy.write('%s\t%1.0f\r\n' % ('BDY file Dee No',m))
        for i in range(0,11):
            EWL = Return[i,j]
            SLR = SeaLevelRise[k]
            Skew = EWL - Tide[149] #Amend to index of high water
            Residual = Surge[:,1]*Skew
            Storm_tide = Tide + Residual + SLR
            bc = Return[i,0]
            Result = np.column_stack([Storm_tide,time])
            bdy = open(a,'a')
            bdy.write(('%s%3.0f\r\n') % ('bc',bc))
            bdy.write(('%3s\t%4s\r\n') % ('397','seconds'))
            for line in Result:
                line.tofile(bdy,sep='\t',format='%2.2f')
                bdy.write('\r\n')
        loga = open(logfile,'a')
        loga.write(('%2.2f\t%2.2f\t%s\r\n') % (EWL,SLR,a))
        loga.close()
        bdy.close()
        m = m+1
```

Note below the layout of the file with the header, followed by the length of the forcing (397 for the standard 99 hr McMillan simulation), the time unit (seconds) and the bci identifier (e.g. bc563 in the demonstrating bci file, but could be anything). Under this follows the time in seconds and the extreme water level (in m ODN).

```
  BDY   file Dee No  2

bc563

397 seconds

4.28    0.00

4.28    900.00

4.23    1800.00

4.12    2700.00

3.95    3600.00

3.72    4500.00

3.44    5400.00

3.12    6300.00

2.74    7200.00

2.33    8100.00
```

The next code is almost exactly the same but in addition to still water forcing, we also add river discharge. CF refers to catchment factor, which converts river discharge up-catchment to the river mouth Marsh and Sanderson [2003]. This method is very limited in that it will not capture discharge from downstream tributaries. Note with loops in Python:

for h in range(1,3) actually means stop before 3 (i.e. at 2) so if you have 5 river scenarios you are testing

you need to change the code to for h in range(1,6).

```
#Creates bdy files based on McMillan scenarios — 21 sea level and 16 return periods, 1 hydrograph
import numpy as np

Tide = np.loadtxt('Tide_Curve_Llandudno.csv')
Surge = np.loadtxt('Surge_Curve_Llandudno.csv',delimiter=',')
Return = np.loadtxt('return_periods.csv',delimiter=',')
SeaLevelRise = np.loadtxt('SLR_NW.csv')
time = np.linspace(0,356400,397) #99 hr simulation at 15 mins temporal resolution
m = 1;
CF = 4.13 #Catchment factor to convert river discharge to coast

logfile = 'lisflood_bdy_log.txt'
log = open(logfile,'w')
log.write('%s\t%s\t%s\t%s\r\n' % ('PQC','EWL','SLR','BDY No'))
log.close()

for h in range(1,3):
    rivfile = 'clwyd_hydro' + str(h) + '.csv'
    Riv = np.loadtxt(rivfile)
    River = Riv*CF
    River_result = np.column_stack([River,time])
    for k in range(0,21):
        for j in range(1,17):
            a = 'dee_clwyd' + str(m) + '.bdy'
            bdy = open(a,'w')
            bdy.write('%s\t%1.0f\r\n' % ('BDY file Dee with River Clwyd No',m))
            bdy = open(a,'a')
            bdy.write(('%s\r\n') % ('RiverClwyd'))
            bdy.write(('%3s\t%4s\r\n') % ('397','seconds'))
            for line in River_result:
                line.tofile(bdy,sep='\t',format='%2.2f')
                bdy.write('\r\n')
            for i in range(0,11):
                EWL = Return[i,j]
                SLR = SeaLevelRise[k]
                Skew = EWL — Tide[149] #Amend to index of high water
                Residual = Surge[:,1]*Skew
                Storm_tide = Tide + Residual + SLR
                bc = Return[i,0]
                Result = np.column_stack([Storm_tide,time])
                bdy = open(a,'a')
                bdy.write(('%s%3.0f\r\n') % ('bc',bc))
                bdy.write(('%3s\t%4s\r\n') % ('397','seconds'))
                for line in Result:
                    line.tofile(bdy,sep='\t',format='%2.2f')
                    bdy.write('\r\n')
            loga = open(logfile,'a')
            loga.write(('%2.2f\t%2.2f\t%2.2f\t%s\r\n') % (h,EWL,SLR,a))
            loga.close()
            bdy.close()
            m = m+1
            h = h+1
```

Should you wish to create start bdyfiles with river discharge, simply add the extra loop, h = h+1 at the bottom, use np.linspace to increment up the discharge and write the values to the file in the same manner as the extreme water levels.

2.4 Creating the parameter files

Now that we have all the necessary input files, we need a way of telling the model which files do what. The parameter file serves this purpose. The manual provides some Python functions for writing parameter files for both start and storm simulations. Again, the code is based on 16 return periods over 21 sea level increments. This section first describes the content of the parameter file before providing the code to create them.

```
  DEMfile                    rhyl.asc #The domain file

resroot                  rhyl_output #The prefix of your output filenames. E.g. ...
    rhyl_output0001.wd

```

```
dirroot                 rhyl #The name of the folder you want LISFLOOD-FP outputs to go into

sim_time                356400.0 #Simulation time. Currently set to McMillan 99 hr duration

initial_tstep           10.0 #Initial time step

mass_int                356.0 #Interval at which the mass file is written to

saveint                 3564.0 #Interval at which .wd files are written. I.e. 100 files for ...
    this simulation

fpfric                  0.035 #Global friction value, could use spatially variable manning's ...
    file. Redundant if manning file specified

bdyfile                 rhyl1.bdy #Name of body file

bcifile                 rhyl.bci #Name of bci file

startfile               rhyl_start1.max #Name of start depth file, i.e. areas already flooded

theta                   0.95 #Aids model stability

adapton                 #Enables adaptive time stepping. DO NOT TOUCH

manningfile             #Enables spatially variable manning value. This file must be same ...
    size and resolution as DEMfile.
acceleration

drycheckoff

hazard                  #Writes out a separate file with maximum flood hazard value for each ...
    cell

voutput                 #Writes out x and y cartesianal velocity at each save interval
```

For further options refer to Bates et al. [2013], particularly in relation to spatially variable Manning's parameter (roughness value). Depending on the floodplain, this may lead to a small amount of uncertainty. For Fleetwood, Prime et al. [2015] found that the total flood extent was not dependent on the Manning's file. Essentially this is another .asc file the same size as the domain with a roughness value for each grid cell. Find below the code to create start parameter files, where the start depth file is the original one created by the start_depth function in the DEM section.

```
def  start_par_writer(resroot,n):
    """Writes out LISFLOOD-FP parameter files to resroot. Based on 16 return periods, 0.1 cm
    SLR increments up to 2 m. Ensure input files are named the same as Resroot. No friction or
    river files"""
    for x in range(1,n):
        fname = resroot + str(x) + '.par'
        f = open(fname,'w')
        f.write('DEMfile                ' + resroot +'.asc\r\n')
        f.write('resroot                ' + resroot + '_start\r\n')
        f.write('dirroot                ' + resroot +'\r\n')
        f.write('sim_time               89100.0\r\n')
        f.write('initial_tstep          10.0\r\n')
        f.write('mass_int               89.1\r\n')
        f.write('saveint                9999999.0\r\n')
        f.write('fpfric                 0.035\r\n')
        f.write('bdyfile                ' + resroot + str(x) + '.bdy\r\n')
        f.write('bcifile                ' + resroot + '.bci\r\n')
        f.write('startfile              ' + resroot + '_start.asc\r\n')
        f.write('theta                  0.95\r\n');
        f.write('adapton\r\n')
        f.write('acceleration\r\n')
        f.write('drycheckoff\r\n')
        f.close()
        return
```

And the code to generate storm simulation files. Note the method to assign each bdy file the appropriate start depth based on having 16 return periods. I.e. files 1-16 have start depth 1, 17-32 have start depth 2 etc. all the way to 321-336 have start depth 21. Add 'y' to end of function to enable hazard and velocity mode.

```
import numpy as np
```

```
def par_writer(resroot,output,hazard):
    """Writes out LISFLOOD-FP parameter files to resroot. Based on 16 return periods, 0.1 cm
    SLR increments up to 2 m. Ensure input files are named the same as Resroot. No friction or
    river files"""
    m = np.linspace(1,337,22) #This assumes 21 sea level scenarios superimposed on 16 (all ...
        McMillan) return periods
    s = 1
    for x in range(1,337):
        #Works out appropriate start depth
        p = m[s]
        if x == p:
            l = s
            s = s+1
        fname = resroot + str(x) + '.par'
        f = open(fname,'w')
        f.write('DEMfile                    ' + resroot +'.asc\r\n')
        f.write('resroot                    ' + output +'\r\n')
        f.write('dirroot                    ' + resroot +'\r\n')
        f.write('sim_time                   356400.0\r\n')
        f.write('initial_tstep              10.0\r\n')
        f.write('mass_int                   356.0\r\n')
        f.write('saveint                    3564.0\r\n')
        f.write('fpfric                     0.035\r\n')
        f.write('bdyfile                    ' + resroot + str(x) + '.bdy\r\n')
        f.write('bcifile                    ' + resroot + '.bci\r\n')
        f.write('startfile                  ' + resroot + '_start' + str(s) + '.max\r\n')
        f.write('#manningfile                ')
        f.write('theta                      0.95\r\n');
        f.write('adapton\r\n')
        f.write('acceleration\r\n')
        f.write('drycheckoff\r\n')
        if hazard == 'y':
            f.write('hazard\r\n')
            f.write('voutput\r\n')
        f.close()
        return
```

2.5 Running the model

LISFLOOD-FP v6.0.2 is not compiled to run on livljobs2. So on the command line enter:

1. ssh livljobs3
2. cd /work/benp/lisflood %Your working directory
3. lisflood_float_rect -v -log filename1.par % where filename1.par is the name of your parameter file.

The simplest way to run multiple scenarios is to use the terminal to create a loop to execute LISFLOOD-FP for each parameter file. This is shown below:

```
for i in {1..n} %Where n is the number of simulations
do
lisflood_float_rect –v –log parameter_filename$i.par
done
```

3 Processing the output

3.1 Calculating resultant velocity

When we want to think about flood hazard, we also need to think about flood velocity, as fast flowing but shallow water can be just as dangerous as deep water, particularly to vulnerable members of society. Standard LISFLOOD-FP output only gives us cartesianal velocity in the x and y directions. To calculate the resultant velocity (V_r), we simply use Pythagoras's Theorem (see Figure 8)

$$V_r = \sqrt{{V_x}^2 + {V_y}^2} \tag{3}$$

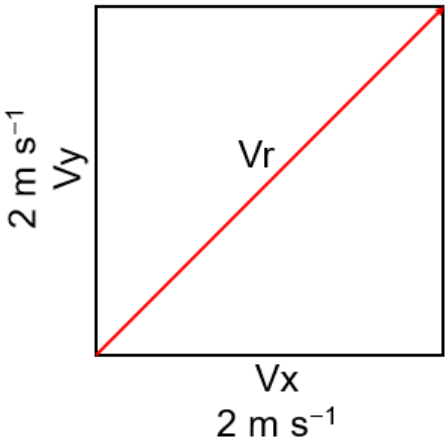

Figure 8: Applying Pythagoras's Theorem to LISFLOOD-FP

```
def Vr(resroot,scenarios,outputnum):
    """Takes Vx and Vy files for each save interval and calculates resultant velocity (VR) using
    Pythagoras's theorem"""
    #Adds one so it loops correctly
    num = scenarios+1
    output = outputnum+1

    for f in range(1,num):
        for i in range(1,output):
            if i < 10:
                a = resroot + str(f) + '-000' + str(i) + '.Vx'
                b = resroot + str(f) + '-000' + str(i) + '.Vy'
                c = resroot + str(f) + '-000' + str(i) + '.Vr'

            elif i < 100:
                a = resroot + str(f) + '-00' + str(i) + '.Vx'
                b = resroot + str(f) + '-00' + str(i) + '.Vy'
                c = resroot + str(f) + '-00' + str(i) + '.Vr'

            else:
                a = resroot + str(f) + '-0' + str(i) + '.Vx'
                b = resroot + str(f) + '-0' + str(i) + '.Vy'
                c = resroot + str(f) + '-0' + str(i) + '.Vr'

            Vx = np.loadtxt(a,skiprows=6)
            Vy = np.loadtxt(b,skiprows=6)
            line1 = lc.getline(a,1)
            line2 = lc.getline(a,2)
            line3 = lc.getline(a,3)
            line4 = lc.getline(a,4)
            line5 = lc.getline(a,5)
            line6 = lc.getline(a,6)
            header = line1 + line2 + line3 + line4 + line5 + line6 #Concatenate header
            Vr = np.hypot(Vx,Vy)
            np.savetxt(c,Vr,header=header,fmt='1.2%f')
```

3.2 Calculating flood hazard

For further details the reader is referred to Surendran et al. [2008]

4 References

References

Paul Bates, Mark Trigg, Jeff Neal, and Amy Dabrowa. LISFLOOD-FP User manual: Code release 5.9.6. Technical Report November, University of Bristol, Bristol, 2013.

Paul D. Bates, Richard J. Dawson, Jim W. Hall, Matthew S. Horritt, Robert J. Nicholls, and Jon Wicks. Simpli-

fied two-dimensional numerical modelling of coastal flooding and example applications. *Coastal Engineering*, 52(9):793–810, sep 2005. ISSN 03783839. doi: 10.1016/j.coastaleng.2005.06.001.

Paul D. Bates, Matthew S. Horritt, and Timothy J. Fewtrell. A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling. *Journal of Hydrology*, 387(1-2):33–45, 2010. ISSN 00221694. doi: 10.1016/j.jhydrol.2010.03.027. URL http://dx.doi.org/10.1016/j.jhydrol.2010.03.027.

P.D Bates and A.P.J De Roo. A simple raster-based model for flood inundation simulation. *Journal of Hydrology*, 236(1-2):54–77, 2000. ISSN 00221694.

K. J. Horsburgh and C. Wilson. Tide-surge interaction and its role in the distribution of surge residuals in the North Sea. *Journal of Geophysical Research*, 112(C8):C08003, aug 2007. ISSN 0148-0227. doi: 10.1029/2006JC004033.

T. J. Marsh and F. J. Sanderson. Derivation of daily outflows from Hydrometric Areas. Technical Report July, CEH Wallingford, Wallingford, 2003.

A. McMillan, C. Batstone, D. Worth, J. Tawn, K. Horsburgh, and M. Lawless. *Coastal flood boundary conditions for UK mainland and islands. Project SC060064/TR2: Design sea levels.* Environment Agency, Bristol, 2011. ISBN 9781849112130. URL http://nora.nerc.ac.uk/19721/.

Thomas Prime, Jennifer M Brown, and Andrew J Plater. Physical and economic impacts of sea-level rise and low probability flooding events on coastal communities. *PLoS ONE*, 10(2):e0117030, 2015.

S. Surendran, G. Gibbs, S. Wade, and H. Udale-Clarke. Supplementary note on flood hazard ratings and thresholds- for development planning and control purpose. Technical Report October 2005, Environment Agency/ HR Wallingford, 2008. URL http://webcache.googleusercontent.com/search?q=cache:JothtGL3zo8J:randd.defra.gov.uk/Document.aspx?Do