

# Grid Generator, Prerequisites

Jonathan Ridenour

August 4, 2016

## 1 Problem Statement

Given a digital elevation model (DEM) of Earth's surface, we need a domain of point coordinates in three dimensions suitable for solving partial differential equations on. The domain extends downward from the DEM surface to a user-defined desired depth.

## 2 Requirements

### 2.1 Inputs

Data to be supplied by the user are:

- a digital elevation model ( $m$  above mean sea level) covering the desired area e.g. a GTOPO30 tile,
- longitude and latitude intervals (in degrees easting and northing),
- a desired depth ( $m$ ).

### 2.2 Outputs

Data to be output by the program are three tab-delimited .txt files containing the separate  $(x, y, z)$ -coordinates of a computational grid spanning the longitude and latitude limits and down to the desired depth.

## 3 Architecture

### 3.1 Class Descriptions

#### 3.1.1 Domain

The **Domain** class is the base class for the 1-, 2-, and 3-dimensional domains from which the **Line**, **Surface**, and **Grid** classes inherit their common functions and data members: numerical constants **Nx**, **Ny**, and **Nz** (number of array elements (**Point** objects) on corresponding sides of the grid block), **xi**, **eta**, and **zeta**

(interpolation constants, see Section 3.2), and dynamically allocated arrays of **Point** objects for coordinates and corner points.

Routines include getters and setters, **showCoordinates()** (virtual) for printing the coordinates of a domain to the terminal, and **printCoordinates()** (virtual) for printing the coordinates to file. The  $(x, y, z)$ -coordinates of the domains are printed to three separate files called **filenameX.txt**, **filenameY.txt**, etc.

### 3.1.2 Grid

This is the 3-dimensional domain which is the goal of this program. The **Grid** object defines a 6-sided computational domain in curvilinear coordinates. The **interpolate()** routine accomplishes 3D transfinite interpolation, as discussed in Section 3.2.2. The **coordinates** array is of size  $N_x \times N_y \times N_z$ , and the **corners** array is of size 8.

### 3.1.3 Surface

This is the 2-dimensional domain which constitutes a boundary for the 3-dimensional domain. The **interpolate()** routine accomplishes 2D transfinite interpolation, as discussed in Section 3.2.1. The **coordinates** array is of size  $N_x \times N_y$ , and the **corners** array is of size 4.

### 3.1.4 Line

This is the 1-dimensional domain which constitutes a boundary for the 2-dimensional domain. The **coordinates** array is of size  $N_x$ , and the **corners** array is of size 2. The idea with **Line** is to initialise it to a specific length, then fill **coordinates** with **Point** objects generated from the DEM.

### 3.1.5 Point

Data members are three doubles that constitute an  $(x, y, z)$ -coordinate.

Public routines are getters, setters, and a **showCoordinate()** function, which displays the coordinate of the point in the terminal.

## 3.2 Mathematics

### 3.2.1 2D Transfinite interpolation

$$\begin{aligned} x(\xi, \eta) = & (1 - \xi)x(0, \eta) + \xi x(1, \eta) + (1 - \eta)x(\xi, 0) + \eta x(\xi, 1) - (1 - \eta)(1 - \xi)x(0, 0) \\ & - \xi(1 - \eta)x(1, 0) - (1 - \xi)\eta x(0, 1) - \eta\xi x(1, 1), \end{aligned}$$

$$\begin{aligned} y(\xi, \eta) = & (1 - \xi)y(0, \eta) + \xi y(1, \eta) + (1 - \eta)y(\xi, 0) + \eta y(\xi, 1) - (1 - \eta)(1 - \xi)y(0, 0) \\ & - \xi(1 - \eta)y(1, 0) - (1 - \xi)\eta y(0, 1) - \eta\xi y(1, 1). \end{aligned}$$

### 3.2.2 3D Transfinite Interpolation

As in [1], the formula for 3D transfinite interpolation is as follows:

$$\begin{aligned}
U(\xi, \eta, \zeta) &= (1 - \xi)X(0, \eta, \zeta) + \xi X(1, \eta, \zeta), \\
V(\xi, \eta, \zeta) &= (1 - \eta)X(\xi, 0, \zeta) + \eta X(\xi, 1, \zeta), \\
W(\xi, \eta, \zeta) &= (1 - \zeta)X(\xi, \eta, 0) + \zeta X(\xi, \eta, 1), \\
UW(\xi, \eta, \zeta) &= (1 - \xi)(1 - \zeta)X(0, \eta, 0) + \zeta(1 - \xi)X(0, \eta, 1) \\
&\quad + \xi(1 - \zeta)X(1, \eta, 0) + \xi\zeta X(1, \eta, 1), \\
UV(\xi, \eta, \zeta) &= (1 - \xi)(1 - \eta)X(0, 0, \zeta) + \eta(1 - \xi)X(0, 1, \zeta) \\
&\quad + \xi(1 - \eta)X(1, 0, \zeta) + \xi\eta X(1, 1, \zeta), \\
VW(\xi, \eta, \zeta) &= (1 - \eta)(1 - \zeta)X(\xi, 0, 0) + \zeta(1 - \eta)X(\xi, 0, 1) \\
&\quad + \eta(1 - \zeta)X(\xi, 1, 0) + \eta\zeta X(\xi, 1, 1), \\
UVW(\xi, \eta, \zeta) &= (1 - \xi)(1 - \eta)(1 - \zeta)X(0, 0, 0) + (1 - \xi)(1 - \eta)\zeta X(0, 0, 1) \\
&\quad + (1 - \xi)\eta(1 - \zeta)X(0, 1, 0) + \xi(1 - \eta)(1 - \zeta)X(1, 0, 0) \\
&\quad + (1 - \xi)\eta\zeta X(0, 1, 1) + \xi(1 - \eta)\zeta X(1, 0, 1) \\
&\quad + (1 - \zeta)\xi\eta X(1, 1, 0) + \xi\eta\zeta X(1, 1, 1).
\end{aligned}$$

Putting these together gives the complete formula:

$$\begin{aligned}
X(\xi, \eta, \zeta) &= U(\xi, \eta, \zeta) + V(\xi, \eta, \zeta) + W(\xi, \eta, \zeta) \\
&\quad - UW(\xi, \eta, \zeta) - UV(\xi, \eta, \zeta) - VW(\xi, \eta, \zeta) \\
&\quad + U VW(\xi, \eta, \zeta).
\end{aligned}$$

## References

- [1] Smith, Robert E (1998) Transfinite Interpolation Generation Systems. In Nigel P., et. al. *Handbook of Grid Generation*, CRC Press.