Empirical modeling of fault regions with natural erosion and fault uplift

Abstract

One of the key factors that affect an earthquake is the stress of the landscape has around the faults. This stress is caused by both the weight of the landscape itself and the uplift of the fault. This paper will create a model that can simulate landscape denudation around a fault line. The model is based on cellular automation comprised with an erosion algorithm, a finite diffusion algorithm and a triangular uplift function. The perpendicular curvature data of the channels from the simulation will be compared against real world data from an area called Dragon's Back Pressure Ridge along the San Andreas fault-line to find any relationship between the curvature data and fault uplift rate as well as to confirm the validity of the model. The parameters of the model are adjusted based on the results from the comparison.

The simulation developed shows no significance relationship between the curvature of the channels and the uplift rate of the fault. The tools used to develop the model, however, can be used as a starting point to develop more accurate methods to analyze uplift rate and simulate future terrain denudation along fault-lines.

Empirical modeling of fault regions with natural erosion and fault uplift

by Jingyu Yao

1. Introduction

1.1. Background

The history of computational model of landscapes begins in 1965 with a paper by Schumm & Licthy [1]. The analysis of the change or the formation of landscapes was more conceptual than quantitative before that time. Many applications of landscape denudation simulation have appeared over the years in areas such as terrain generation, environmental protection and earthquake simulation. However, there are only a few simulations to model regions around a fault line. Terrain denudation modeling for fault regions is important because one of the key factors that affects an earthquake is the stress of the landscape have around the fault. Normal denudation simulations are not sufficient to model fault line regions because the topography around a fault line is always changing due to uplift. But the rate at which the uplift occurs is so slow and it spans over such a long periods time, we do not have enough data recorded within the course of human history to study the pattern of the uplift in a given region. The lack of uplift patterns poses difficulties in confirming the results of the simulation as we have no data to compare against. A new landscape evolution model is needed that can accurately represent the change in land formation caused by fault movement and erosion. The new model will help scientists better understand what caused the strange land formations around fault lines and how fault lines affect earthquakes.

Landscape evolution modeling is separated into three categories: qualitative, physical and surface process models [2]. Qualitative models usually deal with changes in landform over a large scale, usually across continents. They are based on field observations and specific for each case. Physical models concerns with a more detailed representation of terrain. Physical models use field observation and data collections to analysis how different process works. They are first used to understand the formation of channels that occur with natural erosion to many other natural phenomenons. Surface process models focus on a single type of landform and it is strongly linked with field observations. This type of models can process feedbacks both from the surface or tectonic movement and it shows the interaction between the landform and different processes including erosion, diffusion and fault movement. My model belongs to the physical models type.

1.2. Literature review

For development of my model, a strong and simple erosion algorithm is needed as a base to develop a model that can handle complex fault movement as well as a large area. A model cannot be created without either of erosion and fault movement functions. My model is the next step in development in the model proposed by Chase[3]. The model by Chase is based on cellular automata where the terrain is divided into equal sized cells and each cell has 8 neighbors. Drops of rain run down a slope to determine the direction of erosion. The rain deposits a flexible amount of sediment depending on the change in height between each square of the matrix. Chase's model is quite efficient, but it lacks few items that decreased its accuracy. One such item is the different distance the rain will have to travel from going to its neighbors in the corner than the ones right next to it.

The core of an erosion algorithm consists of three components: detachment, transport, and deposition according to [7]. The monograph discusses the two types of algorithms, physical and empirical. Empirical models are different from the previously mentioned models is because they do not take in account of the actual erosion process. Instead they are based on the mathematical relationship between the landscape and erosion. Their empirical model (as well as the model presented in this research) is a lumped-parameter model, which assumes all area of the landscape is homogeneous in constitution. Their model assigned different area with different diffusion properties. The same principle is used for the model developed in [6] as described in the next paragraph, but that model lacked erosion.

In a computer model developed by S.S Egan (1999), it described both two and three dimensional geometric methods of modeling deformation due to fault movement [6]. It performs calculation based on extensional, compressional and strike-slip movement of different layers of rocks that is affected by the fault movement. This models accounts for the different layer types of rocks beneath the surface, and the program calculates different movement based on the rock's characteristics. Although this model takes many parameters into account, it did not consider the effect of erosion on structural restoration. In order to help geographers validate structural interpretations about a landscape, the computer software must able to display information geographers can easily measure, in this case the surface of the terrain.

1.3. Research Problem

Currently, scientists are only able to model given data but are not able validate the use of an algorithm to project current data further in time for complex fault line region to predict future earthquake due to the uncertainly in future fault uplift rates. This hinders the accuracy of future earthquake forecasts. In order to create a comprehensive simulation of future fault lines, methods to analyze a given region to find the uplift pattern as well as accurate landscape denudation methods are needed. The simulated data is then able to help scientists make more accurate earthquake forecasts.

1.4 Significance

Earthquakes are one of the most destructive natural occurrences and yet they are still shrouded in mystery even in today's technological advanced world. According to congressional data from FEMA [4] the country losses about 5.3 billion dollars a year from earthquakes. Long have the scientists try to find better ways to predict the next occurrence and estimate its power. Any improvement in the understanding of earthquakes will have a positive effect on the lives of millions who live near faults or beyond. The landscape evolution models of fault region help scientists better understand earthquakes by using the current data on changes in land formation around a fault line to project backward or forwards in time to see how the area will look like. This information on the land formation has several uses in understanding earthquakes. First, change in sizes of the land form can cause different amount of stresses on the fault lines, the stresses can change the scale of the earthquake. Second, this will offer geologist a more accurate representation of future topography to work with, more accurate data means better chance of predicting earthquakes. Third, once we are able to predict what should happen normally with the landscape, we can compare this to older data, any major discrepancy will be either produced by an earthquake or other natural disaster. These could then be isolated to show the effect of a specific event. Fourth, landscape evolution models can help geologist understand complex feedback between surface and tectonic processes as well as help earth scientist back trace geological event through restoration by computer model.

1.5. Research Hypothesis

To address the problem presented, a DEM (digital elevation model) will be made for Dragon's Back pressure ridge (DBPR). DBPR is an important area to analyze since each segment of the channels away from the fault represents a different time in its uplift period. Thus the channels can be analyzed using space-for-time substitution too understand how uplift, both active and ceased, can affect surface channels and ridges. The data measured for the channels is the perpendicular curvature in the direction of the channels. This shows whether the channel in increasing or decreasing in slope as you move away from it. Then the feedback from the data gathered can be used to adjust and validate my own simulation. My simulation, which includes a combination of erosion and uplift algorithms, should be able to reproduce channels similar to an area such as DBPR. If the model is able to correctly reproduce channels of DBPR, the tools used can then be applied to other areas around the San Andreas Fault to learn how they will look in the future.

2. Methodology

My algorithm solves a partial differential equation for erosion, using the Java platform and Light Weight Java Game Library (LWJGL) for visualization. The program models landscape denudation by simulating the effect of rain by both erosion and diffusion. Drops of "water" will fall randomly at a location in a mesh. The droplets will erode as they slide down to the lowest point. This will act as detachment, transport, and deposition. After every certain interval, a finite diffusion algorithm will be applied to the whole mesh. The second part of the program is fault movements. This is implemented by dividing up the original mesh into many smaller ones, move them individually and then reconnect their borders.

2.1. Program structure

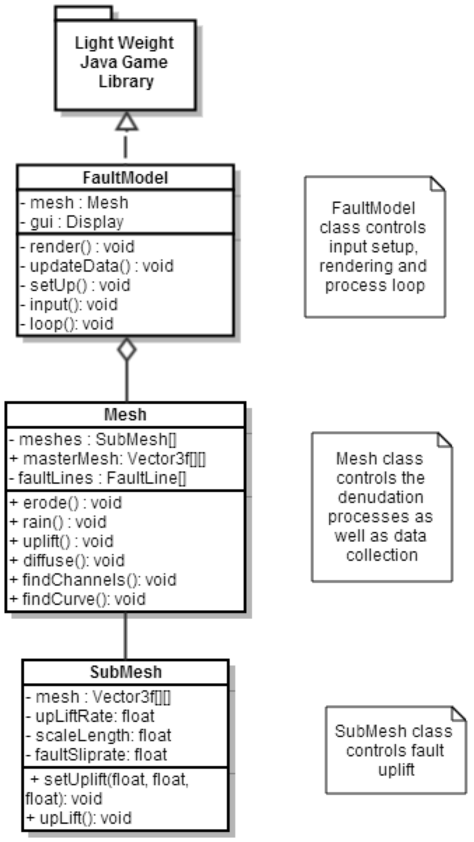


Fig. 2 This figure shows a simplified class diagram for the program. The mesh is broken down into different sub-meshes by the fault lines. The sub-meshes are made from a 2D array of points and the fault lines are defined as a 2D vector. A point is defined as a location in a 3D space with (x,y,z) values. The Mesh class controls the denudation process and SubMesh class controls fault uplift.

2.2 Erosion

The differential scheme for the erosion process is as follows:

$$\Delta S = \frac{wk_a \cdot \Delta h + wk_b}{\Delta h + wk_c} \cdot \Delta h \qquad (1)$$

$$wk_a = wk_{fmax} \qquad (2)$$

$$wk_c = \frac{wk_{fmax} - wk_{fmin}}{wk_{fmax} - 0.5 \cdot wk_{fmin}} \cdot wk_{slp} \qquad (3)$$

$$wk_b = wk_c \cdot wk_{fmin} \qquad (4)$$

where ΔS is sediment removed from the cell, wk_{fmax} and wk_{fmin} are constants that restricts the maximum and minimum values of ΔS and wk_{slp} is the slope between the points and its lowest neighbor.

The rain's movement representation is based on cellular automata, where the terrain is split into many equal dimension square cells and sediments are able to travel freely between its Moore neighborhoods. The main idea of the erosion algorithm starts with a water droplet named walker falling onto a random cell in the mesh at a starting point (P). Then the heights of its surrounding neighbor cells are obtained and differences in height (Δh) between the neighbors and P are calculated by subtracting each neighbor's height from P's height. The largest Δh and its corresponding neighbor (nP) are found with a simple loop. If Δh is negative and the sediments carried by walker are greater than $|\Delta h|$ then enough sediment from walker will be dropped onto P plus a little bit more and the erosion process will start on nP again. But if Δh is negative and the sediments carried by walker are less than $|\Delta h|$, then all the sediments carried by the walker are dumped onto P to end the algorithm. If the largest Δh is positive then the ΔS taken from P is calculated with (1) and added to walker. The higher the slope between P and nP, the more sediments are removed from P. The actual sediments carried from P to nP are only a fraction of the total sediments within walker. This fraction is determined by the value of another constant called $wk_{carrylength}$. The final calculated change is subtracted from walker and P and added to nP. This process now continues on nP. Note that wk_{fmax}, wk_{fmin} and $wk_{carrylength}$ are not universal constants but rather they depend on the climate of the model's location as well as the types of rock being eroded. A walker in a humid environment will to be able to carry more sediment (higher wk_{fmax}) than a walker in a dry environment. And areas of alluvium will have a higher erodibility coefficient ($wk_{carrylength}$) than areas of granite. Carry length is the most important climate variable according to [3].

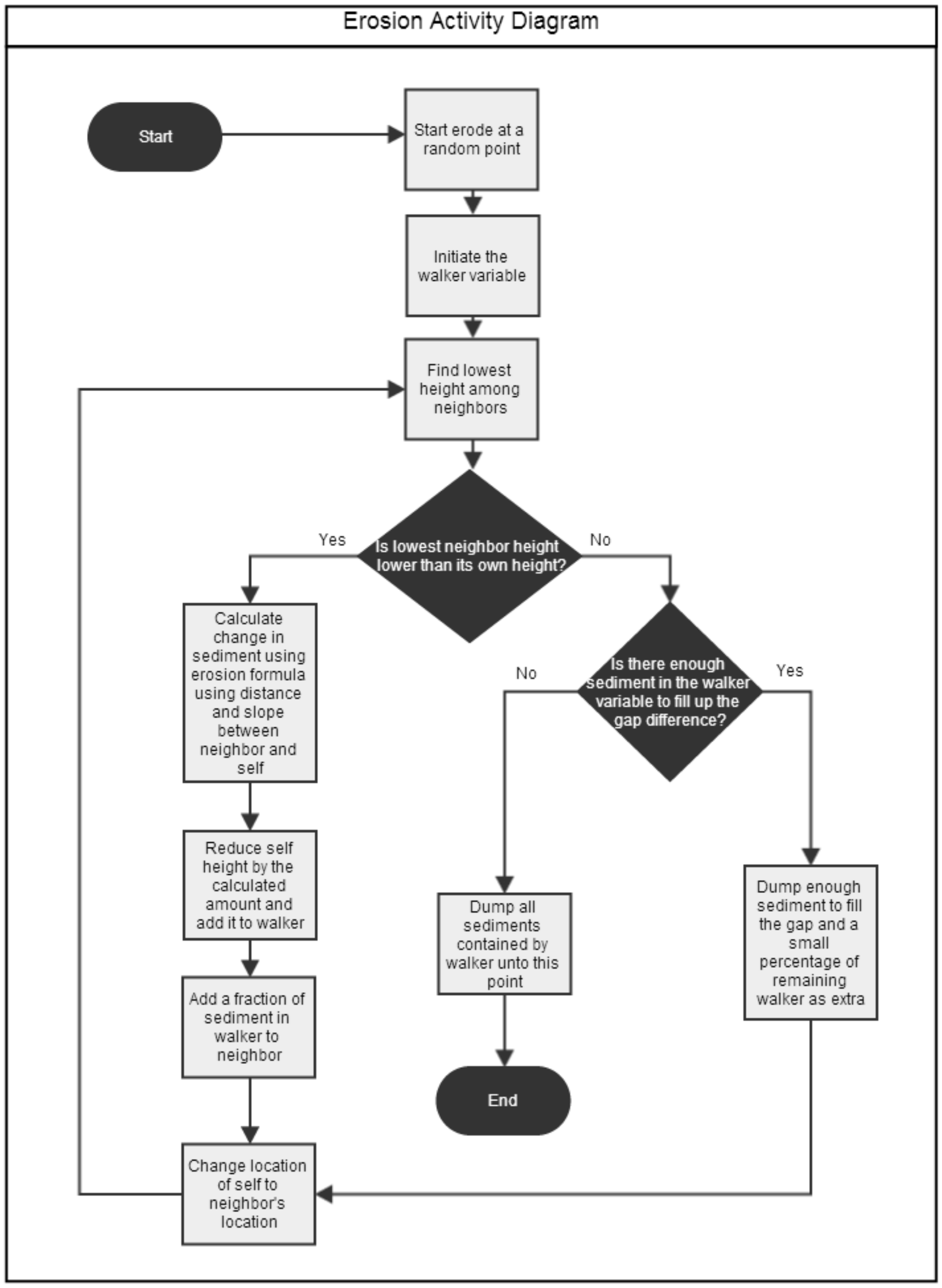

Erosion Activity Diagram
Start
Start erode at a random point
Initiate the walker variable
Find lowest height among neighbors
Is lowest neighbor height lower than its own height?
Yes
No
Calculate change in sediment using erosion formula using distance and slope between neighbor and self
Is there enough sediment in the walker variable to fill up the gap difference?
No
Yes
Reduce self height by the calculated amount and add it to walker
Dump all sediments contained by walker unto this point
Dump enough sediment to fill the gap and a small percentage of remaining walker as extra
Add a fraction of sediment in walker to neighbor
End
Change location of self to neighbor's location

Fig. 2: This figure shows the logic behind the erosion process algorithm. Decisions are represented with diamond shape box and processes are represented with square boxes.

2.3. Diffusion

Since erosion roughens the landscape at all scales but deposition from erosion only smooths the landscape in a large scale[3], diffusion is then needed to smooth the landscape at a small scale. Or else the landscape will look unnatural up close. The diffusion algorithm used in this simulation is a finite diffusion algorithm that only diffuses the sediments on the point to its nearest cells. The diffuse amount is determined by subtracting the average height of the four closest cells (non-diagonal) from the center cell. This amount is then distributed evenly amount the neighbors. This diffusion is applied to every single cell in the mesh after a certain amount of erosion. The ratio between the number of erosion and diffusion also depends on the climate. More diffusion occurs at a more humid climate. The diffusion algorithm used in this simulation is verified to behave correctly as a spreading Gaussian.

Diffusion mask in terms of height:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2.4. Fault uplift

The modeling of the fault movements is achieved first by dividing the mesh into smaller sub-meshes using a predefined fault line. Each sub-mesh will have its own uplift function. The function used to simulate the uplift of DBPR is a triangular uplift function and it is as follows:

$$\frac{dh}{dt} = f(x,y) = g(x - vt)q(y) \tag{5}$$

$$g(x) = \begin{cases} a(L - |x|), & |x| \leq L \\ 0, & |x| > L \end{cases} \tag{6}$$

$$q(y) = \begin{cases} 0, & y \leq 0 \\ L - y, & y < L \\ 0, & y \geq L \end{cases} \tag{7}$$

where $f(x, y)$ is function of change in height over time of that point's distance relative to the fault. The x axis is along the direction of the fault and the y axis is perpendicular to the direction of the fault. Values of *y* are the positive for the side of the fault that is moving. The value of *x* is zero at where the uplift rate is the highest. The constant *a* is the vertical uplift rate, *L* is the scaled length along x in either direction at which the uplift will occur, *v* is the slip rate and *t* is the time. The slip rate is added to the actual x and y coordinates of each point in the direction of the fault. The actual x and y coordinates of each point is check occasionally and its position in the mesh is re-adjusted if needed. The effect of the function is shown by the following diagram:

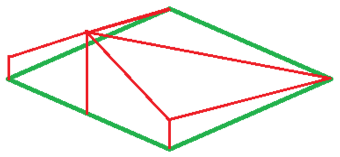


Fig. 3 This is a diagram showing the triangular uplift. Green shows the initial condition and red shows the condition after uplift.

This particular uplift function is used because it resembles the uplift of DBPR.

3. Results

The data gathered from the simulation are the curvatures in the perpendicular direction of the channel. The data is compared to the curvature of DBPR terrain. The curvature of the channels is expected to get higher as the channel approaches the fault-line. The equation used to obtain the curvature is as follows:

$$C_{left} = U_{j-2} - 2U_{j-1} + U_j \qquad (8)$$

$$C_{right} = U_{j+2} - 2U_{j+1} + U_j \qquad (9)$$

These are one dimensional equations of the curvature at U_j. The curvature found in this experiment is the average of (8) and (9) and the direction of curvature in 2D space is determined by the direction of the channel.

The channels are found by using a water flux map based on the ideas as described in [5] but with time being the duration of the whole simulation rather than a short time interval. And instead of adding flux to the map as erosion occurs, the flux map is created by dropping a “fake” rain droplet at each point on the mesh. This gives an equal chance to every point on the mesh so the flux map can more accurately represent the whole mesh. The rain droplet will follow the lowest neighbor and go down a hill but it will not erode anything. The places the rain passed the most will have the densest flux. The channels are found by first filtering out the highest water flux and then when through each single high flux point and backtracking its neighbor next highest flux (in both directions if the point started in the middle of a potential channel). This creates a string a points that represents the location of the channel. The remaining channels are filtered again based on their lengths.

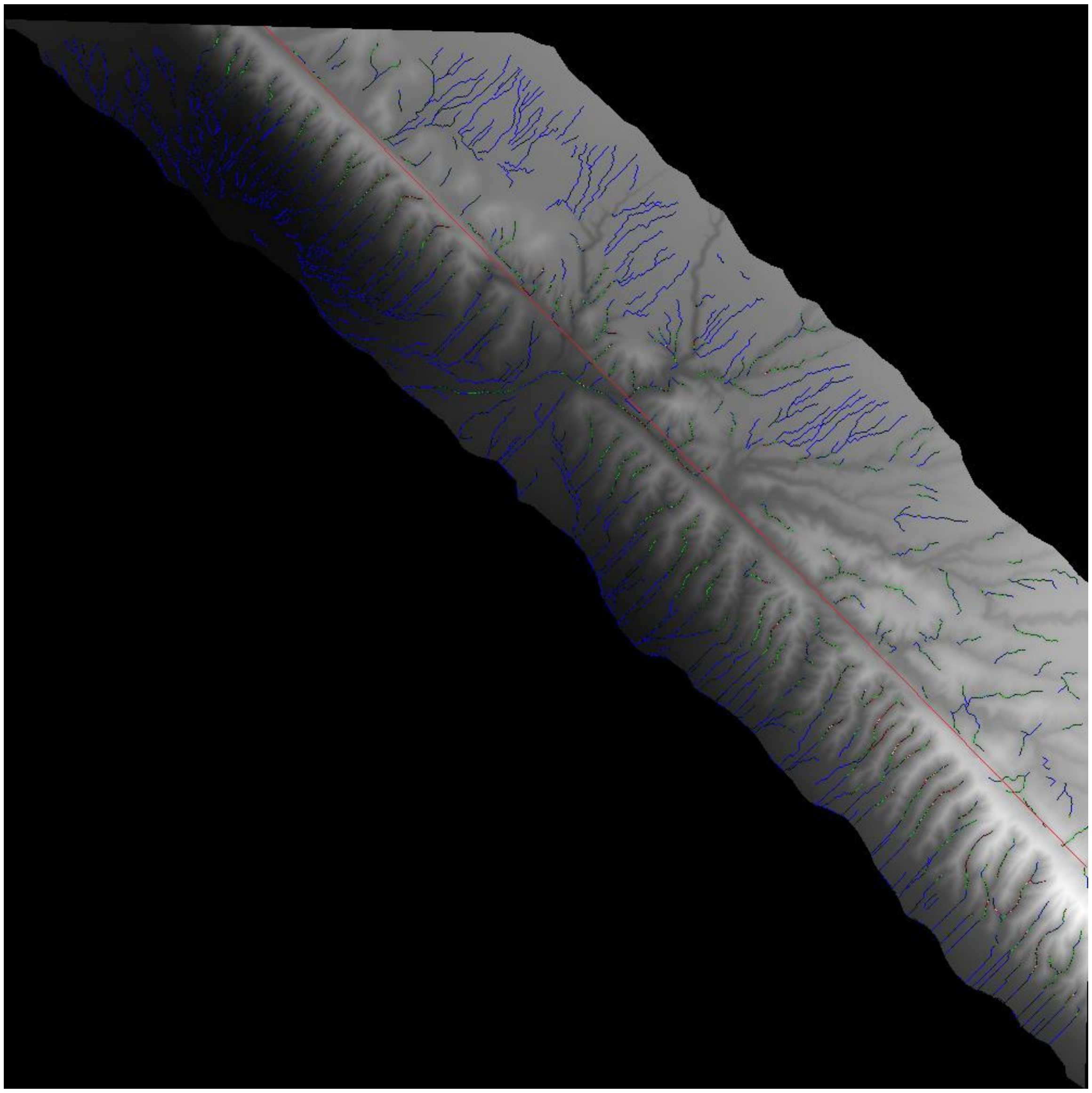


Fig. 4 Curvature overlay of DBPR with red line marking the estimated position of San Andreas Fault. Due to the limitation in the detail of the data used, some channels in the north-east region of DBPR could not be detected.

Curvature color code: blue < 0 < green < 0.1 < red < 1 < pink

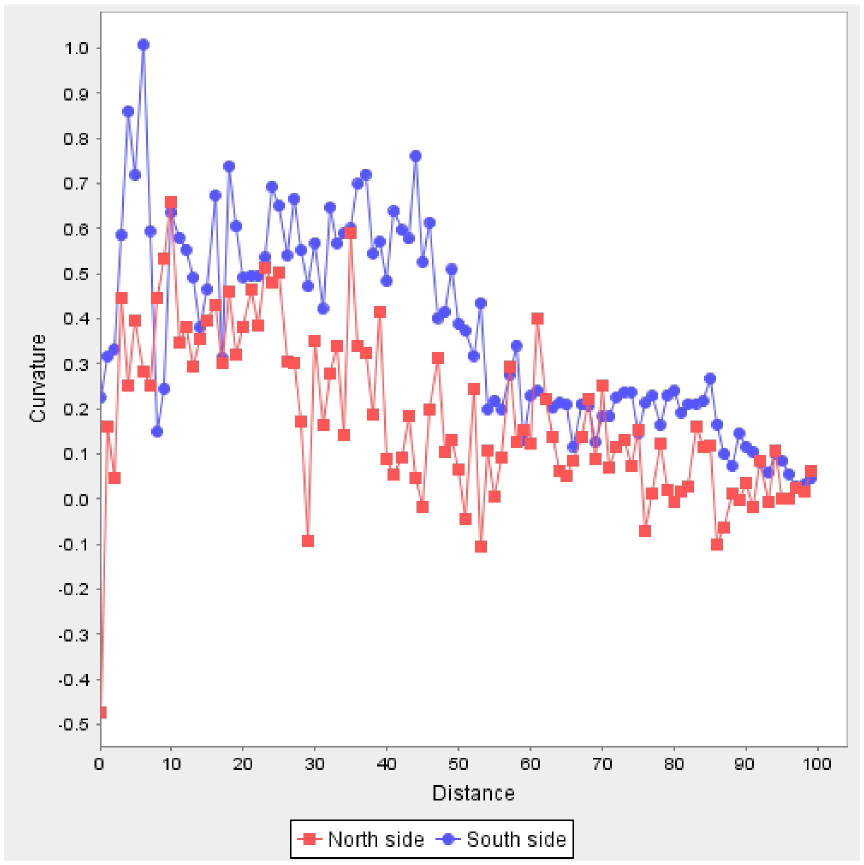
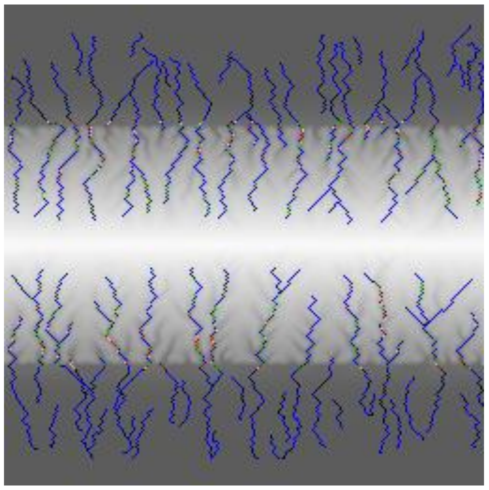


Fig. 5 Curvature data of DBPR, each unit of distance in the graph represents 4m.

Below is one of the simulation data as a time series:



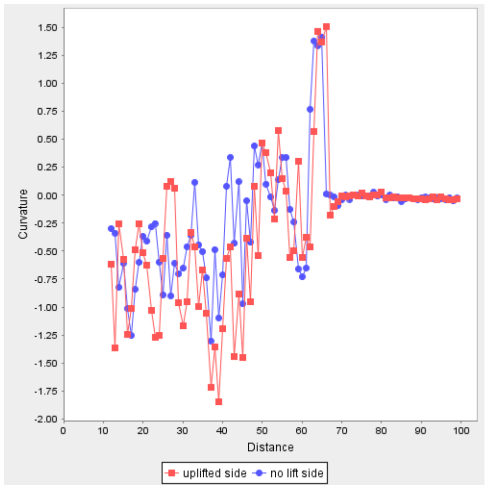
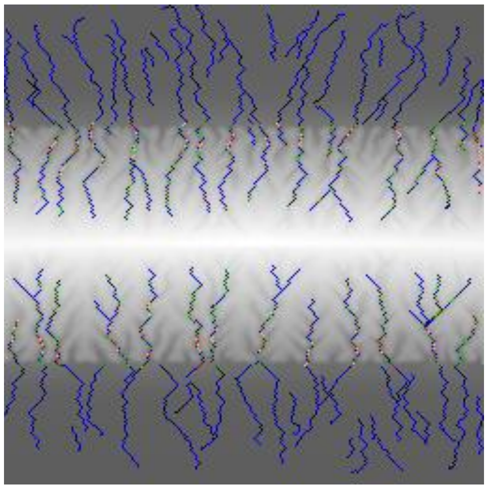


Fig.6 T=0 Curvature overlay

Curvature data



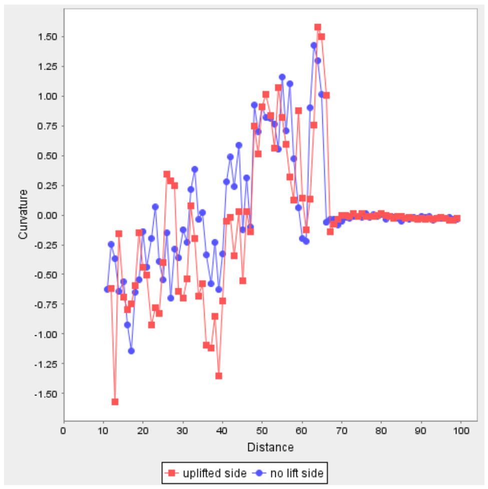
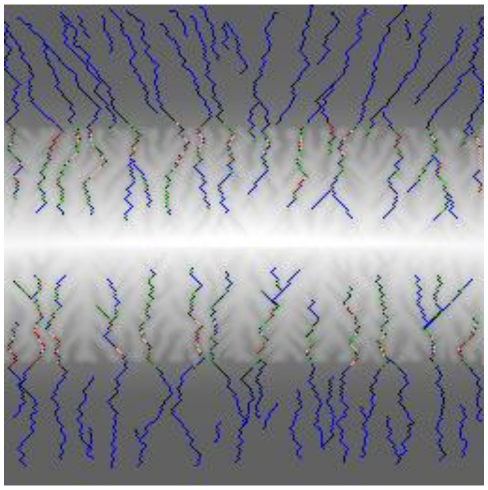


Fig.7 T=1   Curvature overlay

Curvature data



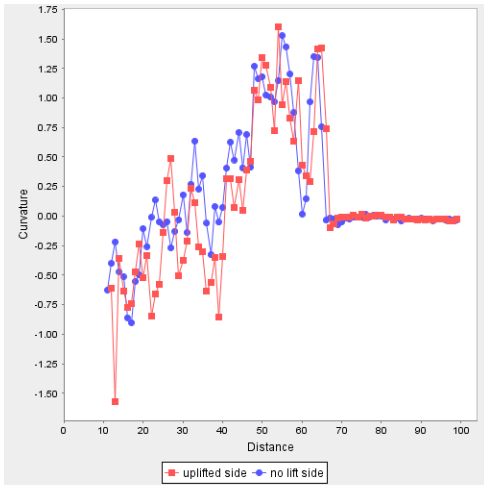
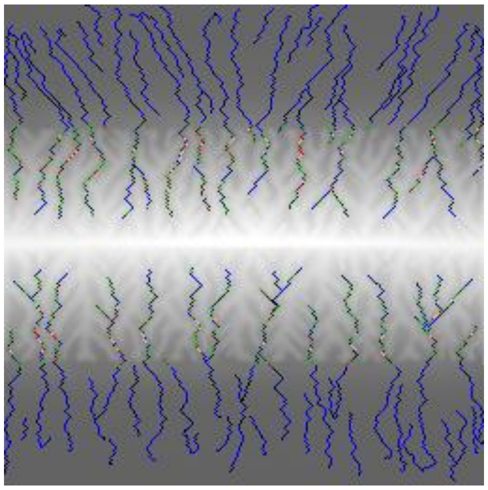


Fig.8 T=2   Curvature overlay

Curvature data



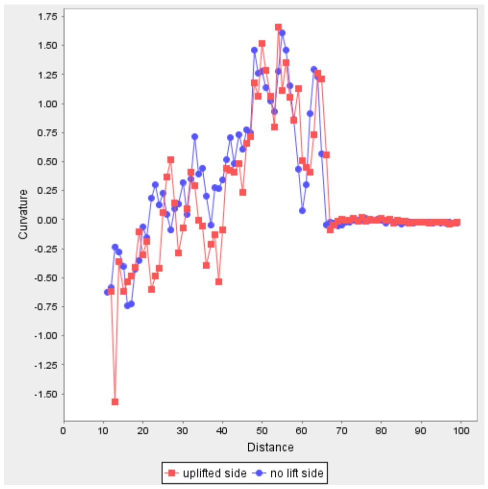
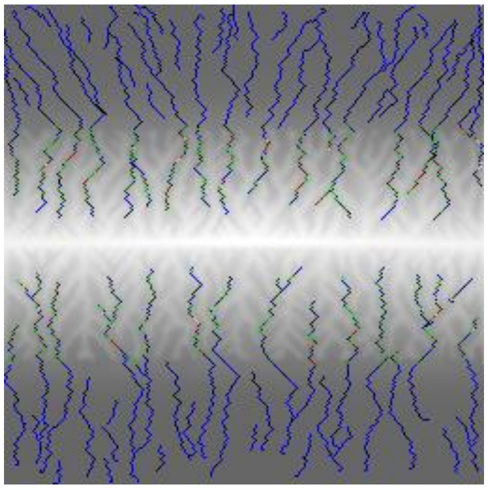


Fig.9 T=3   Curvature overlay

Curvature data



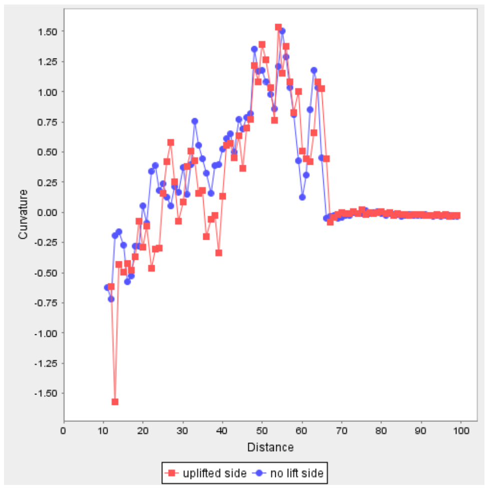


Fig.10 T=4 Curvature overlay *Curvature data*

Overlay color code: blue < 0 < green < 2 < red < 3 < pink

T = 0: Initial setup. Rain dropped: 4 million. Uplift begins for top half of terrain.

T = 1: Rain dropped: 4.5 million.

T = 2: Rain dropped: 5 million. Uplift ends.

T = 3: Rain dropped: 5.5 million.

T = 4: Rain dropped: 6 million.

<u>4. Discussion</u>

To better predict future landscape evolution, a set of tool have to be made to analyze the current landscape as well as to simulate erosion and fault uplift. I have developed set tools to measure the channels and their curvature as well as simulate erosion and fault uplift. Fig. 5 shows the curvature increases as the channel approaches 40 units (each unit is about 4m) and then decreases as it approaches the fault for south (uplifted) side. For north (non-uplifted side) the curvature of the channel increases until 15 units away and then decreases as it approaches the fault. This agrees with the expectation of channel's curvature to increases as it approaches the fault. However, the data from my simulation shows just the opposite. The curvature decreases instead of increases as the channel approaches the fault. Comparing the curvature overlay of my simulation to DBPR data, the height of the ridges at the end of the channels from my simulation is significantly higher than those of DBPR. The structure of the channels is also different. The channels from DBPR are formed from small channels perpendicular to the ridges but the ridges from my simulation does not have such small channels going perpendicular to it. These discrepancies might be caused by the initial topography of the simulation. The initial topography used for the simulation is manually created instead of based off a real world terrain. Even though the general trend from the simulation does not match those of DBPR, the effects of uplift still can be seen from the simulation data. Fig.7 and Fig.8 shows increase in overall curvature for the side of the terrain that is uplifting as well as increase in the difference of the curvature between near the fault and away from the fault. This agrees with the result from [8] that channels respond to uplifts faster than most other natural processes. The difference between the curvature from near the fault and away from the fault decreases when the uplift stops as shown in Fig. 9 and Fig.10. It takes almost one million drops of rain for the channels to resumes linear progression of curvatures. This also confirms that hill slope processes takes longer to respond to change in uplift than channels (where the effects can be seen with only half million drops of rain after uplift started) described in [8]. In another controlled simulation with no uplift, the slope of the curvature graph remains linear, showing a steady progression instead of having such discrepancy in curvature along the channel when it is uplifting. Even though the uplift does cause curvature difference between parts of a channel, the data from the simulation does not indicate any strong correlation between the fault uplift and curvature of the channels. This means that curvature data alone is not sufficient by itself to identify the uplift pattern of a region.

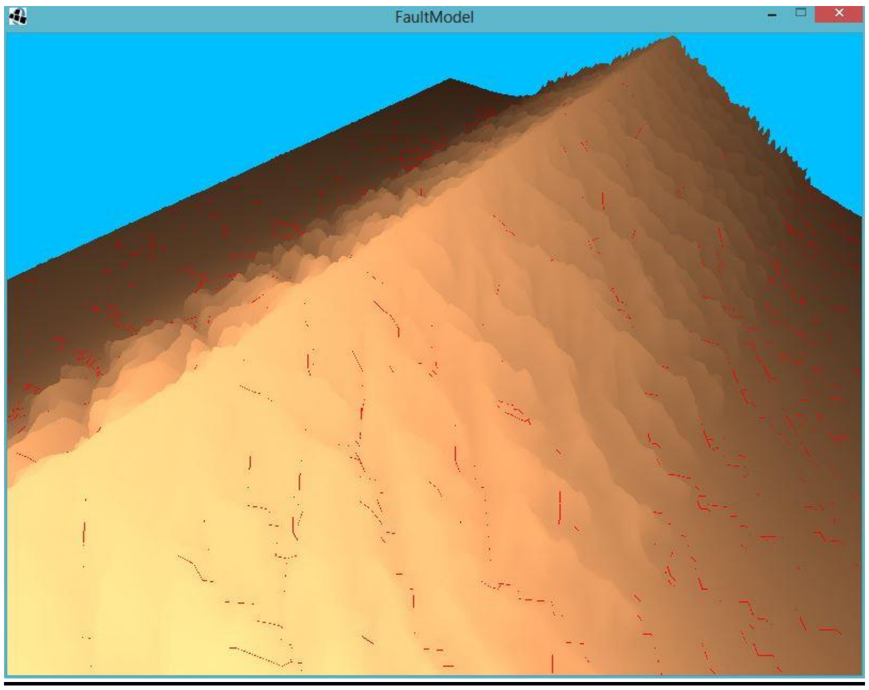
As with previous models[3], my model assumes the temporal variation is the same as spatial variation. Which means it assumes the variation across time and variation across distance are the same. The tools developed in this research can be used as a starting point to build more sophisticated and specific models and data collection method. More data need to be collected from various climate and region in order to prove the validity of the tools. The simulation itself will also require further testing. As the data shown only represents results from one variation of *wk* variables and initial topography. More combinations of *wk* variable that represents different types of climate have to be tested as well as other initial topography for the simulation. The mesh size of the simulation should also be increased since the erosion algorithms assume large topography. Since the tools develop here also takes flexibility into consideration, expansion should not be difficult.

<u>Code Sample</u>

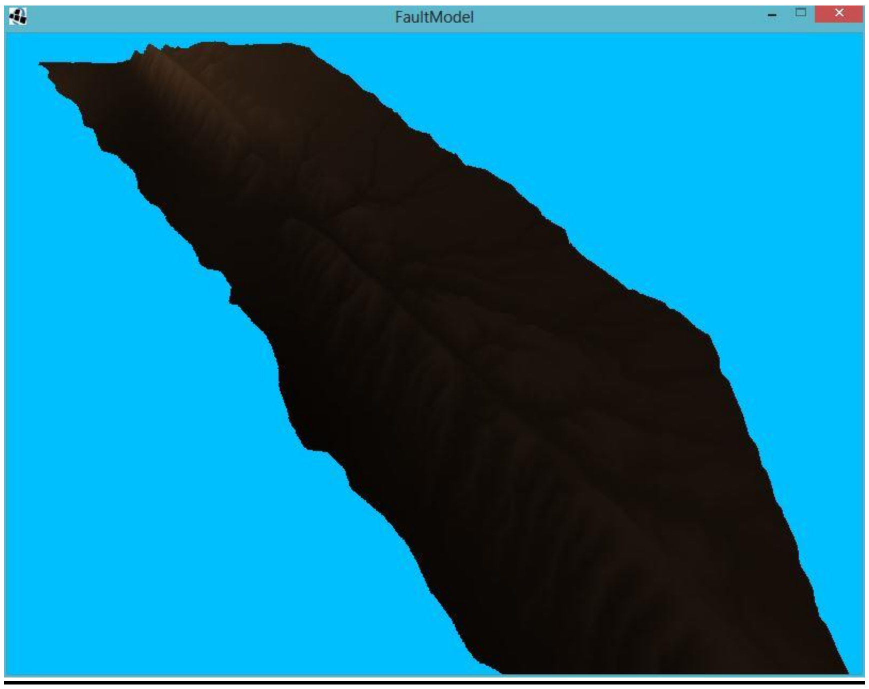
All codes are written by me.

```
//Erosion algorithm, takes a point, returns next point to erode
private Point erode(Point p){
    //position of the point being evaluated
    int i = p.x, j = p.y;
    Vector3f vec = masterMesh[i][j];//vector at this point
    float h = vec.x;//current height
    float delx0, delx1;
    float[] deltaH;//change in height for surrounding
    float greatV;//greatest deltaH
    float distance;//distance between points
    Point nextP;//next point to process
    Vector3f nextVec;//vector at next point

    deltaH = getDeltaH(h,i,j);//get height of its eight surrounding neighbors
    greatV = getGreatV(deltaH);//find the largest height

    //if its not the lowest
    if(greatV > 0){
        //setup next point to process/return
        nextP = new Point(i + posX, j + posY);
        //get the next vector to manupilate value
        nextVec = masterMesh[nextP.x][nextP.y];
        distance = getDistance()
        wk_slp = Math.abs(rise()/run());
        wk_c = (float) ((wk_fmax - wk_fmin) / (wk_fmax - 0.5 * wk_fmin) * wk_slp);

        //add appropriate sediment to walker
        delx0 = (wk_a * greatV + wk_b) / (greatV + wk_c) * greatV * (1/distance);
        walker += delx0;
        //remove 1/wk_carrylength amount of sediment from walker to be changed
        delx1 = walker/wk_carrylength;
        walker -= delx1;

        vec.setX(h - delx0);//actually change the height of the vectors
        nextVec.setX(nextVec.x + delx1);
        return nextP;//move to another point to process
    }
    else{//else if it is the lowest point
        nextP = new Point(i + posX, j + posY);//setup next point to process/return
        //if walker cant complete fill the hole
        if(-greatV > walker){
            vec.setX(h + walker);
            walker = 0; return p;
        }
        //dump just a little more than the lowestest point so walker can keep going
        else{
            delx0 = -greatV; walker -= delx0;
            float littleMore = 0.1f * walker;
            delx0 += littleMore; walker -= littleMore;
            vec.setX(h + delx0);
            return nextP;
        }
    }
}
//uplift function, uplifts this whole submesh
public void upLift(){
    float gX = 0, qY = 0, x = 0,y = 0;
    for(int i = 0; i < mesh.length; i++){
        for(int j = 0; j < mesh[0].length; j++){
            if(mesh[i][j].x != -1){
                x = findPerpendicularX(i,j);
                y = distanceFromFault(i,j);
                if(x < scaleLength){
                    gX = upLiftRate * (x - faultSlipRate*Mesh.getTime());
                }else{
                    gX = upLiftRate * (Math.abs(scaleLength - (x - scaleLength)) - faultSlipRate*Mesh.getTime());
                }
                if(y < scaleLength) qY = scaleLength - y;
                else qY = 0;
                mesh[i][j].x += gX * qY;
                mesh[i][j].y += faultSlipRate;
                mesh[i][j].z += faultSlipRate*Mesh.faultVector.x;
            }
        }
    }
}

//diffuse a single point
public void avgDiffuse(int i, int j){
    Vector3f vec = masterMesh[i][j];//vector at this point
    if(vec.x == -1.0f) return; //don run on "null" points
    float h = vec.x;//cur height
    float ht = 0;//avg height of neighbors
    int avaP = 0;//number of neighbors, prevents excessive diffuse on borders

    ht = findAvgNeighborHeight();

    float hg = (h - ht/avaP) * diffuseConstant;
    if(hg > 0){
        posClean(); posNext();
        float perNeighbor = (0.25f * (hg));
        for(int abc = 0; abc < 4; abc++){
            try{
                if(masterMesh[i + posX][j + posY].x != -1f){
                    masterMesh[i + posX][j + posY].x += perNeighbor;
                    vec.x -= perNeighbor;
                }
            }catch(Exception e){}
            if(abc == 1)posNext();
            else{ posNext();posNext();}
        }
    }
}
```

Program Snapshots



Simulation snapshot (red lines shows the flow of the rain drops)



DBPR snapshot

<u>References</u>

[1] Schumm, S.A. & Licthy, R.W. (1965). Time, space, and causality in geomorphology. American Journal of Science, 263, 110–119.

[2] Pazzaglia, F. "Landscape Evolution Models." *Developments in Quaternary Science* 1 (2003): 247-74. Print.

[3] Chase, Clement G. "Fluvial Landsculpting and the Fractal Dimension of Topography." *Geomorphology* 5 (1992): 39-57. Print.

[4] Schneider, Philip, and Barbara Schauer. *HAZUS MH Estimated Annualized Earthquake Losses for the United States.* Washington, D.C.: FEMA, 2008. Print.

[5] Crave, A., and P. Davy. "A Stochastic ‘‘precipiton’’ Model for Simulating Erosion/sedimentation Dynamics." *Computers & Geosciences* 27 (2001): 815-27. Print.

[6] Egan, S., S, S. Kane, T. Buddin, S, G. Williams, D, and D. Hodgetts. "Computer Modelling and Visualisation of the Structural Deformation Caused by Movement along Geological Faults." *Computers & Geosciences* 25 (1999): 283-97. Print.

[7] Harmon, R. S., and William W. Doe. *Landscape Erosion and Evolution Modeling*. New York: Kluwer Academic/Plenum, 2001. Web.

[8] Hilley, George E., and J. Ramón Arrowsmith. "Geomorphic Response to Uplift along the Dragon's Back Pressure Ridge, Carrizo Plain, California." *Geology* 36.5 (2008): 367. Print.