
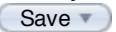


From: Archie Paulson <archie.paulson@colorado.edu>
Subject: Re: negative x_terminus
Date: March 6, 2008 7:15:40 PM MST
To: Chris Malley <cmalley@pixelzoom.com>
Cc: Wendy Kristine Adams <Wendy.Adams@colorado.edu>
 1 Attachment, 14.8 KB 

Chris,

The x_terminus function ends up being a little convoluted in order to accommodate the changing valley floor elevation. But it won't add any significant processing time.

First, let's say the valley floor is evaluated with the following function:

```
F(x) = max( 0,  
           4e3-(x/30.) + exp( -( x - hw_steep ) / hw_length ) )
```

where hw_steep=5e3 and hw_length=1e3
(here 'hw' refers to the headwall, which is the steeper part of the valley at the beginning)

Now define the following function:

```
x_terminus_bulk(ela)= 170.5e3-41.8*ela
```

Next, set a couple of parameters:

```
max_F = max(F)  
x_term_alter_x = max_F - 0.15*hw_length  
x_term_diff = -x_terminus_bulk(max_F) / ( max_F - x_term_alter_x )**2
```

The calculations above need to be done only once, when the sim is started up.

Now, to find x_terminus from a given ela, use the following:

```
if ela > max_F:  
    x_terminus = 0  
    H_max = 0  
    return  
x_terminus = x_terminus_bulk(ela)  
if ela > x_term_alter_x:  
    x_terminus += x_term_diff*( ela - x_term_alter_x )**2  
H_max = max(0.0, 400.-(1.04e-2*ela-23)**2 )
```

The H_max calculation is the same, except that I put in some code to make sure it's never negative.

The attached model.py implements the algorithm above. Let me know if there's any other problems.

-Archie

On Thursday 06 March 2008 17:14:41 Chris Malley wrote:

```
F(x) = Math.max( 0, 4e3 - ( x / 30. ) + Math.exp( -( x - 5e3 ) /  
1e3 ) );
```

On Mar 6, 2008, at 4:48 PM, Archie Paulson wrote:

Chris,

The fit to the x_terminus data depends on the shape of the valley floor.
Would you let me know the function (that I called "F") that you use?

–Archie

On Thursday 06 March 2008 15:31:45 Chris Malley wrote:

Archie,

I opened a ticket for this:

<https://phet.unfuddle.com/p/unfuddled/tickets/show/350/cycle>

Chris

On Mar 5, 2008, at 7:07 PM, Archie Paulson wrote:

Chris,

Yes I see the issue. The x_terminus equation should be such that when the ela equals the altitude of the highest point in the valley, x_terminus=0.

I'll take another look and tweak it a bit so that's true.

About the min/max ELA, note that a lower p0 means more precipitation and therefore a lower ELA. So you should get:

max_ELA = 4518 (t0=20,p_max=0,p0=6000)

min_ELA = 2165 (t0=13,p_max=4,p0=2200)

(see how I swapped the p0 values)

I've attached a document called sample.txt that shows a run of the model.py in the interactive python environment (execute 'ipython -pylab model.py' at the command line). I've also attached a very slightly changed model.py (no change in algorithm) that you should use.

I'll get back to you soon about a better equation for x_terminus.

–Archie

On Wednesday 05 March 2008 16:47:24 Chris Malley wrote:

Archie,

We have a small issue with this part of the model:

$x_{\text{terminus}} = 170.5e3 - 41.8 * \text{ela}$

With our current range of climate parameters,

max_ELA = 4500 (t0=20,p_max=0,p0=2200)

min_ELA = 2606 (t0=13,p_max=4,p0=6000)

The problem occurs as we approach max_ELA. At max_ELA:

$$x_{\text{terminus}} = 170.5e3 - 41.8 * 4500 = -17600$$

We can't have a negative glacier length.

Please confirm that I have the correct min/max ELA values, in case there's a bug in my implementation of the climate model.

Chris

On Mar 2, 2008, at 7:58 PM, Archie Paulson wrote:

Chris,

So here's the new model. I expect this one to be good enough to not change any more. A working implementation of this model is in the attached Python code.

It works like this:

- the climate model (unchanged) is used to compute an ELA
- the entire equilibrium glacier shape is determined from the ELA
- if the climate is changed, the ELA also changes (immediately);
- then a "psuedo-ELA" is computed that exponentially evolves from the former ELA to the new (current) ELA, and the glacier shape is then governed by this psuedo-ELA

The equilibrium shape is calculated as follows:

1. given an ELA (stored in variable `ela`), compute
$$x_{\text{terminus}} = 170.5e3 - 41.8 * ela$$
$$H_{\text{max}} = 400. - (1.04e-2 * ela - 23)^2$$
$$x_{\text{peak}} = 0.5 * x_{\text{terminus}}$$
2. compute the glacier height (H) at each x -value (x), where both H and x arrays are indexed by i , as follows:
 - for each i :
 - if $x[i] < x_{\text{peak}}$:
$$p = 42 - 0.01 * ela$$
$$f = 1.5$$
$$r = f * x_{\text{peak}}$$
$$H[i] = \sqrt{r^2 - (x[i] - x_{\text{peak}})^2} * H_{\text{max}} / r$$
$$H[i] *= (x_{\text{peak}}^p - (\text{abs}(x[i] - x_{\text{peak}})^p)) / x_{\text{peak}}^p$$
 - elif $x[i] < x_{\text{terminus}}$:
$$H[i] = \sqrt{x_{\text{peak}}^2 - (x[i] - x_{\text{peak}})^2} * H_{\text{max}} / x_{\text{peak}}$$
 - else:
$$H[i] = 0.0$$

Obviously the syntax there is pretty much Python. Let me know if anything is ambiguous. Please see the attached python code for a running example.

The ice velocities are computed the same way as before, using the current glacier height.

The three controls govern three variables that I've called t_0 , p_0 and p_{\max} .

These are what we've called "sea level temperature", "snowfall transition elevation" and "max snowfall", respectively. Note that p_{\max} is twice the value of the snowfall at elevation p_0 , which is on the current control.

The ranges for these values should be:

$t_{0_min}, t_{0_max} = 13., 20.$

$p_{0_min}, p_{0_max} = 2.2e3, 6e3$

$p_{\max_min}, p_{\max_max} = 0.0, 4.0$

With these ranges, the longest glacier you get is just shy of 80 km.

Let me know if there's anything not clear.

-Archie<model.py> <lib.py>

<sample.txt> <model.py>



[model.py \(14.8 KB\)](#)