Titan 3.1.1: A modified titan2d for simulating dry avalanche flows over general terrain

Li Yuan, Wei Liu

LSEC and Institute of Computational Mathematics
Academy of Mathematics and Systems Science
Chinese Academy of Sciences, Beijing
Email: lyuan@lsec.cc.ac.cn

March 14, 2016

Contents

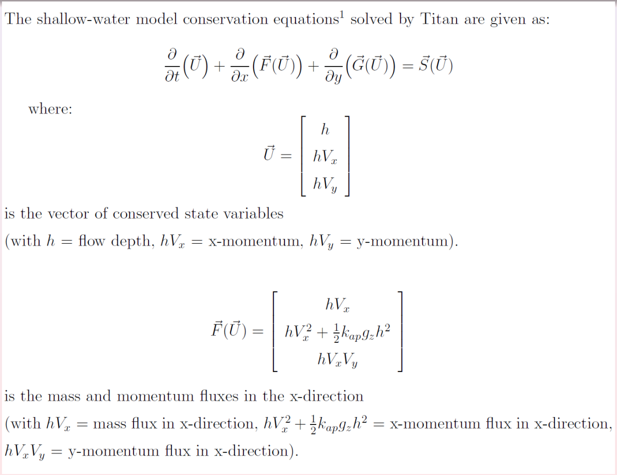
Outline

1. A glimpse of TITAN2D code

- An open source software developed by Geophysical Mass Flow Group, State University of New York at Buffalo. The code (version $\leq$ 3.0.0) is based on a formalism of shallow granular flow model (Iverson & Denlinger 2001JGR) and uses a Godunov type finite volume method incorporated with DEM. Intended for simulating dry granular avalanches over natural terrain.
- Preprocess a GIS-based map layer and extract DEM data in raster format by using GRASS (a free GIS software), then compute slope, curvature in code. One can also create artificial slope in "X|Y|Z" format which can be processed with GRASS.
- Use structured Cartesian grids, parallel adaptive mesh refinement, C++ & Fortran, MPI (+OpenMP).
- Easy to use: vhub.org (online practice on the "Colima" case).

Outline

2. Numerical method: Old equations in TITAN 3.0.0



The shallow-water model conservation equations[1] solved by Titan are given as:

$$\frac{\partial}{\partial t}\left(\vec{U}\right)+\frac{\partial}{\partial x}\left(\vec{F}(\vec{U})\right)+\frac{\partial}{\partial y}\left(\vec{G}(\vec{U})\right)=\vec{S}(\vec{U})$$

where:

$$\vec{U}=\begin{bmatrix} h \\ hV_x \\ hV_y \end{bmatrix}$$

is the vector of conserved state variables

(with h = flow depth, hV_x = x-momentum, hV_y = y-momentum).

$$\vec{F}(\vec{U})=\begin{bmatrix} hV_x \\ hV_x^2+\frac{1}{2}k_{ap}g_z h^2 \\ hV_xV_y \end{bmatrix}$$

is the mass and momentum fluxes in the x-direction

(with hV_x = mass flux in x-direction, $hV_x^2+\frac{1}{2}k_{ap}g_z h^2$ = x-momentum flux in x-direction, hV_xV_y = y-momentum flux in x-direction).

Fig.1: Old equations from Denlinger-Iverson 2001 JGR in a local Cartesian coordinate system, but TITAN2D applies them to global Cartesian system.

$$\vec{G}(\vec{U}) = \begin{bmatrix} hV_y \\ hV_xV_y \\ hV_y^2 + \frac{1}{2}k_{ap}g_zh^2 \end{bmatrix}$$

is the mass and momentum fluxes in the y-direction
(with hV_y = mass flux in y-direction, hV_xV_y = x-momentum flux in y-direction, $hV_y^2 + \frac{1}{2}k_{ap}g_zh^2$ = y-momentum flux in y-direction).

$$\vec{S}(\vec{U}) = \begin{bmatrix} 0 \\ g_xh - hk_{ap}\text{sign}\left(\frac{\partial V_x}{\partial y}\right)\frac{\partial}{\partial y}(g_zh)\sin\phi_{int} - \frac{V_x}{\sqrt{V_x^2+V_y^2}}\max\left(g_z + \frac{V_x^2}{r_x}, 0\right)h\tan\phi_{bed} \\ g_yh - hk_{ap}\text{sign}\left(\frac{\partial V_y}{\partial x}\right)\frac{\partial}{\partial x}(g_zh)\sin\phi_{int} - \frac{V_y}{\sqrt{V_x^2+V_y^2}}\max\left(g_z + \frac{V_y^2}{r_y}, 0\right)h\tan\phi_{bed} \end{bmatrix}$$

is the vector of driving and dissipative source terms
(with g_xh = driving gravitational force in x-direction, $-hk_{ap}\text{sign}\left(\frac{\partial V_x}{\partial y}\right)\frac{\partial}{\partial y}(g_zh)\sin\phi_{int}$ = dissipative internal frictional force in x-direction, $-\frac{V_x}{\sqrt{V_x^2+V_y^2}}\max\left(g_z + \frac{V_x^2}{r_x}, 0\right)h\tan\phi_{bed}$ = dissipative basal frictional force in x-direction; similar terms for y-direction).

Fig.2: Old equations in previous TITAN2D, $g_z = g\cos\theta, g_x = -g_z\frac{\partial b}{\partial x}$.

Review of SH models for arbitrary topography

- Gray (1999), Pudasaini (2003) gave a Savage-Hutter (SH) type formulation in general curvilinear coordinate system fitted to three-dimensional topography. Need grid generation on topography.
- Bouchut (2004) used a global Cartesian coordinate system and used thickness in the direction normal to the bed and velocity in the plane tangential to the bed as solution variables.
- Denlinger (2004) used a global Cartesian coordinate system (with z vertical). Provide canonical conservative flux terms. The basal and internal stresses are calculated with a finite element method.
- Castro-Orgaz(2014) developed Boussinesq type gravity wave theory for granular media.

New equations (Yuan et al, arXiv 1477195, 2016)

Following Denlinger-Iverson (JGR2004), Castro (JGR2014), Patra (2005), Gray (2003), and Bouchut (2004), we derived a new set of momentum equations (3) and (4) in a global Cartesian coordinate system. The normal stress acting on the bed is $\mathbf{n}_b \cdot \boldsymbol{\tau}_b \cdot \mathbf{n}_b = \rho\beta g' h$, where $g' = g + \frac{d\bar{w}}{dt}$, $\beta^{-1} = 1 - \dfrac{\Delta_b w_r}{|\mathbf{v}_r|} \tan\phi_{\text{bed}}, w_r = \bar{u}\partial b/\partial x + \bar{v}\partial b/\partial y$, $\Delta_b = \sqrt{1 + (\partial b/\partial x)^2 + (\partial b/\partial y)^2}$, $|\mathbf{v}_r| = \sqrt{\bar{u}^2 + \bar{v}^2 + w_r^2}$.
By letting $\rho\beta g' h = \rho g h_{\text{n}} \cos\theta$, where h_{n} is the depth in the direction normal to the bed, θ is the angle between the vertical z-axis and the bed normal, we obtain a new formula for the total vertical acceleration

$$g' = \frac{1}{\beta} \frac{g\cos^2\theta}{1 + \omega \left(\frac{\partial h}{\partial x}\frac{\partial b}{\partial x} + \frac{\partial h}{\partial y}\frac{\partial b}{\partial y} \right) \cos^2\theta}. \qquad (1)$$

where $\tan\theta = \sqrt{b_x^2 + b_y^2}$, $\omega = \exp(-\gamma|\mathbf{v}_r||\nabla h|)$, and $\gamma = 1 \sim 3$.

New equations used in Titan 3.1.1

But g' is computed with finite difference in Denlinger (2004) by definition

$$g' = g + \frac{d\bar{w}}{dt}, \text{ where } \frac{d\bar{w}}{dt} = \frac{\partial \bar{w}}{\partial t} + \bar{u}\frac{\partial \bar{w}}{\partial x} + \bar{v}\frac{\partial \bar{w}}{\partial y}. \qquad (2)$$

We add the centrifugal force due to curvature tensor $\mathcal{H}$ to the bed normal stress $\beta g' h$ in the RHS as per Bouchut (2004). In the global horizontal coordinates $Oxyz$, the new momentum equations are (overbars in $\bar{u}, \bar{v}$ have been dropped)

$$\begin{aligned}\frac{\partial (hu)}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}k_{\text{ap}} g' h^2\right) + \frac{\partial (huv)}{\partial y} &= -\frac{S}{2}\frac{\partial (g'h^2)}{\partial y} \\ -\left(\beta g' h + \frac{\mathbf{u}^T\mathcal{H}\mathbf{u}}{c^2}h_{\text{n}}\right)_+ &\left(\frac{u}{|\mathbf{v}_r|}\Delta_b \tan\phi_{\text{bed}} + \frac{\partial b}{\partial x}\right),\end{aligned} \qquad (3)$$

$$\begin{aligned}\frac{\partial (hv)}{\partial t} + \frac{\partial (huv)}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}k_{\text{ap}} g' h^2\right) &= -\frac{S}{2}\frac{\partial (g'h^2)}{\partial x} \\ -\left(\beta g' h + \frac{\mathbf{u}^T\mathcal{H}\mathbf{u}}{c^2}h_{\text{n}}\right)_+ &\left(\frac{v}{|\mathbf{v}_r|}\Delta_b \tan\phi_{\text{bed}} + \frac{\partial b}{\partial y}\right),\end{aligned} \qquad (4)$$

where $S = -\frac{1}{2}[\text{sgn}(\frac{\partial u}{\partial y})k_x + \text{sgn}\left(\frac{\partial v}{\partial x}\right)k_y]\sin\phi_{\text{int}}$, $\mathbf{u} = (u, v)^T$, curvature tensor is

$$\mathcal{H} = c^3\begin{bmatrix} b_{xx} & b_{xy} \\ b_{xy} & b_{yy}\end{bmatrix}, \quad c = \cos\theta. \qquad (5)$$

Numerical scheme: predictor-corrector with HLL flux

Denote $\mathbf{U}^n_{i,j}$ the cell average value of cell (i,j). Rewrite the SH equations as

$$\mathbf{U}_t + \mathbf{A}\cdot\mathbf{U}_x + \mathbf{B}\cdot\mathbf{U}_y = \mathbf{S}(\mathbf{U}) \tag{6}$$

Then

predictor step:

$$\mathbf{U}^{n+\frac{1}{2}}_{i,j} = \mathbf{U}^n_{i,j} - \frac{\Delta t}{2}\left(\mathbf{A}^n_{i,j}\bar{\Delta}_x\mathbf{U}^n_{i,j} + \mathbf{B}^n_{i,j}\bar{\Delta}_y\mathbf{U}^n_{i,j} - \mathbf{S}^n_{i,j}\right) \tag{7}$$

corrector step:

$$\mathbf{U}^{n+1}_{i,j} = \mathbf{U}^{n}_{i,j} - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2}\right] - \frac{\Delta t}{\Delta y}\left[\mathbf{G}_{j+1/2} - \mathbf{G}_{j-1/2}\right] + \Delta t\mathbf{S}_{i,j} \tag{8}$$

where $\bar{\Delta}_x\mathbf{U}, \bar{\Delta}_y\mathbf{U}$ are limited slopes, $\mathbf{F}_{i+1/2} = \mathbf{F}^{\mathrm{HLL}}(\mathbf{U}^l_{i+1/2}, \mathbf{U}^r_{i+1/2})$

$$\mathbf{U}^l_{i+1/2} = \mathbf{U}^{n+1/2}_{i,j} + \frac{\Delta x}{2}\bar{\Delta}_x\mathbf{U}^{n+1/2}_{i,j},\ \mathbf{U}^r_{i+1/2} = \mathbf{U}^{n+1/2}_{i+1,j} - \frac{\Delta x}{2}\bar{\Delta}_x\mathbf{U}^{n+1/2}_{i+1,j} \tag{9}$$

Jacobian matrix $\mathbf{A}_{ij}^n, \mathbf{B}_{ij}^n$

$$
\begin{aligned}
\mathbf{A} = \left.\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\right|_{g'=\text{const}} &= \begin{bmatrix} 0 & 1 & 0 \\ c^2 - u^2 & 2u & 0 \\ -uv & v & u \end{bmatrix}, \quad \begin{aligned} \lambda_1^A &= u - c \\ \lambda_2^A &= u \\ \lambda_3^A &= u + c \end{aligned} \\
\mathbf{B} = \left.\frac{\partial \mathbf{G}}{\partial \mathbf{U}}\right|_{g'=\text{const}} &= \begin{bmatrix} 0 & 0 & 1 \\ -uv & v & u \\ c^2 - v^2 & 0 & 2v \end{bmatrix}, \quad \begin{aligned} \lambda_1^B &= v - c \\ \lambda_2^B &= v \\ \lambda_3^B &= v + c \end{aligned}
\end{aligned}
\tag{10}
$$

where gravity wave speed $c = \sqrt{k_{ap} g' h}$.
The code directly gives analytical expression of $\mathbf{A}_{ij}^n \cdot \bar{\Delta}_x \mathbf{U}_{ij}^n$ in (7).

Option: HLLC flux

By default, TITAN2D uses HLL flux, but fluxes in (8) can also be HLLC flux:

$$\mathbf{F}^{\mathrm{HLLC}}_{i+\frac{1}{2}} = \begin{cases} \mathbf{F}(\mathbf{U}_L), & S_L \geq 0 \\ \mathbf{F}(\mathbf{U}_L) + S_L\left(\mathbf{U}_{*L} - \mathbf{U}_L\right), & S_L \leq 0 \leq S_* \\ \mathbf{F}(\mathbf{U}_R) + S_R\left(\mathbf{U}_{*R} - \mathbf{U}_R\right), & S_* \leq 0 \leq S_R \\ \mathbf{F}(\mathbf{U}_R), & S_R \leq 0 \end{cases} \tag{11}$$

In the x direction:

$$\mathbf{U}_{*K} = \frac{S_K - u_K}{S_K - S_*} \begin{bmatrix} h_K \\ h_K S_* \\ (hv)_K \end{bmatrix}, \quad K = L, R \tag{12}$$

In the y direction:

$$\mathbf{U}_{*K} = \frac{S_K - v_K}{S_K - S_*} \begin{bmatrix} h_K \\ (hu)_K \\ h_K S_* \end{bmatrix}, \quad K = L, R \tag{13}$$

Wave speed estimation

The left and right wave speeds are estimated as

$$\begin{aligned} S_L &= \min(u_L - c_L, u_R - c_R) \\ S_R &= \max(u_L + c_L, u_R + c_R) \end{aligned} \tag{14}$$

The middle wave speed is calculated according to Toro book

$$S_* = \frac{\frac{1}{2}(k_{ap}g'h^2)_R - \frac{1}{2}(k_{ap}g'h^2)_L + h_L u_L(S_L - u_L) - h_R u_R(S_R - u_R)}{h_L(S_L - u_L) - h_R(S_R - u_R)} \tag{15}$$

Outline

3. Code: 3.1 basic structure of TITAN2D

After unzipping the modified package in /titan-3.1.1/, type installation command "./install-titan.sh". **titan_gui.py** script is automatically copied to directory /bin from directory /scripts. The source files of TITAN2D are under different subdirectories of directory /src. These source files can be classified into two parts:

1. preprocess and postprocess parts: include subdirectories /preproc, /stochastic, /vectordataprepoc
2. flow calculation parts: /main + (/adapt, /datastr, /geoflow, /gisapi, /header, /repartition, /tecplot, /useful). Any change of any file in these subdirectories may result in change of execution file "titan" in /main. This requires recompiling "titan" by typing "make" in /src after any change.

The main program is /main/hpfem.C. Headers are mainly in /header like element2.h and geoflow.h, which declare functions. geoflow.h declares 4 extern "C": predict_( ), correct_( ), gmfggetcoef_( ), eigen_( ). They are actually Fortran subroutines under /geoflow: predict.f, correct.f, getcoef.f, eigen.f. The predict.f is called by predict_( ) in step.C, correct.f is called by correct_( ) in corrector.C, and gmfggetcoef( ) in getcoef.f and eigen() in eigen.f are called by get_coef_and_eigen.C. (Notice that the name of a calling-to-Fortran function in C++ has "_").

3.2 Error corrections to titan 3.0.0 (+0)

Important correction 1: Old Titan 3.0.0 often breaks down if using fixed fine meshes. It is found that **/main/init_piles.C** is responsible for this. It is required to refine the initial pile in function *init_piles* to the finest mesh:
old: if(adaptflag) initial_H_adapt
new: initial_H_adapt
(The corrected file with paraboloid is named as **"init_piles.C-new"**)

Important correction 2: in **/datastr/element2.C**, in member function *Element::calc_stop_crit*:
old: effect_tanbedfrict=matprops_ptr→tanbedfrict[material] (the RHS has no assigned value in the beginning).
new: effect_tanbedfrict=tan(effect_bedfrict).

Error corrections to titan 3.0.0 (+1)

Second-important correction 1: also *in Element::calc_stop_crit*, modify variable *"slopetemp"* by changing "×" to "+".
(With above two corrections, Element2.C is named as **element2.C-global**)
Second-important correction 2: in *Element::calc_edge_states*:

1) Numerical fluxes on the left/bottom boundary faces of the domain should not be assigned from opposite faces as the latter may not have value first. Left/bottom boundary fluxes are computed separately using cell average variables now.

2) For the case of S_S_CON, change to elm1→zdirflux(...side+2...), original **side** is wrong. (Suffix **-newbc-** of Element2.C marks corrections 1) and 2))

Second-important correction 3: in *Element::get_slopes*, we also use linear reconstruction in the middle portion of boundary. This can avoid smearing when computing 1D problem. Original TITAN uses constant reconstruction.
(Suffix **-newslope-** of Element2.C marks this correction)
(From **element2.C-global** → **element2.C-global-newslope-mn-newbc-stop** marks the second-important corrections 2 and 3 and stopping criteria in Sec. 3.3)

Error corrections to titan 3.0.0 (+2)

Minor corrections:

1. Old curvature is $\kappa = y''$. new exact curvature formula is $\kappa = y''/[1+(y')^2]^{3/2}$. In **/gisapi/GisApi.C**, modify functions *Get_curvature* and *calculate_curvature*(the latter has no use). And correct the interpolation position in *Get_elevation, Get_slope, Get_curcature.*
   (The corrected file is named **“GisApi.C-new”**)
2. In **/geoflow/predict.f-global & correct.f-global**: modify original curvature effects as $\bar{u}^2 \rightarrow u_\tau^2 = \bar{u}^2(1+b_x^2), \bar{v}^2 \rightarrow v_\tau = \bar{v}^2(1+b_y^2)$, where u_τ, v_τ are tangential velocity components.
3. Further, let $u^*, v^*, h^* \rightarrow u^n, v^n, h^n$ for $\mathbf{S}^n$, and adding stopping criteria, the two files evolve to **“predict.f-global-stop, correct.f-global-stop”.**
4. In **/header/geoflow.h**, old *c_sgn* function contains a cutoff parameter *GEOFLOW_TINY*. Change it to zero.
   (The corrected file is named as **“geoflow.h-new”**)
5. In **/tecplot/tecplot.C**, use $u = (hu)/h$ for consistency rather than *short_speed* got from function *eval_velocity*. New file is **“tectplot.C-new”**.

Error corrections to titan 3.0.0 (+3)
Corrected titan without stopping criteria

The resulting code with all aforementioned corrections and using Titan's old equations to compute a real terrain:
is to use the following candidate files as corresponding work files:
/tecplot/tecplot.C-new; /preprocess/createfunky.C-old;
/main/init_piles.C-new;
/header/: constant.h-1e-5, element2.h-old, geoflow.h-new;
/gisapi/: GisApi.C-new, GisApi.h-old;
/geoflow/: correct.f-global, corrector.C-old, get_coef_and_eigen.C-old, getcoef.f-old, predict.f-global, slopes.C-old, step.C-old; **/datstr/element2.C-global.**
For example, copy **element2.C-global** to **element2.C**.
Then type "make" under /src so as to create "titan" under /main.

3.3 Formula of stopping criteria

According to Mangeney(JGR2003), when the so-called admissible basal stress (also called *net driving force*) is less than the Coulomb friction value $\tau_{\max}$, the granular fluid must stop.
Our criterion follows Phys of Fluids 023302 (2013). First compute an admissible basal friction threshold $\tilde{\mathbf{T}}$ for cell (i, j) at time $n + 1$ by

$$\tilde{\mathbf{q}}_{i,j}^{n+1} = (h\mathbf{v})_{i,j}^{n} + \mathbf{C}_{i,j}^{n+1/2} + \Delta t \mathbf{g}_{i,j}^{n+1/2} \tag{16}$$

$$\tilde{\mathbf{T}}_{i,j}^{n+1} = \sqrt{|\tilde{\mathbf{q}}_{i,j}^{n+1}|^2 + \left[(\tilde{q}_x)_{i,j}^{n+1}(b_x)_{i,j} + (\tilde{q}_y)_{i,j}^{n+1}(b_y)_{i,j}\right]^2} \tag{17}$$

where $\mathbf{C}_{i,j}^{n+1/2}$ is the flux difference terms in (8), and $\mathbf{g}_{i,j}^{n+1/2}$ represents the first terms in RHS of (3) and (4). The stopping criteria says:

- If $|\tilde{\mathbf{T}}_{i,j}^{n+1}| < (\tau_{\max})_{i,j}^{n+1} \Delta t$ & $|\nabla H^{n+1}| < \tan\phi_{\text{bed}}$, then set $\mathbf{v}_{ij} = 0$;
- Otherwise, continue to solve the momentum equations with the Coulomb friction law.

3.4 Code improvement I (adding stopping criteria)

In /header/element2.h, the integer *stoppedflags* has following meaning

$$\text{stoppedflags} = \begin{cases} 0 & \text{not stopped} : \mathbf{v} \neq 0 \\ 1 & \text{not slide but slump} : \mathbf{v} = 0 \text{ but } \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} \neq 0 \\ 2 & \text{stopped: both } \mathbf{v} = 0 \text{ and } \frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = 0 \end{cases} \tag{18}$$

stoppedflags is initialized as 2 in class *Element* and later recomputed in member function *Element::calc_stop_crit*. But it is totally recomputed in the modified *predict.f* and *correct.f* in which the stopping criteria are implemented.

The resulting code with the stopping criteria (has suffix "-stop") and using Titan's old equations uses the same set of files as those for the corrected titan without stopping criteria except the following changes:
/geoflow/: correct.f-global-stop, predict.f-global-stop, /datstr/element2.C-global-newslope-mn-newbc-stop.
("**mn**" in element2.C means minmod limiter).

3.5 Code improvement II: 1) New Cartesian equations with total vertical acceleration (1)

Use equations (3),(4)+ total vertical acceleration (1) with under-relaxation factor ω. The resulting code to compute a real terrain is to use the following candidate files as work files: **/tecplot/tecplot.C-new; /preprocess/createfunky.C-old; /main/Init_piles.C-new; /header/: constant.h-1e-5, element2.h-old-newg, geoflow.h-stop-newg; /gisapi/: GisApi.C-newg-cur, GisApi.h-newg-cur; /geoflow/: corrector.f-global-stop-newg, corrector.C-newg-manning, get_coef_and_eigen.C-newg, getcoef.f-newg, predict.f-global-stop-newg, slopes.C-old, step.C-newg; /datastr/element2.C-global-newslope-mn-newbc-stop-hll-newg. This is the default setup of titan 3.1.1.**

Code improvement II: 2) with total vertical acceleration (2)

If we use equations (3),(4) with finite difference to Denlinger's acceleration (2), we must discard $\partial \bar{w}/\partial t$ in (2), and use limited $\widetilde{g'} = \min(\max(0.01, g'), 10g_0)$ to keep stability. The resulting code for real terrain uses the same set of files as those for the default code with total vertical acceleration (1) except the following changes:
/header/: element2.h-newg-den, geoflow.h-stop-newg-den; /geoflow/: get_coef_and_eigen.C-newg-den, getcoef.f-newg-den, predict.f-global-stop-newg, slopes.C-newg-den-good, step.C-newg-den; /datastr/ element2.C-global-newslope-mn-newbc-stop-hll-newg-den.
(parallel code using this gravity has bug but serial is OK)

3.5 Code improvement II: 3) using Manning friction

If we use the new Cartesian equations (3),(4) with the RHS terms being replaced by pseudo-Manning type terms:

$$s_x = -\frac{u\Delta_b}{\sqrt{u^2+v^2+w^2}}\frac{\tau_B}{\rho_m} - \frac{(g'h)\Delta_b n_m^2 u\sqrt{u^2+v^2+w^2}}{h_{\mathrm{n}}^{4/3}} - (g'h)\frac{\partial b}{\partial x},$$
$$s_y = -\frac{v\Delta b}{\sqrt{u^2+v^2+w^2}}\frac{\tau_B}{\rho_m} - \frac{(g'h)\Delta_b n_m^2 v\sqrt{u^2+v^2+w^2}}{h_{\mathrm{n}}^{4/3}} - (g'h)\frac{\partial b}{\partial y}, \quad (19)$$

where fluid density $\rho_m = 2.19\ \mathrm{kg/m}^3$, pseudo-Manning roughness coefficient $n_m = 0.15\ \mathrm{s\ m}^{-1/3}$ and Bingham yield stress $\tau_B = 1020\ \mathrm{N/m}^2$ in **corrector.f-global-stop-newg-Manning_im, predict.f-global-stop-newg-Manning**.
g' is the total vertical acceleration similar to (1) but is computed from Manning friction law. $g'h$ can have curvature effects as in (3),(4) .

Code using Manning friction law

If we use the new Cartesian equations (3),(4) with equation (19) being the RHS, then a resulting code uses uses the same set of files as those for the default code with total vertical acceleration (1) except the following changes:
/geoflow/: corrector.f-global-stop-newg-Manning_im, getcoef.f-newg(copy this to getcoef.f, and then modify $kactx = kacty = epsilon$ in it), predict.f-global-stop-newg-Manning; /datastr/element2.C-global-newslope-vl-newbc-stop-newg-manning.
("**vl**" in element2.C means Van Leer limiter)

3.6 Change code for different computational domain offsets and initial piles

1. Sometimes, you can obtain a full size of computational domain (produced in *funky0000.inp* by *titan_gui.py*) equal to original DEM region: modify variables (*xlength, ylength, xmin, xmax, ymin, ymax* = respective DEM boundary values) in **/preproc/createfunky.C**, then recompile file *titan_preprocess*, and then copy it to /bin to replace the old one. Script *titan_gui.py* will call *titan_preprocess*. This is useful for obtaining a full computation domain rather the default 0.98*DEM region.

   (New file is **"createfunky.C-wide"**, default is **"createfunky.C-old"**)

2. Change initial pile shape: modify file **/main/init_piles.C** for various shapes of the initial pile. Note that you need to modify both *#define* and function *elliptical_pile_height* in this file. There are already several candidate files for corresponding problems.

Outline

Future work

The grid convergence has been improved after discarding the momentums evaluated via “shortspeed” and modifying the interpolated points for elevation, slope, and curvature as relative to the left/top boundary cell’s centers rather than relative to the left/top boundaries in functions *Get_elevation*, *Get_slope*, *Get_curvature* in */gisapi/GisApi.C*.-new
But there are still other problems like contour distributions do not keep symmetry and regularity well.
Possible reasons are
1) decimal accuracy problem
2) dry-wet interface problem
3) ......

-end-