`nevis` : Documentation

This model is mostly as described in Hewitt (2013).[1] Continuum variables on a two-dimensional domain (hydraulic potential ϕ and water sheet thickness h) are discretised on a rectangular mesh of nodes. These nodes also form the basis of a discrete network of conduits (cross-sectional area S), connecting the nodes across each of the 8 compass points. Water is exchanged between englacial storage, water sheet and conduits such that the hydraulic potential is continuous.

Equations

Variables and parameters are summarized in Table 1.

Atmospheric and overburden potentials (corresponding to $p_w = 0$ and $p_w = p_i \equiv \rho_i g(Z_s - Z_b)$ respectively) are defined by

$$\phi_a(x,y) = \rho_w g Z_b, \qquad \phi_0(x,y) = \rho_i g Z_s + (\rho_w - \rho_i) g Z_b. \tag{1}$$

Ice pressure, water pressure and effective pressure are related to hydraulic potential $\phi(x,y,t)$ through

$$p_i(x,y) = \phi_0 - \phi_a, \qquad p_w(x,y,t) = \phi - \phi_a, \qquad N(x,y,t) = \phi_0 - \phi. \tag{2}$$

The distributed sheet is separated into a cavity sheet, with thickness $h_{cav}(x,y,t)$, and an elastic sheet, with thickness $h_{el}(x,y,t)$. By default these are added together to obtain the overall sheet thickness $h(x,y,t) = h_{cav} + h_{el}$, though they can alternatively be treated separately, with different discharge in each. The cavity sheet evolves according to

$$\frac{\partial h_{cav}}{\partial t} = \frac{\rho_w}{\rho_i} m + U_b (h_r - h_{cav})_+ / \ell_r - \hat{A} h_{cav} |N|^{n-1} N. \tag{3}$$

The elastic sheet has thickness related directly to the water pressure

$$h_{el} = h_c \left(\frac{p_w}{p_i} \right)^{\gamma} + h_{\varepsilon 2} \left[-N_- + \tfrac{1}{2} N_{\varepsilon 2} \left(1 - N_+ / N_{\varepsilon 2} \right)_+^2 \right], \tag{4}$$

where $N_- = \min(N,0)$ and $N_+ = \max(N,0)$. The second term here is designed to increase rapidly when N is negative, as a crude representation of hydraulic jacking. Discharge $\mathbf{q}(x,y,t)$ is given by

$$\mathbf{q} = -K_s h^{\alpha_s} |\nabla \phi|^{\beta_s - 1} \nabla \phi. \tag{5}$$

Basal melting in the sheet $m(x,y)$ is either prescribed, or should be coupled to ice sliding speed and basal shear stress according to (any conduction into the ice is absorbed into G here)

$$m = \frac{G + \boldsymbol{\tau}_b \cdot \mathbf{u}_b}{\rho_w L}. \tag{6}$$

Each conduit has cross-sectional area $S(s,t)$ (s is distance along the conduit) evolving according to

$$\frac{\partial S}{\partial t} = \frac{\rho_w}{\rho_i} M + U_b h_{rc} (1 - S/S_{rc})_+ - \tilde{A} S |N|^{n-1} N. \tag{7}$$

Discharge $Q(s,t)$ is given by

$$Q = -K_c S^{\alpha_c} \left| \frac{\partial \phi}{\partial s} \right|^{\beta_c - 1/2} \frac{\partial \phi}{\partial s}. \tag{8}$$

[1]Hewitt, I.J. 2013 Seasonal Changes in Ice Sheet Motion due to Meltwater Lubrication *Earth and Planetary Science Letters*, **371–372**, 16-25.

Melting $M(s,t)$ is given by

$$M = \frac{1-\rho_w c\beta}{\rho_w L}\left|Q\frac{\partial\phi}{\partial s}\right| - \frac{\rho_w g c\beta}{L}Q\frac{\partial Z_b}{\partial s} + \lambda\frac{|\mathbf{q}\cdot\nabla\phi|}{\rho_w L}. \tag{9}$$

Mass conservation is expressed as

$$\frac{\partial h}{\partial t} + \nabla\cdot\mathbf{q} + \left[\frac{\partial S}{\partial t} + \frac{\partial Q}{\partial s}\right]\delta(\mathbf{x}_c) + \frac{\partial\Sigma}{\partial t} = m + M\delta(\mathbf{x}_c) + E, \tag{10}$$

where englacial storage $\Sigma(x,y,t)$ is a function of water pressure

$$\Sigma = \sigma\frac{p_w}{\rho_w g} + A_m\frac{p_w}{\rho_w g}\delta(\mathbf{x}_m), \tag{11}$$

and the delta functions apply along the (linear) positions of the conduits $\mathbf{x}_c(s)$ and the (point) positions of the moulins $\mathbf{x}_m$. The source term $E(x,y,t)$ can include both distributed and moulin point sources.

Boundary conditions are required on either the pressure or the discharge at all boundaries of the domain. Initial conditions are required for ϕ, h and S.

Numerical structure of the model

An example model run is given in `nevis_example.m`.

The computations are all performed using non-dimensional variables; that is, each variable is scaled by an appropriate value. This non-dimensionalization is performed before calculations begin, and the variables that are subsequently used and saved are all non-dimensional. In order to plot the dimensional quantities, the quantities must be multiplied by the appropriate scale.

Everything is stored in Matlab structures:

`pd` dimensional parameters
`oo` options
`ps` scales used for non-dimensionalization
`pp` parameters (non-dimensional)
`gg` grid and discretization (non-dimensional)
`aa` prescribed fields and boundary conditions (non-dimensional)
`vv` current solution variables (non-dimensional)
`tt` time series of solution summary (non-dimensional)

Default parameters and options are assigned and then non-dimensionalized by

```
>pd = struct; oo = struct;
>[pd,oo] = nevis_defaults(pd,oo);
>[ps,pp] = nevis_nondimension(pd);
```

Non-default values can be assigned to `pd` or `oo` before calling `nevis_defaults`.

A grid is set up by

```
>x = linspace(0,10000/ps.x,50); y = linspace(0,10000/ps.x,50);
>gg = nevis_grid(x,y,oo);
>gg = nevis_mask(gg,nout);
>gg = nevis_label(gg,nbdy);
```

`nevis_grid` populates `gg` with the coordinates of nodes and edges, as well as derivative and mean matrix operators. `x` and `y` define the x and y coordinates for the rectangular grid (note the scaling

Primary variables [vv]	$\phi(x, y, t)$	phi	Hydraulic potential
	$h_s(x, y, t)$	hs	Cavity sheet depth
	$S(s, t)$	Sx,Sy,Ss,Sr	Conduit cross-sectional area
Derived variables [vv]	$h_e(x, y, t)$	he	Elastic sheet depth
	$N(x, y, t)$	N	Effective pressure in sheet
	$Q(s, t)$	Qx,Qy,Qs,Qr	Discharge in channel
	$\mathbf{q}(x, y, t)$	qx,qy	Discharge in sheet
Prescribed fields [aa]	$Z_b(x, y)$	b	Bed elevation
	$Z_s(x, y)$	s	Surface elevation
	$\phi_a(x, y)$	phi_a	Atmospheric potential at bed elevation
	$\phi_0(x, y)$	phi_0	Overburden potential
	$m(x, y)$	m	Basal source
	$E(x, y, t)$	E	Englacial source
	$\sigma(x, y)$	sigma	Englacial void fraction connected to sheet
	$\lambda(x, y)$	lcx,lcy,lcs,lcr	Sheet width contributing to conduit melting
Parameters [pd]	ρ_w	rho_w	Density of water [1000 kg m^{-3}]
	ρ_i	rho_i	Density of ice [910 kg m^{-3}]
	g	g	Gravitational acceleration [9.81 m s^{-2}]
	L	L	Latent heat of melting [3.35×10^5 J kg^{-3}]
	c	c	Specific heat capacity of water [4200 J kg^{-1} K^{-1}]
	β	gamma_cc	Melting point pressure gradient [0 K Pa^{-1}]
	G	G	Geothermal heat flux [0.063 W m^{-2}]
	n	n_Glen	Ice flow law exponent [3]
	A	A	Ice flow law coefficient [6.8×10^{-24} Ps^{-3} s^{-1}]
	$\tilde{A}$	K_s	Modified ice flow law coefficient in channel [5.04×10^{-25} Ps^{-3} s^{-1}]
	$\hat{A}$	K_c	Modified ice flow law coefficient in sheet [5.04×10^{-25} Ps^{-3} s^{-1}]
	α_c	alpha_c	Conduit flux exponent [5/4]
	β_c	beta_c	Conduit pressure gradient exponent [1/2]
	α_s	alpha_s	Sheet flux exponent [3]
	β_s	beta_s	Sheet pressure gradient exponent [1]
	K_c	k_c	Conduit flux coefficient [0.1 m s^{-1} $\mathrm{Pa}^{-1/2}$]
	K_s	k_s	Sheet flux coefficient [10^{-4} Pa^{-1} s^{-1}]
	h_r	h_r	Sheet roughness height [0.1 m]
	ℓ_r	l_r	Sheet roughness length [10 m]
	λ	l_c	Default sheet width contributing to conduit melting [10 m]
	U_b	u_b	Default basal sliding speed [60 m/ y]
	σ	sigma	Default englacial void fraction [0]
	m	melt	Default basal melt rate [0.0059 m/ y]
	A_m	A_m	Moulin cross-sectional area [10 m^2]
	h_{rc}	h_rc	Conduit roughness height [0 m]
	S_{rc}	S_rc	Conduit area cutoff [0 m^2]
	γ	gamma_e	Elastic sheet exponent [1]
	h_c	c_e_power	Elastic sheet depth scale [0 m]
	$h_{\varepsilon 2}$	c_e_reg2	Uplift regularization rate [0 m Pa^{-1}]
	$N_{\varepsilon 2}$	N_reg2	Regularizing pressure for uplift regularization [10^3 Pa]
	Ψ_ε	Psi_reg	Regularizing potential gradient [0.1 Pa m^{-1}]
	N_ε	N_reg	Regularizing pressure for elastic sheet [10^3 Pa]
	p_ε	p_a_reg	Pressure tolerance for boundary adjustment [9810 Pa]

Table 1: Variables, prescribed inputs, and parameters with default values. Note that several of the parameters have different names in the code due to historical legacies.

by `ps.x` to make the quantities non-dimensional). `oo` can include optional flags for how to assign the coordinates.

`nevis_mask` adds labels to `gg` to identify which nodes and edges are inside, outside, and on the boundary of the domain, using the node indices `nout` to define the region outside the domain.

`nevis_label` re-labels the boundary nodes `nbdy` at which the hydraulic potential is to be prescribed, and the adjoining edges as inside, outside or on the boundary of the domain.

The definition of `nout` and `nbdy` depend on the specific problem and may for instance be defined based on ice thickness, or using a pre-existing mask.

Prescribed fields and initial conditions are assigned by

```
>b = (0/ps.z)*gg.nx.^0;
>s = (100/ps.z)*gg.nx.^0;
>[aa,vv] = nevis_initialize(b,s,gg,pp,oo);
```

`nevis_initialize` populates `aa` with default prescribed fields, and `vv` with default initial conditions for the primary variables (generally these won't be sensible initial conditions, so new ones should be assigned subsequently). `b` and `s` are matrices or vectors defining the bed and surface elevation, here taken to be uniform at 0 m and 100 m for illustration.

The model is solved by

```
>[tt,vv] = nevis_timesteps(t_span,vv,aa,pp,gg,oo);
```

`nevis_timesteps` solves the equations over the timespan `t_span`, with initial conditions defined by input `vv`. The output `vv` contains the solution at the final time, and `tt` contains basic summary information about the solution at each timestep. Depending on the options in `oo`, the solution at intermediate times in `t_span` is also saved to file.

The current solution (defined by contents of `vv`) can be plotted using

```
>nevis_plot;
```

(what is actually plotted depends on options in `oo`; see `nevis_plot.m`), or by typing things like

```
>imagesc(ps.x*gg.nx,ps.x*gg.ny,ps.phi*reshape(phi,gg.nI,gg.nJ));
```

(note how the scalings in `ps` are used to convert the non-dimensional quantities to dimensional ones).

Domain and boundary conditions

Nodes are labelled as inside the solution domain (indices `gg.ns`) or outside (`gg.nout`). Similar labels are assigned to the edges and corners. Those indices on which boundary conditions are prescribed are labelled `gg.nbdy`, `gg.ebdy`, or `gg.fbdy`. The method of labelling is described below.

By default, the hydraulic potential on indices `gg.nbdy` is set to `phi_a`, and the discharge on all boundary edges `gg.ebdy`, `gg.fbdy` is set to `0`. If other values are to be assigned, these should be included in `aa`; *e.g.* `aa.phi` can be defined as a list of values for the hydraulic potential on the nodes `gg.nbdy`.

By default the exterior boundaries of the grid set up by `nevis_grid` are reflecting and there are no boundary nodes or edges (the exterior boundaries can instead be made periodic using `oo.xperiodic` and `oo.yperiodic`). Usually, however, a solution domain that is strictly inside the rectangular grid is defined, so that these exterior boundaries are irrelevant. This is performed using

```
>gg = nevis_mask(gg,nout);
```

`nevis_mask` labels node indices `nout` as outside the domain, the remaining nodes as inside the domain, edges that are connected to at least one inside node as inside the domain, and corners that are surrounded by inside nodes as inside the domain. It also identifies which nodes are on the boundary of the domain (`gg.n1`), which edges connect inside nodes to outside nodes (`gg.e1`, `gg.f1`), and by default it labels these edges as the boundary edges `gg.ebdy`, `gg.fbdy` on which discharge is prescribed.

To prescribe pressure on certain nodes (usually those at the margin), use

```
>gg = nevis_label(gg,nbdy);
```

which assigns the list of node indices `nbdy` as the boundary nodes `gg.nbdy` and removes the adjoining edges from the labels `gg.ebdy`, `gg.fbdy`.

If the option `oo.adjust_boundaries` is set, the labelling of boundary nodes may be adjusted dynamically during timestepping (see below). In that case, a list of current boundary nodes is stored in `vv.nbdy`, and the grid labelling is updated at each timestep using `nevis_label`. This option is incompatible with prescribing non-default boundary values in `aa`.

Initial conditions

The default initial conditions assigned by `nevis_initialize` have zero sheet and conduit sizes, which will usually not work. Starting with a uniform sheet and with pressure at some percentage of overburden often seems to work,

```
>vv.phi = aa.phi_a + 0.9*(aa.phi_0-aa.phi_a);
>vv.hs = (0.1/ps.h)*gg.nx.^0;
```

It is usually a good idea to run the model with a constant input for some time before beginning calculations in earnest, to avoid a significant transient from the initial condition.

Moulins

Moulins are defined with a list of node indices `pp.ni_m` for the moulins. In addition, either a function handle `pp.input_function(t)` defining the input to the moulins, or a matrix `pp.sum_m` defining the catchment areas, should be provided. A list of corresponding moulin labels `pp.num_m` can also be defined. The function

```
>[pp.ni_m,pp.sum_m] = nevis_moulins(x_m,y_m,gg,oo);
```

finds the node indices and the catchment areas (based on a Voronoi tesselation) for moulins with coordinates `x_m`,`y_m`. If `oo.random_moulins` is non zero, that number of moulins will be located at random.

Inputs

The input `aa.E` is updated at each time `t` using the function

```
>aa = nevis_inputs(t,aa,pp,gg,oo);
```

`nevis_inputs` assigns the input in different ways depending on the options in `oo`:

If no moulins have been defined, the default is to use function handle `pp.meltE(t)` and lapse rate `pp.E_lapse`, modulated by diurnal amplitude `pp.E_amp`, to define a distributed runoff.

If moulins have been defined (*i.e.* `pp.ni_m` is non-empty), the default is to calculate the same runoff but concentrate it into the moulins according to catchment areas defined by `pp.sum_m` (distributed input can still be enforced by setting `oo.distributed_input`).

If `oo.runoff_function`, the runoff is instead calculated using the function handle `pp.runoff_function(t)` instead (this function should return runoff on each node).

If `oo.input_function`, the function handle `pp.input_function(t)` is used to define the input to the moulins (this function should return runoff to each moulin). This overrides the options above.

Timestepping

Timestepping is performed using

```
>[tt,vv] = nevis_timesteps(t_span,vv,aa,pp,gg,oo);
```

The start and end times are defined by the first and last entries of vector `t_span` and the initial conditions are defined by `vv`. The initial timestep is defined by `oo.dt`. The procedure at each time step is first to update the inputs for the current time; then calculate the summary information save in `tt` (see below); possibly write the current solution to file (see below); then attempt a timestep (with a Newton iteration performed by the function `nevis_timestep`; adjusting the timestep if required, and increasing the suggested next timestep if the iteration was very quick); then finally adjusting the boundary nodes for the next timestep (see below).

The bulk of the work in solving the equations is in the function `nevis_timestep`, which performs a Newton iteration and calls on the function `nevis_backbone` to evaluate the residuals and Jacobian. This function also calculates the derived variables such as discharge (on the nodes) and effective pressure. They can be added to the solution structure `vv` by

```
>[vv] = nevis_backbone(inf,vv,vv,aa,pp,gg,oo);
```

If `oo.change_timestep`, the timestep may be adjusted based upon the failure or success of previous iterations (see `nevis_timesteps.m`).

The structure `tt` stores summary information for every timestep taken. This includes total inflow `Q_in`, `m` and `E`, total outflow `Q_out`, spatially averaged hydraulic potential `phi` and effective pressure `N`, total cavity sheet volume `hs` and channel volume `S`.

If the option `oo.save_pts_all`, the hydraulic potential `pts_phi` and cavity sheet depth `pts_hs` are saved at nodes defined by `pp.pts_ni` (this is useful for storing a time series of pressure in moulins, for example).

At each time in the vector `t_span`, the current solution `vv` and the summary structure `tt` are saved to file, with filename defined with the string `oo.fn`. In addition, if the option `oo.save_timesteps`, the current solution `vv` is saved to a file in a directory with name `oo.fn`, each such file being labelled sequentially. Thus the number and frequency of entries in `t_span` determines how often the solution is saved.

If `oo.adjust_boundaries`, the discharge into and out of the boundary nodes `gg.nbdy` is evaluated after each timestep. If there is inflow, this suggests the pressure should not be prescribed there and such nodes are eliminated from `gg.nbdy` for the next timestep. Conversely, if the pressure at a node that is on margin of the domain (defined by indices `gg.n1m`) is above atmospheric pressure, it suggests that pressure should be prescribed there and such nodes are added to `gg.nbdy` for the next timestep. Since the list of boundary nodes now changes in time, the current list is stored as `vv.nbdy`.

Plotting

The current solution (contained in `vv`) is rescaled and plotted using

```
>nevis_plot;
```

By default, this plots the average magnitude of the discharge on each node, expressed as an areal discharge (units $\mathrm{m}^2\ \mathrm{s}^{-1}$) as if the conduit discharge were spread out over each grid cell. This average is calculated and added to `vv` using

```
>vv = nevis_nodedischarge(vv,aa,pp,gg,oo);
```

Other things such as the topography can be plotted by setting, *e.g.*, `oo.topography` (see `nevis_plot.m`).

A series of timesteps (say 10 of them) saved into a directory with name `fn` can be animated using

```
>nevis_animate(fn,1:10,1,0);
```

The second input here is the timesteps to plot, the third input is an option to plot the discharge, and the fourth is an option not to save any of the frames.