ISSM Mathematical Reference Guide

Ice-sheet and Sea-level System Model
Source: `/g/data1b/au88/jh7060/ISSM/`

Generated from ISSM source code

2026-06-06

Contents

1 Overview & Architecture

ISSM solves the coupled ice-sheet system by composing **Analysis** objects. Each analysis contributes:

- element-level matrix/vector contributions (`CreateKMatrix`, `CreatePVector`)
- boundary conditions (`CreateConstraints`, `CreateLoads`)
- parameter updates (`UpdateParameters`, `UpdateInputFromSolution`)

The call chain is:

```
transient_core / stressbalance_core / ...
    ↪ FemModel::Solve(analysistype)
          ↪ solutionsequence_*(...)        ← solver strategy
                ↪ Analysis::*KMatrix / *PVector ← per element
                    assembly
```

Table 1: Key source directories

Path	Content
`src/c/analyses/`	One `.cpp/.h` pair per governing equation
`src/c/solutionsequences/`	Linear, nonlinear, Newton, FCT, ... solvers
`src/c/cores/`	Top-level dispatchers (`stressbalance_core`, ...)
`src/c/classes/Elements/`	`Tria`, `Penta`, `Seg` FEM elements
`src/m/parameterization/`	MATLAB model-setup utilities
`src/m/classes/`	MATLAB model class definitions

2 Stress-Balance / Ice Dynamics

Primary source: `src/c/analyses/StressbalanceAnalysis.cpp`
Additional: `StressbalanceSIAAnalysis.cpp`, `StressbalanceVerticalAnalysis.cpp`

2.1 General Governing Equations

ISSM solves the **Stokes momentum balance** (or an approximation thereof):

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \rho_{\text{ice}}\, \mathbf{g} = \mathbf{0} \qquad \text{(momentum balance)} \tag{1}$$

$$\boldsymbol{\sigma} = -p\,\mathbf{I} + 2\eta\,\mathbf{D}(\mathbf{u}) \qquad \text{(constitutive law)} \tag{2}$$

$$\mathbf{D}(\mathbf{u}) = \tfrac{1}{2}\big(\boldsymbol{\nabla}\mathbf{u} + (\boldsymbol{\nabla}\mathbf{u})^T\big) \qquad \text{(strain-rate tensor)} \tag{3}$$

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \qquad \text{(incompressibility)} \tag{4}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor, p the isotropic pressure, η the effective viscosity, $\mathbf{u} = (v_x, v_y, v_z)$ the ice velocity, $\rho_{\text{ice}} \approx 917\ \text{kg}\,\text{m}^{-3}$ the ice density, and $\mathbf{g}$ the gravitational acceleration vector.

2.2 Flow Approximations

ISSM supports six flow-equation approximations, selected per element via `md.flowequation`.

2.2.1 SIA — Shallow Ice Approximation

DoFs: 2 per node (v_x, v_y). **Domain:** 2D horizontal (most common). **Physics:** Retains only vertical shear stress; neglects longitudinal stresses.

$$\frac{\partial \sigma_{xz}}{\partial z} = \rho_{\text{ice}}\, g\, \frac{\partial s}{\partial x} \tag{5}$$

$$\frac{\partial \sigma_{yz}}{\partial z} = \rho_{\text{ice}}\, g\, \frac{\partial s}{\partial y} \tag{6}$$

Depth-integrated basal velocity:

$$u_b = u_s - 2A\,(\rho_{\text{ice}}\, g)^n\, |\boldsymbol{\nabla} s|^{n-1} \int_b^s (s-z)^n\, dz \tag{7}$$

2.2.2 SSA — Shallow Shelf Approximation

DoFs: 2 per node (v_x, v_y). **Domain:** Floating ice shelves, fast-flowing streams. **Physics:** Depth-averaged, hydrostatic vertical; retains longitudinal deviatoric stresses.

$$\frac{\partial}{\partial x}\left(2\eta H\left(2\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)\right)+\frac{\partial}{\partial y}\left(\eta H\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)-\tau_{bx}=\rho_{\text{ice}}\, g\, H\frac{\partial s}{\partial x} \tag{8}$$

$$\frac{\partial}{\partial y}\left(2\eta H\left(2\frac{\partial v}{\partial y}+\frac{\partial u}{\partial x}\right)\right)+\frac{\partial}{\partial x}\left(\eta H\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)-\tau_{by}=\rho_{\text{ice}}\, g\, H\frac{\partial s}{\partial y} \tag{9}$$

Basal friction (Weertman sliding law):

$$\boldsymbol{\tau}_b = -C\,|\mathbf{u}_b|^{m-1}\mathbf{u}_b \tag{10}$$

where C is the friction coefficient and m the sliding exponent.

SSA viscosity:

$$\eta = \frac{B}{2\,|\dot{\varepsilon}_e|^{1/n-1}} \tag{11}$$

where B is the ice rheology parameter (temperature-dependent), $n \approx 3$ (Glen's flow law), and the *effective strain rate* is:

$$|\dot{\varepsilon}_e|^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + \frac{\partial u}{\partial x}\frac{\partial v}{\partial y} + \frac{1}{4}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2 \tag{12}$$

2.2.3 HO — Higher-Order (Blatter–Pattyn)

DoFs: 2 per node per vertical level (v_x, v_y). **Domain:** 3D meshes. **Physics:** Omits vertical momentum equation; retains vertical gradients of horizontal velocity.

$$\frac{\partial}{\partial x}\left(4\eta\frac{\partial u}{\partial x}+2\eta\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial y}\left(\eta\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)+\frac{\partial}{\partial z}\left(\eta\frac{\partial u}{\partial z}\right)=\rho\, g\,\frac{\partial s}{\partial x} \tag{13}$$

$$\frac{\partial}{\partial y}\left(4\eta\frac{\partial v}{\partial y}+2\eta\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial x}\left(\eta\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)+\frac{\partial}{\partial z}\left(\eta\frac{\partial v}{\partial z}\right)=\rho\, g\,\frac{\partial s}{\partial y} \tag{14}$$

2.2.4 L1L2 — Level-1.5 Approximation

DoFs: 2 per node. **Physics:** Blends SSA and HO; incorporates a vertical shear term through an analytical integration.

$$|\dot{\varepsilon}_e|^2 = |\dot{\varepsilon}_{\text{SSA}}|^2 + |\dot{\varepsilon}_{\text{shear}}(z)|^2 \tag{15}$$

Viscosity function `ViscosityL1L2()` in `src/c/classes/Materials/`.

2.2.5 MOLHO — Multi-Layer Higher-Order

DoFs: 4 per node ($v_{x,\text{base}}$, $v_{y,\text{base}}$, $v_{x,\text{shear}}$, $v_{y,\text{shear}}$). **Physics:** Separates basal sliding and shear deformation components analytically.

$$u(z) = u_{\text{base}} + u_{\text{shear}}(z), \qquad u_{\text{shear}}(z) = \int_b^z 2\,A(T)\,\sigma_{xz}^n\,dz' \tag{16}$$

`ViscosityMOLHO()` returns four viscosity values for the base and shear components.

2.2.6 FS — Full Stokes

DoFs: 5 per node (v_x, v_y, v_z, p, optional). **Domain:** 3D. **Physics:** Complete Navier-Stokes for slow viscous flow ($\text{Re} \approx 0$).

Supported mixed FEM element pairs (configured via `md.flowequation.fe_FS`):

Table 2: Full Stokes mixed FEM element pairs

Code	Velocity	Pressure	Notes
`P1P1`	P1	P1	Equal-order, needs stabilisation
`P1P1GLS`	P1	P1	+ Galerkin Least Squares stab.
`MINI`	P1+bubble	P1	Stable mini element
`MINIcondensed`	P1+bubble	P1	Bubble condensed at element level
`TaylorHood`	P2	P1	Classical inf-sup stable
`XTaylorHood`	P2	P1	Extended Taylor-Hood
`CrouzeixRaviart`	CR	P1-DG	Non-conforming, locally conservative
`LACrouzeixRaviart`	LA-CR	P1-DG	Local assembly variant
`OneLayerP4z`	P4 vertical	P1	Spectral vertical treatment

2.3 Weak / Variational Form

Multiplying the momentum equation by test functions $\mathbf{v}$ (velocity) and q (pressure) and integrating by parts:

$$\int_\Omega 2\eta\,\mathbf{D}(\mathbf{u}){:}\mathbf{D}(\mathbf{v})\,d\Omega - \int_\Omega p\,\boldsymbol{\nabla}{\cdot}\mathbf{v}\,d\Omega + \int_\Omega q\,\boldsymbol{\nabla}{\cdot}\mathbf{u}\,d\Omega + \int_{\Gamma_b} \boldsymbol{\tau}_b \cdot \mathbf{v}\,d\Gamma = \int_\Omega \rho\mathbf{g}\cdot\mathbf{v}\,d\Omega + \int_{\Gamma_N} \mathbf{T}_{\text{ext}}\cdot\mathbf{v}\,d\Gamma \tag{17}$$

where Γ_b is the basal boundary and Γ_N the Neumann boundary (calving front, $\mathbf{T}_{\text{ext}} = -p_{\text{ocean}}\mathbf{n}$).

B matrix (strain-displacement, Voigt notation, 3D):

$$\mathbf{B} = \begin{pmatrix} \partial\phi/\partial x & 0 & 0 \\ 0 & \partial\phi/\partial y & 0 \\ 0 & 0 & \partial\phi/\partial z \\ \partial\phi/\partial y & \partial\phi/\partial x & 0 \\ \partial\phi/\partial z & 0 & \partial\phi/\partial x \\ 0 & \partial\phi/\partial z & \partial\phi/\partial y \end{pmatrix} \tag{18}$$

Assembled in `StressbalanceAnalysis.cpp` lines 6021–6232.

2.4 Boundary Conditions

Table 3: Stress-balance boundary conditions

Type	Variable	Description
Dirichlet	`md.stressbalance.spcvx/spcvy/spcvz`	Fixed velocity components
FS pressure	$p = \rho g(s-z)/F_{\text{recon}}$	Hydrostatic pressure at surface
Neumann (ocean)	$\mathbf{T} = -p_{\text{ocean}}\mathbf{n}$	Ocean pressure at calving front
Basal friction	$\boldsymbol{\tau}_b = -C\lvert\mathbf{u}\rvert^{m-1}\mathbf{u}$	Applied as load term

Approximation coupling (e.g. SSA–HO, HO–FS) uses zero-velocity constraints at interface nodes flagged as `SSAHOApproximationEnum`, `HOFSApproximationEnum`, etc.

2.5 Glen's Flow Law & Rheology

The ice rheology parameter $B(T)$ follows the Arrhenius relation:

$$A(T^*) = A_0 \exp\left(-\frac{Q}{RT^*}\right) \tag{19}$$

$$B = A^{-1/n} \tag{20}$$

$$T^* = T + \beta\, p \qquad \text{(pressure-corrected temperature)} \tag{21}$$

where A_0 is the pre-exponential factor, Q the activation energy ($60\,\text{kJ}\,\text{mol}^{-1}$ for $T < 263\,\text{K}$, otherwise $139\,\text{kJ}\,\text{mol}^{-1}$), R the gas constant, β the Clausius-Clapeyron gradient, and $n = 3$ (Glen exponent, default).

3 Mass Transport — Ice Thickness Evolution

Primary source: `src/c/analyses/MasstransportAnalysis.cpp`

3.1 Governing PDE

The **depth-integrated continuity equation** (thickness evolution):

$$\frac{\partial H}{\partial t} + \boldsymbol{\nabla}_H \cdot (H\bar{\mathbf{u}}) = \dot{m}_s - \dot{m}_b \tag{22}$$

where H is the ice thickness, $\bar{\mathbf{u}} = (v_x, v_y)$ the depth-averaged horizontal velocity, $\dot{m}_s$ the surface mass balance (positive = accumulation), and $\dot{m}_b$ the basal melt rate (positive = melting). Expanded:

$$\frac{\partial H}{\partial t} + \frac{\partial (Hv_x)}{\partial x} + \frac{\partial (Hv_y)}{\partial y} = \dot{m}_s - \dot{m}_b \tag{23}$$

3.2 Weak Form

Multiplying by test function ϕ and integrating:

$$\int_\Omega \phi \frac{\partial H}{\partial t} d\Omega + \int_\Omega \phi\, \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} H \, d\Omega + \int_\Omega \phi H (\boldsymbol{\nabla} \cdot \bar{\mathbf{u}}) d\Omega = \int_\Omega \phi (\dot{m}_s - \dot{m}_b) d\Omega \tag{24}$$

Implicit (backward Euler) discretisation:

$$\int_\Omega \frac{H^{n+1} - H^n}{\Delta t} \phi \, d\Omega + \int_\Omega \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} H^{n+1} \phi \, d\Omega + \int_\Omega H^{n+1} (\boldsymbol{\nabla} \cdot \bar{\mathbf{u}}) \phi \, d\Omega = \int_\Omega \phi (\dot{m}_s - \dot{m}_b) d\Omega \tag{25}$$

Transient term assembled at `MasstransportAnalysis.cpp` lines ∼420–422.

3.3 Stabilisation Options

Controlled by `md.masstransport.stabilization`:

Table 4: Mass-transport stabilisation schemes

Value	Method	Notes
0	None	Pure Galerkin (may oscillate)
1	SSA Anisotropic Diffusion	Diffusion along streamlines
2	Streamline Upwind	Classical SU scheme
3	Discontinuous Galerkin	DG with numerical flux
4	FCT	Flux Corrected Transport (monotone)
5	SUPG	Streamline Upwind Petrov-Galerkin

DG numerical flux (lines ∼583–642):

$$F^* = \{H \bar{\mathbf{u}} \cdot \mathbf{n}\} + \frac{|\bar{\mathbf{u}} \cdot \mathbf{n}|}{2} [H] \tag{26}$$

where $\{\cdot\}$ is the average and $[\cdot]$ is the jump across element faces.

SUPG test function:

$$\psi_i = \phi_i + \tau_{\text{SUPG}}\, \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} \phi_i, \qquad \tau_{\text{SUPG}} = \frac{h_e}{2|\bar{\mathbf{u}}|} \tag{27}$$

3.4 Key Variables

Table 5: Mass-transport code symbols

Code enum	Symbol	Description
`VxAverageEnum`, `VyAverageEnum`	$\bar{\mathbf{u}}$	Depth-averaged velocity
`ThicknessEnum`	H	Ice thickness
`SmbMassBalanceEnum`	$\dot{m}_s$	Surface mass balance
`BasalforcingsGroundediceMeltingRateEnum`	$\dot{m}_b$	Basal melt rate
`MasstransportMinThicknessEnum`	$H_{\min}$	Minimum thickness floor

4 Thermal & Enthalpy Analysis

Sources:

- `src/c/analyses/EnthalpyAnalysis.cpp` — enthalpy formulation
- `src/c/analyses/ThermalAnalysis.cpp` — classical temperature formulation

4.1 Enthalpy Formulation

The **enthalpy** is defined as:

$$E = \begin{cases} c_p\,(T - T_{\mathrm{ref}}) & T < T_{\mathrm{pmp}} \quad \text{(cold ice)} \\ c_p\,(T_{\mathrm{pmp}} - T_{\mathrm{ref}}) + L\,\omega & T = T_{\mathrm{pmp}} \quad \text{(temperate ice)} \end{cases} \tag{28}$$

where $c_p \approx 2009\ \mathrm{J\,kg^{-1}\,K^{-1}}$ is the specific heat capacity, T_{pmp} the pressure-melting-point temperature, $L \approx 334\ \mathrm{kJ\,kg^{-1}}$ the latent heat, and $\omega \in [0, 1]$ the water fraction.

4.2 Governing PDE

Enthalpy conservation:

$$\rho \frac{\partial E}{\partial t} + \rho\,\bar{\mathbf{u}} \cdot \boldsymbol{\nabla} E - \boldsymbol{\nabla} \cdot (\kappa\,\boldsymbol{\nabla} T) = 2\eta|\dot{\varepsilon}|^2 + Q_{\mathrm{geo}} \tag{29}$$

where $\kappa \approx 2.1\ \mathrm{W\,m^{-1}\,K^{-1}}$ is the thermal conductivity (lower in temperate ice), $2\eta|\dot{\varepsilon}|^2$ the viscous heating, and Q_{geo} the geothermal heat flux at the bed.

The pressure-melting point varies with pressure:

$$T_{\mathrm{pmp}} = T_0 - \beta\,p \qquad (\beta \approx 7.4 \times 10^{-8}\ \mathrm{K\,Pa^{-1}}) \tag{30}$$

4.3 Classical Temperature Formulation

`ThermalAnalysis.cpp` solves the simpler form (no water fraction):

$$\rho c_p \frac{\partial T}{\partial t} + \rho c_p\,\bar{\mathbf{u}} \cdot \boldsymbol{\nabla} T - \boldsymbol{\nabla} \cdot (\kappa\,\boldsymbol{\nabla} T) = 2\eta|\dot{\varepsilon}|^2 \tag{31}$$

4.4 Weak Form

$$\int_\Omega \rho c_p\,\phi \frac{\partial T}{\partial t} d\Omega + \int_\Omega \rho c_p\,\bar{\mathbf{u}} \cdot \boldsymbol{\nabla} T\,\phi\,d\Omega + \int_\Omega \kappa\,\boldsymbol{\nabla} T \cdot \boldsymbol{\nabla}\phi\,d\Omega = \int_\Omega 2\eta|\dot{\varepsilon}|^2\phi\,d\Omega + \int_\Gamma q_n\,\phi\,d\Gamma \tag{32}$$

4.5 Boundary Conditions

Table 6: Thermal boundary conditions

Location	Condition	Code parameter
Ice surface	Dirichlet: $T = T_{\mathrm{surface}}$	`md.thermal.spctemperature`
Ice base (frozen)	Dirichlet: $T = T_{\mathrm{pmp}}$	`md.thermal.spctemperature`
Ice base (temperate)	Neumann: $\kappa\partial T/\partial n = Q_{\mathrm{geo}} - Q_f$	Dynamic BC flag
Geothermal	Neumann: $q = Q_{\mathrm{geo}}$	`BasalforcingsGeothermalfluxEnum`

A penalty load (`Pengrid`) prevents $T > T_{\mathrm{pmp}}$ anywhere (`ThermalAnalysis.cpp` lines 78–91).

Table 7: Hydrology analysis files

File	Model
`HydrologyGlaDSAnalysis.cpp`	GlaDS (Glacier Drainage System)
`HydrologyDCEfficientAnalysis.cpp`	Distributed Channel — efficient
`HydrologyDCInefficientAnalysis.cpp`	Distributed Channel — inefficient
`HydrologyShaktiAnalysis.cpp`	SHAKTI (explicit lake drainage)
`HydrologyShreveAnalysis.cpp`	Shreve routing
`HydrologyPismAnalysis.cpp`	PISM hydrology
`HydrologyArmapwAnalysis.cpp`	ARMA subglacial lakes
`HydrologyTwsAnalysis.cpp`	Two-way Scheme
`HydrologyPrescribeAnalysis.cpp`	Prescribed hydraulics

5 Hydrology Models

5.1 GlaDS Model

Based on Werder et al. (2013). Solves a coupled sheet–channel system.

State variables: hydraulic potential Φ, sheet thickness h_s, channel cross-sectional area S (on mesh edges).

Hydraulic potential:

$$\Phi = \rho_w g z_b + p_w \tag{33}$$

Sheet flow equation (mass conservation):

$$\frac{\partial h_s}{\partial t} + \boldsymbol{\nabla} \cdot \mathbf{q}_s = m_s + m_r - c_s h_s \tag{34}$$

$$\mathbf{q}_s = -k_s h_s^{\alpha} \left|\boldsymbol{\nabla}\Phi\right|^{\beta-2} \boldsymbol{\nabla}\Phi \qquad \text{(Darcy-Weisbach type)} \tag{35}$$

Channel flow equation (on 1D edges):

$$\frac{\partial S}{\partial t} + \frac{\partial Q_c}{\partial s} = m_c - c_c(N)\, S \tag{36}$$

$$Q_c = -k_c S^{\alpha_c} \left|\frac{\partial \Phi}{\partial s}\right|^{\beta_c - 2} \frac{\partial \Phi}{\partial s} \tag{37}$$

Closure by viscous creep:

$$c_c(N) = 2\, A_s\, n^{-n}\, N^n\, S \qquad \text{(channel creep closure)} \tag{38}$$

$$c_s h_s = A_s\, n^{-n}\, N^n\, h_s \qquad \text{(sheet creep closure)} \tag{39}$$

Effective pressure: $N = p_{\text{ice}} - p_w = \rho_{\text{ice}} g H - (\Phi - \rho_w g z_b)$

Melt production (viscous + geothermal):

$$m = \frac{|\mathbf{q}_s|\,|\boldsymbol{\nabla}\Phi| + G + \boldsymbol{\tau}_b \cdot \mathbf{u}_b}{L_{\text{heat}}} \tag{40}$$

Table 8: GlaDS model parameters (`HydrologyGlaDSAnalysis::UpdateParameters`)

Code name	Symbol	Description
`sheet_conductivity`	k_s	Sheet hydraulic conductivity
`channel_conductivity`	k_c	Channel hydraulic conductivity
`sheet_alpha/beta`	α, β	Sheet flow exponents
`channel_alpha/beta`	α_c, β_c	Channel flow exponents
`cavity_spacing`	l_c	Basal cavity spacing
`pressure_melt_coefficient`	c_t	$\partial T_{\text{pmp}}/\partial p$
`omega`	ω	Melt discharge exponent

5.2 Distributed Channel Model

Separates efficient (channelised) and inefficient (distributed) drainage.

Efficient system (lower pressure, channels):

$$\boldsymbol{\nabla} \cdot (K_{\text{eff}} \, \boldsymbol{\nabla} \Phi_{\text{eff}}) + \Lambda(\Phi_{\text{ineff}} - \Phi_{\text{eff}}) = 0 \tag{41}$$

Inefficient system (linked cavities / till):

$$S_s \frac{\partial \Phi_{\text{ineff}}}{\partial t} + \boldsymbol{\nabla} \cdot (K_{\text{ineff}} \, \boldsymbol{\nabla} \Phi_{\text{ineff}}) - \Lambda(\Phi_{\text{ineff}} - \Phi_{\text{eff}}) = m_{\text{input}} \tag{42}$$

where Λ is the exchange coefficient between the two systems.

5.3 SHAKTI Model

Explicit model for subglacial lake evolution (`HydrologyShaktiAnalysis.cpp`):

- Tracks lake extent and pressure as discrete drainage events
- Couples lake pressure to ice overburden: $p_{\text{lake}} = N_{\text{crit}} \rho_{\text{ice}} g H$
- Solves sediment transport and deformation thickness

Effective pressure fed to ice dynamics: $N = \rho_{\text{ice}} g H - p_w$

6 Elastic Sea-Level Adjustment (ESA/GIA)

Primary source: `src/c/analyses/EsaAnalysis.cpp`
Also: `LoveAnalysis.cpp`, `SealevelchangeAnalysis.cpp`

6.1 Love Number Formalism

The Earth's elastic response to surface ice-mass changes is computed using **spherical harmonic Green's functions**.

Vertical displacement:

$$U(\theta) = (M_e R_e)^{-1} \sum_{n=0}^{N_{\max}} h_n \, G_U(\theta) \tag{43}$$

Geoid perturbation:

$$\mathcal{N}(\theta) = M_e^{-1} \sum_{n=0}^{N_{\max}} (1 + k_n) \, G_{\mathcal{N}}(\theta) \tag{44}$$

Horizontal displacement:

$$H(\theta) = (M_e R_e)^{-1} \sum_{n=0}^{N_{\max}} l_n \, G_H(\theta) \tag{45}$$

where h_n, k_n, l_n are elastic Love numbers, $P_n(\cos\theta)$ Legendre polynomials, θ the angular distance, M_e and R_e Earth mass and radius.

Green function assembly (`EsaAnalysis.cpp` lines ∼96–143):

$$\delta h_n = h_n - h_{N-1} \tag{46}$$

$$\delta k_n = k_n - k_{N-1} \tag{47}$$

$$\delta l_n = l_n - l_{N-1} \frac{N-1}{n+\varepsilon} \tag{48}$$

$$U_{\text{elastic}} \mathrel{+}= \delta h_n \, P_n(\cos\alpha) \tag{49}$$

$$H_{\text{elastic}} \mathrel{-}= \sin(\alpha) \, \delta l_n \, \frac{dP_n}{d\alpha} \tag{50}$$

$$G_{\text{elastic}} \mathrel{-}= \left[2\delta h_n - n \, \delta k_n - \delta k_n\right] P_n(\cos\alpha) \tag{51}$$

Legendre recursion:

$$P_n = \frac{(2n-1)\cos(\alpha) \, P_{n-1} - (n-1) \, P_{n-2}}{n} \tag{52}$$

$$\frac{dP_n}{d\alpha} = \frac{(2n-1)\left(P_{n-1} + \cos(\alpha)\frac{dP_{n-1}}{d\alpha}\right) - (n-1)\frac{dP_{n-2}}{d\alpha}}{n} \tag{53}$$

6.2 Sea-Level Fingerprints

Barystatic sea level change:

$$\Delta SL = -\frac{\Delta M}{\rho_{\text{ocean}} \, A_{\text{ocean}}} \tag{54}$$

Combined relative sea-level (RSL) fingerprint:

$$\text{RSL}(\theta) = -\frac{\Delta\Phi}{g} + U + \Delta SL \tag{55}$$

6.3 GIA Viscoelastic Component

`LoveAnalysis.cpp` computes time-dependent Love numbers for a Maxwell viscoelastic Earth model, enabling long-term glacial isostatic adjustment (GIA).

7 Damage Mechanics

Primary source: `src/c/analyses/DamageEvolutionAnalysis.cpp`

7.1 Governing Equation

The damage parameter $D \in [0, 1]$ (0 = intact, 1 = fully fractured) evolves by advection with a source/sink:

$$\frac{\partial D}{\partial t} + \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} D = F(\sigma_{\text{von}}, D) - \Gamma(N) \, D \tag{56}$$

where F is a stress-driven source, $\Gamma(N)D$ the pressure-driven healing term, and σ_{von} the von Mises stress invariant.

7.2 Damage Source Laws

Controlled by `md.damage.law`.

Law 0 — Threshold:

$$F = \begin{cases} 0 & \sigma_{\text{von}} < \sigma_{\text{th}} \\ c_1 & \sigma_{\text{von}} \geq \sigma_{\text{th}} \end{cases} \tag{57}$$

Law > 0 — Smooth arctan:

$$F = c_1 \cdot \frac{2}{\pi} \arctan\left[c_2 \, \frac{\sigma_{\text{von}} - \sigma_{\text{th}}}{\sigma_{\text{bound}}} \right] \tag{58}$$

Parameters (`DamageEvolutionAnalysis.cpp` lines 82–116):

Table 9: Damage model parameters

Parameter	Description
`c1`, `c2`	Damage rate coefficients
`stress_threshold` (σ_{th})	Yield stress for damage onset
`stress_ubound` (σ_{bound})	Saturation stress
`kappa`	Healing rate coefficient

7.3 Effect on Rheology

Damage reduces the effective viscosity:

$$\eta_{\text{eff}} = (1 - D)\, \eta_{\text{intact}} \tag{59}$$

8 Age Tracking

Primary source: `src/c/analyses/AgeAnalysis.cpp`

8.1 Governing Equation

The ice age τ satisfies the advection equation with a unit source:

$$\frac{\partial \tau}{\partial t} + \mathbf{u} \cdot \boldsymbol{\nabla} \tau = 1 \tag{60}$$

Ice deposited at the surface has $\tau = 0$; it increases by 1 year per year as it advects downward and laterally.

8.2 Weak Form & Assembly

$$\int_\Omega \phi \, \frac{\tau^{n+1} - \tau^n}{\Delta t} d\Omega + \int_\Omega \phi \, \mathbf{u} \cdot \boldsymbol{\nabla} \tau^{n+1} d\Omega = \int_\Omega \phi \, d\Omega \tag{61}$$

Convection matrix (lines 157–163):

$$A_{ij} = \int_\Omega \phi_i \left(v_x \frac{\partial \phi_j}{\partial x} + v_y \frac{\partial \phi_j}{\partial y} + v_z \frac{\partial \phi_j}{\partial z} \right) d\Omega \tag{62}$$

Mass matrix (transient) (lines 165–170):

$$M_{ij} = \int_\Omega \phi_i \, \phi_j \, d\Omega \tag{63}$$

8.3 Boundary Conditions

- **Surface:** $\tau = 0$ (fresh accumulation, `md.age.spcage`)
- **Inflow boundaries:** $\tau = \tau_{\text{inflow}}$

8.4 SUPG Stabilisation (lines 172–192)

$$\tau_{\text{SUPG}} = \frac{h_e}{2|\mathbf{u}|}, \qquad \psi_i = \phi_i + \tau_{\text{SUPG}}\, \mathbf{u} \cdot \boldsymbol{\nabla}\phi_i \tag{64}$$

9 Balance Thickness & Balance Velocity

Sources: `BalancethicknessAnalysis.cpp`, `Balancethickness2Analysis.cpp`, `BalancevelocityAnalysis.`

9.1 Steady-State Thickness

At **steady state**, the continuity equation becomes:

$$\boldsymbol{\nabla} \cdot (H\mathbf{u}) = \dot{m}_s - \dot{m}_b \tag{65}$$

Given a known velocity field $\mathbf{u}$, the **balance thickness** H_{bal} satisfies:

$$\mathbf{u} \cdot \boldsymbol{\nabla} H_{\text{bal}} + H_{\text{bal}}(\boldsymbol{\nabla} \cdot \mathbf{u}) = \dot{m}_s - \dot{m}_b \tag{66}$$

9.2 Weak Form

$$\int_\Omega \phi\, \mathbf{u} \cdot \boldsymbol{\nabla} H_{\text{bal}}\, d\Omega + \int_\Omega \phi\, H_{\text{bal}}(\boldsymbol{\nabla} \cdot \mathbf{u}) d\Omega = \int_\Omega \phi(\dot{m}_s - \dot{m}_b) d\Omega \tag{67}$$

CG tangent matrix (`CreateKMatrixCG`, lines ∼154–200):

$$K_{ij} = \int \phi_i \left(v_x \frac{\partial \phi_j}{\partial x} + v_y \frac{\partial \phi_j}{\partial y} \right) dA + \int \phi_i\, \phi_j \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} \right) dA \tag{68}$$

9.3 Balance Velocity

The depth-averaged balance velocity $\bar{\mathbf{u}}_{\text{bal}}$ satisfying steady-state mass conservation:

$$\boldsymbol{\nabla} \cdot (\bar{\mathbf{u}}_{\text{bal}}\, H) = \dot{m}_s - \dot{m}_b, \qquad \bar{\mathbf{u}}_{\text{bal}} = -\frac{\int_{\text{upstream}} (\dot{m}_s - \dot{m}_b)\, dA}{H} \tag{69}$$

Solved using a flux-routing approach along the surface gradient.

10 Adjoint / Inverse Methods

Sources: `AdjointHorizAnalysis.cpp`, `AdjointBalancethicknessAnalysis.cpp`, `src/c/cores/control_co`

10.1 Adjoint Formulation

ISSM uses adjoint-based data assimilation to infer basal friction C and rheology parameter B from observed surface velocities/strain rates.

Cost functional:

$$\mathcal{J}(m) = \frac{1}{2} \sum_{\text{obs}} \frac{(\mathbf{u}_{\text{obs}} - \mathbf{u}_{\text{mod}})^2}{\sigma_{\text{obs}}^2} + \mathcal{R}(m) \tag{70}$$

Tikhonov regularisation:

$$\mathcal{R}(m) = \frac{\alpha_1}{2} \int_\Omega (\boldsymbol{\nabla} m)^2 d\Omega + \frac{\alpha_2}{2} \int_\Omega (m - m_{\text{prior}})^2 d\Omega \tag{71}$$

Adjoint equation:

$$\mathbf{K}^T \boldsymbol{\lambda} = -\frac{\partial \mathcal{J}}{\partial \mathbf{u}} \tag{72}$$

Gradient of cost:

$$\frac{\partial \mathcal{J}}{\partial m} = \frac{\partial (\mathbf{K}\mathbf{u})}{\partial m} \cdot \boldsymbol{\lambda} + \frac{\partial \mathcal{R}}{\partial m} \tag{73}$$

10.2 Optimisation Backends

Table 10: Optimisation backend files

Core file	Method
`controladm1qn3_core.cpp`	L-BFGS (m1qn3 variant)
`controlm1qn3_core.cpp`	M1QN3 solver
`controltao_core.cpp`	PETSc/TAO optimiser

11 Grounding Line & Free Surfaces

Sources: `LevelsetAnalysis.cpp`, `FreeSurfaceTopAnalysis.cpp`, `FreeSurfaceBaseAnalysis.cpp`, `GLheightadvectionAnalysis.cpp`

11.1 Level-Set Grounding Line

The grounding line position is tracked via a level-set function ϕ_{GL}:

$$\phi_{\text{GL}} \begin{cases} > 0 & \text{grounded ice (bed in contact)} \\ < 0 & \text{floating ice shelf} \\ = 0 & \text{grounding line} \end{cases} \tag{74}$$

Flotation criterion:

$$\rho_{\text{ice}}\, H + \rho_{\text{water}}\, z_b < 0 \implies \text{floating} \tag{75}$$

Level-set evolution:

$$\frac{\partial \phi_{\text{GL}}}{\partial t} + \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} \phi_{\text{GL}} = 0 \tag{76}$$

11.2 Free Surface Evolution

Top surface ($s = b + H$):

$$\frac{\partial s}{\partial t} = w_s - \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} s + \dot{m}_s \qquad \text{(kinematic BC)} \tag{77}$$

Basal surface:

$$\frac{\partial b}{\partial t} = w_b - \bar{\mathbf{u}} \cdot \boldsymbol{\nabla} b - \dot{m}_b \qquad \text{(for marine ice; zero for bedrock)} \tag{78}$$

12 Solution Sequences & Solvers

Directory: `src/c/solutionsequences/`

12.1 Linear Solve (`solutionsequence_linear.cpp`)

```
1. Assemble K, f   (global stiffness matrix and load vector)
2. Partition:  K_ff u_f = f_f − K_fs u_s  (free/specified DoF split)
3. Solve:  K_ff u_f = p_f  (direct or iterative)
4. Merge:  u = [u_f; u_s]
5. UpdateInputFromSolution(u)
```

12.2 Nonlinear Fixed-Point / Picard (`solutionsequence_nonlinear.cpp`)

```
u^(0) = initial guess
for k = 0, 1, ..., max_iter:
    Assemble K(u^(k)), f(u^(k))
    Solve   K u^(k+1) = f
    if   ||u^(k+1) − u^(k)|| / ||u^(k)|| < tol: break
```

12.3 Newton–Raphson (`solutionsequence_newton.cpp` lines 53–109)

```
u^(0) = initial guess
for k = 0, 1, ..., max_iter:
    r^(k) = K(u^(k)) u^(k) − f(u^(k))  (residual)
    J^(k) = ∂r/∂u |_{u^(k)}  (Jacobian = tangent stiffness)
    Solve   J^(k) Δu = −r^(k)
    u^(k+1) = u^(k) + Δu
    if   ||r^(k)|| / ||f|| < ε_res  AND
         ||Δu|| / ||u|| < ε_rel: break
```

12.4 Theta (θ) Method for Time Integration

Generalised trapezoidal rule:

$$\left(\frac{\mathbf{M}}{\Delta t} + \theta \mathbf{K}\right) \mathbf{u}^{n+1} = \left(\frac{\mathbf{M}}{\Delta t} - (1-\theta)\mathbf{K}\right) \mathbf{u}^{n} + \theta \mathbf{f}^{n+1} + (1-\theta)\mathbf{f}^{n} \tag{79}$$

Table 11: θ-method schemes

θ	**Scheme**	**Order**	**Stability**
0	Forward Euler	1st	Conditionally stable
0.5	Crank-Nicolson	2nd	Unconditionally stable
1	Backward Euler	1st	Unconditionally stable

12.5 Uzawa Pressure Iteration

For mixed saddle-point systems (Full Stokes velocity–pressure coupling):

$$\begin{pmatrix} \mathbf{K} & \mathbf{B}^T \\ \mathbf{B} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{0} \end{pmatrix} \tag{80}$$

Uzawa iteration:

$$\mathbf{K}\,\mathbf{u}^{(k+1)} = \mathbf{f} - \mathbf{B}^T p^{(k)} \quad \text{(velocity sub-solve)} \tag{81}$$

$$p^{(k+1)} = p^{(k)} - \rho\,(\mathbf{B}\,\mathbf{u}^{(k+1)}) \quad \text{(pressure correction)} \tag{82}$$

Converges when $\rho \in \left(0,\ 2/\lambda_{\max}(\mathbf{K}^{-1}\mathbf{B}\mathbf{K}^{-1}\mathbf{B}^T)\right)$.

12.6 FCT — Flux Corrected Transport (`solutionsequence_fct.cpp`)

Monotone transport scheme:

```
1. Compute low-order (diffusive) solution u_L (upwind scheme)
2. Compute high-order (Galerkin) solution u_H
3. Anti-diffusive fluxes F = M_L(u_H − u_L)
4. Limit: F* = αF, α ∈ [0,1] (Zalesak limiter)
5. u = u_L + M_L^{-1} F*
```

12.7 Schur Complement CG (`solutionsequence_schurcg.cpp`)

For coupled 2×2 block systems (e.g., GlaDS sheet + channel):

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix} \tag{83}$$

$$\mathbf{S} = \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \qquad \text{(Schur complement)} \tag{84}$$

$$\mathbf{S}\,p = \mathbf{g} - \mathbf{C}\mathbf{A}^{-1}\mathbf{f} \qquad \text{(reduced system, solved by CG)} \tag{85}$$

$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{f} - \mathbf{B}\,p) \qquad \text{(back-substitute)} \tag{86}$$

12.8 Convergence Criteria

All nonlinear solvers check (any one criterion satisfied ⇒ converged):

$$\varepsilon_{\text{rel}} = \frac{\left\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\right\|_2}{\left\|\mathbf{u}^{(k+1)}\right\|_2} < \text{tol}_{\text{rel}} \tag{87}$$

$$\varepsilon_{\text{res}} = \frac{\|\mathbf{K}\mathbf{u} - \mathbf{f}\|_2}{\|\mathbf{f}\|_2} < \text{tol}_{\text{res}} \tag{88}$$

$$\varepsilon_{\text{abs}} = \left\|\mathbf{u}^{(k+1)} - \mathbf{u}^{(k)}\right\|_2 < \text{tol}_{\text{abs}} \tag{89}$$

13 Finite Element Infrastructure

Directory: `src/c/classes/Elements/`

13.1 Element Hierarchy

```
Element  (base, abstract)
|-- Tria   -- 2D triangle (2D/basal problems)
|-- Penta  -- 3D 6-node prism (3D ice volume)
`-- Seg    -- 1D segment (boundary integrals, channels)
```

13.2 Reference Elements & Basis Functions

13.2.1 Linear Triangle (P1)

Natural coordinates (ξ, η) on the unit simplex:

$$\phi_1(\xi, \eta) = 1 - \xi - \eta \tag{90}$$
$$\phi_2(\xi, \eta) = \xi \tag{91}$$
$$\phi_3(\xi, \eta) = \eta \tag{92}$$

Physical gradients: $\boldsymbol{\nabla}\phi = \mathbf{J}^{-T}\hat{\boldsymbol{\nabla}}\phi$ where $\mathbf{J}$ is the Jacobian of the physical-to-reference map.

13.2.2 Quadratic Triangle (P2)

6 nodes (3 corner + 3 edge-midpoint); used for velocity in Taylor-Hood elements.

13.2.3 Prism Element (Penta, P1 × P1)

Tensor product of triangle and linear vertical:

$$\phi_{i,k}(\xi, \eta, \zeta) = \phi_i^{\text{tri}}(\xi, \eta) \cdot \phi_k^{1D}(\zeta) \tag{93}$$

Higher-order vertical: P1-P2 (9 nodes), P1-P4 (15 nodes), etc.

13.3 Quadrature Rules

Table 12: Gaussian quadrature for triangles (`GaussTria`)

Order	Points	Exactness
1	1 (centroid)	P1 polynomials
2	3	P2
3	4	P3
4	6	P4
5	7	P5

Prisms (`GaussPenta`) use a product rule: $n_{\text{tri}} \times n_{\text{vert}}$ Gauss points (typically $4 \times 3 = 12$ for P1-P2 approximations).

13.4 Jacobian Computation

Line element (1D):

$$|J| = \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2 + \left(\frac{\partial z}{\partial \xi}\right)^2} \tag{94}$$

Surface element (2D):

$$\mathbf{J} = \begin{pmatrix} \partial x/\partial \xi & \partial x/\partial \eta \\ \partial y/\partial \xi & \partial y/\partial \eta \\ \partial z/\partial \xi & \partial z/\partial \eta \end{pmatrix}, \qquad |J| = \left\| \frac{\partial \mathbf{r}}{\partial \xi} \times \frac{\partial \mathbf{r}}{\partial \eta} \right\| \tag{95}$$

Volume element (3D):

$$\mathbf{J} = \begin{pmatrix} \partial x/\partial \xi & \partial x/\partial \eta & \partial x/\partial \zeta \\ \partial y/\partial \xi & \partial y/\partial \eta & \partial y/\partial \zeta \\ \partial z/\partial \xi & \partial z/\partial \eta & \partial z/\partial \zeta \end{pmatrix}, \qquad |J| = \det(\mathbf{J}) \tag{96}$$

13.5 Input Interpolation at Gauss Points

All physical fields are stored as `Input` objects:

```
Input* inp = element->GetInput(ThicknessEnum);
inp->GetInputValue(&H, gauss);                  // scalar at Gauss pt
inp->GetInputDerivativeValue(&grad[0], xyz, gauss);  // gradient
```

Supports P0 (constant), P1 (linear), P2 (quadratic) and vertex-to-Gauss interpolation.

14 Parameterisation Layer (MATLAB/Python)

Directory: `src/m/parameterization/`

14.1 `setflowequation.m` — Assigning Flow Approximations

```
% Flag elements per approximation
SIAflag  = FlagElements(md, 'SIA_domain');
SSAflag  = FlagElements(md, 'SSA_domain');
HOflag   = FlagElements(md, 'HO_domain');
FSflag   = FlagElements(md, 'FS_domain');

% Fill unflagged elements with a default
SSAflag(~(SIAflag | HOflag | FSflag)) = 1;

% Validate: each element gets exactly one flag
assert(all(SIAflag + SSAflag + HOflag + FSflag == 1));

% Build coupling zones (e.g. HO-FS overlap)
nodeonHOFS = intersect(nodeonHO, nodeonFS);
HOFSflag(any(ismember(md.mesh.elements, nodeonHOFS), 2)) = 1;
```

Coupling modes: `penalties` (Lagrange multiplier) or `tiling` (element partitioning).

14.2 `setmask.m` — Grounded/Floating Ice

```
% Ocean levelset: negative = grounded, positive = floating
md.mask.ocean_levelset = md.geometry.base - md.geometry.bed;

% Floating base driven by hydrostasy
base_hydro = -rho_ice/rho_water * H;
md.mask.ocean_levelset(floating) = base_hydro - bed;
```

14.3 Effective Pressure

```
P_ice   = rho_ice * g * H;
P_water = rho_water * g * max(0, z_sl - bed);
N       = max(0, P_ice - P_water);   % no tension
```

14.4 Geometry Conventions

$$
\begin{aligned}
s &= \text{surface elevation (top of ice)} \\
b &= \text{base elevation (bottom of ice)} \\
H &= s - b \quad \text{(ice thickness)} \\
\text{bed} &= \text{bedrock elevation} \\
h &= s - z \quad \text{(depth below surface)}
\end{aligned}
$$

For floating ice: $b = -(\rho_{\text{ice}}/\rho_{\text{water}})\, H$ (hydrostatic equilibrium).

15 Summary Table

Table 13: Analysis summary

Analysis	Governing Equation	Domain	FEM Basis	Time Scheme
Stress Balance (SSA)	Depth-int. momentum	2D horiz	P1/P2	Steady (Newton)
Stress Balance (HO)	Higher-order momentum	3D	P1 prism	Steady (Newton)
Stress Balance (FS)	Full Stokes	3D	Mixed P1-P2	Steady (Newton/Uzawa)
Mass Transport	$\partial H/\partial t + \nabla \cdot (H\mathbf{u}) = \dot{m}_s - \dot{m}_b$	Basal 2D	P1/P1-DG	Implicit Euler
Enthalpy	$\rho \partial E/\partial t + \rho \mathbf{u} \cdot \nabla E - \nabla \cdot (\kappa \nabla T) = Q$	3D	P1	Implicit Euler
Temperature	$\rho c_p \partial T/\partial t + \ldots = 2\eta \lvert\dot{\varepsilon}\rvert^2$	3D	P1	Implicit Euler
Age	$\partial \tau/\partial t + \mathbf{u} \cdot \nabla \tau = 1$	3D	P1	Implicit Euler
Balance Thickness	$\mathbf{u} \cdot \nabla H + H \nabla \cdot \mathbf{u} = \dot{m}_s - \dot{m}_b$	Basal 2D	P1/DG	Steady

continued on next page

Analysis	Governing Equation	Domain	FEM Basis	Time Scheme
GlaDS Hydrology	Sheet + channel potential	2D+1D edges	P1	Implicit
Damage	$\partial D/\partial t + \mathbf{u}\cdot\nabla D = F - \Gamma D$	3D	P0/P1	Implicit Euler
ESA/GIA	Elastic Love number series	Global spherical	Spectral	Steady
Sea-level Change	$\mathrm{RSL} = -\Delta\Phi/g + U + \Delta SL$	Global	Spectral	Steady
Grounding Line	$\partial\phi/\partial t + \mathbf{u}\cdot\nabla\phi = 0$	Basal 2D	P1	Implicit
Adjoint	$\mathbf{K}^T\lambda = -\partial\mathcal{J}/\partial\mathbf{u}$	3D/2D	As forward	Steady

Complete Analyses Directory Listing

`src/c/analyses/` — all 52 analysis files:

```
AdjointBalancethickness2Analysis.cpp/h
AdjointBalancethicknessAnalysis.cpp/h
AdjointHorizAnalysis.cpp/h
AgeAnalysis.cpp/h
Analysis.h
Balancethickness2Analysis.cpp/h
BalancethicknessAnalysis.cpp/h
BalancethicknessSoftAnalysis.cpp/h
BalancevelocityAnalysis.cpp/h
DamageEvolutionAnalysis.cpp/h
DebrisAnalysis.cpp/h
DepthAverageAnalysis.cpp/h
EnthalpyAnalysis.cpp/h
EnumToAnalysis.cpp/h
EsaAnalysis.cpp/h
ExtrapolationAnalysis.cpp/h
ExtrudeFromBaseAnalysis.cpp/h
ExtrudeFromTopAnalysis.cpp/h
FreeSurfaceBaseAnalysis.cpp/h
FreeSurfaceTopAnalysis.cpp/h
GLheightadvectionAnalysis.cpp/h
HydrologyArmapwAnalysis.cpp/h
HydrologyDCEfficientAnalysis.cpp/h
HydrologyDCInefficientAnalysis.cpp/h
HydrologyGlaDSAnalysis.cpp/h
HydrologyPismAnalysis.cpp/h
HydrologyPrescribeAnalysis.cpp/h
HydrologyShaktiAnalysis.cpp/h
HydrologyShreveAnalysis.cpp/h
HydrologyTwsAnalysis.cpp/h
L2ProjectionBaseAnalysis.cpp/h
L2ProjectionEPLAnalysis.cpp/h
LevelsetAnalysis.cpp/h
LoveAnalysis.cpp/h
MasstransportAnalysis.cpp/h
MeltingAnalysis.cpp/h
MmemasstransportAnalysis.cpp/h
OceantransportAnalysis.cpp/h
SamplingAnalysis.cpp/h
```

```
SealevelchangeAnalysis.cpp/h
SmbAnalysis.cpp/h
SmoothAnalysis.cpp/h
StressbalanceAnalysis.cpp/h
StressbalanceSIAAnalysis.cpp/h
StressbalanceVerticalAnalysis.cpp/h
ThermalAnalysis.cpp/h
UzawaPressureAnalysis.cpp/h
analyses.h
```

This guide was generated directly from the ISSM source code at `/g/data1b/au88/jh7060/ISSM/`. All equation numbers refer to the weak form as implemented in the corresponding C++ file. Source line references are approximate — use `grep` or your editor search to locate exact implementations.