SEACAS Analysis Workflow Codes/Capabilities

Gregory D. Sjaardema

October 7, 2025

Contents

This was originally written September 2015 prior to assembly and change-set support so those are not covered. Io_modify was also written after this document.

1 High-Level Analysis Workflow Components Overview

1.1 Generate mesh

Codes: cubit, pointwise, patran, gridgen, ...

1.2 Manipulate and modify mesh (subset, transform, edit, ...)

- Subsetting of entities (blocks, sets)
- Mesh Transformations (rotate, scale, translate, mirror, warp, adjust)1
- Mesh Improvement (move sidesets to touch or avoid overlap; snap nodes to surface)
- Equivalence nodes based on tolerance (make mesh contiguous)
- Delete elements, merge specific nodes, reposition nodes
- Smooth to improve element shape metrics
- Combine entities (blocks, sets)
- Change attribute values
- Change entity ids, names
- Create/Modify set distribution factors
- Reverse sideset normals
- Deform mesh (modify coordinates based on nodal displacement variable at specific time)

Codes: grepos, mapvar, io_shell, gjoin, ejoin, simba

1.3 Verify model fidelity

- Bounding box of blocks, model
- Mass properties of blocks, model
- Examine/Query bulkdata and metadata information from database
  - nodal coordinates[1]
  - Node and element maps
  - element connectivity
  - results variable values
  - attribute values
  - inverse connectivity
  - coordinate frames
  - locate nodes/elements by various geometric queries
  - subset nodes/elements/entities that information will be displayed for
- cavity volume

[1] If mesh has variables, are they also transformed. Automatically?, Manually?

- overlap of contact surfaces
- time step estimation for explicit dynamics
- gap between two surfaces
- element shape metrics
- skew, aspect ratio, jacobian
- sum element variables over mesh optionally multiplied by volume
- nodal volume
- check for disconnected nodes
- adjacencies (blocks touching sidesets and other blocks)

Codes: explore, grepos, io_info, cubit, numbers, blot

1.4 Combine multiple meshes into single analysis mesh

- combine nodes within tolerance or with same id (optionally filtered by nodeset/sideset/?; should not create invalid topology)
- transform (translate, scale, rotate, ...) before combining
- combine similar blocks/sets into single block/set or retain as multiple block/set
- subset entities (user-selectable blocks/sets are combined)
- renumber, rename entities
- should handle meshes with results data
- *would be desirable to be able to do this "on-the-fly" at analysis time.*

Codes: ejoin, gjoin, cubit

1.5 Refine mesh

- uniform refinement by integer (2^3, 4^3, ...) or arbitrary scale value.
- snap new nodes to geometry
- map results data onto new mesh

Codes: adapt, scale, doublemesh, percept

1.6 Decompose into file-per-processor for parallel runs

- *not needed if using Ioss with auto-decomp.*
- homogeneous systems or heterogeneous, hierarchical decompositions
- arbitrary weightings and graph connectivity
- if done internal to application, can leverage rebalance infrastructure
- mesh may have results data which must also be spread

Codes: nem_slice, nem_spread, decomp, io_shell, slice, yada, yada-yada, penso

1.7 Run analysis

- 1 ⇒ N parallel decomposition "on-the-fly".
  Codes: Ioss
- restart with or without offline adaptivity; possible change in processor count
- mesh adaptation driven by quantity on mesh (error indicator)
  Codes: adapt, scale,
- (N ⇒ M)
  Codes: epu, decomp, nem_slice, nem_spread, Ioss
- loose coupling with results transfer.
  Codes: cth pressure map, shell to hex, mapvar, percept, algebra, ...
- embedded visualization. Codes: catalyst
- N ⇒ 1 parallel recomposition "on-the-fly".
  Codes: Ioss

1.8 Compose file-per-processor output from analysis code into one or more files

- not needed if using Ioss with auto-join
- subset times, blocks/sets, variables

Codes: epu, io_shell, nem_join

1.9 Compose file-per-topology-change output into single file

- subset times, blocks/sets, variables
- merging of element/node status variables
- desirable if Exodus did not require this. Prototype using netcdf groups. (Ioss, io_shell, io_info can demonstrate)

Codes: conjoin

1.10 Query, display, and post process results files

- xy plot (variable vs variable)
- deformed mesh plot with variable
- variable vs time
- mesh info (like explore)
- csv output
- x11, postscript output
- contour
- paint; either discrete or blended colors
- multiple color scale modes
- shading
- transparency

- vector
- pathline, streamlines, level sets
- cut planes
- compare results from two similar analyses
- create new variables via algebraic combinations of old variables
- subsetting mesh by block
- subsetting variables
- subsetting time
- "if" constructs
- envelope of results
- Examine/Query information from database
- minmax (min and max value of given variable at step or over all times)
- tensor principal values and magnitude
- cavity volume changing over time
- feature extraction or identification (fragments, cracks)

Codes: blot, ensight, paraview, explore, numbers, algebra, exodiff, io_info, io_shell,

1.11 Translate to different format

- reverse translation is also sometimes needed.

Codes: exotxt/txtexo, exo2mat/mat2exo, exotec2, exomatlab, ncdump/ncgen, ex2ex1v2/ex1ex2v2, io_shell, blot (csv, xmgr), Ioss (spyplot, csv, history)

1.12 Others, no specific category or all categories

- output bit size for integers and floating point
- output underlying format (netcdf, hdf, ...)
- output netcdf mode (classic, 64, ...)
- compression settings and statistics

Codes: exo_format, ncdump

2 Exodus

A brief summary of the Exodus format used as the default database output for all Sierra applications and many other Sandia and non-Sandia analysis codes. Not all capabilities are used or supported by all applications which can cause compatibility issues when analysts are attempting complicated workflows.

2.1 Format

- netcdf-based
- classic, 64-bit offset[2], hdf5-classic, hdf5[3]
- integer size: 32-bit or 64-bit for bulk, ids, maps[4]
- floating point size: 32-bit or 64-bit
- compression of integer and floating point data supported with hdf5-based formats.
- parallel io—pnetcdf, mpiio, mpi-posix, file-per-processor

2.2 Entities

- A mesh-entity is an individual node, edge, face, or element.
- An entity is a set or block consisting of a single mesh-entity type.
- Each entity can have variables, maps, and attributes which contain an entry per mesh-entity.
- Each entity has an optional name and a required id (32-bit or 64-bit)which is non-negative.
- A mesh-entity can be in one and only one entity block,
- A mesh-entity can be in zero or more entity sets.
- Currently there is only a single implicit node block containing all nodes in the model.

2.3 Names

- The following can be named: entities (element, face, edge, node blocks and sets), attributes, maps, and entity variables.
- the names of the above can be any length up to 255 (which is limit of hdf5)
- Names of QA strings limited to 32 characters
- Element type names limited to 32 characters
- Info records limited to 80 characters
- title limited to 80 characters

2.4 Attributes

- Attributes are optional floating point data defined on an entity with a value for each mesh-entity in that entity.
- Common element attributes are shell-thickness, particle diameter, and rod cross-sectional area.
- No limit to the number of attributes.

2.5 Maps

- There are zero or more named maps for each of node, edge, face, and element.
- A reserved map is used for node and element global ids.

[2]NOTE: the 64-bit offset format has nothing to do with the size of integers stored in the database.

[3]The classic format handles models up to about 30 million nodes; the 64-bit offset will handle models up to about 540 million nodes and a maximum of 134 million hexes per element block; the hdf5-based formats are virtually unlimited...

[4]If 64-bit integers are used, then the netcdf format must be hdf5-based

2.6 Arbitrary Polyhedra

- Exodus supports arbitrary polygon (2D) and arbitrary polyhedra (3D) elements.
- Capability has been used by some 1400 projects, at LANL, and some external customers.
- I think it is supported by Paraview and Ensight
- None of the Seacas codes currently handle this capability.

2.7 Compatibility

- All codes should support reading and/or writing Exodus database with:[5]
  - less than 2^{31} nodes or elements
  - 32-character or less variable names
  - global, nodal, and element variables
  - one node block, multiple element blocks, sidesets, and nodesets.
  - classic or 64-bit-offset netcdf format.
- The other netcdf formats should be handled seamlessly if the code is linked to a current netcdf library built with netcdf-4 support. However, it is probably that those codes might not output the same format unless they have been modified to select that format.
- Support for 64-bit integers is provided in most of the Seacas codes.
  - All fortran-based Seacas codes should automatically support this;
  - most of the C-based and C++-based Seacas codes have been modified to support this;
  - Support by external codes is unknown.
- Support for long names is provided in many of the Seacas codes.
  - A code linked to a current Exodus library will not crash if long names are used; however, all names longer than the previous default of 32-characters will be truncated on input and output.
  - A code linked to an older Exodus library will crash if it tries to read a database containing names longer than 32 characters.
  - Many of the "legacy" Seacas codes do not understand or use entity names.
- Most of the "legacy" Seacas codes do not recognize nodeset or sideset variables.
- Most of the "legacy" Seacas codes do not recognize face and edge sets or blocks or variables on those sets and blocks.
- Most of the "legacy" Seacas codes do not recognize attributes on anything other than element blocks. Some of those also are restricted to the "standard" attributes.

2.8 Limitations

- Does not handle mesh topology changes. A new file must be created each time the mesh topology changes. If run in file-per-processor mode, then each processor creates a new file resulting in too many files which can and has overwhelmed and crashed filesytems. (The new change-set capability begins to address this)

[5] In other words, if a code only reads Exodus, it should be able to process or understand all of the data in the file if the file only contains these items. If the code read and writes Exodus, then the output database should contain all data on the input database if the file contains only these items. This may be further limited in that some codes only support a limited set of element types.

- Default file-per-processor mode requires additional tool support and limits analysts when compared to the single file, decomposition-on-the-fly mode that most analysts prefer and most commercial and external codes support.
- Would be good to have hierarchical grouping of entities with variables definable on the groups. (The new assembly capability addresses this)
- Can be slow on some systems; however, this is (hopefully) due to insufficient tuning on those systems due to limited IO/EXODUS support over the past few years.
- Need explicit composite fields (scalar, vector, tensor, quaternion) instead of current "heuristic" method based on field name suffix. (The new DG / Enhanced-Field metadata capability addresses this)
- Need integer fields.

3 Codes and Capabilities

(S) In SEACAS, both external `GitHub` distribution and sierra
(D) Deprecated. Was previously in SEACAS; no longer supported
(DS) In SEACAS, but should be deprecated
(X) Experimental. Under development or trying out some concepts
(E) External. Not in SEACAS, but reads and/or writes EXODUS files.

3.1 Graphics

- blot (S)
  - xy plot
  - deform plot
  - var vs time
  - mesh info (like explore)
  - csv output
  - x11, postscript output
  - contour
  - paint; either discrete or blended colors
  - multiple color scale modes
  - shading (limited)
  - vector
  - pathline
  - node/element id from location
  - global, nodal, and element variables; does not support nodeset and sideset and others.
- ensight (E)
- paraview (E)
- catalyst (E)

3.2 Mesh Modification

- algebra (S)
  - create new variables via algebraic combinations of old variables
  - subsetting mesh by block
  - delete specified elements by id or value of element variable
  - subsetting variables
  - subsetting time
  - "if" constructs
  - envelope
  - tensor principal values and magnitude
  - global, nodal, and element variables; does not support nodeset and sideset and others.
- conex (D)
  - all capabilities are provided by conjoin, deprecated.
- conjoin (S)
  - join two or more EXODUS databases temporally; similar topology
  - used to join outputs from restarted analyses or the multiple files arising from mesh topology changes which are not handled by EXODUS.
  - mesh topology can change over time
- exodiff (S)
  - compare two EXODUS files with similar topology
  - NaN detection
  - many options
- io_shell (S)

3.3 Mesh Generation

- fastq (DS)
- gen3d (DS)
- genhxshl (DS)
- genshell (DS)
- Cubit (E)
- Pointwise (E)
- Patran (E)
- Gridgen (E)
- io_shell (S) (using generated option to give rectangular "regular" mesh)

3.4 Mesh Manipulation

- ejoin (S)

  - join 2 or more mesh with optional results into single file
  - combine nodes within tolerance or with same id (cannot filter by nodeset)
  - produces an output entity for each input entity on each input database (no coalescing)
  - can subset entities
  - offset input mesh (all but first offset by same value)
  - can subset variables

- gjoin (D)

  - join 2 or more mesh (no results) files into single file
  - combine entities (element blocks, node sets, side sets) into single entity
  - delete entities
  - renumber entity ids
  - user-specified tolerance
  - limit nodes by nodeset
  - transform (scale, rotate, translate, mirror) input mesh
  - all capabilities should be supported in ejoin...

- grepos (S)

  - Mesh Orientation:
    - reposition or modify database
    - adjust
    - explode
    - mirror
    - move sidesets to touch
    - offset
    - randomize (random offsets to nodal coordinates)
    - revolve
    - scale
    - scale attributes
    - laplacian smoothing
    - snap (move sideset nodes to lie on surface of another sideset) k
    - warp (maps e.g., tire tread onto carcass)
    - zero (sets all nodal coordinates with tolerance of zero to zero)
  - Mesh Modification:
    - change ids
    - change names
    - change attribute values
    - combine entities into a single entity
    - delete entities
    - delete qa, info
    - equivalence nodes
    - increment entity ids
    - swap sidesets (reverse normal)
    - elementize (convert nodal to element variable)
    - centroids (create element variable showing element centroid location)

        * deform (modify coordinates based on nodal variable at specific time)
        * merge (combine two nodes into single node)
    - Mesh Information:
        * limits (bounding box)
        * list info about various entities

- mapvar (S)

- mapvar-kd (S)

    - transfer solutions results from one mesh to another
    - multiple schemes (nodal average, liner constrained least squares, direct transfer)
    - mapvar-kd basically same as mapvar except uses kd-tree search for better speed.

- io_shell (S)

- io_modify (S)

3.5 Mesh Query

- exo_format (S)

    - output bit size for integers and floating point
    - output underlying format (netcdf, hdf, ...)
    - output netcdf mode (classic, 64, ...)
    - should probably be combined into io_info or explore...

- explore (S)

    - Examine/Query information from database
    - subset nodes/elements/entities that information will be displayed for
    - user-settable floating point precision
    - database summary information
    - node and element maps
    - blocks, type, connectivity
    - attribute values
    - results variables values
    - sets
    - entries and df
    - qa and info
    - output to screen or file
    - check (check validity of database)
    - minmax (min and max value of given variable at step or over all times)
    - supports nodeset and sideset variables
    - show block membership of nodes (inverse connectivity)
    - limits (bounding box)
    - list coordinate frames

- numbers (S)

  - mass properties
  - locate nodes/elements by various geometric queries
  - cavity volume
  - overlap of contact surfaces
  - time step estimation for explicit dynamics
  - gap between two surfaces
  - limits
  - bounding boxes of element blocks.
  - element shape metrics
  - skew, aspect ratio, jacobian
  - sum element variables over mesh optionally multiplied by volume
  - nodal volume
  - hex (3D) and quad (2D) meshes only (recent support for tetrahedral elements added)

- io_info (S)
  - part of Ioss test programs, but provides some useful information about database
  - check for disconnected nodes
  - adjacencies (blocks touching sidesets and other blocks)
  - computes element volume (hex only)
  - bounding box of each element block
  - summary of input mesh
- io_modify (S)

3.6 Format Conversion

- exotxt (D)
- txtexo (D)
  - convert Exodus to/from text file (can lose precision, doesn't support all of Exodus). Use ncdump and ncgen instead.
- exotec2 (D)
  - convert Exodus to tecplot
- mat2exo (S)
- exo2mat (S)
  - convert Exodus file to matlab database
  - binary or text
- exomatlab (S)
  - convert Exodus global variables to matlab-readable format
  - should also do specified nodal and element variables (not implemented)
- nas2exo (S)
  - convert nastran output to Exodus
- ncdump (E)
- ncgen (E)

  - netcdf-supplied routines for converting EXODUS file to/from text

- ex1ex2v2 (D)
- ex2ex1v2 (D)
  - convert to/from current EXODUS format and old format. There should be no need to use these unless you have an archived database that you need to access.
- io_shell (S)
- struc_to_unstruc (S)
  - convert structured mesh to unstructured mesh.

3.7 Parallel Decomposition

- nem_slice (S)
  - determine parallel decomposition of input mesh for specified number of processors
  - many decomposition options:

    chaco: multikl, spectral, inertial, random, brick, zpinch
    zoltan: hsfc, rib, rcb, rcb ignore z,
    other: linear, random, scattered
  - weighting supported
  - handles spheres
  - optionally output a visualization mesh showing decomposition
- nem_spread (S)
  - read mesh database and corresponding nemesis file created by nem_slice and output a file-per-processor based on the decomposition.
  - handles 64-bit integer files
- io_shell (S)
  - io_shell reads a mesh and writes same mesh using Ioss
  - since Ioss has auto-decomp and auto-join option, can use io shell to decompose and join meshes in parallel
  - supports: rcb, rib, hsfc, metis sfc, kway, kway geom, linear, cyclic, random
  - will also do N $\Rightarrow$ 1 (auto-join)
  - handles 64-bit integer files
- decomp (S)
  - new script that drives nem_slice and nem_spread.
- loadbal(D)
  - old script that drives nem_slice and nem_spread.
- slice (X)
  - experimental

  - decomposition and file-per-processor output
  - decomposition options: linear, scattered, random, rb, kway, file
  - handles 64-bit ids
  - contiguous decomposition option
  - each piece is contiguous
  - subsetting of entities

- stk_decomp

- yada-yada

- yada

- penso

3.8 Parallel Recomposition

- epu (S)

  - join file-per-processor output files into single file
  - "parallel option" (subcyle)
  - can append to existing file with the same mesh topology

- io_shell (S)

  - io_shell reads a mesh and writes same mesh using Ioss
  - since Ioss has auto-decomp and auto-join option, can use io shell to join meshes in parallel
  - handles 64-bit integer files

- nem_ join (D)

  - deprecated. All capabilities (and more) should be provided by epu

3.9 Mesh Conversion or Refinement

- sphgen3d (D) convert hex mesh to spherical elements
- exodus.py (S) python api for accessing Exodus databases
- exomerge.py (S) python api for accessing Exodus databases
- adapt (E)
- percept (E)
- scale (E)
- cth pressure map (S)
- shell to hex (S)

More detail