Oasys

Oasys GSA

Export Reference

Oasys
YOUR IDEAS BROUGHT TO LIFE

8 Fitzroy Street
London
W1T 4BJ
Telephone: +44 (0) 20 7755 4515

Central Square
Forth Street
Newcastle Upon Tyne
NE1 3PL
Telephone: +44 (0) 191 238 7559

e-mail: oasys@arup.com
Website: oasys-software.com

Oasys GSA

Contents

GSA export

It is possible to export GSA data and results for creation of a database. Certain model data is exported followed by results for nodes, elements and assemblies. Results for analysis cases are exported. Results for combination cases are exported for cases which have a single set of results, e.g. simple combinations. Envelope combinations are output as a max and a min value.

The extraction creates a folder with the same name as the GSA model and information if written to a single file for each data or results type.

The extraction uses GsaShell with the command line:

gsashell --action export-database --gsafile <gsa_file> --filetype <type> --input <in_file>

or

gsashell --action export-database --gsafile <gsa_file> --filetype <type> --points <num> --include-interesting <option>

where

- gsa_file is the file to be exported,
- type is csv or sql and
- in_file is an input file give details of what is to be exported
- num is the number of points along a beam element
- option is true or false for including interesting points in the export

The input file is a simple text file of key value pairs separated by a tab. The following options can be specified:

- node – node list (default all)
- element – element list (default all)
- member – member list (default all)
- assembly – assembly list (default all)
- case – case list (default all)
- num_points – number of points for output along beam elements (default 5)
- interesting_points – true/false to include/exclude results at interesting points on beam elements (default false)
- delimiter – the field delimiter for csv files (typically , | or tab)

The output of model data is unaffected by the entity lists.

COM

The export is available through COM using the command

```
ExportToCsv(BSTR pathname, LONG numPoint, VARIANT_BOOL interestingPoints,
      VARIANT_BOOL combinations, BSTR delimiter, SHORT* retCode)
```

With arguments

`pathname` – folder for output

`numPoint` – number of points along beam elements

`interestingPoints` – flag to include interesting points

`combinations` – flag to include combinations (not used at present)

`delimiter` – field separator (usually a comma)

Data

Model and results data can be exported to csv or sql files. They create a series of tables as detailed below.

Model (model)

- job_number
- initials
- title
- subtitle
- calculation_header
- steel_code
- concrete_code
- notes
- guid – model guid

Location (location)

- country – ISO abbreviation
- north – angle to north
- elevation
- latitude
- longitude
- currency

History (history)

- date
- time
- user

- notes

Axis (axis)

- id – axis id
- name
- type
- origin_x, origin_y, origin_z – origin
- xaxis_x, xaxis_y, xaxis_z – x vector
- yaxis_x, yaxis_y, yaxis_z – y vector
- zaxis_x, zaxis_y, zaxis_z – z vector

Where type is one of CARTESIAN, CYLINDRICAL or SPHERICAL.

Grid plane (grid_plane)

- id – grid plane id
- name
- type
- axis
- elevation

Where type is STOREY or PLANE.

Lists (list)

- id – list id
- name
- type
- description
- expanded – expanded description

Where type is NODE, ELEMENT or MEMBER and expanded is an expanded list separated by commas.

Nodes (node)

- id – node id
- name
- colour
- position _x, position _y, position _z – position
- axis
- restraint
- spring – spring property id
- mass – mass property id
- damper – damper property id

Where restraint is a string containing x, y, z, xx, yy, zz for restrained degrees of freedom of pin (equivalent to xyz) or fix (equivalent to xyzxxyyzz)

Eléments (element)

- id – element id
- name
- colour
- type
- prop_1d_id – 1D property id
- prop_2d_id – 2D property id
- prop_3d_id – 3D property id
- group_id
- num_node – number of nodes
- node_1 ... node_8 – nodes
- orientation_angle
- dummy
- parent_member
- member_type – parent member type
- offset_x1
- offset_x2
- offset_y
- offset_z
- release_1 – (string, only for beam elements)
- release_2

Where nodes are 8 fields filled as appropriate to the element type. Element types are:

- BEAM
- BAR
- ROD
- TIE
- STRUT
- TRI3
- QUAD4
- TRI6
- QUAD8
- TETRA4
- PYRAMID5
- WEDGE6
- BRICK8
- CABLE
- SPRING
- DAMPER
- LINK

All with two nodes unless the number is specified in the type. If a beam type element has three nodes the third node is the orientation node.

And the parent member types are

- BEAM
- COLUMN
- 1D_GENERIC
- SLAB
- WALL
- 2D_GENERIC
- 3D_GENERIC

The property ID is null if not relevant for this element.

Members (member)

- id – member id
- name
- colour
- type
- prop_1d_id – 1D property id
- prop_2d_id – 2D property id
- prop_3d_id – 3D property id
- group_id
- topology – (string)
- type_1d
- type_2d
- mesh_size
- orientation_angle
- dummy
- offset_x1
- offset_x2
- offset_y
- offset_z
- release_1 – (string, only for 1D members)
- release_2
- BIM id

Where the member types are

- BEAM
- COLUMN
- 1D_GENERIC
- SLAB
- WALL
- 2D_GENERIC
- 3D_GENERIC

And the type 1D options are from the 1D element types above, and the type 2D options are

- LINEAR
- QUADRATIC

- RIGID

Assemblies (assembly)

- Id – assembly id
- name
- storey – (TRUE or FALSE)
- geometry
- node_1
- node_2
- node_3
- internal – (string)
- dim_y – dimension y
- dim_z – dimension z

Where the geometry options are LINEAR, CIRCULAR and LAGRANGE

1D properties (section)

- id – property id
- name
- colour
- material – material type
- gr
- grade_id – material grade id
- description
- perimeter – (string in form (x_0,y_0) (x_1,y_1)...)
- pool
- area
- iyy – I_{yy}
- izz – I_{zz}
- j – J
- ky – k_{yy}
- kz – k_{zz}
- volume_per_length – volume per unit length

2D properties (prop_2d)

- id – property id
- name
- colour
- material – material type
- grade_id – material grade id
- description
- thickness

3D properties (prop_3d)

- id - property id
- name
- colour
- material – material type
- grade_id – material grade id

Results

Results should be available for analysis case and combination cases with syntax

- A*i* for analysis case *i*
- C*i* for combination case *i* where the combination case is simple
- C*i*p*j* for combination case *i*, permutation *j*

Results denoted by a * mean the result is output as a value/permutation pair. The permutation field has the same title as the field to which it relates but has '_perm' appended. For analysis cases and simple combinations the permutation will be blank.

Cases (case)

- case_id
- case_type – (Analysis, Combination)
- case_number
- case_permutation
- description

Permutations (permutation)

- case_id
- case_number
- case_permutation
- description

Global (result_global)

- case_id
- case_type – (Analysis, Combination)
- case_number
- case_permutation
- load_x, load_y, load_z, load_xx, load_yy, load_zz – total load
- reaction_x, reaction_y, reaction_z, reaction_xx, reaction_yy, reaction_zz – total reaction
- mode
- frequency
- load_factor
- modal_stiff – modal stiffness

- modal_geo_stiff - modal geometric stiffness
- modal_mass
- effective_mass_x, effective_mass_y, effective_mass_z, effective_mass_xx, effective_mass_yy, effective_mass_zz

Where items from mode down only have values for dynamic and buckling analysis as appropriate

Nodes (result_node)

- id – node id
- case_id
- case_type – (Analysis, Combination)
- case_number
- disp_x, disp_y, disp_z, disp_x, disp_y, disp_zz*
- reaction_x, reaction_y, reaction_z, reaction_xx, reaction_yy, reaction_zz*
- constraint_x, constraint_y, constraint_z, constraint_xx, constraint_yy, constraint_zz*
- vel_x, vel_y, vel_z, vel_xx, vel_yy, vel_zz – velocity*
- acc_x, acc_y, acc_z, acc_xx, acc_yy, acc_zz - acceleration*

1D elements (result_elem_1d)

- id – element id
- case_id
- case_type – (Analysis, Combination)
- case_number
- position_r – relative position (r in range [0:1])
- disp_x, disp_y, disp_z – displacement*
- force_x, force_y, force_z*
- moment_x, moment_y, moment_z*

2D elements (result_elem_2d)

- id – element id
- case_id
- case_type – (Analysis, Combination)
- case_number
- position_r, position_s – relative position (r,s in range [0:1])
- disp_x, disp_y, disp_z – displacement*
- force_xx, force_yy, force_xy – in-plane force N*
- moment_xx, moment_yy, moment_xy – moment M*
- shear_x, shear_y – shear force*
- stress_top_xx, stress_top_yy, stress_top_zz, stress_top_xy, stress_top_yz, stress_top_zx*
- stress_middle_xx, stress_middle_yy, stress_middle_zz, stress_middle_xy, stress_middle_yz, stress_middle_zx*
- stress_bottom_xx, stress_bottom_yy, stress_bottom_zz, stress_bottom_xy, stress_bottom_yz, stress_bottom_zx*

- *strain_top_xx, strain_top_yy, strain_top_zz, strain_top_xy, strain_top_yz, strain_top_zx**
- *strain_middle_xx, strain_middle_yy, strain_middle_zz, strain_middle_xy, strain_middle_yz, strain_middle_zx**
- *strain_bottom_xx, strain_bottom_yy, strain_bottom_zz, strain_bottom_xy, strain_bottom_yz, strain_bottom_zx**
- *pore_pressure*

3D elements (result_elem_3d)

- id – element id
- case_id
- case_type (Analysis, Combination)
- case_number
- position_r, position_s, position_t – relative position (r,s,t in range [0:1])
- disp_x, disp_y, disp_z – displacement*
- stress_xx, stress_yy, stress_zz, stress_xy, stress_yz, stress_zx*
- *strain_xx, strain_yy, strain_zz, strain_xy, strain_yz, strain_zx**
- *pore_pressure*

Springs (result_spring)

- id – entity id
- type – (NODE or ELEMENT)
- case_id
- case_type – (Analysis, Combination)
- case_number
- position_r – relative position (r as 0 or 1)
- disp_x, disp_y, disp_z – displacement*
- force_x, force_y, force_z*
- moment_x, moment_y, moment_z*

Assembly (result_assembly)

- id – assembly id
- case_id
- case_type – (Analysis, Combination)
- case_number
- position_r – relative position (r in range [0:1])
- disp_x, disp_y, disp_x – displacement*
- rotn_x, rotn_y, rotn_z – rotation*
- drift_x, drift_y, drift_z*
- force_x, force_y, force_z*
- moment_x, moment_y, moment_z*

Element positions & interpolation functions

Results on elements need to be located at a given position. This can be done with respect to an embedded axis system

1D elements

The position is given by a coordinate r in the range [0:1]

The interpolation function is given by

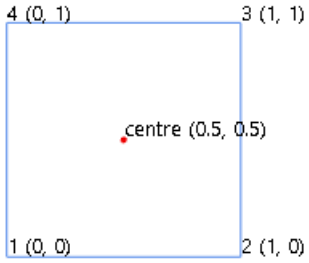
$$x(r) = \sum_i h_i x_i$$

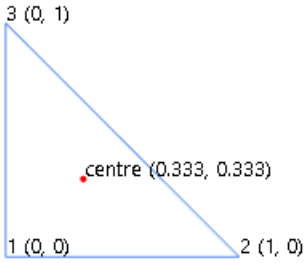
where

$$h_1 = (1 - r)$$
$$h_2 = r$$

2D elements

The position is given by a coordinates r and s in the range [0:1]





The interpolation function is given by

$$x(r,s) = \sum_i h_i x_i$$

Where the quad interpolation functions are

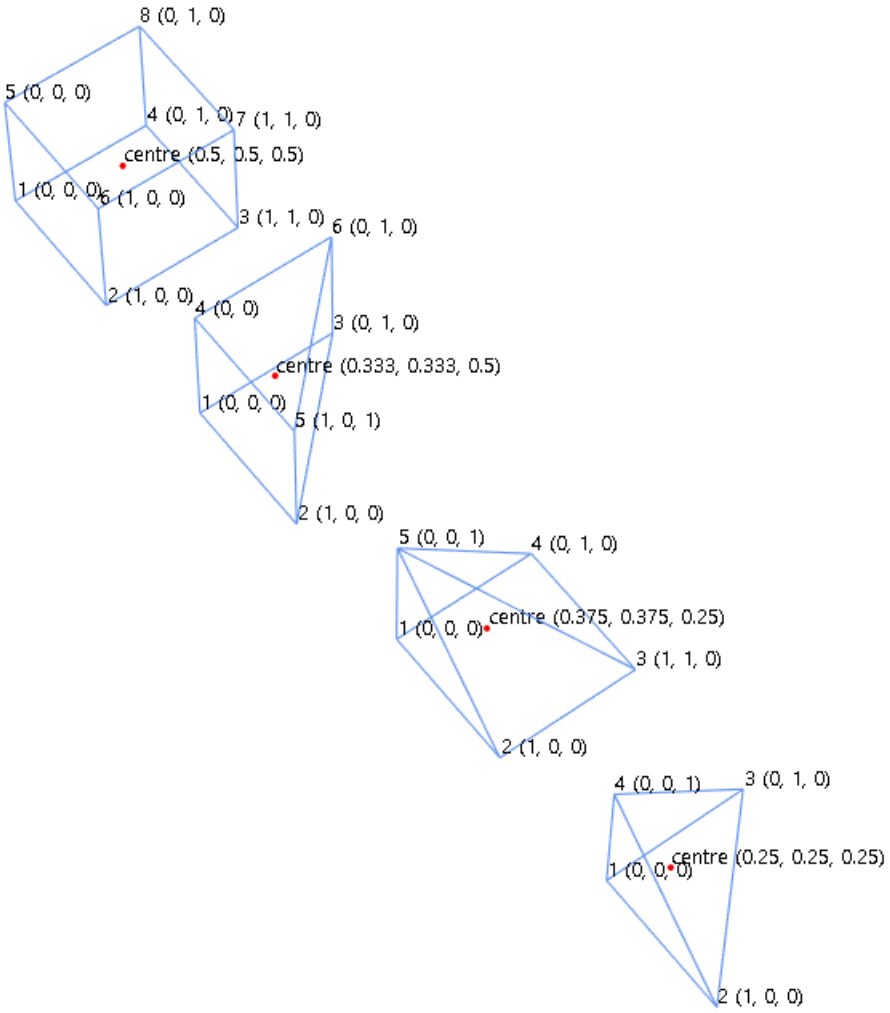
$$h_1 = (1-r)(1-s)$$
$$h_2 = r(1-s)$$
$$h_3 = r\,s$$
$$h_4 = (1-r)\,s$$

and the triangle interpolation functions are

$$h_1 = (1-r)(1-s)$$
$$h_2 = r(1-s)$$
$$h_3 = (1-r)\,s$$

3D elements

The position is given by a coordinate r, s and t in the range [0:1]



The interpolation function is given by

$$x(r,s,t) = \sum_i h_i x_i$$

Where the brick interpolation functions are

$$h_1 = (1-r)(1-s)(1-t)$$
$$h_2 = r(1-s)(1-t)$$
$$h_3 = r \cdot s(1-t)$$
$$h_4 = (1-r)s(1-t)$$
$$h_5 = (1-r)(1-s)t$$
$$h_6 = r(1-s)t$$
$$h_7 = r \cdot s \cdot t$$
$$h_8 = (1-r)s \cdot t$$

the wedge interpolation functions are

$$h_1 = (1-r)(1-s)(1-t)$$
$$h_2 = r(1-s)(1-t)$$
$$h_3 = (1-r)s(1-t)$$
$$h_4 = (1-r)(1-s)t$$
$$h_5 = r(1-s)t$$
$$h_6 = (1-r)s \cdot t$$

the pyramid interpolation functions are

$$h_1 = (1-r)(1-s)(1-t)$$
$$h_2 = r(1-s)(1-t)$$
$$h_3 = r \cdot s(1-t)$$
$$h_4 = (1-r)s(1-t)$$
$$h_5 = (1-r)(1-s)t$$

and the tetra interpolation functions are

$$h_1 = (1-r)(1-s)(1-t)$$
$$h_2 = r(1-s)(1-t)$$
$$h_3 = (1-r)s(1-t)$$
$$h_4 = (1-r)(1-s)t$$