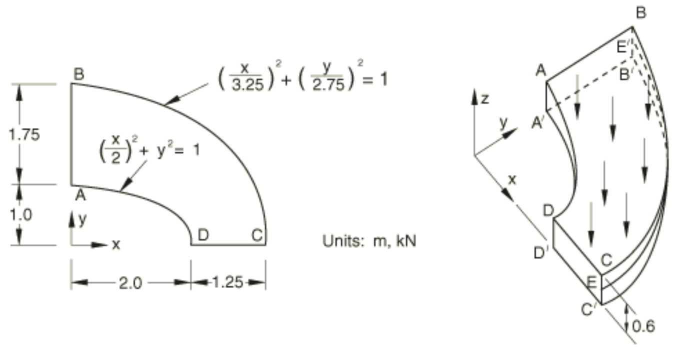
NAFEMS Benchmark Test LE10

Problem description



Model:

Thick plate under uniform pressure.

Mesh:

A coarse and a fine mesh are tested.

Material:

Linear elastic, Young's modulus = 210 GPa, Poisson's ratio = 0.3, density = 7800 kg/m^3.

Boundary conditions:

$u_y = 0$ on face DCD′C′. $u_x = 0$ on face ABA′B′. $u_x = u_y = 0$ on face BCB′C′. $u_z = 0$ on line EE′ (E is the midpoint of edge CC′; E′ is the midpoint of edge BB′).

Loading:

Uniform normal pressure of 1.0 MPa on the upper surface of the plate.

Reference solution

This is a test recommended by the National Agency for Finite Element Methods and Standards (U.K.): Test LE10 from NAFEMS Publication TNSB, Rev. 3, "The Standard NAFEMS Benchmarks," October 1990.

Target solution: Direct stress, $\sigma_{yy} = 5.38$ MPa at point D.

Patch

- A patch is a basic geometry object, and is defined as a collection of elements, connected at nodes (control points), with a basis assigned to it. If geometry is imported from another data source, each spline object or mesh object imported will be its own patch. Patches can be imported into the JSON file using the Include keyword. A patch can define an entire U-spline or single Bézier element. Geometry is stored at the patch level. Each patch is associated to a unique patch_id.
- Coreform provides a selection of patch creation tools that will help users to more easily define geometry. This library is not intended to be comprehensive but does begin to build some of the foundational tools for CAD. The first section is a library of typical geometries encountered and understood in IGA. The second section is a library of operations such as sweeps, revolves, etc. that allow for modifcation or enhancement of patches.
- This version of the software also includes some special parameterized objects that can be used to create helical geometries, etc. These are prototypes of the possibilities of more specialized primitives that can be created as IGA objects for use both in CAD and simulation.
- In the future, support will also be added to create unstructured U-spline primitives.

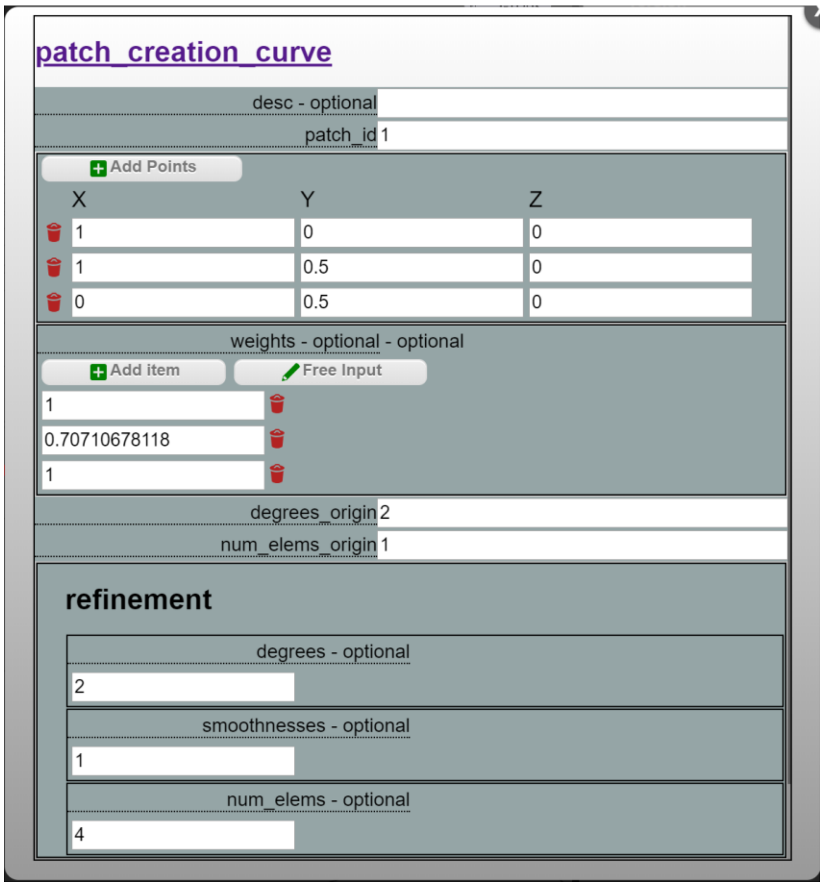
Documentation: `patch_creation_curve`

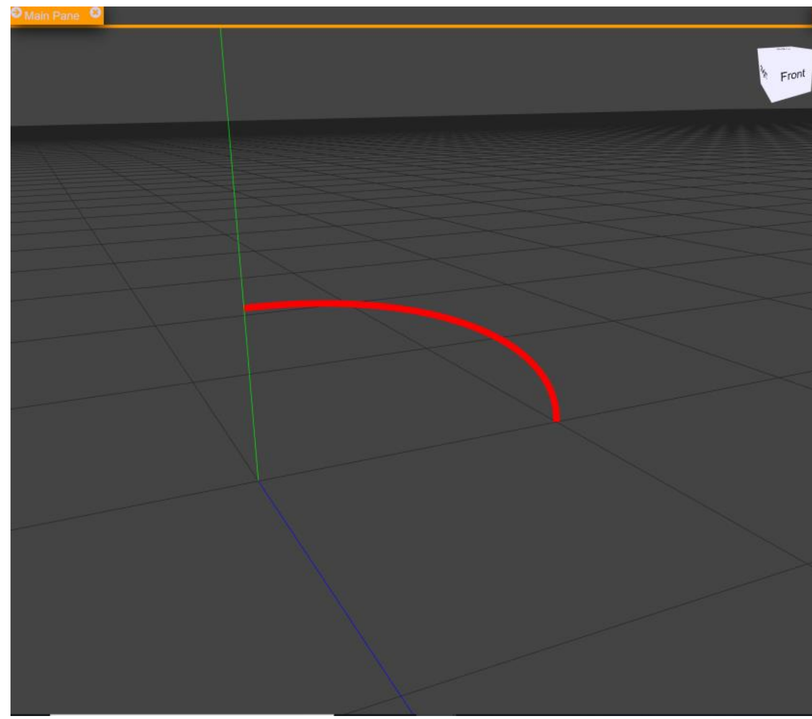
patch_creation_curve

Define a linear segment between two points.

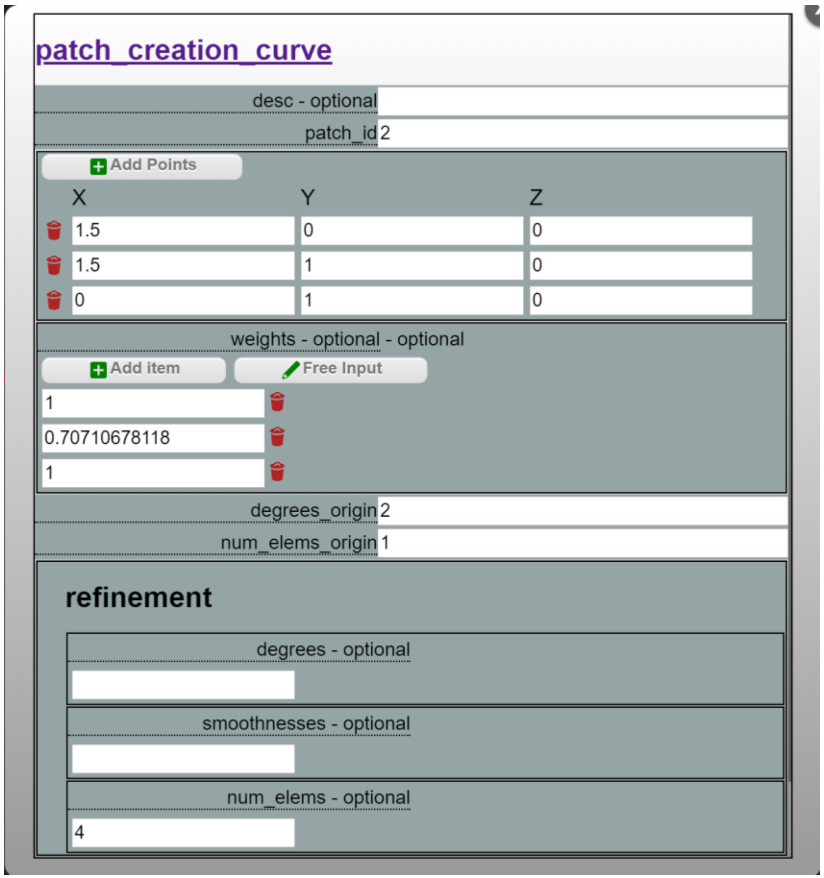
<table>
<tr><th>Variable</th><th>Value Description</th></tr>
<tr><td>patch_id</td><td>A unique ID number used to identify the patch.</td></tr>
<tr><td>points</td><td>Array of the control points (X, Y, Z components) of the patch. These are in physical coordinates.</td></tr>
<tr><td>weights</td><td>The map from patch node IDs to nodal weights. If the is_rational flag is true then this array must be present.</td></tr>
<tr><td>degrees_origin</td><td>Degree of the original curve.</td></tr>
<tr><td>num_elems_origin</td><td>Number of element of the original curve.</td></tr>
<tr><td>refinement</td><td><table>
<tr><th>Object Member</th><th>Value Description</th></tr>
<tr><td>degrees</td><td>A vector defining the desired degree in each parametric direction.</td></tr>
<tr><td>smoothnesses</td><td>A vector defining the desired smoothness in each parametric direction. By default, degrees - 1.</td></tr>
<tr><td>num_elems</td><td>A vector defining the number of elements desired in each parametric direction.</td></tr>
</table></td></tr>
</table>

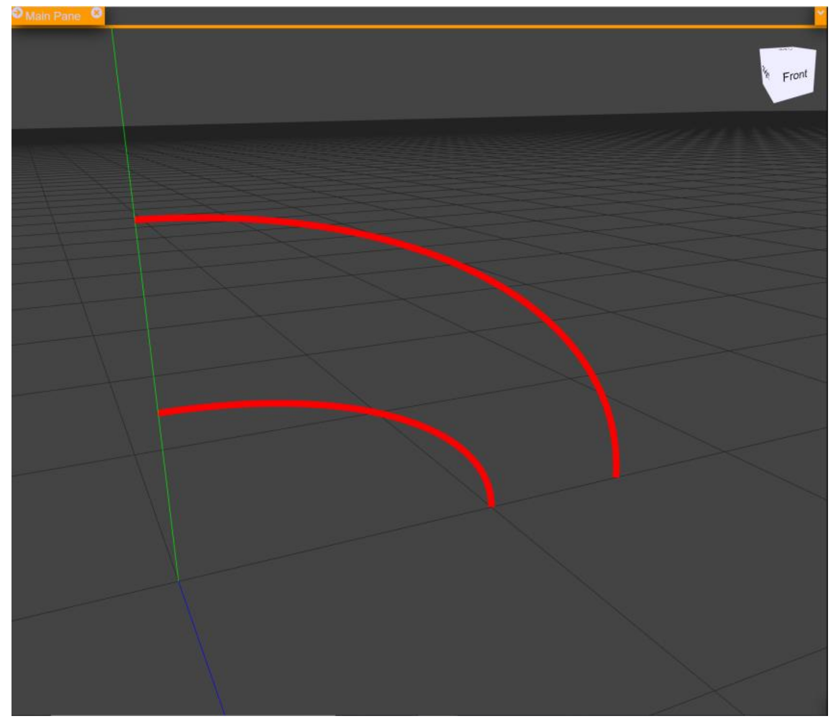
Create inner arc



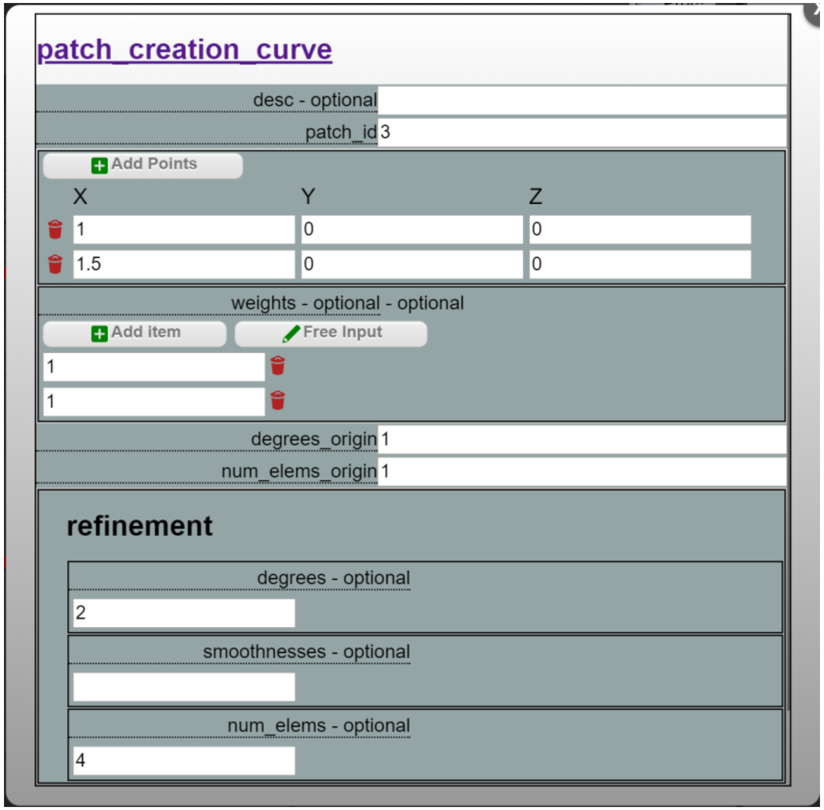


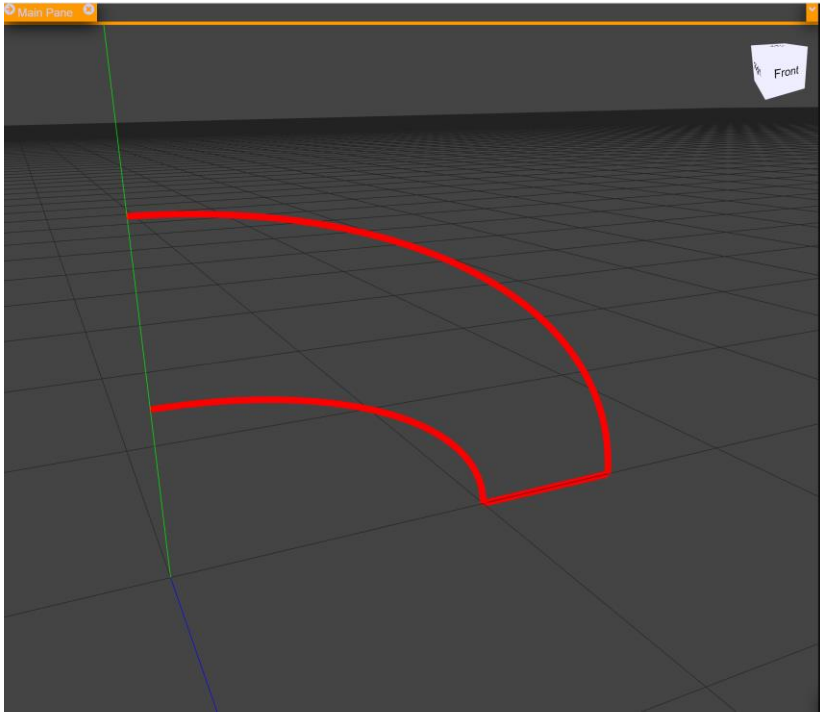
Create outer arc



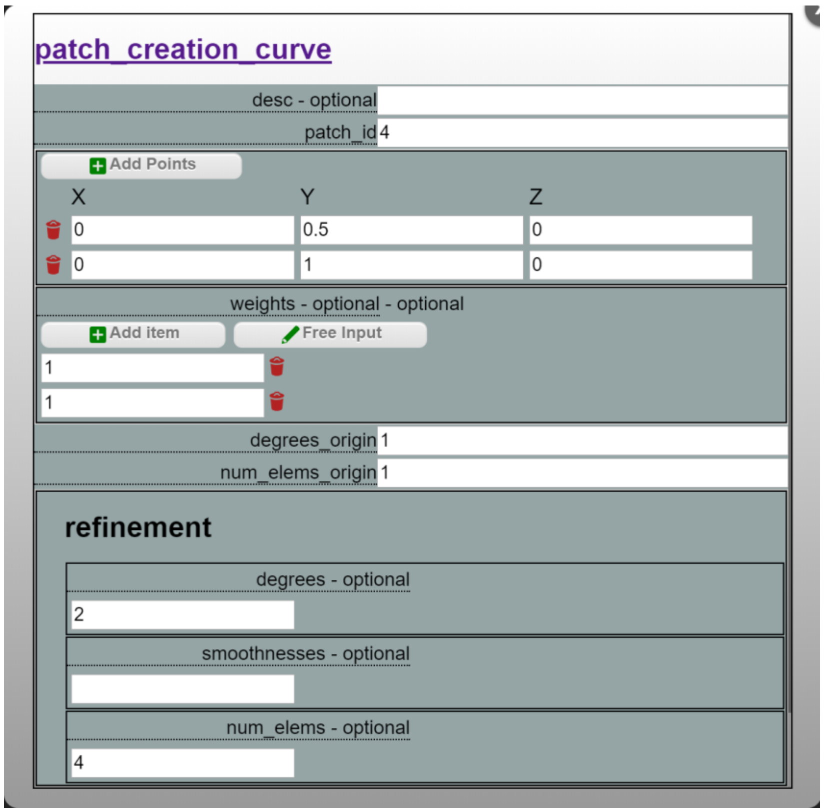


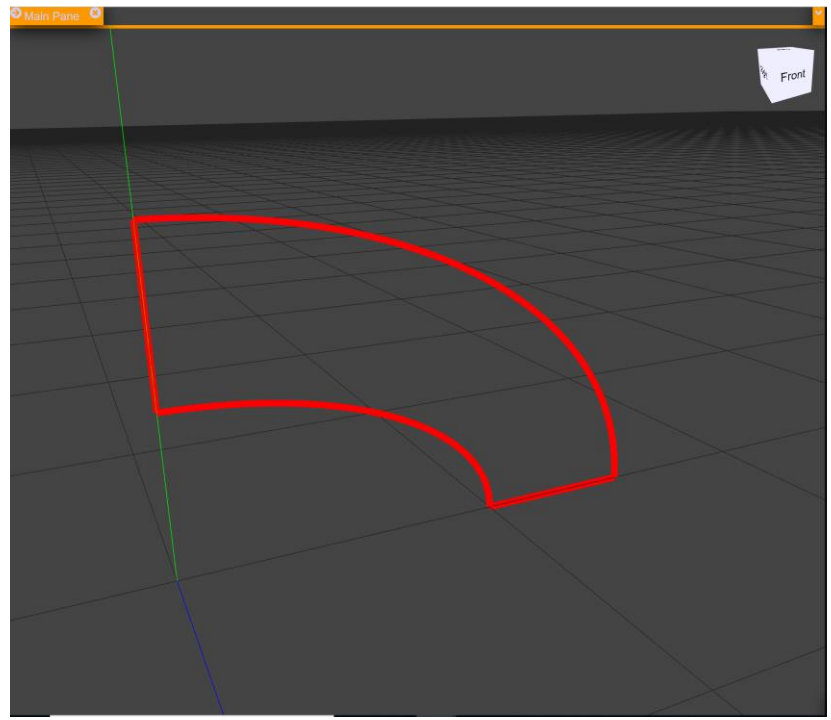
Create connecting curve





Create connecting curve





Documentation: patch_operation_coons

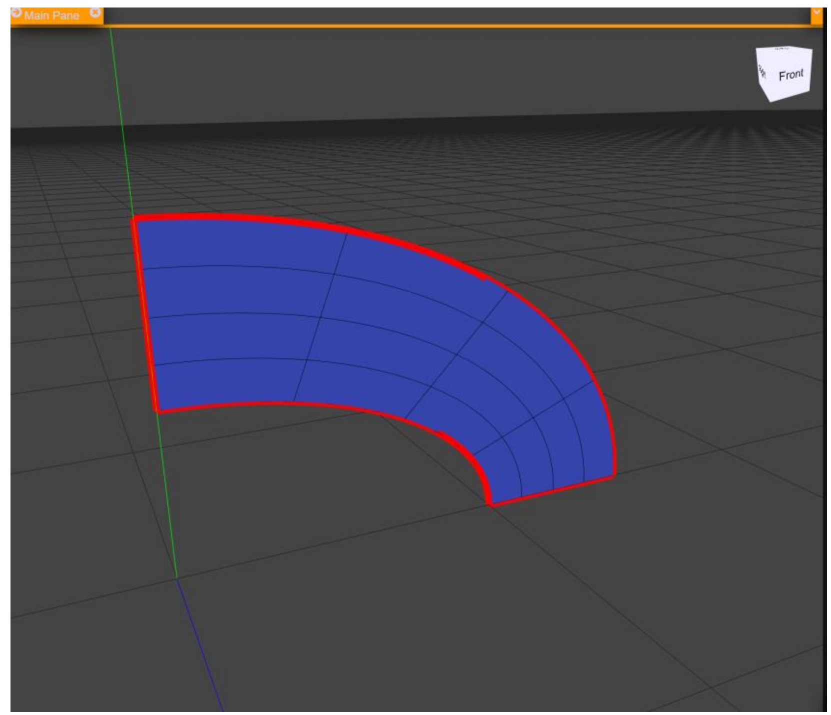
patch_operation_coons

Create a Coons surface patch from four curve patches describing its contour.

Variable	**Value Description**
patch_operation_id	A unique ID number used to identify the patch operation.
patch_id	A unique ID number used to identify the new patch created in this keyword.
patch_id_origin_1	Patch ID of the left curve (s=0, t).
patch_id_origin_2	Patch ID of the right curve (s=1, t).
patch_id_origin_3	Patch ID of the bottom curve (s, t=0).
patch_id_origin_4	Patch ID of the top curve (s, t=1).

patch_operation_coons

desc - optional	
patch_operation_id	1
patch_id	5
patch_id_origin_1	1
patch_id_origin_2	2
patch_id_origin_3	3
patch_id_origin_4	4




version
patch_creation_curve
patch_creation_curve
patch_creation_curve
patch_creation_curve
patch_operation_coons

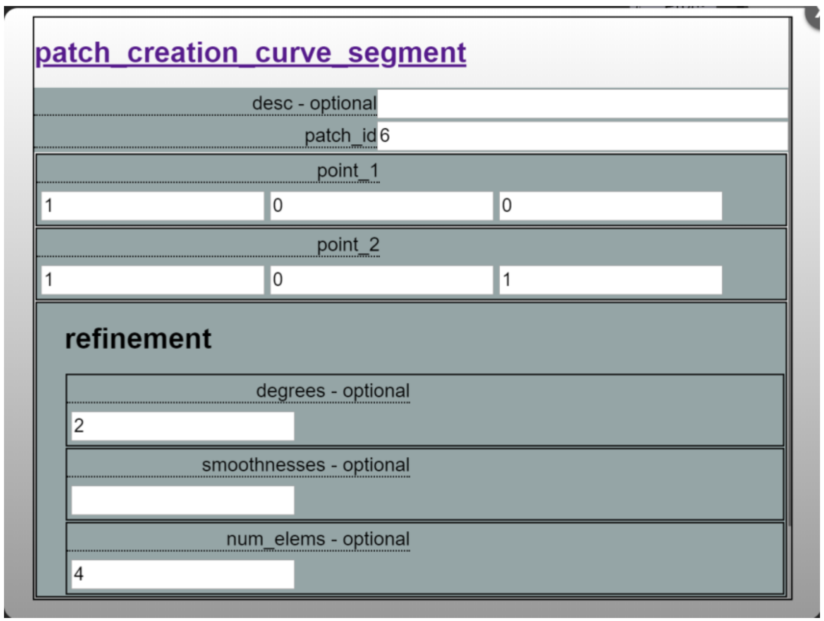

honeywell.coreform.com says
An error occurred on the backend c++ api server {"error":"IGXException \u001b[0;31;1mNo Patch,Operation Order Found at static void patch::FnPatchFromJSON::parseOperations(const util::AbstractSyntaxTree&, patch::Bases&, std::map<patch::PatchIdBase<unsigned int, (patch::PatchIdType)1>, std::unique_ptr<patch::Patch> >&) in /codes/patch/src/FnPatchFromJSON.cpp:2067\u001b[0m\n"}
OK

Documentation: `patch_creation_curve_segment`

patch_creation_curve_segment

Define a linear segment between two points.

Variable	Value Description	
patch_id	A unique ID number used to identify the patch.	
point_1	3D coordinates of the 1st point.	
point_2	3D coordinates of the 2nd point.	
refinement	**Object Member**	**Value Description**
	degrees	A vector defining the desired degree in each parametric direction.
	smoothnesses	A vector defining the desired smoothness in each parametric direction. By default, degrees - 1.
	num_elems	A vector defining the number of elements desired in each parametric direction.



patch_creation_curve_segment

desc - optional

patch_id 6

point_1

1 | 0 | 0

point_2

1 | 0 | 1

refinement

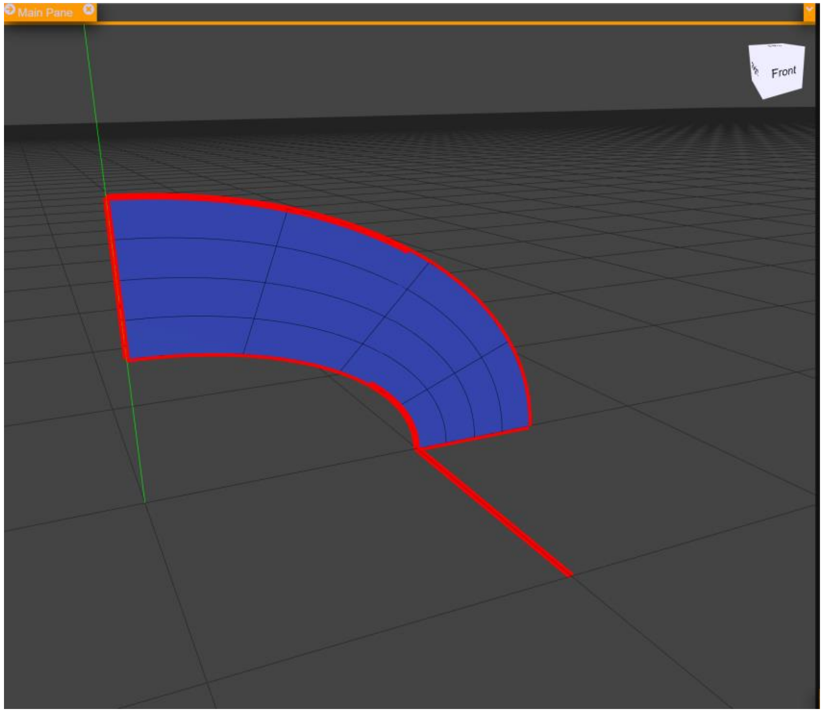
degrees - optional

2

smoothnesses - optional

num_elems - optional

4



Documentation:
patch_operation_translational_sweep

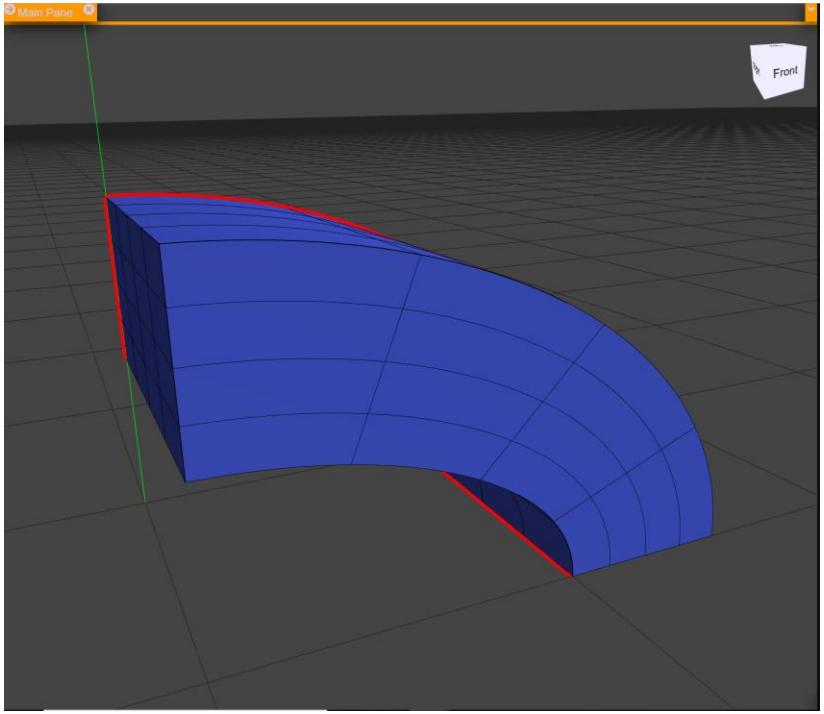
patch_operation_translational_sweep

Create a translational sweep surface/solid patch from a curve/surface patch given a sweeping curve patch. Unlike "patch_creation_from_frame_sweep", the cross section is not rotated following the path, but just translated. Note also that the two patches should start at the same location. The new sweeping parametric dimension will be placed last. Note that the two curve/surface bases must be identical.

Variable	Value Description
patch_operation_id	A unique ID number used to identify the patch operation.
patch_id	A unique ID number used to identify the new patch created in this keyword.
patch_id_origin	Patch ID of the curve/surface to be swept.
patch_id_sweeping_curve	Patch ID of the sweeping curve.

patch_operation_translational_sweep

desc - optional	
patch_operation_id	2
patch_id	7
patch_id_origin	5
patch_id_sweeping_curve	6



Documentation: `patch_operation_order`

patch_operation_order

Once a patch is created, it can be transformed (translated, scaled, mirrored...), refined, or its dimension can be extended. The order of operations matters. For instance, a rotation followed by a translation won't give the same result as a translation followed by a rotation. For this reason, all the operations in the "transformation" class must be associated to a unique operation_id. This card specifies in which order these operations should be performed.

Variable	Value Description
patch_operation_ids	A vector of operation_ids listing the order of operations.

patch_operation_order

desc - optional

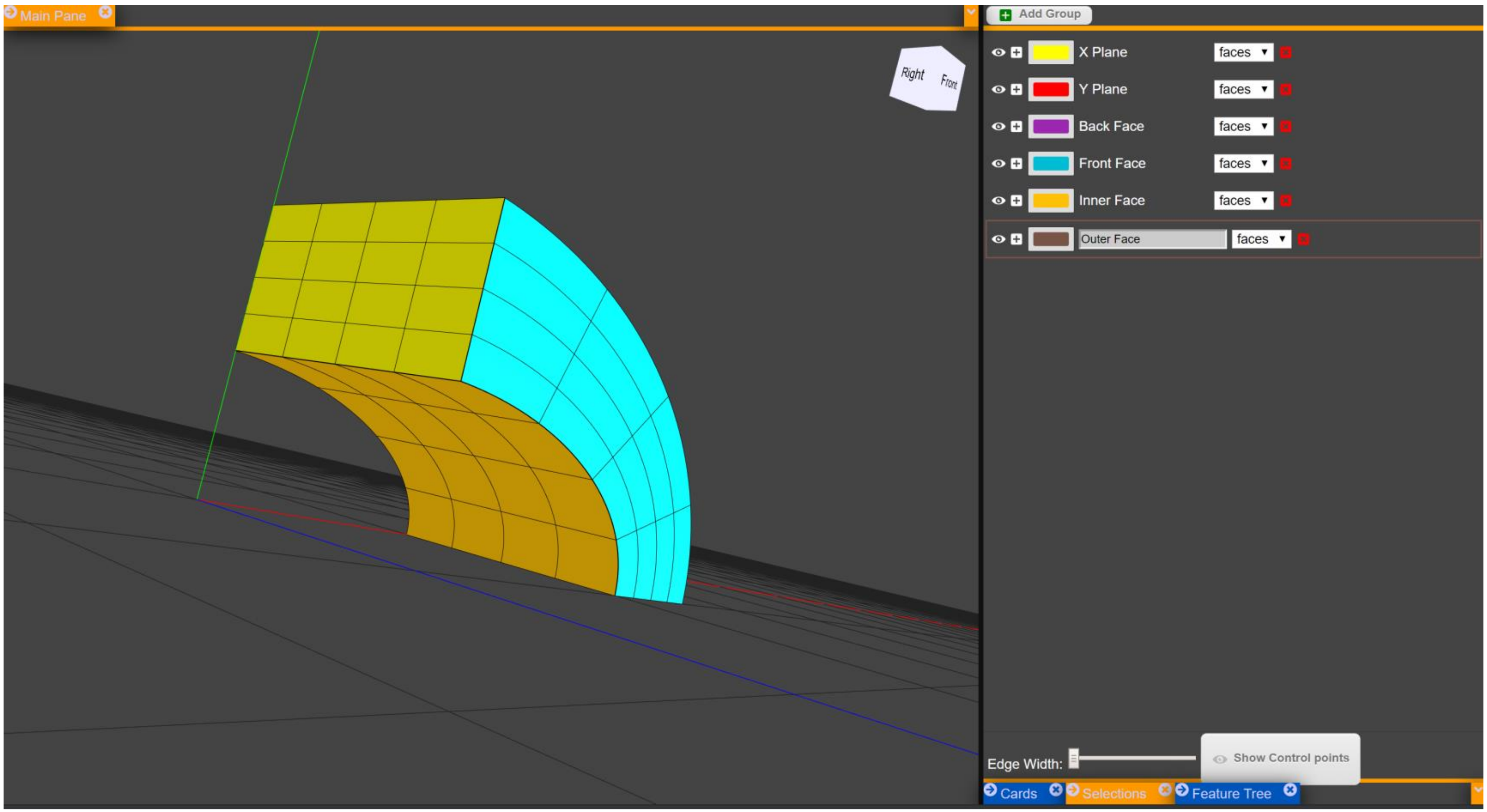
patch_operation_ids

Add item | Free Input

1

2

Create Geometry


Main Pane
Right
Front
Add Group
X Plane
faces
Y Plane
faces
Back Face
faces
Front Face
faces
Inner Face
faces
Outer Face
faces
Edge Width:
Show Control points
Cards
Selections
Feature Tree

What is a “domain”?

- A domain is a patch with additional analysis attributes attached to it.
- A domain defines how the patch will be used in the simulation and assigns unique global IDs to the nodes in the patch for use in the simulation.
- There is a one-to-one correspondence between each patch used in the simulation and a domain.
- Domains describe how geometry will be formed and includes a map from the nodes defined in to a global node id for the whole problem.

Documentation: domain_spline_solid

domain_spline_solid

Define a solid domain defined by a spline patch.

<table>
<tr><th>Variable</th><th colspan="2">Value Description</th></tr>
<tr><td>domain_id</td><td colspan="2">A unique domain ID.</td></tr>
<tr><td>patch_id</td><td colspan="2">The patch ID which defines the geometry of the domain.</td></tr>
<tr><td>use_parent_basis</td><td colspan="2">FOR INTERNAL USE ONLY. Whether or not a parent basis will be built for the domain.
default: false</td></tr>
<tr><td rowspan="5">node_map</td><td colspan="2">Alternatives</td></tr>
<tr><td colspan="2">The map from domain node IDs to global node IDs for the patch associated to this domain. This mapping serves the same purpose as an element connectivity array in FEA. Example: [0, 1, 2, 3, 4, 5, 6]</td></tr>
<tr><td>Object Member</td><td>Value Description</td></tr>
<tr><td>tol</td><td>Instead of defining the node_map by an array, the node_map can be setup automatically. For multple domains, some nodes might be shared between multiple domains. The geometric tolerence (tol) defines when two close points between two domains are considered commun. tol is only optional if the problem is composed by only one domain, or multi-domains that are contained in domain_ids_excluded.
default: 1e-8</td></tr>
<tr><td>domain_ids_excluded</td><td>Let assume that the current domain (domain_id = 1) is in contact with a second domain (domain_id = 2). Due to contact, some nodes of domain 1 and 2 might be at the same position, but since we want to investigate the evolution of the contact, the nodes should not be merged. To avoid that, we set domain_ids_excluded = [2]. When setting the card for domain_id = 2, it is also required to setup domain_ids_excluded = [1].
default:</td></tr>
</table>

domain_spline_solid

desc - optional	
domain_id	1
patch_id	7
use_parent_basis - optional	false - default ▼

node_map

Add item | Free Input

node_map

tol - optional	

domain_ids_excluded - optional - optional

Add item | Free Input

Documentation: subdomain_domains

subdomain_domains

Defines a subdomain through a set of domain boundaries. The interior of a domain can also be included in this subdomain.

Variable	Value Description
subdomain_id	A unique subdomain ID
domain_segments	The segments in the subdomain are the specific boundaries (and/or the interior) of different domains that are included in the subdomain. These segments are represented as an array of arrays with two values. The first value is the domain ID and the second value is the specific boundary ID of the domain included in the subdomain. Boundary IDs (i.e. 0, 1, 2, ...)are assigned to each boundary, when "-1" is used as the boundary ID, it refers to the interior of the domain.

subdomain_domains

desc - optional	
subdomain_id	0

Add DomainSegments

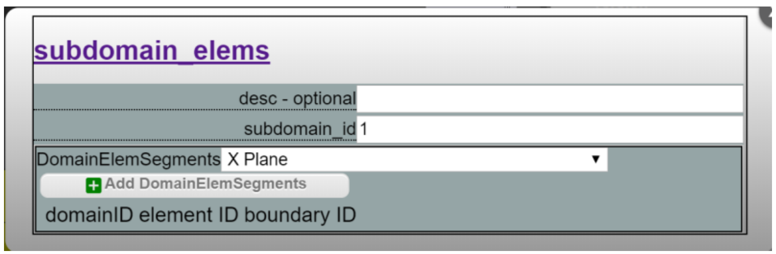
domain ID	domain boundary ID
1	-1

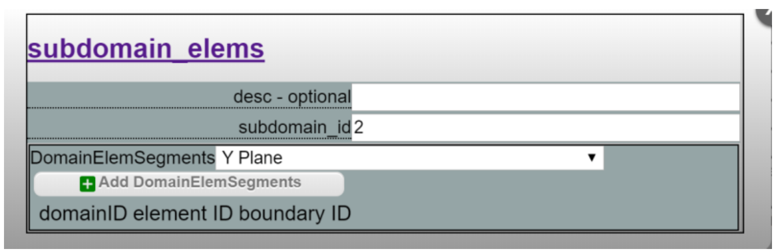
Documentation: subdomain_elems

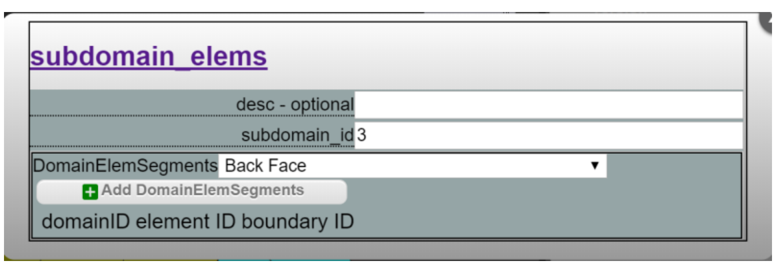
subdomain_elems

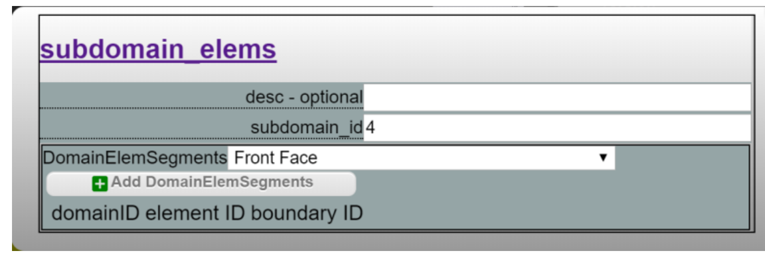
Defines a subdomain through a set of element segments. Both element boundaries and element interiors can be included in this subdomain.

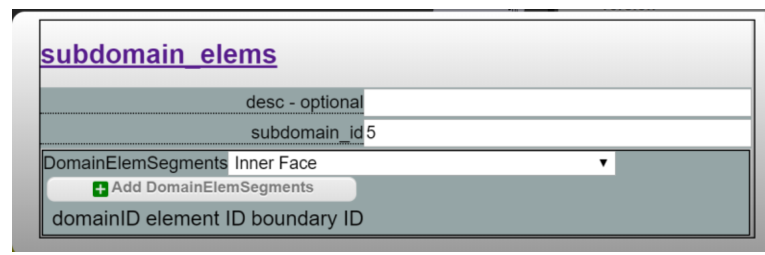
Variable	Value Description
subdomain_id	A unique subdomain ID
domain_elem_segments	The segments of the subdomain are the specific element boundaries or interiors included in the subdomain. These segments are represented by an array of arrays with three values. The first value is a domain ID, the second value is an element ID, and the third values is an element boundary ID. The element boundary ID can be one of the following values. For a cube: S0, S1, T0, T1, U0, U1. For a quad face: S0, S1, T0, T1. For a triangle: A0, B0, C0. For a tet: A0, B0, C0, D0. For the interior of any object: -1.

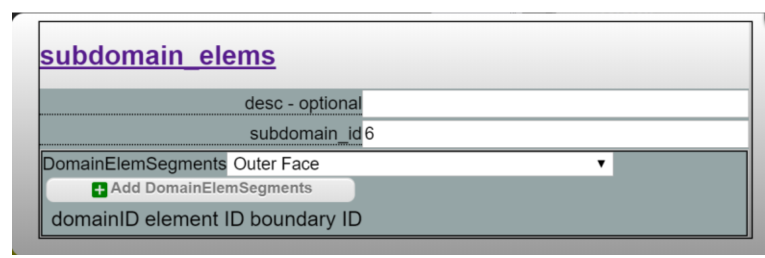

subdomain_elems
desc - optional
subdomain_id 1
DomainElemSegments X Plane
Add DomainElemSegments
domainID element ID boundary ID


subdomain_elems
desc - optional
subdomain_id 2
DomainElemSegments Y Plane
Add DomainElemSegments
domainID element ID boundary ID


subdomain_elems
desc - optional
subdomain_id 3
DomainElemSegments Back Face
Add DomainElemSegments
domainID element ID boundary ID


subdomain_elems
desc - optional
subdomain_id 4
DomainElemSegments Front Face
Add DomainElemSegments
domainID element ID boundary ID


subdomain_elems
desc - optional
subdomain_id 5
DomainElemSegments Inner Face
Add DomainElemSegments
domainID element ID boundary ID


subdomain_elems
desc - optional
subdomain_id 6
DomainElemSegments Outer Face
Add DomainElemSegments
domainID element ID boundary ID

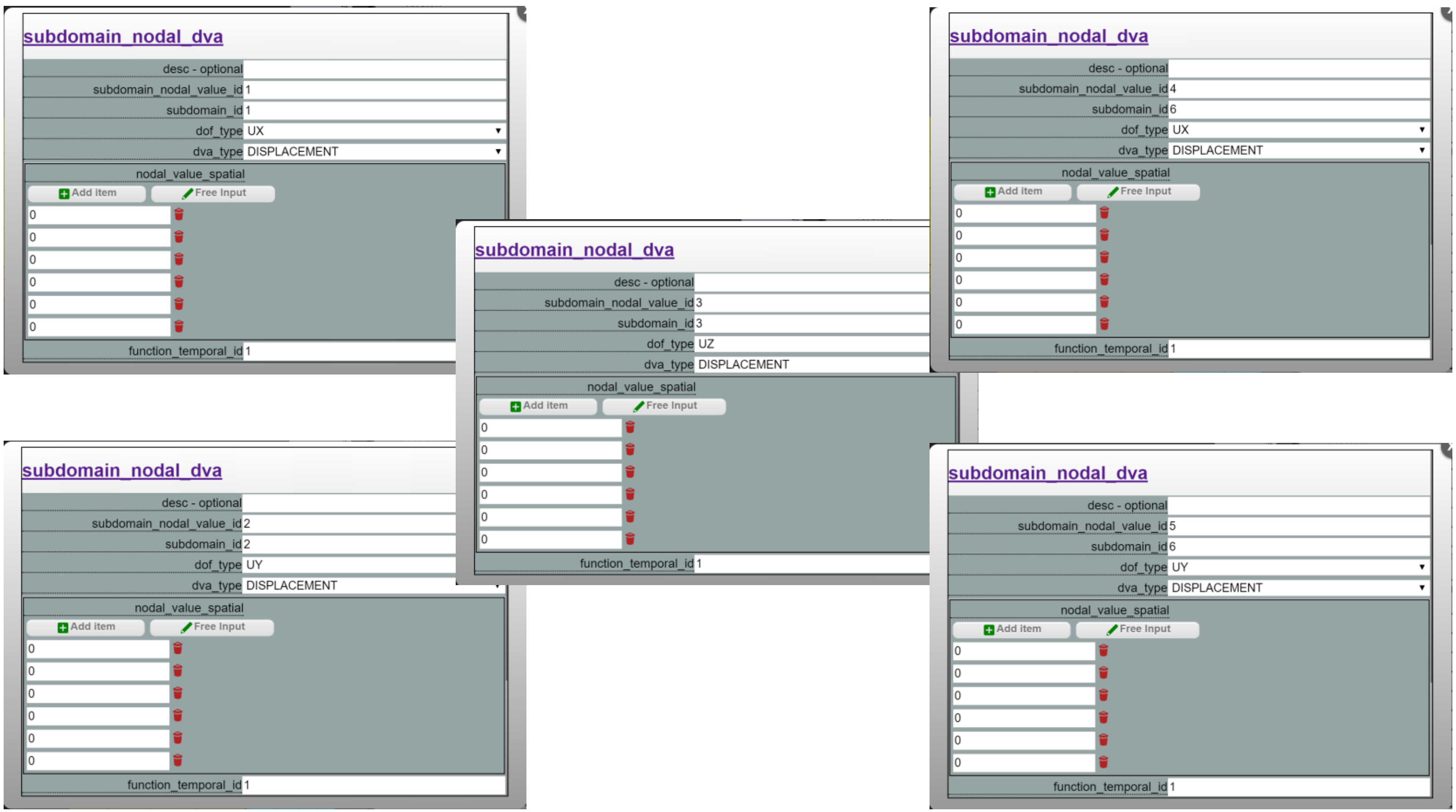
Documentation: subdomain_nodal_dva

subdomain_nodal_dva

A nodal displacement, velocity, or acceleration (i.e., dva) field defined over a subdomain.

Variable	Value Description
subdomain_nodal_value_id	A unique subdomain field value ID.
subdomain_id	The subdomain ID of the subdomain the DVA is defined over.
dof_type	The degrees-of-freedom which are involved in the subdomain calculations. Applicable degrees-of-freedom: **Enumeration** — **Value Description** **UX** — x-translational degree-of-freedom, **UY** — y-translational degree-of-freedom, **UZ** — z-translational degree-of-freedom, **RX** — x-rotational degree-of-freedom, **RY** — y-rotational degree-of-freedom, **RZ** — z-rotational degree-of-freedom, **U2** — all translational degree-of-freedom in 2d, **U3** — all translational degree-of-freedom in 3d, **R3** — all rotational degree-of-freedom in 3d, **UR3** — all translational and rotational degrees-of-freedom in 3d.
dva_type	Whether the field corresponds to displacement, velocity, or acceleration degree-of-freedom: **Enumeration** — **Value Description** **DISPLACEMENT** — displacement, **VELOCITY** — velocity, **ACCELERATION** — acceleration.
nodal_value_spatial	The spatial value of each node in the corresponding subdomain.
function_temporal_id	The temporal function ID which defines the temporal behavior of the load.

Boundary Conditions



Documentation: subdomain_field_load

subdomain_field_load

A load field defined over a subdomain.

Variable	Value Description
subdomain_field_value_id	A unique subdomain field value ID.
subdomain_id	The subdomain ID of the subdomain the load is defined over.
load_type	The load type being applied to the subdomain: **Enumeration** / **Value Description** **force:** force loading, **moment:** moment loading, **pressure:** pressure loading.
function_spatial_id	The spatial function ID which defines the spatial behavior of the load.
function_temporal_id	The temporal function ID which defines the temporal behavior of the load.

subdomain_field_load

desc - optional	
subdomain_field_value_id	1
subdomain_id	4
load_type	force ▼
function_spatial_id	1
function_temporal_id	2

What is a “function” card?

- Cards in the function class describe the spatial or temporal behavior of the elements in the simulation.

Documentation: function_temporal_constant

function_temporal_constant

Define a constant temporal function.

Variable	Value Description
function_temporal_id	A unique function temporal ID.
value	The constant value of the function.
birth	Time that the boundary condition is applied.
death	Time that the boundary condition is removed.
tol	A tolerance to use when determining if the function is alive.

function_temporal_constant

desc - optional	
function_temporal_id	1
value	1
birth	0
death	1000000
tol	1e-9

Documentation: function_temporal_linear_interpolation

function_temporal_linear_interpolation

Define a temporal function using linear interpolation.

Variable	Value Description
function_temporal_id	A unique function temporal ID.
birth	Time that the boundary condition is applied.
death	Time that the boundary condition is removed.
tol	A tolerance to use when determining if the function is alive.
graph	A \$(t, f(t))\$ function graph.

function_temporal_linear_interpolation

desc - optional	
function_temporal_id	2
birth	0
death	1000000
tol	1e-9

Add Graph

t	f(t)
0	0
1	1

Documentation: function_spatial_constant

function_spatial_constant

Define a constant spatial function.

Variable	**Value Description**
function_spatial_id	A unique function spatial ID.
domain_type	Specifies whether the function is defined in the reference or current configuration.
value	The constant value of the function. This should be a vector with three components.
magnitude	If this optional parameter is specified then the value of the function is interpreted as a direction and will be normalized.

function_spatial_constant

desc - optional	
function_spatial_id	1
domain_type	reference ▼

value		
0	0	1

magnitude - optional	-100

What are Material cards?

- Material property cards specify the physical attributes of the materials to be used in the formulation. Material properties include Young’s modulus (a measure of the stiffness of a solid material), Poisson’s ratio (the ratio of transverse strain to axial strain), and the mass density.

Documentation: material_isotropic_linear_elastic

material_isotropic_linear_elastic

Variable	Value Description
material_id	Material ID.
E	Young's modulus.
nu	Poisson's ratio.
rho	Mass density.
thermal_expansion	The thermal expansion coefficient. default: 0.0

material_isotropic_linear_elastic

desc - optional	
material_id	1
E	30e6
nu	0.29
rho	7e-4
thermal_expansion - optional	0.0

What is a Formulation?

- A formulation designates the dimension, quadrature, and material for the type of simulation to be run on each part. Each type of formulation (beam, contact, phase field fracture, shell, and solid) contains unique physical properties. Each part has only one formulation, but multiple parts can share the same formulation.

Documentation: formulation_solid

formulation_solid

Define a solid formulation.

<table>
<tr><th>Variable</th><th>Value Description</th></tr>
<tr><td>formulation_id</td><td>Formulation ID. formulation_id is referenced through the part keyword. A formulation_id must be specified.</td></tr>
<tr><td>formulation_type</td><td>The physical formulation of the part:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>solid_1d</td><td>a one-dimensional solid,</td></tr>
<tr><td>solid_2d</td><td>a two-dimensional solid,</td></tr>
<tr><td>solid_3d</td><td>a three-dimensional solid,</td></tr>
</table></td></tr>
<tr><td>quadrature</td><td>The part quadrature rule:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>Q1</td><td>1 point gauss quadrature rule,</td></tr>
<tr><td>Q2</td><td>2 point gauss quadrature rule,</td></tr>
<tr><td>Q3</td><td>3 point gauss quadrature rule,</td></tr>
<tr><td>Q4</td><td>4 point gauss quadrature rule,</td></tr>
<tr><td>Q5</td><td>5 point gauss quadrature rule,</td></tr>
<tr><td>Q6</td><td>6 point gauss quadrature rule,</td></tr>
<tr><td>Q7</td><td>7 point gauss quadrature rule,</td></tr>
<tr><td>Q8</td><td>8 point gauss quadrature rule,</td></tr>
<tr><td>Q9</td><td>9 point gauss quadrature rule,</td></tr>
<tr><td>Q10</td><td>10 point gauss quadrature rule,</td></tr>
<tr><td>QP0</td><td>p point gauss quadrature rule,</td></tr>
<tr><td>QP1</td><td>$p+1$ point gauss quadrature rule,</td></tr>
<tr><td>QNU</td><td>non-uniform reduced gauss quadrature rule</td></tr>
</table></td></tr>
<tr><td>material_id</td><td>Material ID defined in a material properties keyword.</td></tr>
</table>

formulation_solid

desc - optional	
formulation_id	1
formulation_type	solid_3d ▼
quadrature	QP1 ▼
material_id	1

What is a Part?

- A part describes all the physical and computational properties for a given set of geometries. A formulation and a subdomain are necessary to define a part.

Documentation: part

part

Variable	Value Description
part_id	The Part ID.
formulation_id	A formulation defined in a formulation keyword.
subdomain_ids	A list of subdomains defined in a subdomain keyword.
temperature_id	If some materials of the current part are temperature dependent, a temporal tempertaure description should be provided and is defined in a temperature keyword.

part

desc - optional	
part_id	1
formulation_id	1

subdomain_ids

Add item Free Input

0

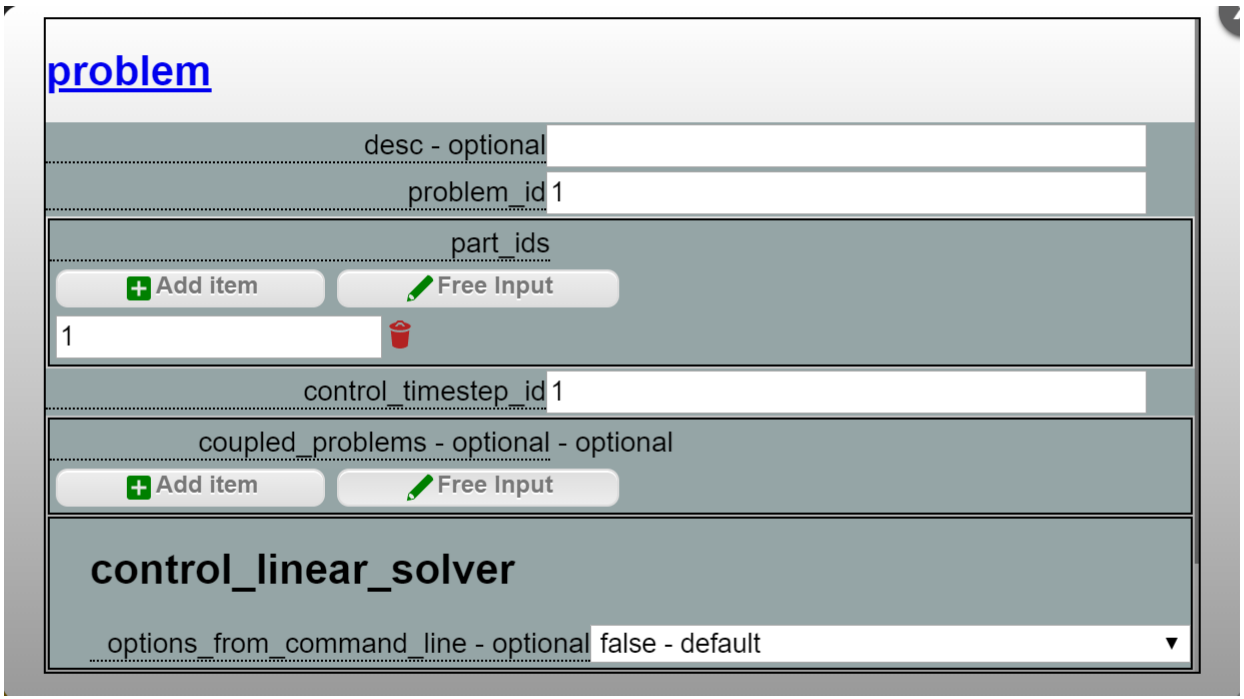
temperature_id - optional	

Documentation: problem

problem

The problem card is used to associate a part with dynamic parameters and indicate the desired output. Attributes like boundary conditions are not included in the problem card definition; they are linked by referencing the problem_id in the problem_boundary_condition.

<table>
<tr><th>Variable</th><th>Value Description</th></tr>
<tr><td>problem_id</td><td>A unique problem ID.</td></tr>
<tr><td>part_ids</td><td>An array of part ids that make up the domain of the problem.</td></tr>
<tr><td>control_timestep_id</td><td>The ID of the timestep control for this problem.</td></tr>
<tr><td>coupled_problems</td><td>The problem IDs which couple to this problem.</td></tr>
<tr><td>control_linear_solver</td><td><table>
<tr><th>Object Member</th><th>Value Description</th></tr>
<tr><td>options_from_command_line</td><td>If this is option is set to true then the linear solver options are set from command line arguments. See the Petsc manual for details.
default: false</td></tr>
</table></td></tr>
</table>


problem
desc - optional
problem_id 1
part_ids
Add item
Free Input
1
control_timestep_id 1
coupled_problems - optional - optional
Add item
Free Input
control_linear_solver
options_from_command_line - optional
false - default

Documentation: problem_boundary_condition

problem_boundary_condition

Variable	Value Description
problem_id	The problem to which these boundary conditions are assigned.
subdomain_nodal_value_ids	The subdomain nodal value IDs which defines the boundary condition.

problem_boundary_condition

desc - optional	
problem_id	1

subdomain_nodal_value_ids

Add item | Free Input

1

2

3

4

5

Documentation: problem_field_load

problem_field_load

Variable	Value Description
problem_id	The problem to which the load is assigned.
subdomain_field_value_ids	The subdomain field value IDs which defines the load.

problem_field_load

desc - optional	
problem_id	1

subdomain_field_value_ids

Add item | Free Input

1

Documentation: control_timestep_quasistatic

control_timestep_quasistatic

This card contains information to control time stepping for linear quasistatic problems.

Variable	**Value Description**
control_timestep_id	A unique control timestep ID.

control_timestep_quasistatic

desc - optional	
control_timestep_id	1

Documentation: control_model

control_model

This card contains a number of somewhat unrelated variables that control how dynamic simulations progress, as well as visualization options that apply to both static and dynamic simulations.

Variable	Object Member	Object Member	Value Description
control_time	initial_time_step		The initial time step. If **initial_time_step = 0.0** then the initial time step is determined automatically (only for explicit transient solutions). default: 1.0
	termination_time		The termination time for the simulation. default: 1.0
	adaptive_timestep	iteration_optimal	The targeted number of nonlinear iterations desired within a timestep.
		iteration_window	The interval around the optimal iteration number within which the time step will not be modified.
		growth_factor	If the number of iterations for convergence is less than the minimum desired increase the time step according to this factor.
		reduction_factor	If the number of iterations for convergence is greater than the maximum desired decrease the time step according to this factor.
		delta_t_min	Minimum time step.
		delta_t_max	Maximum time step.
control_problem			The problem_id of the problem that controls the timestep loop (e.g. time step size) and output.
enable_parent_basis			FOR INTERNAL USE ONLY. If true then a parent basis will be computed for each spline domain. This should only be used if the mesh is very structured. For example, a mesh composed of uniform B-splines. default: false
enable_output			If true then ouput will be written. default: true
enable_output_restart			If true then restart ouput will be written. default: false
output_restart_file_name_prefix			The restart output file name prefix. default: result
output_restart_delta_t			The restart output is written every delta_t time (time zero is always written). If this is set to 0 then restart output is written at every step. default: 0.0

control_model

desc - optional	

control_time

initial_time_step - optional	1.0
termination_time - optional	1.0

adaptive_timestep

iteration_optimal	
iteration_window	
growth_factor	
reduction_factor	
delta_t_min	
delta_t_max	

control_problem	1
enable_parent_basis - optional	false - default ▼
enable_output - optional	true - default ▼
enable_output_restart - optional	false - default ▼
output_restart_file_name_prefix - optional	result
output_restart_delta_t - optional	0.0

Documentation: subdomain_output_field

subdomain_output_field

<table>
<tr><th>Variable</th><th>Value Description</th></tr>
<tr><td>subdomain_output_id</td><td>A unique subdomain output ID</td></tr>
<tr><td>subdomain_ids</td><td>A unique subdomain ID that defines the domain over which quantities will be output</td></tr>
<tr><td>function_temporal_id</td><td>The temporal function ID which defines the temporal behavior of the output.</td></tr>
<tr><td>field_types</td><td>The field types which will be output:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>displacement</td><td>The components of displacement u_x, u_y, u_z.</td></tr>
<tr><td>velocity</td><td>The components of velocity v_x, v_y, v_z.</td></tr>
<tr><td>acceleration</td><td>The components of acceleration a_x, a_y, a_z.</td></tr>
<tr><td>strain</td><td>The components of strain.</td></tr>
<tr><td>stress</td><td>The components of stress.</td></tr>
<tr><td>vm_stress</td><td>The von Mises stress.</td></tr>
<tr><td>eps</td><td>The equivalent plastic strain.</td></tr>
<tr><td>effective_plastic_work</td><td>The accumulated plastic work measure that contributes to crack growth for phase-field simulations.</td></tr>
<tr><td>effective_driving_energy</td><td>The total effective energy density that drives the crack growth. Can have contributions from elastic strain energy and effective plastic work.</td></tr>
<tr><td>weight</td><td>The rational weighting of the geometry.</td></tr>
<tr><td>phase_field</td><td>Scalar phase-field value.</td></tr>
<tr><td>phase_field_rate</td><td>The rate of change of the phase-field value. Valid only for dynamic phase-field problems.</td></tr>
</table></td></tr>
<tr><td>One Of</td><td>
<table>
<tr><th>Variable</th><th>Value Description</th></tr>
<tr><td>delta_time</td><td>The output is written every delta_time time between birth and death (the birth time step is always written). If this is set to 0 then output is written at every step between birth and death.</td></tr>
<tr><td>delta_step</td><td>The output is written every delta_step time steps between birth and death (the birth time step is always written). If this is set to 0 then output is written at every step between birth and death.</td></tr>
</table></td></tr>
<tr><td>file_name_prefix</td><td>The output file name prefix.
default: results</td></tr>
<tr><td>file_type</td><td>The type of file format that will be written:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>vtk</td><td>VTK file format,</td></tr>
<tr><td>hdf5</td><td>HDF5 file format.</td></tr>
</table>
default: vtk</td></tr>
<tr><td>sample_type</td><td>The type of sampling which will be performed on each element:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>UNIFORM1</td><td>each element is uniformly subdivided into a 1 by 1 submesh for visualization,</td></tr>
<tr><td>UNIFORM2</td><td>each element is uniformly subdivided into a 2 by 2 submesh for visualization,</td></tr>
<tr><td>UNIFORM3</td><td>each element is uniformly subdivided into a 3 by 3 submesh for visualization,</td></tr>
<tr><td>UNIFORM4</td><td>each element is uniformly subdivided into a 4 by 4 submesh for visualization,</td></tr>
<tr><td>UNIFORM5</td><td>each element is uniformly subdivided into a 5 by 5 submesh for visualization,</td></tr>
<tr><td>UNIFORM6</td><td>each element is uniformly subdivided into a 6 by 6 submesh for visualization,</td></tr>
<tr><td>UNIFORM7</td><td>each element is uniformly subdivided into a 7 by 7 submesh for visualization,</td></tr>
<tr><td>UNIFORM8</td><td>each element is uniformly subdivided into a 8 by 8 submesh for visualization,</td></tr>
<tr><td>UNIFORM9</td><td>each element is uniformly subdivided into a 9 by 9 submesh for visualization,</td></tr>
<tr><td>UNIFORM10</td><td>each element is uniformly subdivided into a 10 by 10 submesh for visualization,</td></tr>
<tr><td>UNIFORM15</td><td>each element is uniformly subdivided into a 15 by 15 submesh for visualization,</td></tr>
<tr><td>UNIFORM20</td><td>each element is uniformly subdivided into a 20 by 20 submesh for visualization,</td></tr>
<tr><td>UNIFORM25</td><td>each element is uniformly subdivided into a 25 by 25 submesh for visualization,</td></tr>
<tr><td>UNIFORM30</td><td>each element is uniformly subdivided into a 30 by 30 submesh for visualization.</td></tr>
</table>
default: UNIFORM1</td></tr>
<tr><td>cache_basis_evals</td><td>Whether or not basis evaluations required to generate output will be cached and reused.
default: true</td></tr>
<tr><td>include_elem_outlines</td><td>Whether or not element outlines will be included in the output.
default: true</td></tr>
<tr><td>solution_type</td><td>The solution configuration which will be used for output:
<table>
<tr><th>Enumeration</th><th>Value Description</th></tr>
<tr><td>current</td><td>The current converged solution,</td></tr>
<tr><td>alpha</td><td>an alpha level solution,</td></tr>
<tr><td>next</td><td>the next predicted solution.</td></tr>
</table>
default: current</td></tr>
</table>

subdomain_output_field

desc - optional
subdomain_output_id 1

subdomain_ids
Add item
Free Input
0

function_temporal_id 1

Add FieldTypes
- optional displacement
- optional stress
- optional vm_stress

delta_time 0
delta_step

file_name_prefix - optional results
file_type - optional vtk - default
sample_type - optional UNIFORM4
cache_basis_evals - optional true - default
include_elem_outlines - optional true - default
solution_type - optional current - default

Initializing the Solve

```
/////////////////////////////////////////////////////
=====> STARTING SOLVE
=====================================================
=====> Initializing solver...
=====> Setting initial conditions...
=====> Writting initial output...
=====> Writting output...
Writing to file: results_ts000000.vtu
=====> Initializing time steps...
=====> Start time: 0
=====> End time: 1
=====> Initial time step: 1
```

Solving a timestep

```
//////////////////////////////////////////////////////////////////////
=====> Starting time step 1
==========> Current time: 0
==========> Step size: 1
======================================================================
=====> Starting problem 1
==========> Time step: 1
==========> Current time: 0
==========> Step size: 1
=====> Adding kinematic boundary conditions...
=====> Computing external force...
=====> Computing internal force...
=====> Assembling the stiffness matrix...
=====> Solving the linear system...
=====> Writting output...
Writing to file: results_ts000001.vtu
//////////////////////////////////////////////////////////////////////
=====> TIME STEP COMPLETE
```

Time Report

```
=====> ACCUMULATED EXECUTION TIME REPORT
Total elapsed time (secs):    2.09325
Output (secs):                1.94749                          (93.0366%)
Restart (secs):               0                                (0%)

Report for problem 1
Total problem time (secs):    0.12587                          (6.01286%)
Time integrator (secs):       0.125811                         (6.01004%)
Corrector iteration (secs):   0                                (0%)
External F assembly (secs):   0.00522304                       (0.249506 %)
Internal F assembly (secs):   0.00451088                       (0.215486 %)
Stiffness assembly (secs):    0.0462561                        (2.20967 %)
Total assembly (secs):        0.05599                          (2.67466 %)
Linear solve (secs):          0.06021                          (2.87625 %)
```