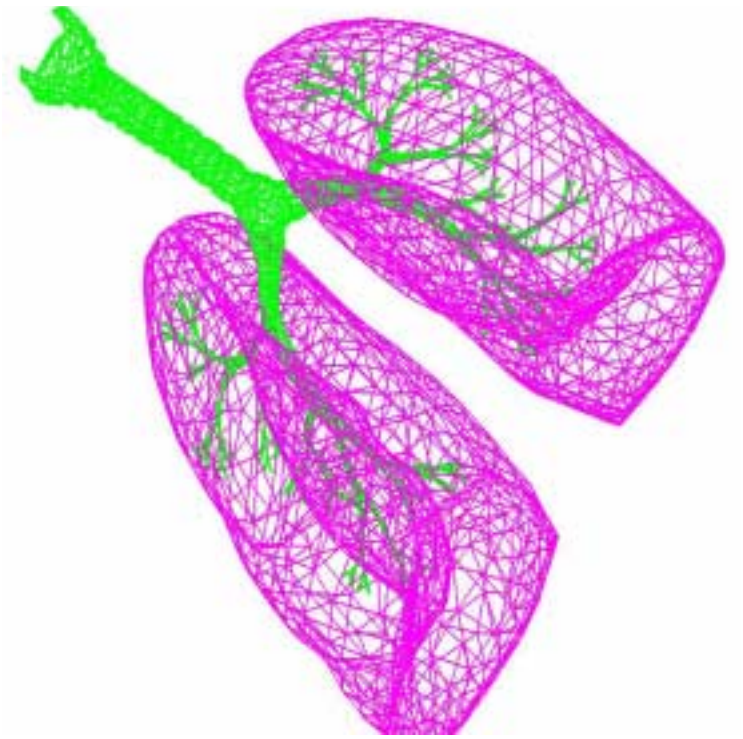
NWGrid
Data Structures Reference

The *N*orth*W*est *Grid* Generation Code

Web Site: http://www.emsl.pnl.gov/nwgrid



Dr. Harold E. Trease
Lynn L. Trease
Applied Mathematics Group
Theory, Modeling, and Simulation Directorate
***William R. Wiley*, Environmental Molecular Sciences Laboratory**
Pacific Northwest National Laboratory
Operated by Battelle for the US Department of Energy

TABLE OF CONTENTS

Mesh Object Definitions 3
Mesh Object Scalar Properties 3
Mesh Object Node Properties 4
Mesh Object Node Coordinates 6
Mesh Object Element Properties 7
Mesh Object - Miscellaneous Properties 8
Fortran Interface 11
Mesh Object Connectivity 12
Geometries 15
Documentation 15
Contact 15

Mesh Object Definitions

The data structure which contains the information necessary to define a mesh is called a Mesh Object. A Mesh Object consists of attributes. There is a default template for a Mesh Object composed of the following attributes:

Mesh Object Scalar Properties

Attribute	Type	Description/Default
name	char*32	Mesh object name.
nnodes	integer	Number of nodes in mesh.
nelements	integer	Number of elements in the mesh, e.g. triangles, tetrahedral
nfaces	integer	Number of unique topological facets in the mesh, e.g. number of edges in 2D or number of element faces in 3D
nedges	integer	Number of unique edges in mesh.
mbndry	integer	Value signifying that if the node number is greater than **mdndry** then the node is a boundary node. (default is 16,000,000) Must be greater than 48***nnodes** and may be reset by **connect.**
ndimensions_topo	integer	Topological dimensionality, 1, 2, or 3, i.e., a non-planar surface would have **ndimensions_topo** = 2 and **ndimensions_geom**. = 3.
ndimensions_geom	integer	1, 2 or 3 for dimension of geometry (default is 3)
nodes_per_element	integer	The value is dependent on the type of mesh; e.g. for tetrahedral mesh the value will be 4.
edges_per_element	integer	The value is dependent on the type of mesh; e.g. for tetrahedral mesh the value will be 6.
faces_per_element	integer	Topological number of facets per element, i.e. in 1D this number is always 2, for 2D use the number of edges of the element, for 3D use the number of faces of the element. The value is dependent on type of mesh; e.g. for tetrahedral mesh the value will be 4.

Mesh Object Node Properties

Attribute	Type	Description/Default
isetwd	Vector of integers Vector length nnodes	Array containing **pset** membership.
ialias	Vector of integers Vector length nnodes	Array of alternate node numbers, i.e. for merged points.
imt1	Vector of integers Vector length nnodes	Array of node material.

itp1	Vector of integers Vector length nnodes	Array of node type. If type > 20 node will be invisible.

Node Type	Name	Description
0	int	Interior
2	ini	Interface
3	vrt	Virtual
4	vin	Virtual + interface
8	vif	Virtual + interface + free
9	alb	Virtual + interface + free + reflective
10	rfl	Reflective boundary node
11	fre	Free boundary node
12	irb	Interface node on reflective boundary
13	ifb	Interface node on free boundary
14	rfb	Node on intersection of free boundary and reflective boundary
15	irf	Interface node on intersection of free boundary and reflective boundary
16	vrb	Virtual node on reflective boundary
17	vfb	Virtual node on free boundary
18	vrf	Virtual node on free + reflective boundary
19	vir	Virtual + interface node on reflective boundary
20	mrg	Merged node
21	dud	Dudded node
41	par	Parent node

icr1	Vector of integers Vector length nnodes	Array of constraint numbers for nodes; the value of this array is an index into the **icontab** table of node constraints.
isn1	Vector of integers Vector length nnodes	Array of child, parent node correspondence. Points on material interfaces are given point type 41 (parent). One child point is spawned for each material meeting at the parent point. The **isn1** field of the parent point will contain the point number of the first child point. The **isn1** field of the first child will contain the point number of the next child. The **isn1** field of the last child will contain the point number of the parent. The point types of the child points will be 2, 12, 13, 15 or 19 depending on whether the interface point is also on an exterior boundary. This parent, child relationship is established by the **settets** command.
ign1	Vector of integer Vector length nnodes	Array of generation numbers for nodes.

Mesh Object Node Coordinates

Attribute	Type	Description/Default
xic	Vector of reals Vector length nnodes	Array of x node coordinates.
yic	Vector of reals Vector length nnodes	Array of y node coordinates.
zic	Vector of reals Vector length nnodes	Array of z node coordinates

Mesh Object Element Properties

<table>
<tr><th>Attribute</th><th>Type</th><th>Description/Default</th></tr>
<tr><td>itetclr</td><td>Vector of integers
Vector length
nelements</td><td>Array of element material.</td></tr>
<tr><td>itettyp</td><td>Vector of integers
Vector length
nelements</td><td>Array of element shape.
<table>
<tr><th>Name</th><th>Value</th><th>Description</th></tr>
<tr><td>ifelmpnt</td><td>1</td><td>point</td></tr>
<tr><td>ifelmlin</td><td>2</td><td>line</td></tr>
<tr><td>ifelmtri</td><td>3</td><td>triangle</td></tr>
<tr><td>ifelmqud</td><td>4</td><td>quadrilateral</td></tr>
<tr><td>ifelmtet</td><td>5</td><td>tetrahedron</td></tr>
<tr><td>ifelmpyr</td><td>6</td><td>pyramid</td></tr>
<tr><td>ifelmpri</td><td>7</td><td>prism</td></tr>
<tr><td>ifelmhex</td><td>8</td><td>sbinit_nosh.f (hexahe-dron)</td></tr>
<tr><td>ifelmhyb</td><td>9</td><td>hybrid</td></tr>
<tr><td>ifelmply</td><td>10</td><td>polygon</td></tr>
</table></td></tr>
<tr><td>itetoff</td><td>Vector of integers
Vector length
nelements</td><td>Index into itet array for an element.</td></tr>
<tr><td>jtetoff</td><td>Vector of integers
Vector length
nelements</td><td>Index into jtet array for an element.</td></tr>
<tr><td>itet</td><td>Vector of integers
Vector length
nelements</td><td>Array of node vertices for each element. Contains the node numbers of element j.</td></tr>
</table>

jtet	Vector of integers Vector length nelements	Array of element connectivity. Contains the element neighbors of element j element number and local face number.
xtetwd	Vectors of reals Vector length nelements	Array containing eset membership information (64 sets).

Mesh Object – Miscellaneous Properties

Attribute	Type	Description/Default
ipolydat	char*32	Flag to **dump/gmv** to output polygon data. (default is yes)
vor2d	char*32	Flag to **dump/gmv** to output voronoi cells and median mesh cells for 2D meshes. (default is yes)
vor3d	char*32	Flag to **dump/gmv** to output voronoi cells and median mesh cells for 3D meshes. (default is no)
epsilon	real	Value of machine epsilon which will be calculated by the code.
epsilon1	real	Value of smallest edge that the code can distinguish will be set internally by the code (see **setsize**).
epsilona	real	Value of smallest area that the code can distinguish will be set internally by the code.
epsilonv	real	Value of smallest volume that the code can distinguish will be set internally by the code.
ipointi	integer	Node number of the first node of the last set of nodes generated, used by the 0,0,0 **pset** syntax.
ipointj	integer	Node number of the last node of the last set of nodes generated, used by the 0,0,0 **pset** syntax.
idebug	integer	Debug flag values greater than 0 produce increasing levels of output.
itypconv_sm	integer	
maxiter_sm	integer	Number of smoothing iterations in the **smooth** and **radapt** routines. (default is 25)

tolconv_sm	real	
nnfreq	integer	Flag to control reconnection after refine - set to zero to turn off reconnection. (default is 1)
ivoronoi	integer	Flag to control reconnection criterion - 1 means restore delaunay, -2 means improve geometric quality of the elements, 3 means adaptive reconnection with user supplied routine. (see recon) (default is 1)
iopt2to2	integer	Flag to contol boundary flips during reconnection (0=reflective boundaries, 1=interfaces, 2=reflective boundaries and interfaces, 3=all) (default is 2)
dumptype	char*32	Type of gmv file to write. (default is binary)
velname	char*32	Name of velocity attribute. (default is vels)
densname	char*32	Name of density attribute. (default is ric)
presname	char*32	Name of pressure attribute. (default is pic)
ener name	char*32	Name of energy attribute. (default is eic)
xmin	real	Minimum x coordinate of nodes in mesh set by **setsize**. (default is 1)
xmax	real	Maximum x coordinate of nodes in mesh set by **setsize**. (default is 1)
ymin	real	Minimum ycoordinate of nodes in mesh set by s**et-size**. (default is 1)
ymax	real	Maximum y coordinate of nodes in mesh set by **setsize**. (default is 1)
zmin	real	Minimum z coordinate of nodes in mesh set by **setsize**. (default is 1)
zmax	real	maximum z coordinate of nodes in mesh set by **setsize**. (default is 1)
kdtree_level	integer	Resolution level of kdtree -- 0 means terminal nodes contain 1 member. (default is 0)

max_number_of_sets	integer	Number of **pset** and **eset** allowed - currently restricted to 32. (default is 32)
number_of_psets	integer	Number of defined psets in the mesh.
number_of_esets	integer	Number of defined eltsets in the mesh.
geom._name	char*32	Name of geometry associated with this mesh. (see **geom**) (default is defaultgeom)

The current state of a mesh object can be displayed by the **cmo/status** command Note: Many commands and the cmo_get_info subroutine accept **itp** as equivalent to **itp1; icr** to **icr1, isn** to **isn1**; **imt** to **imt**1. The user should never add an attribute whose name is **itp,imt,icr,isn.**

The default Mesh Object can be expanded by adding user defined attributes (see **cmo/addatt**).

The value of parameters can be changed by the cmo/setatt command.
(e.g. **cmo/setatt/**/epsilonl/1.d-9)

NWGrid will add attributes to the mesh object in certain instances. For example, if there are any constrained surfaces, reflect, virtual or intrcons types, the following attributes are added to the mesh object:

NCONBND	number of combinations of constrained surfaces
ICONTAB(50,NCONBND)	
ICONTAB(1,i)	number of surfaces contributing to the ith constraint
ICONTAB(2,i)	degree of freedom of the ith constraint
ICONTAB(2+j,i)	Surface number of the jth surface contributing to the ith constraint

In order to determine which constraint entry applies to node ip, retrieve the value i=icr1 (ip), i.e. ICONTAB(1, icr1(ip)) gives the number of surfaces that ip is `on'. If icr1(ip) is zero there is no constraint on that node. The number of the surfaces that ip is 'on' are stored in ICONTAB(3,icrl(ip))...ICONTAB(2 + ICONTAB(1,icr1(ip))).

TENSOR	Dimension of XCONTAB
XCONTAB (TENSOR, NPOINTS)	This is a 3x3 matrix which multiplied by the velocity vector, constrains the velocity to the degrees of freedom possessed by the node. May be constructed by calls to constrainv.

FORTRAN Interface

Mesh Object attribute data are accessed through a set of subroutines. An example of accessing an existing Mesh Object and creating a new mesh object is given in the Code Development Reference, under Utility Subroutines; that example should be used as a template when operating with Mesh Objects. The subroutine set includes:

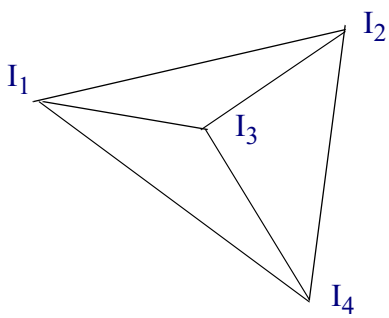
cmo_get_name	Retrieve active mesh object name.
cmo_set_name	Set active mesh object name.
cmo_get_info	Retrieve mesh object data.
cmo_set_info	Set mesh object data.
cmo_newlen	Adjust mesh object memory allocation.

Only data from the active Mesh Object may be retrieved. Calling cmo_set_name will make the referenced Mesh Object active. Scalar quantities are retrieved and stored using cmo_get_intinfo, cmo_get_attinfo, and cmo_set_info. Vector quantities are referred to by their pointers. The length of the vectors is calculated internal to *NWGrid* based on the values of the scalar mesh object attributes. Memory allocation for a new mesh object or for a mesh object which will grow in size is accommodated by first setting the appropriate scalars for the Mesh Object by using cmo_set_intinfo with the new number of elements and/or nodes and then calling cmo_newlen. These two steps must be taken before adding to the size of a Mesh Object.

Mesh object parameters are retrieved with the cmo_get_attparam subroutine.

Mesh Object Connectivity

The Mesh Object attributes**, itettyp, itetoff, jtetoff, itet,** and **jtet** along with the variables contained in the include file local_element.h completely describe the mesh connectivity. The following discussion is based on the concept of local facets and local edges for an element. The nodes comprising a given element are always specified in a well-defined order; hence when one references the 'second facet' of an element, one references a pre-defined set of points. Consider a tetrahedral element, with nodes labeled as in the diagram:



The points are oriented so that the triple product I1I2 x I2I3 x I2I4 is positive, and the volume of the tet is one-sixth of the triple product. The local facets are defined as follows:

F_1	=	I_2	I_3	I_4
F_2	=	I_1	I_4	I_3
F_3	=	I_1	I_2	I_4
F_4	=	I_1	I_3	I_2

The local edges for a tetrahedral are defined as follows:

E_1	=	I_1	I_2
E_2	=	I_1	I_3
E_3	=	I_1	I_4
E_4	=	I_2	I_3
E_5	=	I_2	I_4
E_6	=	I_3	I_4

Similarly, local facets and local edges are defined for all element types.

(see supported element types)

itettyp(it) gives the element type of element **it.**
itetoff(it) gives the offset to the first node in element **it.**
itet(itetoff(it)+j) gives the jth node of element **it.**
nelmnen(itettyp(it)) gives the number of nodes of element **it.**

To loop through all the nodes of all elements in the mesh:

```
do it=1,ntets
  do j=1,nelmnen(itettyp(it))
     k=itet(itetoff(it)+j)
  enddo
enddo
```

nelmnef(itettyp(it)) gives the number of facets of element **it**.
ielmface0(iface,itettyp(it)) gives the number of nodes on facet **iface** of element **it**. **ielmface1(local_node,iface,itettyp(it))** gives the increment to the node number **(local_node)** on facet **iface** of element **it.**

To loop through all the nodes, **k,** of all elements in the mesh by facets**:**

```
do it=1,ntets
  do i=1,nelmnef(itettyp(it))
    do j=1,ielmface0(i,itettyp(it))
      k=itet(itetoff(it)+
      ielmface1(j,i,itettyp(it)))
    enddo
  enddo
enddo
```

nelmnee(itettyp(it)) gives the number of edges of element **it.**
ielmface2(inode,iface,itettyp(it)) gives the edge number associated with **inode** on facet **iface** of element **it.**
ielmedge1(1|2,iedge,itettyp(it)) gives the node offset associated with edge **iedge** of element **it.**

To loop through all pairs of edge nodes **(i1,i2)** of all elements in the mesh :

```
do it=1,ntets
 do i=1,nelmnee(itettyp(it))
     i1=itet(itetoff(it)+
     ielmedge1(1,i,itettyp(it)))
     i2=itet(itetoff(it)+
     ielmedge1(2,i,itettyp(it))
  enddo
 enddo
```

To loop through all pairs of edge nodes (**i1,i2**) of all elements in the mesh by facets**:**

```
 do it=1,ntets
   do i=1,nelmnef(itettyp(it))
     do j=1,ielmface0(i,itettyp(it))
       ie=ielmface2(j,i,itettype(it))
       i1=itet(itetoff(it)+
       ielmedge1(1,ie,itettyp(it)))
       i2=itet(itetoff(it)+
       ielmedge1(2,ie,itettyp(it)))
     enddo
   enddo
 enddo
```

jtet(itetoff(it)+j) gives the element number and local facet number of the neighbor to element **it**, facet **j**.

To loop to find all neighbors of elements **(jt** is neighbor element number**, jf** is facet of neighboring element) if **jt** is a neighbor of element **it** and local face **if** of **it** is the same as local face **jf** of **jt** then **jtet (jtetoff(it)** + **if) = nef_cmo * (jt-**1)+**jf,** where **nef_cmo i**s the number of faces per element. Similarily, **jtet (jtetoff(jt)** + **jf)= nef_cmo * (it -1)** + **if.) (mbndry** is the value added to **jtet** if element **it** is on a boundary or interface; the **jtet** value of an element **it** with facet **j** on an exterior boundary will be exactly **mbndry;** the **jtet** value of an element it with facet **j** on an interior interface will be **mbndry** + the **jtet** value calculated from the neighboring element number and neighbor element local_facet number):

```
c get number of faces per element for this mesh object
call cmo_get_info(`faces_per_element',cmo_name,
```

```
nef_cmo,ilen,ity,ics)
do it=1,ntets
 do i=1,nelmnef(itettyp(it))
c check if element face is on an external boundry
   if(jtet(jtetoff(it)+i).eq.mbndry) then
     jt=0
     jf=0
c check if element face is on an internal boundry
     elseif(jtet(jtetoff(it)+i).gt.mbndry)then
    jt=1+(jtet(jtetoff(it)+i)-mbndry-1)/nef_cmo
    jf=jtet(jtetoff(it)+i)-mbndry-nef_cmo*(jt-1)
C Volume element
       else
    jt=1+(jtet(jtetoff(it)+i)-1)/nef_cmo
    jf=jtet(jtetoff(it)+i)-nef_cmo*(jt-1)
   endif
 enddo
enddo
```

Geometries

A geometry consists of a set of surfaces and regions that are defined in terms of existing surfaces. A name is associated with the geometry data and the default name is - default geom-. A geometry may be named using the **geometry/create** command which must precede the **surface**, **region**, and **mregion** commands that define this geometry. A geometry may be associated with more than one mesh object. **cmo/geometry** and **cmo/constraint** command are used to associate geometries with mesh objects.

The data for a geometry is contained in a set of memory managed arrays whose partition name is the geometry name and in common blocks geom and cgeom defined in the geom_lg_h file. This file also contains the pointer statements that can be used to access the data arrays directly.

At present access routines to the geometry information have not been written except for getregv which accepts as input the coordinated of a set of query points and which returns a list of region numbers into which each query point falls.

Documentation

The following ***NWGrid*** documents can be found on the internal, ***NWGrid/NWPhys*** web site. **http://www.emsl.gov/nwgrid**

NWGrid Installation Guide
NWGrid Users Manual
NWGrid Tutorial
NWGrid Examples
NWGrid Command Reference
NWGrid Programmers Reference

Contact

Contact Lynn Trease, llt@pnl.gov, for further assistance and questions.