Rocmop User's Guide

Center for Simulation of Advanced Rockets
University of Illinois at Urbana-Champaign
2270 Digital Computer Laboratory
Urbana, IL

Title:	Rocmop User's Guide
Author:	Phillip Alexander and Pornput Suriyamongkol (revision 3)
Subject:	This document describes the use of the Rocmop module
Revision:	3.0
Revision History	Revision 0: Initial Release: March 2005 Revision 1: Reformatted: December 2005 Revision 2: Documentation Week Update: December 2006 Changed to Microsoft Word Format Revision 3: Reorganized: March 2007 Distinguished between Rocmop1 and Rocmop2
Effective Date:	05/02/2007

1.0 Introduction

1.1 Rocmop Overview

Rocmop (mesh optimization) is a Roccom module which improves the quality of unstructured volume meshes through nodal repositioning, a process referred to as *mesh smoothing*. Mesquite is an externally developed serial mesh-smoothing utility which Rocmop instantiates to improve each pane of the mesh in isolation. The module implements a simple averaging scheme to realign shared nodes at pane interfaces. Tetrahedral and hexahedral elements are supported in Rocmop. Currently it makes use of Mesquite version 0.95.

Currently Rocmop has 2 versions: Rocmop1 and Rocmop2. Rocmop1 *requires* that input meshes contain complete ghost information, i.e., all 5 *pconn* blocks described in the Roccom user's guide must be present. Rocmop2 *requires only* shared node information (*pconn* block 1) since it will construct the rest itself. See section 3.0, “Building and Running” for instructions on compiling with the different versions of Rocmop.

Rocmop is written in C++ and uses the Roccom framework. It builds on all platforms supported by *Rocstar* 3. A control file can be used to set Rocmop's run-time options; alternatively, some options can be set directly through the Rocmop API. Rocmop will operate on any unstructured tetrahedral volume mesh which is properly registered through Roccom, and contains the required *pconn* information.

1.2 Related Documents

The information in this guide is supplemented by the following documents:

- “Roccom User's Guide”
- “Rocin User's Guide”
- “Rocketeer User's Guide”
- “Rocmop Developer's Guide”
- “*Rocstar* 3 User's Guide”

2.0 Purpose and Methods

Rocmop is intended to provide parallel online smoothing of meshes during the course of a simulation. The goals of this operation are twofold. First, it seeks to prolong the life of the simulation by delaying element inversion, an event which currently terminates the run. Secondly, the length of the time step which the solvers can handle is limited by mesh quality, so mesh smoothing might enable larger time steps, reducing wall-clock time. A standalone version of

Rocmop is also provided which is useful for improving meshes without running a full-blown *Rocstar* 3 simulation.

Figure 1 illustrates the complete smoothing process.

2.1 Serial Pane Smoothing via Mesquite

Mesquite is a robust mesh smoothing package with a wide variety of smoothing algorithms, quality measures, and termination criteria. Rocmop uses this tool to perform optimization via a single iteration of a Feasible Newton solver, using the inverse mean ratio as a quality measure. A separate instance of Mesquite is created to handle each pane of the mesh. When using Rocmop1, these panes must include ghost cell information in the form of a Roccom *pconn* attribute. Rocmop2 ignores any preexisting ghost information (*pconn* blocks 2-4), and generates this information itself. The nodes on the pane boundary are fixed in place while Mesquite operates on the volume interior.

2.2 Pane Interface Node Realignment

Nodes on the boundaries of two panes are often interior to both pane's meshes when ghost information is considered. Therefore, smoothing of the mesh panes in isolation often results in disagreement between panes on the location of a shared node. To address this issue, after smoothing in isolation, each pane updates shared-node coordinates by setting them equal to the average position of the node across all incident panes.

2.3 Diagrams

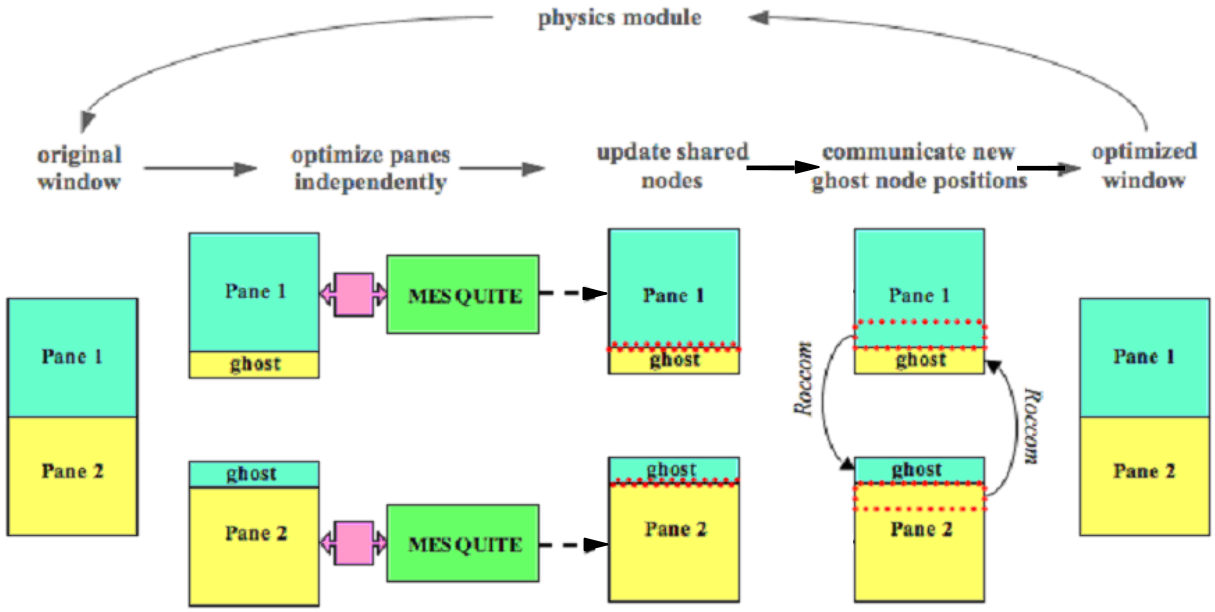

Figure 1 Rocmop smoothing scheme

3.0 Building and Running

Rocmop is written in C++ and uses the *Rocstar* 3 makefile infrastructure. The source and makefiles are found at /Rocstar/Rocmop/Codes in the CSAR CVS repository with the following subdirectories:

```
Rocmop/Codes/Makefile
Rocmop/Codes/Makefile.basic
Rocmop/Codes/Makefile.in
Rocmop/Codes/include
Rocmop/Codes/src
Rocmop/Codes/test
Rocmop/Codes/util
```

A separate directory, `Rocmop/Codes/External,` contains Mesquite's source code. Rocmop currently uses the 0.95 release of Mesquite, but the repository also contains v. 0.9, v. 1.1. and v. 1.1.4 for future development.

3.1 Library Dependencies

Rocmop is integrated into *Rocstar* 3, but it may also be built separately. Because Rocmop links to several of Roccom's submodules - Rocsurf, Rocblas, Rocin, etc. - Roccom should be built

before Rocmop. Rocin uses NCSA's HDF4 library and/or CGNS for managing file I/O. Instructions on obtaining and installing HDF4 are found in the *Rocstar* 3 User's Guide. The following commands will checkout the Roccom and Rocmop source files into ./Roccom and ./Rocmop respectively and will build Roccom's default dynamic libraries:

```
>$ cvs co -d Roccom Rocstar/Roccom/Codes
>$ cvs co -d Rocmop Rocstar/Rocmop/Codes
>$ cd Roccom
Roccom>$ gmake
```

Note that *make* may be used instead of *gmake* on some machines such as Turing. The *Rocstar* 3 User's Guide has more information on the *Rocstar* 3 makefile infrastructure and build options.

3.2 Building Rocmop

After the prerequisite libraries are built, Rocmop may be built as follows:

```
Roccom>$ cd ../Rocmop
Rocmop>$ gmake
```

This will produce Rocmop library files “libRocmop.dylib” and “libRocmop.so” in a directory named “lib” at the same level as the “Rocmop” directory. In addition to the standard *Rocstar* 3 makefile options described in the *Rocstar* 3 User's Guide, Rocmop provides an additional flag, *ROCMOP*, which builds different versions of Rocmop. The default version is Rocmop1.

Option	***Description***
ROCMOP=Rocmop1	Build Rocmop1 (default)
ROCMOP=Rocmop2	Build Rocmop2

The default target of Rocmop's makefiles is Rocmop's dynamic library. Other targets are used to clean up the Rocmop directory or to build the test programs and utility:

Target	***Description***
clean	Remove files created during building and linking, as well as .hdf files
add_aspect_ratios	A test program which inserts values of aspect ration metric of an existing meshes into new .hdf files
build_meshes	A test program which creates a series of simple .hdf files of various types
metric_demo	A test program which illustrates the use of various quality metrics
smooth_volume	A test program which smooths a .hdf file or series of .hdf files
ChkPconnGRecv	A utility to test meshes if they all have complete *pconns*. Complete

Target	*Description*
	pconn which lists all ghost nodes in its ghost receive block (block 3) is required when using Rocmop2.

3.3 Running Rocmop (the Rocmop API)

This section describes the set of functions which is available through the Roccom framework when Rocmop has been registered as a module. Its use is illustrated in "smooth_volume", a test program to smooth meshes outside of an actual simulation. This program is described in section 5.4.

smooth(const COM::Attribute *pmesh, COM::Attribute *disp)

Parameter	*Description*
pmesh	The Roccom *pmesh* attribute of the mesh to be smoothed. See the Roccom User's guide for a description of the *pmesh* attribute
disp	A nodal buffer with three components of type COM_DOUBLE. It is the responsibility of a caller to allocate memory for *disp*. At exit, it contains a displacement, which, when added to the current nodal coordinates of the mesh, produces a smoothed mesh

set_value(const char* opt, const void* value)

Parameter	*Description*
opt	Name of the option to modify (see the table below)
value	New value of the option

Rocmop options accessible through *set_value( )* include:

Option	*Value Type*	*Values*	*Description*
verbose	int	Integers ≥ 0	Verbosity level (0 - 6), used for debugging. Default value is 0, the lowest level
method	int	0	Perform volume smoothing on an unstructured tetrahedral or hexahedral volume mesh. Currently no other valid options
lazy	int	0,1	Check mesh quality before smoothing?
tol	float	(0.0,180.0)	If lazy option is set, mesh only smoothed if maximum dihedral angle exceeds *tol*
maxdisp	float	≥ 0.0	Scale nodal displacement returned by Rocmop so that no node is displaced more than *maxdisp*. If set to 0.0, no scaling is performed
inverted	int	0,1	Set to 1 if tetrahedral node ordering is inverted according to the standards provided in the Roccom User's Guide. Rocflu is an example of a module that uses inverted convention.

4.0 Input and Output (User Interface)

Rocmop is typically loaded and invoked through the Roccom API. Options may be set through *set_value( )*, but the module also looks for a control file, and , if it is present, sets a subset of the run-time options based on the contents of that file.

4.1 Use as a Roccom Module

This C++ example shows how to use Rocmop through the Roccom framework:

```
// Load Rocmop into the Roccom framework
COM_LOAD_MODULE_STATIC_DYNAMIC( Rocmop, "MOP");

// Get function handle for Rocmop::smooth(...)
int MOP_smth = COM_get_function_handle("MOP.smooth");

// Call Rocmop::smooth.
// VOL_pmesh is a handle for the pmesh attribute of the mesh
// to smooth.
// VOL_disp is a handle to the nodal displacement buffer.
COM_call_function( MOP_smth, &VOL_pmesh, &VOL_disp);
```

The source code for the test programs serve as a more complete template for using Rocmop in the Roccom framework. For examples, *smooth_volume* and *add_aspect_ratios* illustrate how to use Rocmop to smooth meshes and compute mesh quality statistics respectively.

4.2 Control File Format

When the Rocmop module is loaded through Roccom, the library automatically looks for the file "RocmopControl.txt". Figure 2 shows the locations where Rocmop looks for a control file. In the case of an integrated simulation, this file should be placed in Rocmop directory in dataset directory. However, if Rocmop is being run in standalone mode through one of the test programs, the control file should be placed in Rocmop directory in the current directory.

Integrated Simulation	*Standalone Mode*
Data Set ∟ Rocmop ∟ RocmopControl.txt	Current Directory ∟ Rocmop ∟ RocmopControl.txt

Figure 2 Locations where Rocmop control file for integrated simulation (left) and standalone mode (right)

If a control file is found, Rocmop sets an ordered list of options to the values found in the file. Lines beginning with either an empty space or '#' are ignored. The order of the values in the control file corresponds to the following list of Rocmop options: *verbosity, method, lazy, tol, maxdisp, N* and *disp_thresh.* With the exception of the last two entries, these options are also accessible through *set_value( )*. *N* accepts an integer value, and when set to anything other than 0, the mesh is smoothed every Nth step. Rocmop is disabled if *N* is set to 0. *Disp_thresh* takes a float value. When it is set to a value other than 0.0, Rocmop adds the largest magnitude displacement of the input nodal displacements at each call to *smooth( )*. When this running total exceeds *disp_thresh,* mesh smoothing is performed. Here is an example Rocmop control file:

```
# RocmopControl.txt

# Verbosity
1
#Smoothing Method
0
#Lazy?
0
#Tolerance
165.0
#Max disp
0.0
#N (smoothing frequency)
1
```

```
#Disp Tol (only smoothed when this disp reached)
0.0

# End RocmopControl.txt
```

5.0 Examples, Test Problems and Utility

There are five example programs, which *are add_aspect_ratios, smooth_volume, build_meshes, metric_demo, chkPconnGRecv*. The first two programs are in parallel while the rest are in serial. This section describes each program and its usage.

5.1 add_aspect_ratios

Usage:
```
>$add_aspect_ratios <Rocin input file> <material name>
```

This program reads in an .hdf file specified in <Rocin input file>, calculates aspect ratio metrics and associated values on the mesh, and prints out the new .hdf file with the metrics attached as elemental attributes. The new attributes are circumradius (*R*), inradius (*r*), shortest edge length (*l*), *3r/R*, *R/r*, *R/l*, minimum dihedral angle, and maximum dihedral angle. Users specify material name for the output .hdf file through <material name>.

5.2 build_meshes

Usage:
```
>$build_meshes
```

Build_meshes creates a series of small unstructured meshes which are useful for debugging and testing smoothing procedures. By design, each mesh contains at least one poorly placed interior node. These meshes are written out to .hdf files. Partitioned meshes created include the shared-node section of the *pconn*, but none of the ghost sections. See the Roccom User's Guide for a description of the different blocks of the *pconn* attribute.

File	*Mesh Description*
unstr_hex0000.hdf	Serial hexahedral mesh with 8 elements and 27 nodes
unstr_prism0000.hdf	Serial prismatic mesh of 8 elements and 15 nodes
unstr_prism_tet0000.hdf	Serial mixed element mesh consisting of 8 prisms, 4 tetrahedrons, and 16 nodes
unstr_prism_tet_20000.hdf	Partitioned mixed element mesh with 32 prisms, 27 tetrahedrons and 46 nodes per partition.
unstr_pyr0000.hdf	Serial pyramidal mesh with 6 elements and 9 nodes.
unstr_tet_2000.hdf	A two-partition mesh with 27 nodes and 48 tetrahedrons per partition.

5.3 metric_demo

Usage:
```
>$metric_demo
```

This program demonstrates the usage of the mesh quality metrics classes by calculating Algebraic and Geometric mesh quality metrics on various triangles, tetrahedrons, quadrilaterals, and hexahedrons.

5.4 smooth_volume

Usage:
```
>$smooth_volume <Rocin input file> <Output file prefix> <Niter>
<Invert Tets>
```

Smooth_volume is used to smooth a fluids mesh contained in an .hdf file, or series of .hdf files, without running an integrated *Rocstar* 3 simulation. The <Rocin input file> specifies the input. See the Rocin User's Guide for more information on *Rocin* control files. The program also looks for a Rocmop control file in Rocmop directory in the current directory, and if found, sets Rocmop's run-time options from this file as described in section 4.2. Unlike *Rocstar* 3, smooth_volume does not require a process per mesh partition, and may even be used in serial to smooth a partitioned mesh if both Roccom and *Rocstar* 3 are compiled with the *DUMMY_MPI* flag set. The smoothed mesh is written back out to an .hdf file whose name begins with <Output file prefix>. <Niter> sets the number of iterations of smoothing to make. <Invert Tets> should be set to a nonzero value if tetrahedral elements are inverted with respect to Roccom conventions, which is the case with the Rocflu fluids module.

5.5 chkPconnGRecv

Usage:
```
>$chkPconnGRecv <sample filename> <number of partitions> <index
base> <option>
```

chkPconnGRecv is the utility to check *pconn*s in all meshes if they list all the ghost nodes in their ghost receive block (block 3). The completeness of *pconn* is required in order to update ghost nodes through the communication and is crucial when using Rocmop2. The <sample filename> specifies an example of the input filenames. <index base> specifies what the first partition is numbered (0 or 1). If <option> is set to 1 the utility writes out the ghost nodes missing from the *pconn* in Tecplot format.