Rocrem User's Guide

Terry L. Wilmarth
wilmarth@uiuc.edu
Center for Simulation of Advanced Rockets

September 16, 2008

1 Introduction

This document describes the use of CSAR's *Rocrem* suite of tools for off-line and on-line remeshing and parallel solution data transfer.

1.1 Goal and Scope

The goal of this guide is to provide a high-level description of the remeshing and solution data transfer capabilities of *Rocrem* to aid potential users. For details on processes and techniques involved in remeshing and data transfer, please refer to the *Rocrem* Developer's Guide.

1.2 Related Documents

1. *Rocmop* User's Guide: describes mesh smoothing, an approach to try before remeshing; also used before solution transfer to smooth the volume mesh.

2. *Rocprop* User's Guide: describes surface mesh smoothing, which can be used to smooth the new surface mesh before a volume mesh is generated to fill the volume enclosed by the surface mesh.

3. *Rocrep* User's Guide: describes local mesh repair, another approach to try before remeshing, should smoothing with *Rocmop* prove inadequate.

4. *Rocrem* Developer's Guide: gives the low-level implementation details of this remeshing tool.

2 Purpose and Methods

The primary purpose of *Rocrem* is to enable a simulation to continue. When mesh deformation impairs the functioning of a solver, there are three levels of desperation that should be applied to obtain a usable mesh.

1. Smoothing(*Rocmop*): this involves the movement of vertices in the mesh to improve the overall mesh quality. It does not help in the face of severe mesh distortion. It does not modify the mesh dramatically and therefore does not require updating solution data.

2. Local repair(*Rocrep*): this involves rearrangement of nodes and elements in areas of the mesh that are particularly poor. Local repair applies a set of local operations such as refinements, coarsenings or Delaunay-style flips to elements in the distorted areas of the mesh. As elements and vertices are modified, solution data must also be modified to correspond to the altered mesh. While this technique is appropriate for isolated areas of the mesh, it can be expensive to modify and transfer solution at the primitive level of the operations. Thus it should not be used for widespread mesh distortion. *Rocrep* was never fully developed, so it may be wise to skip this approach and move directly to full remeshing via *Rocrem*.

3. Remeshing(*Rocrem*): this approach discards the current mesh topology, and reconstructs an entirely new mesh constrained by the same geometry as the original mesh. Solution data transfer is then performed from the old mesh to the new mesh. This is an expensive operation that should only be used in the worst case of severe widespread mesh distortion.

Typically, a mesh that is suitable for remeshing will have a global distortion that affects many areas of the mesh. Such a mesh is shown in Figure 1. If the affected areas are very limited in range occurring around some distinctive mesh features, local repair is more suitable.

3 Building and Running

3.1 Requirements

Rocrem is an AMPI program that uses Simmetrix for remeshing, ParFUM for parallel mesh manipulation, and CHARM++ for the parallel collision detection used in solution data transfer. It depends on several other *Rocstar* components for various pre- and post-processing and smoothing activities, such as:

- *Roccom*/*Rocin* is used for reading a mesh dump from HDF files.
- *Roctail* is used for post-processing the mesh to produce a restart-ready dataset. It in turn depends on *Rocface* for surface data transfer.

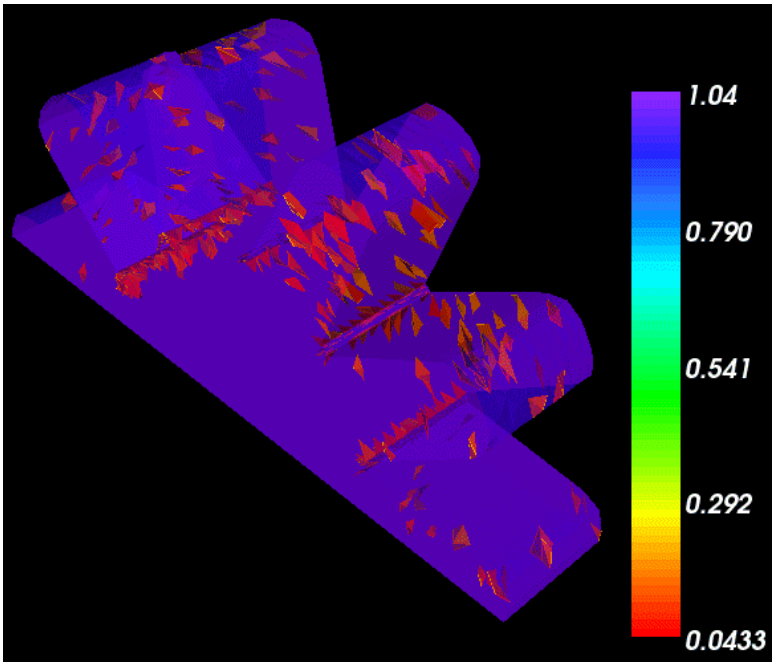

Figure 1: A starslice mesh with bad elements shown in red.

- *Rocmop* is used before the solution transfer stage to smooth out the mesh in a manner acceptable to *Rocstar*.
- *Rocprop* is used after generating the surface mesh with Simmetrix to smooth out the surface in a manner acceptable to *Rocstar*.

3.2 Building

These are the basic steps to building the *Rocrem* tool suite:

1. Install Simmetrix software on your target architecture. For CSAR staff, this entails copying the appropriate binaries from the distribution on turing.
2. Install the latest stable CHARM++ distribution and build the ParFUM version for your target architecture. See `http://charm.cs.uiuc.edu` for more details on selecting a version, downloading and building CHARM++.
3. Check out `genx/Codes` from the CSAR CVS repository.
4. Enter the `genx/Codes` directory and build with the following command:

   ```
   make REMESH=1 CHARM=1 SIMMETRIX=1 CHARM_PATH=<yourCharmPath>
   ```

   You may also wish to specify a `SIMMETRIX_PATH` if Simmetrix cannot be found in the default location expected by the *Rocstar* Makefile.

5. If you would like to build the partitioning and solution transfer tools alone on a system that is not supported by Simmetrix, eliminate the SIMMETRIX=1 directive.

4 User Interface

The *Rocrem* suite of remeshing tools can be used in two primary modes: the stand-alone tools and the on-line mode. The stand-alone tools have several options for performing various meshing related tasks.

4.1 Stand-alone Tools

Limitations on parallelism in Simmetrix currently prevent a fully parallel remeshing tool. Thus, the remeshing phase of *Rocrem* is currently serial. Efforts are ongoing to provide parallel remeshing functionality, and the currently available tools and their limitations are documented herein. To enable the most flexible usage of the tools, they have been provided as modular components with separate executables. The full remeshing process is still available in the `Rocrem` executable, but the user may find it more convenient to invoke the serial remeshing and the parallel solution transfer phases separately.

Remeshing a *Rocstar* dataset follows this process:

1. The old mesh and solution data are read, in parallel, from ordinary output dumps of the integrated code using the Rocin and Roccom interface. This is performed by either the `Rocrem` or `rocprem` executables.

2. Remeshing is performed by reassembling the serial interface mesh, including boundary conditions, and calling the commercial meshing tools from Simmetrix. The `Rocrem` or `rocprem` and `rocparm` programs perform the remeshing, with `Rocrem` being completely serial, while the pairing of `rocprem` for serial surface remeshing with `rocparm` for parallel volume remeshing accomplish as much of the task in parallel as Simmetrix software currently allows.

3. The new mesh and boundary conditions are partitioned using METIS or ParMETIS via ParFUM or the Geometric Partitioner. The METIS/ParMETIS-based partitioning phase can be performed either at the end of `Rocrem`, the beginning of `fem1xfer`, or separately by `partition`. The Geometric Partitioner is used in a separate phase only, `geompartition`. When `rocparm` is used, Simmetrix uses ParMETIS to partition the mesh during mesh generation, so no separate partitioning step is required (although the mesh is repartitioned multiple times in `rocparm` to maintain load balance).

4. Solution data is transferred from the old mesh to the new mesh in parallel, using the Charm++ Parallel Collision Detection Library to match up the old and new mesh. This is accomplished with the `fem1xfer` program.

5. The integrated code is restarted with the new mesh and solution data.

The *Rocrem* suite of remeshing tools consists of the following:

- **Rocrem**: This is the primary remeshing and solution transfer tool. It can be used to perform remeshing alone, or remeshing and solution transfer combined. It is important to note that when using the combined form, parallelism is needed to handle partitioning and solution transfer on large meshes, so the long serial remeshing phase will require significant time during which all the processors but one will remain idle. It is recommended to use this tool serially for remeshing only, and to use `fem1xfer` subsequently to perform the solution transfer quickly in parallel. Most executables require the working directory be the main *Rocstar* directory (e.g. `<mypath>/ACM/024procs/`). The usage is thus:

  ```
  Rocrem [<flags>] +vp<partitions> <solver> <timestep>
  ```

- **rocprem**: This tool is used to prepare a serial mesh for parallel handling in `rocparm`. It performs much the same functionality as the initial phases of `Rocrem`, reading the parallel mesh, assembling the surface on one processor, and remeshing and smoothing that surface. It writes two primary files: `rocrem_discmodel_<timestep>.sdm` and `rocrem_smoothed_surf_<timestep>.sms`. The usage is nearly identical to that of `Rocrem`:

  ```
  rocprem [<flags>] +vp<partitions> <solver> <timestep>
  ```

- **rocparm**: This tool is used to perform volume generation on a mesh that has been prepared with `rocprem`. It writes out a partitioned mesh in ASCII file formats named with the following pattern: `rocrem_encoded_mesh_<timestep>_<vp#>.out`, thus one file for each partition. These files are read automatically by fem1xfer when using the `-q3` option. The usage is nearly identical to that of `Rocrem`, with the exception that the program is a pure MPI code that must be run on the exact same number of processors as the desired number of partitions. Thus there is no `+vp<#vp>` argument.

  ```
  rocparm [<flags>] <solver> <timestep>
  ```

- **partition**: This is a very simple program that reads a serial ParFUM mesh and partitions it with ParMETIS, writing out the partitioned mesh in ParFUM format. It can also be made to use serial METIS, which is particularly convenient when ParMETIS has problems on 64-but machines.

  ```
  partition <timestep> +vp<partitions>
  ```

- **geompartition**: This is a very simple program that reads a serial ParFUM mesh and partitions it with GeomPar, writing out the partitioned mesh in ParFUM format.

  ```
  geompartition [<partition granularity>] <timestep> +vp<partitions>
  ```

- **fem1xfer**: This tool takes a ParFUM mesh (serial or partitioned) or a partitioned ASCII dataset converted to a ParFUM mesh and performs solution transfer from an

HDF dataset to the ParFUM mesh. This is used as the second phase of remeshing, and is also used as the third phase of the local mesh repair process (*Rocrep*).

`fem1xfer [<flags>] +vp<partitions> <solver> <timestep>`

- **smtx2tmsh**: This is a simple serial tool to convert the results of repairing a mesh with *Rocrep* into a form suitable for running solution transfer on via `fem1xfer` (NOT MAINTAINED)

  `smtx2tmsh [<flags>] <solver> <timestep>`

Rocrem's stand-alone tools are invoked on the command line or in a script. Most of the tools have the same set of `<flags>`, some of which are not defined for some tools. Flags that are meaningless to a particular tool are simply ignored.

```
Usage: <exec> <args>
<args> :- [<flags>] +vp<partitions> <solver> <timestep>
 <partitions> is the number of resulting partitions after remesh.
 <solver> is the path to the solver files. Example: Rocflu
 <timestep> is the XX.XXXXXX time used by GENx (default 0).
 <flags> are optional, and include:
  -bs <surfPrefix>     Specify a surface or volume HDF file prefix.
  -bv <volPrefix>      These are handy when trying to read a strangely-named
                       dataset.  These are set to the common case by default.

  -u                   Do not remesh the surface. Uses the raw surface data.

  -k <face> <patch>    Merge model face <face> into patch <patch>. This is
                       used when a feature is disappearing.

  -y                   Shut off Roctail surface transfer.

  -z                   Transfer the surface data via copy (works only with -u).

  -x                   Transfer the surface data via extrusion method. (NYI)

  -f                   Fast (low quality) volume remeshing.

  -l                   Surface mesh modification level.  Default 2.

  -s<sizing> <val>     Specify a sizing method. e.g.:
        -sabs <size>    Use absolute mesh sizing, size is edge length
        -srel <scale>   Use relative mesh sizing, scaled by scale. Default: 1.0
        -savg <scale>   Use averaged mesh sizing, scaled by scale
        -svol <scale>   Scale volume sizing by additional amount. Default: 0.2
```

```
-j                    Force serial partitioning. (ParMETIS is default.)

-mmop <num>           Smooth volume mesh num times with Rocmop before solution
                      transfer.

-mprop <num>          Smooth surface mesh num times with Rocprop before
                      solution transfer.

-q<num> [<path> <dir> <time>]
                      Specify a target destination mesh for solution transfer.
                      e.g.:
      -q0              Reads parallel_fem_*.dat files
      -q1              Reads serial_fem_*.dat files
      -q2 <path> <dir> <time> Reads an HDF dump at timestep <time> from
                       <path>/<dir>.
      -q3              Reads parallel ASCII mesh (parallel remeshing output).

-g<num>               Add n ghost layers to mesh. Default: num=1, 0<=num<=9.

-t                    Skip data transfer.

-e                    Skip extrusion.

-p<num>               Specify number of partitions in original mesh.
                      Default: num=partitions (above).

-i <indir>            Directory in solver dir to read from.
                      Default: indir="Rocout".

-o <outdir>           Directory in solver dir to write to.
                      Default: outdir="Rocout.remesh_<timestamp>"
-1                    That's a one, signifying partition ids start at 1.
                      Default: 0.
-d<num>               Field width for partition id. Default: num=4, 3<=num<=9.

-a<filebase>          Read repaired mesh, perform data transfer.
                      Looks for <filebase>+.sdm, .sms, .f2p and .p2bc.

-v<num>               Set level of debugging verbosity. Default num=0 (off).

-w<num>               Set level of debugging files. Default num=0 (off).
```

```
-c                    Perform extensive self-checking.

--help                Prints this summary.
```

Most of these operations are well-explained by the usage text above, but we will elaborate on these below. We also indicate for which executables these are defined in square brackets. The `partition` and `geompartition` executables take significantly fewer arguments; these are documented separately earlier in this section.

Required Arguments

- `+vp<partitions>`: This is the number of partitions required for the resulting dataset, and in most cases, it is also the number of partitions in the input dataset. Changing the number of partitions mid-stream in the remeshing process is discussed later in this section. This argument is not required by `rocparm` which must be run on the same number of processors as the desired number of partitions.
- `<solver>`: This is the solver from which the mesh comes and for which it is destined. This argument is largely used to find the path to the dataset, but may also be used in future to perform solver-specific functions. This argument is not needed for either of the partitioning programs.
- `<timestep>`: This is the XX.XXXXXX format for a timestep that is used to label input and output files.

General Flags

The general flags as summarized above are used for all codes except the two partitioning programs, `partition` and `geompartition`.

- `-v<num>`: Set level of debugging verbosity. Default num=0 (off). [ALL]
- `-w<num>`: Set level of debugging output files. Default num=0 (off). These files are placed in the `Rocrem` directory within the main *Rocstar* directory. [ALL]
- `-c`: Perform extensive self-checking. Use of this option on the command-line is deprecated as self-checking is being optimized out via a preprocessor flag. [ALL]

Input/Output File Flags

- `-bs<surfPrefix>`: Specify an input surface file prefix to be used as an alternative to '`ifluid`'. [`Rocrem, rocprem, fem1xfer`]
- `-bv<volPrefix>`: Specify a input volume file prefix to be used as an alternative to '`fluid`'. [`Rocrem, rocprem, fem1xfer`]

- `-i<indir>`: Directory in solver dir to read from. By default, indir is '`Rocout`'. [`Rocrem, rocprem, fem1xfer`]
- `-o<outdir>`: Directory in solver dir to write to. By default, outdir is '`Rocout.remesh_<timestamp>`'. This is passed to *Roctail* which handles post-processing and does the actually writing of the new dataset. [`Rocrem, fem1xfer`]
- `-1`: The one signifies partition ids start at 1 instead of 0. [`Rocrem, rocprem, fem1xfer`]
- `-d<num>`: Field width for partition id in the input dump filenames. This id 4 by default, because currently most dumps use 4 digits. [`Rocrem, rocprem, fem1xfer`]
- `-a<filebase>`: Read repaired mesh from *Rocrep* output. Looks for <filebase>+.sdm, .sms, .f2p and .p2bc. [`smtx2tmsh`]

Surface Remeshing

- `-u`: Do not remesh the surface. This will keep the original mesh surface, remeshing the volume only. This limits the possible improvements in mesh quality. [`Rocrem, rocprem`]
- `-l <num>`: Surface remeshing aggressiveness level. If num=1, leave the surface mesh alone. If num=2 (the default), perform reasonable surface mesh modifications. If num=3, aggressivly modify the surface mesh in order to achieve high quality. For num=3, beware of edge flips along model edges; use this level only if num=2 fails, as it can introduce spurious "features" into the geometry which may lead to surface intersections.
- `-mprop <num>`: Smooth the surface mesh with *Rocprop* num times before generating the volume mesh. This seems to iron out some restart issues. [`Rocrem`]
- `-sabs <size>`: Scale the mesh using absolute mesh sizing. The size is specified as the desired edge length. [`Rocrem, rocprem`]
- `-srel <scale>`: Scale the mesh using relative mesh sizing. Edge sizes are scaled by scale. By default, scale is 1.0, so no scaling is done. [`Rocrem, rocprem`]
- `-savg <scale>`: Scale the mesh by setting the desired edge length to the average edge length over the entire mesh multiplied by scale. [`Rocrem, rocprem`]
- `-k <face> <patch>`: Merge a model face `<face>` into a neighboring patch `<patch>`. This is used when a feature is being compressed into a single layer of slivers. The feature is merged with a neighboring region, so that subsequent remeshing can remove the slivers. [`Rocrem, rocprem`]

Volume Remeshing

- `-f`: Fast (low quality) remeshing. This option sacrifices quality and accuracy for a fast remeshing process. Specifically, it skips the refinement, smoothing and optimization phases that are required to produce a high-quality mesh with the proper sizing. This is typically used to obtain quick test results for debugging purposes. [`Rocrem, rocparm`]

- `-svol <scale>`: Volume sizing is first derived from surface sizing. Any given volume edge will be sized similarly to the nearest surface edge. This flag allows for an additional scale factor to be applied to the volume. It is added to a no-scaling factor of 1.0 for the volume scaling. By default, it is set to 0.2, to compensate for Simmetrix's tendency to over refine volume regions. This means if the surface is scaled by x, then the volume in effect scaled by x+0.2. A more accurate way of stating this is to say that however the surface is scaled, the volume mesh is scaled by 1.2 with respect to the surface. A negative value can be passed to obtain a finer volume mesh.[`Rocrem, rocparm`]

Partitioning

- `-j`: When partitioning in `Rocrem` or `fem1xfer`, parallel partitioning via ParMETIS is used by default. Use of this flag forces the use of serial partitioning via METIS. This is particularly useful on machines where ParMETIS is unstable, such as 64-bit machines.

- `-g<num>`: Add n ghost layers to mesh, where n is 1 by default. A single ghost layer for a region is composed of all non-local elements that share at least one node with any local element, plus all non-local nodes that are found in the connectivity of those elements. Subsequent ghost layers are defined similarly on the region bounded by the previous ghost layer. [`Rocrem, fem1xfer, smtx2tmsh`]

- `-p<num>`: Specify the number of partitions in original mesh, in case it differs from the number of target partitions. This option currently does nothing. The number of original partitions is assumed to equal the number of target partitions. [`Rocrem, fem1xfer, smtx2tmsh`]

Solution Transfer

- `-q<num>`: Specify mesh files as the transfer destination. By default, it assumes a parallel ParFUM mesh as the destination (num=0). These files are of the form `parallel_fem_<timestep>_vp<vp#>_<#vp>.dat`. With num=1, it looks for a serial ParFUM mesh, `serial_fem_<timestep>_vp<vp#>_<#vp>.dat`. For num=2, it looks for an HDF dataset. This version takes three additional args that help locate the HDF dataset. Finally, num=3 tells `fem1xfer` to read a partitioned ASCII dataset generated by `rocparm`. [`fem1xfer`]

- `-mmop <num>`: Smooth the volume mesh with *Rocmop* num times before performing solution transfer. This seems to iron out some restart issues. [`Rocrem, fem1xfer`]

- `-z`: Transfer via copy the original surface data (works only with -u). Use of this is deprecated. [`Rocrem`]
- `-x`: Transfer the surface data via extrusion method. Not yet implemented. [`Rocrem, fem1xfer`]
- `-t`: Skip data transfer phase. Writes out the mesh in ParFUM format so that it can be read later by `fem1xfer`. [`Rocrem`]
- `-e`: Skip extrusion. This option is useful when we are sure the original mesh surface corresponds perfectly to the new surface. Extrusion is performed by default to account for variances in the surfaces of the original and new mesh. It improves the accuracy and conservation of solution transfer near the surface when the new surface differs from the old. If the new surface does not differ from the old, specifying this flag speeds up solution transfer slightly. Using this option when the surfaces differ will compromise accuracy and conservation. [`Rocrem, fem1xfer`]

Post-processing

- `-y`: Shut off Roctail surface transfer. This prevents Roctail from attempting to perform surface transfer in cases when we know it will fail. [`Rocrem, fem1xfer`]

Changing Number of Partitions During a Remesh

When processor availability changes in the midst of a simulation, it is often convenient to be able to restart the simulation on a different number of processors. *Rocrem* can be used in this case to remesh the rocket, and generate the new mesh with a different number of partitions from the original input mesh. Given `<old-parts>` as the number of partitions of the original mesh and `<new-parts>` as the desired number of partitions for the remeshed mesh, the process for achieving this repartitioning is as follows:

1. Run `rocprem` serially with `<old-parts>` VPs to read the original mesh: e.g. `rocprem -srel 1.2 -v2 -w1 +vp256 Rocflu 10.230000` would read an initial dataset with 256 partitions and prepare it for remeshing.
2. Run `rocparm` on `<new-parts>` processors. This code is pure MPI, so there is no `+vp` argument. Note the directory change: e.g. `cd Rocrem; mpirun +np 160 rocparm -svol 0.2 -v2 -w1 Rocflu 10.230000` will create a new mesh with 160 partitions.
3. Run `fem1xfer` with `-p <old-parts> <new-parts>` and specify a number of VPs to be equal to the larger of `<old-parts>` and `<new-parts>`. This should run on any number of processors. Again note the directory change back to the main *Rocstar* directory: e.g. `cd ..; fem1xfer -y -q3 -p 256 160 -v2 -w1 +vp256 Rocflu 10.230000` will perform solution transfer from the original 256 partition mesh to the new 160 partition mesh. Note also the use of `-q3` which tells `fem1xfer` to obtain the mesh via the special ASCII partitioned mesh files generated by `rocparm`.

4.2 On-line Remeshing with the *Rocrem* API

The on-line remeshing mode of *Rocrem* is designed to be used inside of *Rocstar* for automatic remeshing. The method of operation is as follows:

1. In *Rocstar*, a solver performs a timestep.
2. A decision process determines if one of the mesh improvement techniques should be triggered. It is possible that more than one technique is applied.
3. If the decision process chooses to remesh, remeshing and solution transfer is invoked through the *Rocrem* API.
4. When *Rocrem* is finished, a new *Rocstar* restart dataset has been created.
5. *Rocstar* performs a warm restart using the new dataset.

The interface with *Rocrem* is defined as follows:

```
// char* solver: Path to Rocout directory (i.e. blah/blah/Rocflu)
// char* solver_indir, solver_outdir: Where to get/put data other than
Rocout
// double time: timestep to read
// bool par_part: true to use parallel partitioning (will become default)
// bool remesh_surf: true=remesh surface false=leave surface alone
// bool transfer_surf: true=transfer surface data if not remeshing surface
// double scaleFactor: scale surface mesh sizing
// int fieldWidth: digits in filename partition id
// int fileBase: 0 is partition ids start at 0, 1 otherwise
// int debug: 0=off, 1 = some, more = more
// int ngLayers: number of ghost layers; 0-9
// returns 1 if success; 0 if fail
int Rocrem_remesh(const char* solver, const char* solver_indir,
                  const char* solver_outdir, double time, bool par_part,
                  bool remesh_surf, bool transfer_surf, double scaleFactor,
                  MPI_Comm myComm, int fieldWidth, int fileBase, int debug,
                  int ngLayers);
```

`Rocrem_remesh` has arguments similar to the command-line flags, but many are missing, since they are mostly irrelevant in the on-line case.

4.3 Example

In the *Rocrem* source directory, there is a directory called `api_test` which contains a small program to make use of the *Rocrem* API. This example is self-contained, and very nearly complete.

5 *Rocrem* Usage in Practice

5.1 Partitioning

The trickiest part of the remeshing process is actually the partitioning. For small meshes, one can get away with using METIS, which is stable and appears to run on most platforms. However, it will run out of memory when trying to partition larger meshes. To partition larger meshes, either ParMETIS or the Geometric Partitioner can be used. ParMETIS has been unstable on 64-bit architectures, whereas the `geompartition` program is sometimes challenging to build with certain compilers.

There are various ways to handle partitioning in the *Rocrem* tools. Partitioning is performed in `Rocrem` immediately after remeshing and before solution transfer. However, it is often preferred to separate the tasks of remeshing, which is serial, and solution transfer, which is parallel. In this case, partitioning can be performed at the end of the serial remeshing stage in `Rocrem`, or at the beginning of the parallel solution transfer phase in `fem1xfer`. We usually select the appropriate partitioner, METIS or ParMETIS, based on which of these we are using respectively. See the options `-q` and `-j` for details.

Alternatively, METIS or ParMETIS can be invoked in an entirely separate phase via the `partition` program, or the Geometric Partitioner can be used via the `geompartition` program. To use these, the `-q` option is used to tell `Rocrem` to generate a serial mesh dump, and to tell `fem1xfer` to read a parallel mesh dump. Both partitioning programs will read the serial dump and generate the parallel dump.

5.2 Feature Removal

When a surface patch of a rocket mesh is deformed to the point where it is very small or narrow, such that the triangles used to represent it are very thin or considerably smaller than the rest of the mesh, and continued deformation is likely to make the patch disappear entirely, we can re-label the patch such that it blends in with the surrounding patch. This allows *Rocrem* to improve the elements of that patch by no longer preserving its boundaries. The simulation can thus progress to remove the patch entirely.

To do this, run `Rocrem` or `premesh` normally with at least `-w1` and it will generate a zoned surface Tecplot file (filenames beginning with `rocrem_zoned_`). The run can be terminated when it starts generating the volume, as by that time, the zoned surface plots will have been written.

Load the zoned file into Tecplot and view by zones, which are labelled with geometric model ID, patch ID and BC, respectively. Look at the patch you want renumbered and get its geometric model ID. Then look for the patch you want to join the patch to be removed with, and get its patch ID. Tecplot recycles colors so you may need to change the zone colors to ensure that you have uniquely identified the two patches in question. These two IDs can then be passed to `Rocrem` or `rocprem` with the -k option. If, for example, the patch you want to remove has geometric model ID 10, and the patch adjacent to it that you want it to join has a patch ID of 2, you should subsequently run `Rocrem` or `rocprem` with `-k 10 2`.

5.3 Parallel Remeshing

Parallel meshing is available in a three stage process. All three stages can be run on a parallel machine.

1. `rocprem`: Reads the parallel surface mesh and stitches it into a serial surface mesh, loads it into Simmetrix, remeshes and resizes the surface, and calls Rocprop smoothing. This dumps out a discrete model and a surface mesh file in Simmetrix format that are read by the second stage. The serial surface remeshing is short, and the initial stitching phase is parallel, so this code could be run either way.

2. `rocparm`: Reads the Simmetrix surface mesh and discrete model, generates a volume mesh in parallel, adapts the mesh in parallel, and performs smoothing and optimization on the volume in parallel. Writes out an encoded mesh dump. This code is pure MPI and must be called on the same number of processors as the number of desired partitions.

3. `fem1xfer`: This is used just like before to perform solution transfer. The only difference is that the `-q` option must be used appropriately to read the input mesh from the encoded mesh dump.