Running Drekar

The Drekar Input File

- XML input
  - Can also use YAML
- First part: Setting up the mesh
  - Here a uniform “inline mesh”
  - Can also input a custom mesh in Exodus format
  - Specifying periodic boundary conditions

```
<ParameterList name="Mesh">

    <Parameter name="Source" type="string" value="Inline Mesh" />

    <ParameterList name="Exodus File">
        <Parameter name="File Name" type="string" value="drekar.exo" />
    </ParameterList>

    <ParameterList name="Inline Mesh">
        <Parameter name="Mesh Dimension" type="int" value="3" />
        <ParameterList name="Mesh Factory Parameter List">
            <Parameter name="X Blocks" type="int" value="1" />
            <Parameter name="Y Blocks" type="int" value="1" />
            <Parameter name="Z Blocks" type="int" value="1" />
            <Parameter name="X Elements" type="int" value="10" />
            <Parameter name="Y Elements" type="int" value="40" />
            <Parameter name="Z Elements" type="int" value="40" />
            <!--
            <Parameter name="X Procs" type="int" value="8" />
            <Parameter name="Y Procs" type="int" value="2" />
            <Parameter name="Z Procs" type="int" value="2" />
            -->
            <Parameter name="X0" type="double" value="0.0" />
            <Parameter name="Y0" type="double" value="-1.0" />
            <Parameter name="Z0" type="double" value="-1.0" />
            <Parameter name="Xf" type="double" value="5.0" />
            <Parameter name="Yf" type="double" value="1.0" />
            <Parameter name="Zf" type="double" value="1.0" />
              <ParameterList name="Periodic BCs">
                <Parameter name="Count" type="int" value="1" />
                <Parameter name="Periodic Condition 1" type="string" value="yz-coord 1e-8: left;right"/>
              </ParameterList>
        </ParameterList>
    </ParameterList>

</ParameterList> <!--Mesh-->
```

The Drekar Input File

- Pin pressure at specific node
- Set up the physics blocks
  - Most of the time, just have 1
  - Here specified to be a fluid
  - Could have different physics in different parts of the problem
- Specify variables in the equations

```
<ParameterList name="Adapters">
    <ParameterList name="Pinning">
        <Parameter name="Node Set" type="string" value="origin"/>
        <Parameter name="Element Block" type="string" value="eblock-0_0_0"/>
        <Parameter name="Field Name" type="string" value="PRESSURE"/>
        <Parameter name="Value" type="double" value="0.0"/>
    </ParameterList>
</ParameterList>
```

```
<ParameterList name="Block ID to Physics ID Mapping">
    <Parameter name="eblock-0_0_0" type="string" value="fluid"/>
</ParameterList>
```

```
<ParameterList name="Assembly">
    <Parameter name="Field Order" type="string" value="UX UY UZ PRESSURE"/>
    <Parameter name="Workset Size" type="int" value="500"/>
</ParameterList>
```

The Drekar Input File

- What equations will be solved
  - Here, we have a “fluid” block
- Continuity and momentum
- We specify FEM parameters
  - Element type (HGrad)
  - Element order (first order; linear)
  - Quadrature points (2)
- Includes stabilization information
  - Very important, but don’t worry about for now

```
<ParameterList name="Physics Blocks">

    <ParameterList name="fluid">
        <ParameterList>
            <Parameter name="Type" type="string" value="Continuity"/>
            <Parameter name="Basis Type" type="string" value="HGrad"/>
            <Parameter name="Basis Order" type="int" value="1"/>
            <Parameter name="Integration Order" type="int" value="2"/>
            <Parameter name="Model ID" type="string" value="fluid model"/>
            <Parameter name="Prefix" type="string" value=""/>
            <ParameterList name="Options">
                <Parameter name="TAU_C" type="string" value="SHAKIB"/>
                <Parameter name="PSPG STABILIZATION" type="string" value="ON"/>
            </ParameterList>
        </ParameterList>

        <ParameterList>
            <Parameter name="Type" type="string" value="Momentum"/>
            <Parameter name="Basis Type" type="string" value="HGrad"/>
            <Parameter name="Basis Order" type="int" value="1"/>
            <Parameter name="Integration Order" type="int" value="2"/>
            <Parameter name="Model ID" type="string" value="fluid model"/>
            <Parameter name="Prefix" type="string" value=""/>
            <ParameterList name="Options">
                <Parameter name="TAU_M" type="string" value="SHAKIB"/>
                <Parameter name="SUPG STABILIZATION" type="string" value="ON"/>
            </ParameterList>
        </ParameterList>
    </ParameterList>

</ParameterList>
```

The Drekar Input File

- Closure models
- How to treat the stress tensor
- Sets problem parameters
  - Density
  - Newtonian stress tensor
  - Dynamic viscosity
  - Forcing terms
- Print out statistics of variables
  - Volume average
  - Max, Min

```
<ParameterList name="Closure Models">

    <ParameterList name="fluid model">

        <ParameterList name="DENSITY">
            <Parameter name="Value" type="double" value="1.0"/>
        </ParameterList>
        <ParameterList name="STRESS_TENSOR">
            <Parameter name="Value" type="string" value="NEWTONIAN"/>
        </ParameterList>
        <ParameterList name="VISCOSITY">
            <Parameter name="Value" type="double" value="3.0e-1"/>
        </ParameterList>
        <ParameterList name="SOURCE_UX">
            <Parameter name="Value" type="double" value="500.0"/>
        </ParameterList>
        <ParameterList name="SOURCE_UY">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
        <ParameterList name="SOURCE_UZ">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>

        <ParameterList name="Global Statistics">
            <Parameter name="Value" type="string" value="UX,UY,UZ,PRESSURE"/>
        </ParameterList>

    </ParameterList>

</ParameterList>
```

The Drekar Input File

- Boundary conditions
- Basic ingredients:
  - Type
  - Boundary
  - Block ID
  - Equation
  - Value
- You can define non-constant and more general BCs as well
- eblock-0_0_0 is the default value, but it can be changed

```
<ParameterList name="Boundary Conditions">

        <!-- No Slip Set on Top and Bottom of channel -->

    <ParameterList>
        <Parameter name="Type" type="string" value="Dirichlet"/>
        <Parameter name="Sideset ID" type="string" value="top"/>
        <Parameter name="Element Block ID" type="string" value="eblock-0_0_0"/>
        <Parameter name="Equation Set Name" type="string" value="UX"/>
        <Parameter name="Strategy" type="string" value="Constant"/>
        <ParameterList name="Data">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
    </ParameterList>

    <ParameterList>
        <Parameter name="Type" type="string" value="Dirichlet"/>
        <Parameter name="Sideset ID" type="string" value="top"/>
        <Parameter name="Element Block ID" type="string" value="eblock-0_0_0"/>
        <Parameter name="Equation Set Name" type="string" value="UY"/>
        <Parameter name="Strategy" type="string" value="Constant"/>
        <ParameterList name="Data">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
    </ParameterList>

    <ParameterList>
        <Parameter name="Type" type="string" value="Dirichlet"/>
        <Parameter name="Sideset ID" type="string" value="top"/>
        <Parameter name="Element Block ID" type="string" value="eblock-0_0_0"/>
        <Parameter name="Equation Set Name" type="string" value="UZ"/>
        <Parameter name="Strategy" type="string" value="Constant"/>
        <ParameterList name="Data">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
    </ParameterList>
```

The Drekar Input File

- Output
- Where to write solution

```
<ParameterList name="Output">
    <Parameter name="File Name" type="string" value="./Parallel_Database_2/FullyDevRectDuctFlow.exo"/>
</ParameterList>
```

- Output control
- How often to write solution

```
<ParameterList name="Solver Factories">
    <ParameterList name="Rythmos Observers">
        <Parameter name="Write Solution to Exodus File" type="string" value="ON"/>
        <Parameter name="Time Step Interval for Writing Solution" type="int" value="1"/>
        <Parameter name="Write Initial Condition" type="string" value="TRUE"/>
    </ParameterList>
</ParameterList>
```

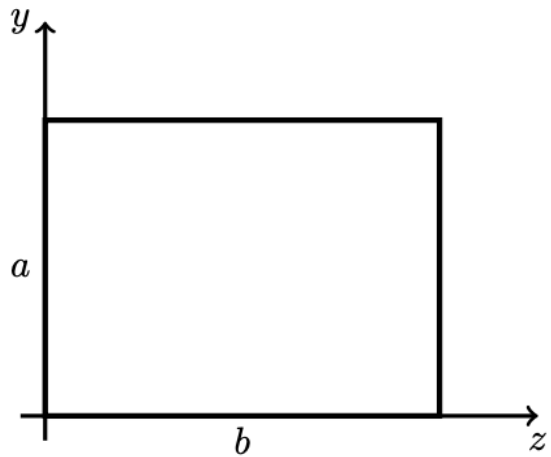
Starting a Simulation Campaign

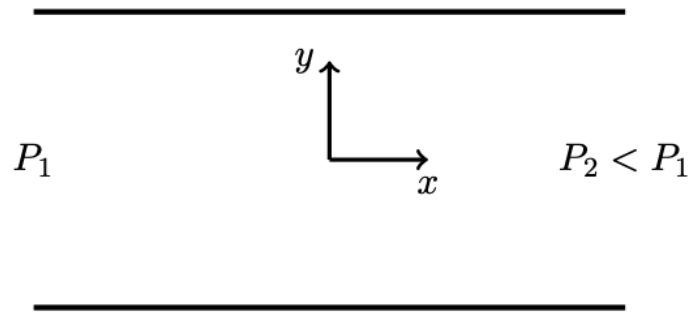
- Organization of a simulation campaign is important
- You can find some tips at Instructions (Part II): Running Drekar
- The basic ideas are:
  - Create a logical directory structure for your jobs
    - From where will you launch your jobs?
    - Where will write solution files out?
  - Recommendation: Create a soft link to the code executable
    - ln -s <target_path> <link_path>
    - Often this is in a /bin directory in your home directory
  - Use a bash script to submit your job and monitor the output
- Pro tip: Keep track of your campaigns in a spreadsheet

Running Jobs

- Follow the instructions at Instructions (Part II): Running Drekar
- You will run the 3D duct flow problem
  - HW1, Problem 2
- The input file can be found at: Lecture 9 Materials
- Try to run with 1 core first
- Next, run with multiple cores
  - Try to think about how many degrees of freedom per core you have

3D Duct Flow





- Recall: The 3D duct flow is a *pressure-driven* flow
- Since the flow is steady, there must be a balance of forces
  - $\underbrace{\text{Surface forces on walls}}_{F_w} = \underbrace{\text{body force}}_{\text{Pressure gradient}}$
- The input file specifies a force in the x-direction for this reason

3D Duct Flow: Reynolds Number

Define the Reynolds number as $Re = \frac{UL}{\nu}$

- U: Velocity scale
  - Average u in duct, centerline streamwise velocity ($u_{\max}$)
- L: Length scale
  - Height of duct, H (Why isn't this the best choice?)
  - Wetted hydraulic diameter: $D = \frac{2ab}{a+b}$
    - $a =$ length of duct in y, $b =$ length of duct in z
    - What is D when $b = a$?
- $\nu = \frac{\mu}{\rho}$ where ρ is the density and μ is the dynamic viscosity

Visualize Results

- Some basic visualization instructions: Instructions (Part III): Visualization
- You can copy your solution to your local machine and run Paraview there
- You can also use netCDF to process the solution files (both on Odyssey and locally)
  - See the Instructions (Part III)
- Some things to plot:
  - Velocity field
  - Velocity profile (different plots for y and z)
  - Compare analytical and numerical solutions (via velocity profiles)
- Other considerations
  - Scaling studies

Scalability Interlude

- A very important concept in research codes is scalability
  - Measured in "time to solution"
- Two primary flavors
  - Strong scaling: Run a problem of a given size using more cores
    - Speedup = $S = \frac{T_1}{T_N}$ where T_1 is the serial time and T_N is the time on N cores
    - Look at $T\left(n_{\text{procs}}\right)$ for a given problem size
  - Weak scaling: Run a bigger problem in the same amount of time
    - Look at $T\left(n_{\text{procs}}\right)$ for a fixed problem size

Total Time and Speedup

The total computation time is

$$T(N) = \frac{p}{N} + s$$

where p is the parallel workload, N is the number of processors, and s is the serial time.

The speedup is therefore

$$S(N) = \frac{p + s}{\frac{p}{N} + s}.$$

Strong Scaling

- Strong scaling is limited by Amdahl's law
- Amdahl's law states that a program is limited by the serial fraction of the workload
- It assumes that the parallel workload is fixed
- Let $f_s = \frac{s}{s+p}$ be the serial fraction of the work
- Then the speedup is

$$S(N) = \frac{N}{1 + (N-1)f_s}$$

- For large N the speedup is $S \to \frac{1}{f_s}$ for large N

Remember, strong scaling means using more processors for a given problem size.

Weak Scaling

How does the solution time scale with number of processors for *fixed* problem size per processor?

Goal: Do more tasks of size t in the *same* amount of time.

Compare with strong scaling which looks at speedup on a fixed workload.

Weak scaling is concerned with how much *work* gets done.

Custom Mesh

Drekar can handle a custom mesh.

```
<Parameter name="Source" type="string" value="Exodus File" />

<ParameterList name="Exodus File">
    <Parameter name="File Name" type="string" value="duct_mesh.gen" />
    <ParameterList name="Periodic BCs">
       <Parameter name="Count" type="int" value="1" />
       <Parameter name="Periodic Condition 1" type="string" value="y-coord 1e-8: surface_2;surface_1"/>
    </ParameterList>
</ParameterList>
```

Be sure to change the following:

- "right" to "surface_2" and "left" to "surface_1"
- "eblock-0_0_0" to "block_1"
- "origin" to "nodelist_1"
- "top" to "surface_4" and "bottom" to "surface_3"
- "front" to "surface_5" and "back" to surface_6"

Creating a Custom Mesh

There are programs that can generate a custom mesh (e.g. Cubit).

In this class, you will generate the mesh manually.

This is not efficient, but it is a common workflow.

First, put the following lines in your .bashrc file and source it:

1. export PATH=$HOME/Drekar/drekar-gcc-RELEASE/packages/seacas/applications/nem_spread:$PATH
2. export PATH=$HOME/Drekar/drekar-gcc-RELEASE/packages/seacas/applications/nem_slice:$PATH

Custom Mesh

- Modify the mesh.cpp script. Compile and run to generate a .p2e file.
- Update the mesh.inp file
  - input.inp describes each line in mesh.inp in detail
- Use the provided p2e executable
  - This was built on Odyssey for you to use
  - Run p2e by typing (./p2e)
  - You will be prompted for the mesh file (<mesh_file>.p2e) and the input file (<input_file>.inp)
    - Note: You need the extensions of both of these!
  - You will be prompted for an output file name
- You should now have a .exo file

Decomposing a Custom Mesh

- Copy your new .exo mesh file into a directory of your choosing.
- Rename your .exo mesh to a .gen mesh.
- Run slicerbase: ./slicerbase <num_procs> <meshfile>
  - Note: You may need to turn slicerbase into an executable
    - chmod +x slicerbase
  - Note: <meshfile> shouldn't have an extension. Slicerbase expects a .gen file to be found.
- Now you should have <num_procs> files representing your mesh on different parts of the domain.

Restarts

Drekar can start from a previous simulation result.

```
<Parameter name="Source" type="string" value="Exodus File" />

<ParameterList name="Exodus File">
    <Parameter name="File Name" type="string" value="duct_flow.exo"/>
    <Parameter name="Restart Index" type="int" value="30000"/>
    <ParameterList name="Periodic BCs">
       <Parameter name="Count" type="int" value="1" />
       <Parameter name="Periodic Condition 1" type="string" value="y-coord 1e-8: surface_2;surface_1"/>
    </ParameterList>
</ParameterList>
```

Again, make sure your boundary conditions, etc are consistent with the mesh.

Custom Initial Conditions

You can specify C code in your input file.

This feature can be used to define the initial conditions.

```
<ParameterList name="Initial Conditions">
    <ParameterList name="block_1">
        <ParameterList name="UX">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
        <ParameterList name="UY">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
        <ParameterList name="PRESSURE">
            <Parameter name="Value" type="double" value="0.0"/>
        </ParameterList>
        <ParameterList name="TEMPERATURE">
            <Parameter name="Type" type="string" value="RTC" />
            <Parameter name="Basis Field" type="string" value="TEMPERATURE"/>
            <Parameter name="Body" type="string" value='
            double pi = 4.0 * atan(1.0);
            TEMPERATURE = -(yin - 1.0) + 0.1 * cos(pi * xin) * cos(pi * (2.0 * yin - 1.0) / 2.0);
            ' />
        </ParameterList>
    </ParameterList>
</ParameterList>
<!-- End: Initial Conditions -->
```

Today's Goals

1. Run the 3D duct flow problem
    - Use different numbers of cores for different jobs
    - Run at different Reynolds numbers (keep them low!)
2. Combine the solution files into one .exo file
3. Visualize the results using Paraview on your local machine
4. Extract the fields in Python using netCDF
    - Plot the velocity profile and compare to analytical solution

Before Next Time

1. Generate a custom mesh
    1. Note: You can visualize the mesh with Paraview before running a job
2. Use that custom mesh for the duct flow problem
3. Try different initial conditions for the velocity field

Note: The mesh.cpp file provided uses a uniform mesh in z. You may want to use a non-uniform mesh in z similar to the one in y.