Arches Handoff* Instructions

Jeremy Thornock - June 3, 2020

*Using the InteriorFace boundary specification

Comments

- The process described here is utilized in “production Arches”.
- The use of a handoff is not required - one may use the Dirichlet condition for uniform boundary conditions.

Handoff File Format

Example

```
mixture_fraction
0.0966796875 0.0966796875
289
0 0 0 0.0011099316797418823
0 0 1 4.0749386606806856e-05
0 0 2 0.0
0 0 3 0.0
0 0 4 0.0
0 0 5 0.0
0 0 6 0.0
0 0 7 0.0
0 0 8 0.0
0 0 9 0.0
0 0 10 0.0
0 0 11 0.0
0 0 12 0.0
0 0 13 0.0
0 0 14 0.0
0 0 15 3.767877759372962e-05
0 0 16 0.0010993949168518651
0 1 0 4.074938660680576e-05
```

Format

```
Variable Name (for user reference)
Intended grid spacing
Number of points
i j k value
.  .  .  …
.  .  .  …
```

Note that the boundary condition is specified for a two dimensional plane. The third index (k in the example) is constant and ignored by the code.

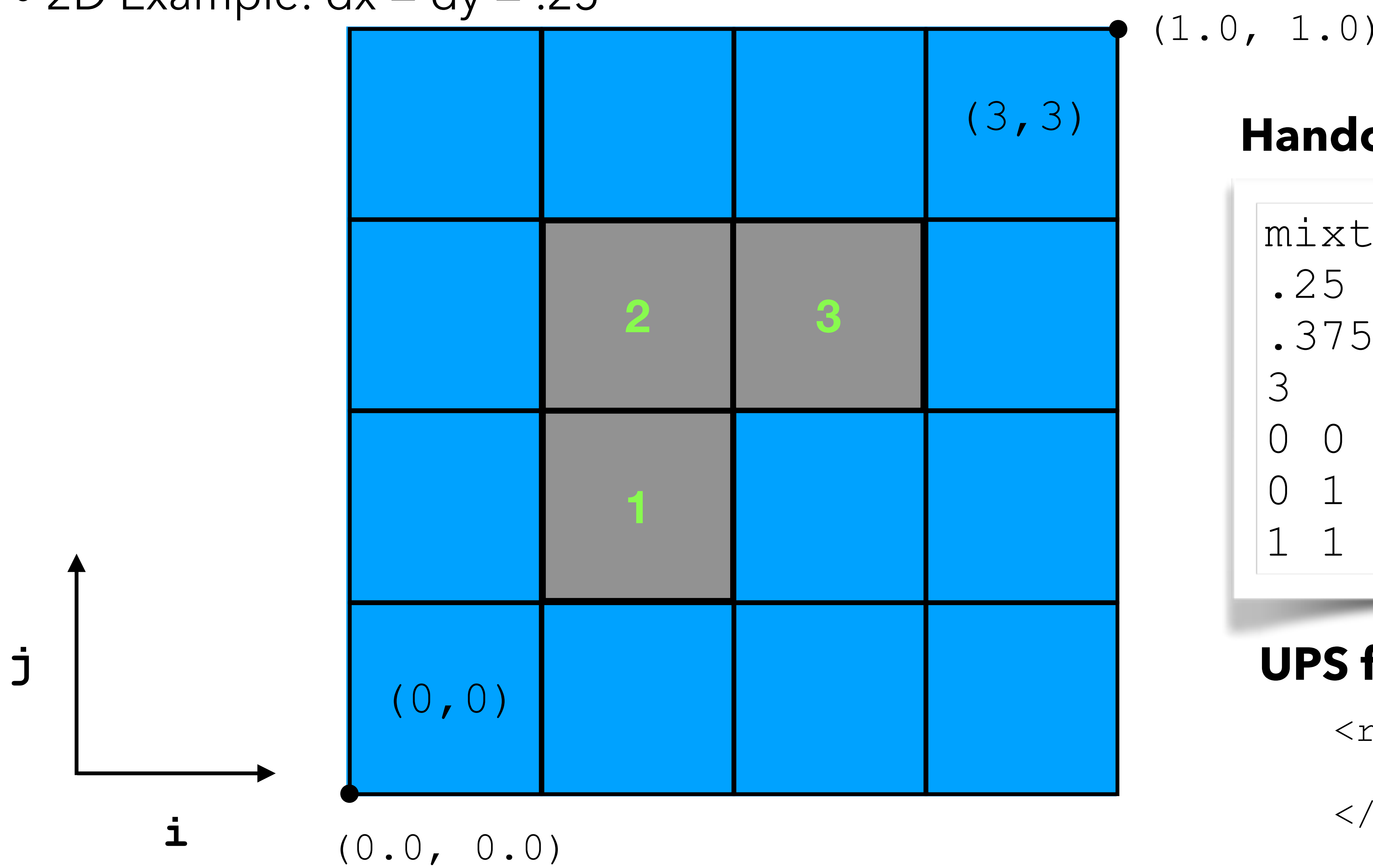
Units

The number in the handoff is a FLUX: $\frac{\phi}{t \cdot A}$

$\phi =$ units of the transported variable
$t =$ time (s)
$A =$ area (m^2)

Relative X,Y,Z

- The relative x,y,z location sets the position of the cell-centered (0,0,0) index and is entered in the input file.
- The same handoff file can be used multiple times in a single UPS file yet translate the starting relative index position.
- 2D Example: dx = dy = .25



(green values correspond to handoff file values)

Handoff file contents:

```
mixture_fraction
.25 .25
.375 .375 0
3
0 0 0 1
0 1 0 2
1 1 0 3
```

UPS file spec:

```
<relative_xyz>
  [.375, .375, 0]
</relative_xyz>
```

UPS Specification

(see arches_spec.xml for additional details)

- The handoff information provides a source to the transport RHS
- The location of the handoff is specified using the <InteriorFace> Uintah spec.
- A transport source term is created using information from the <InteriorFace> spec.
- The new source is attached to a transport equation.

<InteriorFace>

```
<BoundaryConditions>
  <DefaultMaterial>0</DefaultMaterial>
  <!-- Secondary, Co-Flow Inlet -->
  <InteriorFace rectangle="z-" lower="1 1 1" upper="2 2 1">
    <BCType label="myInjector" var="Custom">
      <value>1.</value>
    </BCType>
    <BCType label="mom_inject_x" var="Dirichlet">
      <value>1.</value>
    </BCType>
  </InteriorFace>
```

condition #1

condition #2

- var="Custom" for "myInjector" indicates use of a handoff file. (<value> required, but ignored)
- var="Dirichlet" for "mom_inject_x" is a simple Dirichlet condition (no handoff, <value> used for the flux.)

UPS Specification, con’t

<Sources>

```
<Sources>
  <src label="myInjector" type="cc_inject_src">
    <inputfile>inputs/Arches/handoff/input/handoff_f.dat</inputfile>
  </src>
  <src label="mom_inject_x" type="fx_inject_src">
  </src>
</Sources>
```

- The “label” needs to match the name specified in the <InteriorFace> specification.
- “myInjector” is a “Custom” BCType, so it must specify the location of the handoff file.
- “mom_inject_x” is a “Dirichlet” BCType, so no other spec is required. The <value> specification in the BCType will provide the value.
- Note the “type” specification. Options are cc_inject_src (cell-centered), fx_inject_src (face-x centered), fy_inject_src (face-y centered), fz_inject_src (face-z centered),

UPS Specification, con't

<Eqn>

```
<TransportEqns>
  <Eqn label="mixture_fraction" type="CCscalar">
    <doDiff>true</doDiff>
    <doConv>true</doConv>
    <conv_scheme>super_bee</conv_scheme>
    <determines_properties/>
    <initialization type="constant">
      <constant>.0</constant>
    </initialization>
    <Clipping>
      <low>0.0</low>
      <high>1.0</high>
    </Clipping>
    <src label="myInjector"/>
  </Eqn>
```

```
<MomentumSolver>
  <wall_closure type="constant_coefficient">
    <wall_csmag>0.4</wall_csmag>
  </wall_closure>
  <convection_scheme>central</convection_scheme>
  <src label="mom_inject_x"/>
</MomentumSolver>
```

- Attach the source terms to the appropriate transport equations using the label.
- Note that in the case of momentum the grid type is deduced automagically for you and only added to the corresponding momentum component (i.e., only y-momentum sources will be added to the y-momentum equation)

Generating a Handoff File from Arches

- Handoff files are generated *for* coarse simulations *from* fine resolution simulations
- Using **VisIt**, the general process is as follows:
  1) Run a fine resolution simulation
  2) Visualize the result of the fine simulation in VisIt
  3) Extract a plane of values from a pseudo-color plane plot using VisIt at the desired location using the Xmdv file format
  4) See StandAlone/inputs/Arches/handoff/genHandoffExample/README of an example of generating handoff files.
- Using **lineextract**, the general process is as follows:
  1) Run a fine resolution simulation
  2) Run lineextract on the plane of data you want to extract from the UDA
  3) Change the header of the lineextract output to have:
     Line 1: Name of the variable
     Line 2: Original grid spacing
     Line 3: Number of points
     Lines 4-N: i j k value
  4) See StandAlone/inputs/Arches/handoff/genHandoffExample/README of an example of generating handoff files from this point.

NOTES

- A python tool is provided to map fine data to coarse data in inputs/ARCHES/handoff/scripts/ArchesHandoffGenerator.py
- The tool, ArchesHandoffGenerator.py, may be extended to cover additional methods of mapping fine to coarse data - there may be multiple ways to to this (as there are in AMR methods)
- The example, make_handoff.py, is *one* way to do the mapping. Note the conservation choices made in the script. Other choices may be made regarding how to map fine to coarse data.

An Example

Run an example case (8-cores), extract the data with VisIt, coarsen the data to make handoff files, and finally run the coarse simulation with the handoff information. This problem simulates a methane fire using the RCCE model. The physics and the quality of the simulation itself is ignored. This example just demonstrates the procedure for creating and using a handoff file.

From the **`Uintah/src/Standalone`** directory:

- Execute `sus` using `inputs/ARCHES/handoff/genHandoffExample/methaneFire_fine.ups`
- Visualize with VisIt and select an x-cutting plane at coord (0.15,0,0) at the last output timestep and add a threshold to narrow in on the input with y =[.75, 2.25] and z=[.75, 2.25]
- Using the filename "fine", export [u,v,w]VelocitySPBC, mixture_fraction, mixture_fraction_2, enthalpy and heat_loss in Xmdv format, which should produce three files (4 x fine.00*.okc and 1 x fine.visit files). Make sure to export the coordinates with the data when prompted.
- Run the `inputs/ARCHES/handoff/scripts/okcProc.py` python script, passing the `fine.visit` file as the argument. This produces seven `*.dat` files corresponding to the five variables of interest. These files are the combined information across all patches.

An Example Continued

- Note that PDF files are created to aid in seeing the extracted variables are the extracted surface with all the combined information
- Now run `python inputs/ARCHES/handoff/genHandoffExample/make_handoff.py` which will read in the combined fine grid simulation information and make coarse_grid handoff files (refinement ration = 2). The `make_handoff.py` script is specific to this example. Currently, one would need to make her/his own script and deploy her/his desired strategy for conservation at the fine/coarse interface.
- When running the `make_handoff.py` script, contour plots are shown for each variable with the coarse data plotted as a contour and the fine data as a filled contour. This provides a sense of the how the interpolator is working.
- Finally, run `sus` on `inputs/ARCHES/handoff/genHandoffExample/methaneFire_coarse.ups` and visualize the output
- The two cases are stepping and producing output at the same frequency. Compare the two and see how the coarsening operation changes the results.