ENERGY FLAGSHIP
www.csiro.au

NUMBAT

High-resolution simulations of density-driven convective mixing

User manual

Christopher P. Green and Jonathan Ennis-King

Report Number ****
September 30, 2015

CSIRO Energy Flagship
71 Normanby Road, Clayton VIC, 3168, Australia
Private Bag 10, Clayton South VIC, 3169, Australia
Telephone: +61 3 9545 2777
Fax: +61 3 9545 8380

Copyright and disclaimer

Important disclaimer

Contents

1 Introduction

Numbat is a finite element application for solving the coupled Darcy and convection-diffusion equations for density-driven convective mixing in porous media. Numbat is built on the MOOSE Framework (www.mooseframework.com), and leverages multiple powerful features from this foundation. It features mesh adaptivity, high-order finite elements, is massively parallel, and uses a simple plain text input file.

2 Theory

The governing equations for density-driven flow in porous media are Darcy's law

$$\mathbf{u} = -\frac{\mathbf{K}}{\mu} \left(\nabla P - \rho(c) g \hat{\mathbf{z}} \right), \tag{2.1}$$

where $\mathbf{u}$ is the velocity vector, P is the fluid pressure, $\rho(c)$ is the fluid density as a function of solute concentration c, g is gravity, and $\hat{\mathbf{z}}$ is the unit vector in the z direction.

The fluid velocity must also satisfy the continuity equation

$$\nabla \cdot \mathbf{u} = 0, \tag{2.2}$$

and the solute concentration is governed by the convection - diffusion equation

$$\phi \frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \phi D \nabla^2 c, \tag{2.3}$$

where ϕ is the porosity, t is time and D is the diffusivity.

Darcy's law and the convection-diffusion equations are coupled through the fluid density, which is given by

$$\rho(c) = \rho_0 + \frac{c}{c_0} \Delta \rho, \tag{2.4}$$

where c_0 is the equilibrium concentration, and $\Delta \rho$ is the increase in density of the fluid at equilibrium concentration.

3 Installation

Install MOOSE

Numbat is a MOOSE application. In order to install Numbat, the MOOSE framework must first be installed. Detailed instructions are available at www.mooseframework.com/getting-started.

Clone Numbat

The next step is to clone Numbat from GitHub. In the following, it assumed that MOOSE was installed to the directory `~/projects`. If MOOSE was installed to a different directory, the following instructions must be modified accordingly.

To clone Numbat, use the following commands at the command line:

```
cd ~/projects
git clone https://github.com/cpgr/numbat.git
cd numbat
git checkout master
```

At this stage, there should be a `~/projects/numbat` directory.

Compile Numbat

Next, compile Numbat using

```
make -jn
```

where `n` is the number of processing cores on the computer. If everything has gone well, Numbat should compile without error, resulting in a binary named `numbat-opt`.

Test Numbat

Finally, to test that the installation worked, the small test suite can be run using

```
./run_tests -jn
```

where `n` is the number of processing cores on the computer. If everything has worked, the automatic tests should run and pass, and you are ready to use Numbat to undertake high-resolution simulations of density-driven convective mixing in porous media.

4 Input file syntax

The input file for a Numbat simulation is a simple hierarchical, block-structured plain text file identical to the MOOSE input file.

4.1 2D models

The main blocks required to implement a 2D simulation of density-driven convective mixing are now discussed.

Variables

For a 2D model, the simulation must have two variables: a *concentration* variable, and a *streamfunction* variable. These can be implemented in the input file using the following code:

```
[Variables]
  [./concentration]
  [../]
  [./streamfunction]
  [../]
[]
```

Kernels

The kernels block are where the physics of the problem are specified. Three individual kernels are required for a 2D model: a *DarcyDDC* kernel for the *streamfunction* variable, a *ConvectionDiffusionDDC* kernel for the *concentration* variable, and a *TimeDerivative* kernel also for the *concentration* variable. An example for an isotropic model is

```
[Kernels]
  [./TwoDDarcyDDC]
    type = DarcyDDC
    variable = streamfunction
    concentration_variable = concentration
  [../]
  [./TwoDConvectionDiffusionDDC]
    type = ConvectionDiffusionDDC
    variable = concentration
    streamfunction_variable = streamfunction
```

```
    coeff_tensor = '1 0 0 0 1 0 0 0 1'
  [../]
  [./TimeDerivative]
    type = TimeDerivative
    variable = concentration
  [../]
[]
```

AuxVariables

The velocity components in the x and y directions in a 2D model can be calculated using the auxiliary system. These velocity components are calculated using the *streamfunction*, see the mathematical model for details.

In the 2D case, two auxiliary variables, u and w, can be defined for the horizontal and vertical velocity components, respectively. Importantly, these auxiliary variables must have *constant monomial* shape functions (these are referred to as elemental variables, as the value is constant over each mesh element). This restriction is due to the gradient of the *streamfunction* variable being undefined for nodal auxiliary variables (for example, those using linear Lagrange shape functions). Auxilliary variables for the velocity components can be defined using

```
[AuxVariables]
  [./u]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./w]
    order = CONSTANT
    family = MONOMIAL
  [../]
[]
```

AuxKernels

The velocity components are calculated by *VelocityDDCAux* AuxKernels, one for each component. For the 2D case, the input syntax is

```
[AuxKernels]
  [./uAux]
    type = VelocityDDCAux
    variable = u
    component = x
    streamfunction_variable = streamfunction
  [../]
  [./wAux]
    type = VelocityDDCAux
    variable = w
    component = y
    streamfunction_variable = streamfunction
  [../]
[]
```

4.2 3D models

Variables

For a 3D model, three variables are required: one *concentration* variable and two *streamfunction* variables corresponding to the x and y components. This can be implemented in the input file using:

```
[Variables]
  [./concentration]
  [../]
  [./streamfunctionx]
  [../]
  [./streamfunctiony]
  [../]
[]
```

Kernels

Four individual kernels are required for a 3D model: a *DarcyDDC* kernel for each *streamfunction* variables, a *ConvectionDiffusionDDC* kernel for the *concentration* variable, and a *TimeDerivative* kernel also for the *concentration* variable. An example of the kernels block for a 3D isotropic model is

```
[Kernels]
  [./ThreeDDarcyDDCx]
    type = DarcyDDC
    variable = streamfunctionx
    concentration_variable = concentration
    component = x
  [../]
  [./ThreeDDarcyDDCy]
    type = DarcyDDC
    variable = streamfunctiony
    concentration_variable = concentration
    component = y
  [../]
  [./ThreeDConvectionDiffusionDDC]
    type = ConvectionDiffusionDDC
    variable = concentration
    streamfunction_variable = 'streamfunctionx streamfunctiony'
    coeff_tensor = '1 0 0 0 1 0 0 0 1'
  [../]
  [./TimeDerivative]
    type = TimeDerivative
    variable = concentration
  [../]
[]
```

In the 3D case, it is important to note that the *DarcyDDC* kernel must specify the component that it applies to, and that the *streamfunction_variable* keyword in the *ConvectionDiffusionDDC* kernel must contain both *streamfunction* variables ordered by the x component then the y component.

AuxVariables

For the 3D case, there is an additional horizontal velocity component (v), so the input syntax is

```
[AuxVariables]
```

```
  [./u]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./v]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./w]
    order = CONSTANT
    family = MONOMIAL
  [../]
[]
```

AuxKernels

For the 3D case, three *AuxKernels* are required. Note that both *streamfunction* variables must be given, in the correct order (x then y).

```
[AuxKernels]
  [./uAux]
    type = VelocityDDCAux
    variable = u
    component = x
    streamfunction_variable = 'streamfunctionx streamfunctiony'
  [../]
  [./vAux]
    type = VelocityDDCAux
    variable = v
    component = y
    streamfunction_variable = 'streamfunctionx streamfunctiony'
  [../]
  [./wAux]
    type = VelocityDDCAux
    variable = w
    component = z
    streamfunction_variable = 'streamfunctionx streamfunctiony'
  [../]
[]
```

CONTACT US

t 1300 363 400
+61 3 9545 2176
e enquiries@csiro.au
w www.csiro.au

YOUR CSIRO

Australia is founding its future on science and innovation. Its national science agency, CSIRO, is a powerhouse of ideas, technologies and skills for building prosperity, growth, health and sustainability. It serves governments, industries, business and communities across the nation.

FOR FURTHER INFORMATION

CSIRO Energy Flagship

Chris Green
t +61 3 9545 8371
e chris.green@csiro.au
w www.csiro.au