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 ****
October 1, 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

A simple illustration of the problem under consideration is presented in Figure 2.1 for a two-dimensional representation. In this case, we have a

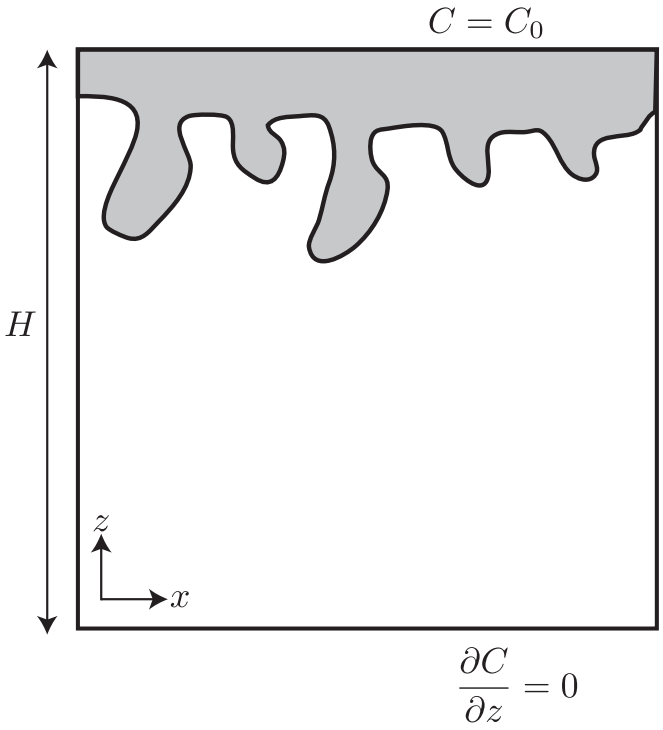

Figure 2.1: Schematic of 2D model. No-flow boundary conditions are imposed at the bottom boundary, and periodic boundary conditions are applied along the lateral boundaries. A constant concentration boundary condition is applied at the top boundary.

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

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

where $\mathbf{u} = (u, v, w)$ is the velocity vector, $\mathbf{K}$ is permeability, μ is the fluid viscosity, P is the fluid pressure, $\rho(c)$ is the fluid density as a function of solute concentration c, g is gravity, and $\hat{\mathbf{k}}$ 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.

The boundary conditions are

$$w = 0, \quad z = 0, -H, \tag{2.5}$$

$$\frac{\partial c}{\partial z} = 0, \quad z = -H, \tag{2.6}$$

$$c = c_0, \quad z = 0, \tag{2.7}$$

which correspond to impermeable boundary conditions at the top and bottom boundaries ($z = 0$ and $z = -H$, respectively), and a saturated condition at the top boundary.

Initially, there is no solute in the model

$$c = 0, \quad t = 0. \tag{2.8}$$

The solution of the governing equations differs in 2D and 3D. As a result, we shall consider the two cases separately.

2.1 2D model

If we consider an anisotropic model, with vertical and horizontal permeabilities given by k_z and k_x, respectively, we can non-dimensionalise the governing equations in 2D following Ennis-King and Paterson (2005). Defining the anisotropy ratio γ as

$$\gamma = \frac{k_z}{k_x}, \tag{2.9}$$

we scale the variables using

$$x = \frac{\phi\mu D}{k_z \Delta\rho g \gamma^{1/2}}\hat{x}, \quad z = \frac{\phi\mu D}{k_z\Delta\rho g}\hat{z}, \quad u = \frac{k_z\Delta\rho g}{\mu\gamma^{1/2}}\hat{u}, \quad w = \frac{k_z\Delta\rho g}{\mu}\hat{w}$$
$$t = \left(\frac{\phi\mu}{k_z\Delta\rho g}\right)^2 \hat{t}, \quad c = c_0\hat{c}, \quad P = \frac{\mu\phi D}{k_z}\hat{P}, \tag{2.10}$$

where $\hat{x}$ refers to a dimensionless variable. The governing equations in dimensionless form are then

$$\mathbf{u} = -\left(\nabla P + c\hat{\mathbf{k}}\right), \tag{2.11}$$

$$\mathbf{u} = 0, \tag{2.12}$$

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \gamma \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial z^2}, \tag{2.13}$$

where we have dropped the hat on the dimensionless variables for brevity.

The dimensionless boundary conditions are

$$w = 0, \quad z = 0, -Ra, \tag{2.14}$$

$$\frac{\partial c}{\partial z} = 0, \quad z = -Ra, \tag{2.15}$$

$$c = 1, \quad z = 0, \tag{2.16}$$

where Ra is the Rayleigh number, defined as

$$Ra = \frac{k_z \Delta \rho g H}{\phi \mu D}. \tag{2.17}$$

In this form, the Rayleigh number only appears in the boundary conditions as the location of the lower boundary. Therefore, Ra can be interpreted in this formalism as a dimensionless model height, and can be varied in simulations by simply changing the height of the mesh.

Finally, the dimensionless initial condition is

$$c = 0, \quad t = 0. \tag{2.18}$$

For isotropic models, where $k_x = k_z$ and hence $\gamma = 1$, we recover the dimensionless equations given by Slim (2014).

The coupled governing equations must be solved numerically. To simplify the numerical analysis, we introduce the streamfunction $\psi(x, z, t)$ such that

$$u = -\frac{\partial \psi}{\partial z}, \quad w = \frac{\partial \psi}{\partial x}. \tag{2.19}$$

This definition satisfies the continuity equation, Eq. (2.12), immediately.

The pressure P is removed from Eq. (2.11) by taking the curl of both sides and noting that $\nabla \times \nabla P = 0$ for any P, to give

$$\nabla^2 \psi = -\frac{\partial c}{\partial x}, \tag{2.20}$$

where we have introduced the streamfunction ψ using Eq. (2.19).

The convection-diffusion equation, Eq. (2.13) becomes

$$\frac{\partial c}{\partial t} - \frac{\partial \psi}{\partial z}\frac{\partial c}{\partial x} + \frac{\partial \psi}{\partial x}\frac{\partial c}{\partial z} = \gamma \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial z}. \tag{2.21}$$

The boundary conditions become

$$\frac{\partial \psi}{\partial x} = 0, \quad z = 0, -Ra, \tag{2.22}$$

$$\frac{\partial c}{\partial z} = 0, \quad z = -Ra, \tag{2.23}$$

$$c = 1, \quad z = 0, \tag{2.24}$$

while the initial condition is still given by Eq. (2.18).

In two dimensions, Numbat solves Eq's. (2.20) and (2.21) using the finite element method.

2.2 3D model

We now consider the case of a three-dimensional model. For simplicity, we consider the case where all lateral permeabilities are equal ($k_y = k_x$). The governing equations for the 3D model are identical to the 2D model. In dimensionless form, they are given by Eq's. (2.11) to (2.13), with boundary conditions given by Eq's. (2.14) to (2.16), and initial condition given by Eq. (2.18).

To solve these governing equations in 3D, a different approach must be used as the streamfunction ψ is not defined in three dimensions. Instead, we define a vector potential $\Psi = (\psi_x, \psi_y, \psi_z)$ such that

$$\mathbf{u} = \nabla \times \Psi. \tag{2.25}$$

It is important to note that the vector potential is only known up to the addition of the gradient of a scalar ζ as

$$\nabla \times (\Psi + \nabla\zeta) = \nabla \times \Psi \quad \forall\zeta, \tag{2.26}$$

as $\nabla \times \nabla\zeta = 0$ for any scalar ζ. This uncertainty is referred to as guage freedom, and is common in electrodynamics. Taking the curl of Eq. (2.11) and substituting Eq. (2.25), we have

$$\nabla(\nabla \cdot \Psi) - \nabla^2\Psi = \left(\frac{\partial c}{\partial y}, -\frac{\partial c}{\partial x}, 0\right), \tag{2.27}$$

where we have again used the fact that $\nabla \times \nabla P = 0$. If we choose $\nabla \cdot \Psi = 0$ to specify the guage condition, this simplifies to

$$\nabla^2\Psi = \left(-\frac{\partial c}{\partial y}, \frac{\partial c}{\partial x}, 0\right). \tag{2.28}$$

As shown in E and Liu (1997), $\nabla \cdot \Psi = 0$ is satisfied throughout the domain if

$$\begin{aligned} \psi_x = \psi_y = 0, \quad z = 0, -Ra, \\ \frac{\partial \psi_z}{\partial z} = 0, \quad z = 0, -Ra. \end{aligned} \tag{2.29}$$

The governing equations are then

$$\nabla^2\Psi = \left(-\frac{\partial c}{\partial y}, \frac{\partial c}{\partial x}, 0\right), \tag{2.30}$$

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \gamma\left(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}\right) + \frac{\partial^2 c}{\partial z^2}, \tag{2.31}$$

where the continuity is satisfied automatically because $\nabla \cdot (\nabla \times \Psi) = 0$ for any Ψ.

Finally, it is straightforward to show that $\psi_z = 0$ in order to satisfy $\nabla^2\psi_z = 0$ and $\frac{\partial \psi_z}{\partial z} = 0$, which means that the vector potential has only x and y components,

$$\Psi = (\psi_x, \psi_y, 0), \tag{2.32}$$

and therefore the fluid velocity $\mathbf{u} = (u, v, w)$ is

$$\mathbf{u} = \left(-\frac{\partial \psi_y}{\partial z}, \frac{\partial \psi_x}{\partial z}, \frac{\partial \psi_y}{\partial x} - \frac{\partial \psi_x}{\partial y}\right). \tag{2.33}$$

Note that if there is no y dependence, Eq's. (2.30) and (2.31) reduce to

$$\nabla^2\Psi = \left(0, \frac{\partial c}{\partial x}, 0\right), \tag{2.34}$$

$$\frac{\partial c}{\partial t} + \mathbf{u} \cdot \nabla c = \gamma\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial z^2}. \tag{2.35}$$

It is simple to show that $\nabla^2 \psi_x = 0$ and $\psi_x = 0$ at $z = 0, -Ra$ are only satisfied if $\psi_x = 0$ in the entire domain. In this case, the governing equations reduce to the two-dimensional formulation, as expected.

In three dimensions, Numbat solves Eq's. (2.30) and (2.31) using the finite element method.

3 Installation

Install MOOSE

Numbat is a MOOSE application. In order to run 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 *stream-function* 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'
  [../]
[]
```

PostProcessors

The flux across the top boundary is of significant interest. This can be calculated using a *SideFluxIntegral* postprocessor. This postprocessor integrates the concentration variable over the top boundary. Note that this postprocessor comes with the standard MOOSE distribution, and requires a diffusivity to be entered. As diffusivity has been scaled out of the problem, we can simple use a constant value of 1 in this postprocessor.

```
[Postprocessors]
  [./boundaryfluxint]
    type = SideFluxIntegral
    variable = concentration
    boundary = top
    diffusivity = 1
  [../]
[]
```

References

W. E and J.-G. Liu, *Finite difference methods for 3D viscous incompressible flows in the vorticity-vector potential formulation on nonstaggered grids*, J. Comp. Phys., 138, 57–82 (1997).

J. Ennis-King and L. Paterson, *Role of convective mixing in the long-term storage of carbon dioxide in deep saline aquifers*, SPE J., 10, 349–356 (2005).

A.C. Slim, *Solutal-convection regimes in a two-dimensional porous medium*, J. Fluid Mech., 741, 461–491 (2014).

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