porousIcoFoam - Documentation

Callum Bruce

October 20, 2021

Abstract

Documentation for porousIcoFoam, a transient, incompressible OpenFOAM solver for the laminar flow of Newtonian fluids including regions of clear fluid and porous media including:

- Details of the Darcy Brinkman Forchheimer governing equations.
- Details of the solution algorithm and finite volume discretization relevant to the implementation of porousIcoFoam.
- Calculation of the forces and moments acting on porous bodies.

Chapter 1

The Darcy Brinkman Forchheimer governing equations

The unsteady, incompressible Darcy Brinkman Forchheimer (DBF) continuity and momentum equations are given by:

$$\nabla \cdot \mathbf{u} = 0 \tag{1.1}$$

$$\frac{1}{\phi}\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{\phi^2}(\mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p^* + \frac{\nu}{\phi}\nabla^2 \mathbf{u} - \frac{\nu}{K}\mathbf{u} - \frac{c_F}{\sqrt{K}}|\mathbf{u}|\mathbf{u} \tag{1.2}$$

where $\phi = V_f/V$ is porosity, a dimensionless scalar field defined as the ratio of fluid volume, V_f to total volume, V inside a finite control volume. p^* is kinematic pressure $[m^2 s^{-2}]$. K is permeability $[m^2]$. The value of the coefficient K depends on the geometry of the medium. For the special case where the porous media is isotropic, K is a scalar field; for the more general case of anisotropic porous media, K is a tensor field. c_F is a dimensionless form-drag constant.

Equation 1.2 is a macroscopic momentum equation. It should be noted $\mathbf{u} = \phi \mathbf{u}_f$ is the "Darcy velocity" and similarly $p^* = \phi p_f^*$, where $\mathbf{u}_f$ and p_f^* are the volumetric average (macroscopic)[1] fluid velocity and kinematic pressure respectively. Hsu and Cheng [1] derive equation 1.2 by volume averaging of the microscopic Navier Stokes conservation equations (2.1, 2.2) over a representative volume.

[1]The volumetric average (macroscopic) fluid velocity and pressure are those averaged over V in contrast to the true local pore scale fluid velocity and pressure inside the porous media which the macroscopic momentum equation does not resolve.

Chapter 2

Solving the system of equations in OpenFOAM

2.1 icoFoam solver

Lets consider the unsteady, incompressible Navier Stokes continuity and momentum equations:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p^* + \nu \nabla^2 \mathbf{u} + S_i \tag{2.2}$$

where S_i represents source terms. All other symbols take their usual meaning. Hence we have four equations and four unknowns (u, v, w, p^*). There is no equation for pressure. This is the system of equations solved for in OpenFOAM for the unsteady, incompressible solver icoFoam.

To proceed, the momentum equation (2.2) is constructed in matrix form as:

$$\mathbf{MU} = -\nabla p^* \tag{2.3}$$

Considering the x components of the momentum equation; in expanded form we have n equations, one for each cell centroid:

$$\begin{pmatrix} M_{1,1} & M_{1,2} & M_{1,3} & \dots & M_{1,n} \\ M_{2,1} & M_{2,2} & M_{2,3} & \dots & M_{2,n} \\ M_{3,1} & M_{3,2} & M_{3,3} & \dots & M_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ M_{n,1} & M_{n,2} & M_{n,3} & \dots & M_{n,n} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_n \end{pmatrix} = \begin{pmatrix} (\partial p^*/\partial x)_1 \\ (\partial p^*/\partial x)_2 \\ (\partial p^*/\partial x)_3 \\ \vdots \\ (\partial p^*/\partial x)_n \end{pmatrix} \tag{2.4}$$

We require to solve for the unknown vector $\mathbf{U}$. Note that $\mathbf{U}$ is a vector containing $\mathbf{u}$ at all the cell centroids. The matrix of coefficients, $\mathbf{M}$, is known and populated using the finite volume method. Terms on the diagonal of $\mathbf{M}$ pertain to contributions by the owner cell while off diagonal terms contain contributions from neighboring cells.

To proceed, the matrix of coefficients, $\mathbf{M}$ is separated into diagonal components, $\mathbf{A}$ and off-diagonal components, $\mathbf{H}$. Equation 2.3 then takes the form:

$$\mathbf{AU} - \mathbf{H} = -\nabla p^* \tag{2.5}$$

The matrix $\mathbf{A}$ contains the diagonal components of $\mathbf{M}$:

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & 0 & 0 & \dots & 0 \\ 0 & A_{2,2} & 0 & \dots & 0 \\ 0 & 0 & A_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & A_{n,n} \end{pmatrix} \tag{2.6}$$

The matrix $\mathbf{A}$ is sometimes denoted a_p in the literature. $\mathbf{A}$ is easily inverted resulting in $\mathbf{A}^{-1}$:

$$\mathbf{A}^{-1} = \begin{pmatrix} 1/A_{1,1} & 0 & 0 & \dots & 0 \\ 0 & 1/A_{2,2} & 0 & \dots & 0 \\ 0 & 0 & 1/A_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1/A_{n,n} \end{pmatrix} \tag{2.7}$$

The matrix $\mathbf{A}^{-1}$ is sometimes denoted $1/a_p$ in the literature. The matrix $\mathbf{H}$ is evaluated explicitly using the velocity from the previous iteration. Hence it is known:

$$\mathbf{H} = \mathbf{A}\mathbf{U} - \mathbf{M}\mathbf{U} \tag{2.8}$$

We proceed with deriving a pressure equation. This is done by multiplying both sides of equation 2.5 by $\mathbf{A}^{-1}$ and rearrange for $\mathbf{U}$:

$$\mathbf{U} = \mathbf{A}^{-1}\mathbf{H} - \mathbf{A}^{-1}\nabla p^* \tag{2.9}$$

Substituting equation 2.9 into the continuity equation (2.1) and rearranging we arrive at a Poisson equation for pressure:

$$\nabla \cdot (\mathbf{A}^{-1}\nabla p^*) = \nabla \cdot (\mathbf{A}^{-1}\mathbf{H}) \tag{2.10}$$

where p^* is unknown. Equation 2.10 is used to couple pressure and velocity given $\mathbf{A}^{-1}$ and $\mathbf{H}$.

Notation for $\mathbf{M}$, $\mathbf{A}$ etc. used in `icoFoam.C` is documented in table 2.1.

Algebraic	C++
$\mathbf{M}$	`UEqu`
$\mathbf{A}$	`UEqu.A()`
$\mathbf{H}$	`UEqu.H()`
$\mathbf{A}^{-1}$	`rAU`
$\mathbf{A}^{-1}\mathbf{H}$	`HbyA`

Table 2.1: Notation used in `icoFoam.C`.

The volVectorField, `HbyA`, is also represented as a surfaceScalarField (i.e. the flux of `HbyA`) by interpolating onto cell faces and taking the dot product of the face normal vectors. This surfaceScalarField is denoted `phiHbyA` and is used in solving the pressure Poisson equation.

2.2 PISO algorithm

The icoFoam solver uses the PISO (Pressure-Implicit with Splitting of Operators) pressure-velocity coupling algorithm to solve the governing equations and iterate forward in time.

2.2.1 PISO algorithm steps

Each step of the algorithm is documented here with reference to the icoFoam source code.

Step 1: Solve the momentum predictor

Given the current pressure and velocity fields; solve the momentum equation (2.3) for **U**:

```
fvVectorMatrix UEqn
(
    fvm::ddt(U)               // Local acceleration term
    + fvm::div(phi, U)        // Convective acceleration term
    - fvm::laplacian(nu, U)   // Viscous term
);

if (piso.momentumPredictor())
{
    solve(UEqn == -fvc::grad(p));
}
```

This yields a velocity field which does not satisfy the continuity equation (2.1).

Step 2: Calculate $\mathbf{A}^{-1}$ and $\mathbf{A}^{-1}\mathbf{H}$ fields

Calculate the $\mathbf{A}^{-1}$ and $\mathbf{A}^{-1}\mathbf{H}$ fields from the current velocity field:

```
volScalarField rAU(1.0/UEqn.A());

volVectorField HbyA(constrainHbyA(rAU*UEqn.H(), U, p));
```

Step 3.1: Calculate `phiHbyA`

Calculate the flux of `HbyA` as the surfaceScalarField `phiHbyA`:

```
surfaceScalarField phiHbyA
(
    "phiHbyA",
    fvc::flux(HbyA)
    + fvc::interpolate(rAU)*fvc::ddtCorr(U, phi)
);
```

The second term in `phiHbyA` is the Rhie-Chow corrector [2]???

Step 3.2: Adjust boundary faces

Adjust the balance of fluxes on boundary faces (inlets and outlets) to obey continuity:

```
adjustPhi(phiHbyA, U, p);
```

Update the pressure on boundary faces where pressure flux is imposed:

```
constrainPressure(p, U, phiHbyA, rAU);
```

Step 4: Solve the pressure equation (non-orthogonal corrector loop)

Solve the Poisson equation (2.10) for the updated pressure field. Repeat for n non-orthogonal corrector loops:

```
fvScalarMatrix pEqn
(
    fvm::laplacian(rAU, p) == fvc::div(phiHbyA)
);

pEqn.setReference(pRefCell, pRefValue);

pEqn.solve();
```

On the final non-orthogonality correction, correct the flux using the most up-to-date pressure field:

```
if (piso.finalNonOrthogonalIter())
{
    phi = phiHbyA - pEqn.flux();
}
```

Step 5: Using updated pressure field, calculate flux-corrected velocity field

With the updated pressure field, calculate the corrected velocity field via equation 2.9:

```
U = HbyA - rAU*fvc::grad(p);
```

Impose boundary conditions on bounding faces:

```
U.correctBoundaryConditions();
```

The resulting velocity field now satisfies the continuity equation (2.1).

Step 6: Using flux-corrected velocity field repeat PISO corrector loop

Using the flux-corrected velocity field return to Step 2 and repeat Step 2 to Step 5 for n PISO corrector loops. This way the momentum predictor is solved once per iteration. Several PISO corrector inner loops are preformed per iteration. In practice, two PISO corrector loops are typically performed per iteration to arrive at a "converged" pressure field.

2.2.2 PISO algorithm functions

Details of PISO loop functions found in `icoFoam.C` are documented in table 2.2.

Function	Description
`constrainHbyA()`	Ensure equation 2.9 is satisfied on boundary faces.
`fvc::flux()`	Return surfaceScalarField representing cell face flux obtained from a given volVectorField.
`fvc::interpolate()`	Interpolate cell centered field onto cell faces.
`fvc::ddtCorr()`	Included as part of the correction to the Rhie-Chow interpolation [2]???
`adjustPhi()`	Adjust balance of fluxes on boundary faces (inlets and outlets) to obey continuity.
`constrainPressure()`	Update the pressure on boundary faces where a pressure flux is imposed.

Table 2.2: Table to test captions and labels.

2.3 porousIcoFoam solver

Following the same solution algorithm detailed in §2.1 and §2.2, a new solver has been implemented in OpenFOAM to solve the DBF governing equations (1.1, 1.2). This solver is a modified version of the built in icoFoam solver called porousIcoFoam. The main modification is in the definition of the `UEqu` in `porousIcoFoam.C` which becomes:

```
fvVectorMatrix UEqn
(
    (1.0/pty)*fvm::ddt(U)                  // Local acceleration term
    + (1.0/sqr(pty))*fvm::div(phi, U)      // Convective acceleration term
    - (1.0/pty)*fvm::laplacian(nu, U)      // Viscous term
    + fvm::SuSp(nu/K_, U)                  // Darcy term
    + fvm::SuSp((cf/sqrt(K_))*mag(U), U)   // Forchheimer term
);
```

where `pty` and `K_` are porosity (volScalarField) and permeability (volScalarField) respectively and `cf` is the dimensionless form-drag constant (dimensionedScalar). These new parameters are read into the solver via the `creatFields.H` header file and must be correctly defined in the `./0/*` and `./constant/transportProperties` dictionaries in case directory.

Source code for the porousIcoFoam solver can be downloaded from:
https://github.com/c-bruce/porousIcoFoam.

2.3.1 poroucIcoFoam case setup

2.3.2 poroucIcoFoam verification

Neale and Nadar [3] present an analytical solution for the velocity profile for flow within a channel bound above by a solid wall ($y = h$) and below by a porous medium ($0 > y > H$) for the Brinkman extended Darcy law (i.e. dropping the material derivative and Forchheimer terms in equation 1.2). For the channel flow problem considered here, the DBF momentum

equation takes the form:

$$\frac{\mu}{K}u + \frac{\mu}{\phi}\frac{d^2u}{dy^2} = \frac{dp}{dx} \tag{2.11}$$

The analytical solution is analogous to the solution of Poiseuille pipe flow. The no slip condition is applied on the upper bounding wall ($y = h$) in the clear fluid region and a slip condition is applied on the lower wall ($y = H$) of the porous media region. Full expressions for the analytical solution for the velocity profiles in the clear fluid and porous media regions are given in Neale and Nadar [3] but are omitted here for brevity.

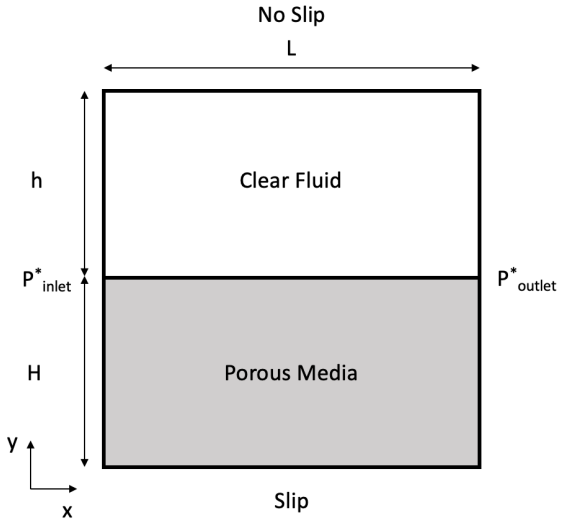


Figure 2.1: Verification domain.

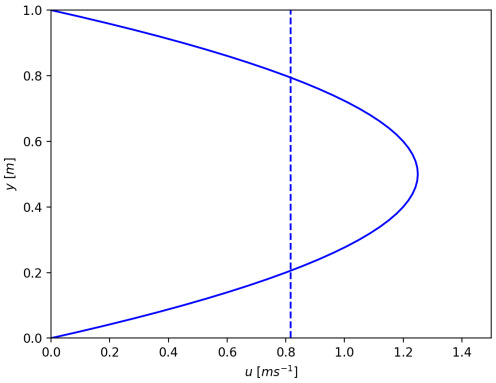
It is possible to solve the channel flow problem numerically applying a constant pressure gradient along the length of a two-dimensional domain and no slip/slip boundary conditions at the upper and lower bounds of the domain respectively. A simulation domain is set up with $h = H = 1\ m$, $L = 2\ m$ and $dP^*/dx = 1\ ms^{-2}$. Kinematic viscosity, ν, is equal to $0.1\ m^2s^{-1}$. Density, ρ is set to unity, hence $\mu \equiv \nu$ and $p \equiv p^*$. The numerical solution is solved for on an orthogonal, uniform 200×200 rectilinear grid with $0.01\ m$ cell size. A matrix of analytical and numerical solutions for the velocity profile in the channel are presented in figure 2.2 for $\phi = [0.75,\ 0.95]$ and $Da = [10^{-2},\ 10^{-4}]$ where $Da = K/l^2$ is the Darcy number, a dimensionless number describing porous media permeability. l is the characteristic length and is set equal to h. Results are presented in non-dimensional form with the characteristic velocity, U_∞, taken as the mean flow velocity in the clear fluid region when it is bound above and below by a solid wall with the no slip boundary condition (Poiseuille pipe flow, see Figure 2.2a). $U_\infty = 0.8167\ ms^{-1}$ and hence, the Reynolds number, $Re = 8.167$. Neale and Nadar [3] provide an expression for mass flow rate per unit width, M, expressed as M/M_0 where M_0 denotes the flow which would prevail in the clear fluid region if the lower wall were impermeable (Poiseuille pipe flow, see Figure 2.2a). Results for $M/M_0|_{analytical}$ are compared against those for $M/M_0|_{numerical}$ in table 2.3.

Varying Da has a greater impact on the velocity profile compared with varying ϕ. With decreasing Da the problem approximates Poiseuille pipe flow where there is negligible fluid flow in the porous media region. It is evident from the velocity profiles in figure 2.2 and

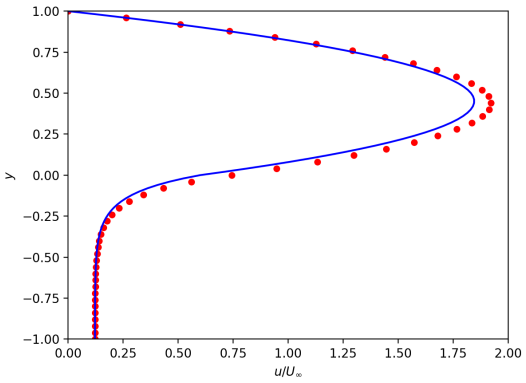
	$M/M_0\|_{analytical}$	$M/M_0\|_{numerical}$	Difference (%)
$\phi = 0.75$ $Da = 10^{-2}$	1.294319	1.346897	3.90
$\phi = 0.75$ $Da = 10^{-4}$	1.026353	1.033239	0.67
$\phi = 0.95$ $Da = 10^{-2}$	1.321106	1.317734	0.26
$\phi = 0.95$ $Da = 10^{-4}$	1.030367	1.029552	0.08

Table 2.3: Mass flow rate per unit width, M/M_0 results.

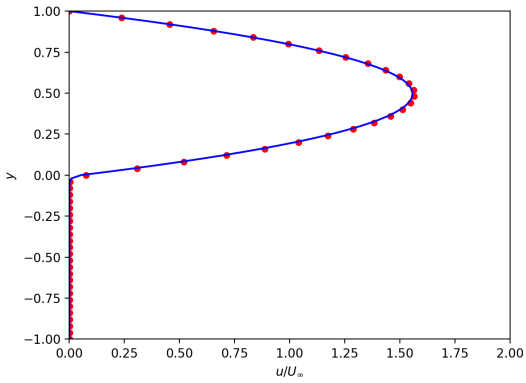
M/M_0 results in table 2.3 show that the numerical solutions do not match exactly the analytical solutions. It is posed that these discrepancies originate from the implementation of the convective term in equation 1.2. According to the Dupuit-Forchheimer relationship a factor of $1/\phi$ should be taken inside the gradient term to account for non-homogeneous porosity in the single domain model; neglecting to do so is a simplification which assumes negligible spatial variation in porosity. The largest discrepancy between analytical and numerical solutions being in the case of $\phi = 0.75,\ Da = 10^{-2}$ corroborates this. The effect of porosity variation is not required for a high-porosity medium, but it should be considered for a dense porous medium [4].



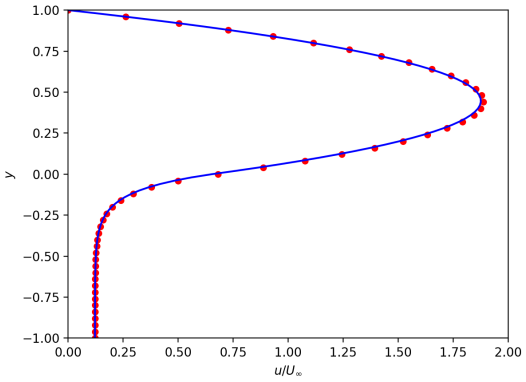
(a) Poiseuille pipe flow. Velocity profile, u(y) (solid blue line). Mean velocity, U_∞ (dashed blue line).



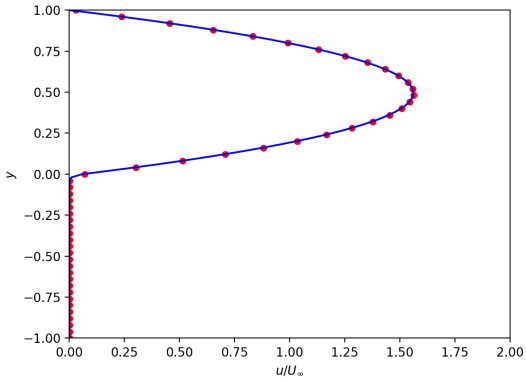
(b) $\phi = 0.75$ $Da = 10^{-2}$



(c) $\phi = 0.75$ $Da = 10^{-4}$



(d) $\phi = 0.95$ $Da = 10^{-2}$



(e) $\phi = 0.95$ $Da = 10^{-4}$

Figure 2.2: porousIcoFoam verification cases. Analytical velocity profile (solid blue line), numerical velocity profile (red dots) for cases b), c), d) and e).

Chapter 3

Forces and moments acting on porous bodies

3.1 Conventional forces and moments calculation

The conventional forces and moments calculation considers the various contributions to the total forces and moments acting on a porous body as described by the DBF momentum equation (1.2). These include pressure, viscous and porous (Darcy and Forchheimer) contributions.

$$\mathbf{F}_t = \mathbf{F}_p + \mathbf{F}_v + \mathbf{F}_d + \mathbf{F}_f \tag{3.1}$$

3.1.1 Pressure contribution

The normal pressure force contribution is calculated as a surface integral at the clear fluid - porous media interface. This is written in discrete form as:

$$d\mathbf{F}_p|_i = \rho \mathbf{A}_i (p_i^* - p_{ref}^*) \tag{3.2}$$

$$\mathbf{F}_p = \sum_i d\mathbf{F}_p|_i \tag{3.3}$$

where $\mathbf{A}_i$ is the i^{th} boundary face area vector. The normal pressure moment contribution is given, in discrete form, as:

$$d\mathbf{M}_p|_i = \mathbf{r}_i \times d\mathbf{F}_p|_i \tag{3.4}$$

$$\mathbf{M}_p = \sum_i d\mathbf{M}_p|_i \tag{3.5}$$

where $\mathbf{r}_i$ is the i^{th} boundary face position vector.

3.1.2 Viscous contribution

The tangential force contribution is calculated as a surface integral at the clear fluid - porous media interface. This is written in discrete form as:

$$d\mathbf{F}_v|_i = \mathbf{A}_i \cdot \boldsymbol{\tau}_i \tag{3.6}$$

$$\mathbf{F}_v = \sum_i d\mathbf{F}_v|_i \tag{3.7}$$

where $\boldsymbol{\tau}_i$ is the viscous stress tensor on the boundary face. The tangential viscous moment contribution is given, in discrete form, as:

$$d\mathbf{M}_v|_i = \mathbf{r}_i \times d\mathbf{F}_v|_i \tag{3.8}$$

$$\mathbf{M}_v = \sum_i d\mathbf{M}_v|_i \tag{3.9}$$

3.1.3 Porous (Darcy) contribution

The Darcy force contribution is calculated as a volume integral. Considering the third term on the RHS of equation 1.2 the Darcy force contribution is given, in discrete form, as:

$$d\mathbf{F}_d|_i = \rho \mathbf{V}_i (\frac{\nu}{K}\mathbf{u}) \tag{3.10}$$

$$\mathbf{F}_d = \sum_i d\mathbf{F}_d|_i \tag{3.11}$$

where $\mathbf{V}_i$ is the i^{th} volume element (inside the porous media). The Darcy moment contribution is calculated, in discrete form, as:

$$d\mathbf{M}_d|_i = \mathbf{r}_i \times d\mathbf{F}_d|_i \tag{3.12}$$

$$\mathbf{M}_d = \sum_i d\mathbf{M}_d|_i \tag{3.13}$$

3.1.4 Porous (Forchheimer) contribution

The Forchheimer force contribution is calculated as a volume integral. Considering the forth term on the RHS of equation 1.2 the Forchheimer force contribution is given, in discrete form, as:

$$d\mathbf{F}_f|_i = \rho \mathbf{V}_i (\frac{c_F}{\sqrt{K}}|\mathbf{u}|\mathbf{u}) \tag{3.14}$$

$$\mathbf{F}_f = \sum_i d\mathbf{F}_f|_i \tag{3.15}$$

The Forchheimer moment contribution is calculated, in discrete form, as:

$$d\mathbf{M}_f|_i = \mathbf{r}_i \times d\mathbf{F}_f|_i \tag{3.16}$$

$$\mathbf{M}_f = \sum_i d\mathbf{M}_f|_i \tag{3.17}$$

3.1.5 forcesConventional functionObject

An OpenFOAM functionObject, forcesConventional, has been written to calculate the forces and moments acting on a porous body. This functionObject is intended for use with the porousIcoFoam solver. Pressure, viscous and porous (Darcy and Forchheimer) force contributions are calculated using the definitions given in §3.1.1, 3.1.2, 3.1.3, 3.1.4.

To use the forcesConventional functionObject the following dictionary must be added to `./system/controlDict/functions`:

```
forces_conventional
{
// Mandatory entries
type forcesConventional;
libs ("libmyForcesConventionalFunctionObject.so");
// Patches
patches (interface1);
// rho
rho 1.0;

// Optional entries
// Field names
p p;
U U;
K_ K_;
// pRef
pRef 0.0;
// Calculate for a porous body?
porosity true;
porousZone porousMedia;
}
```

Properties in the forcesConventional dictionary entry are documented in table 3.1.

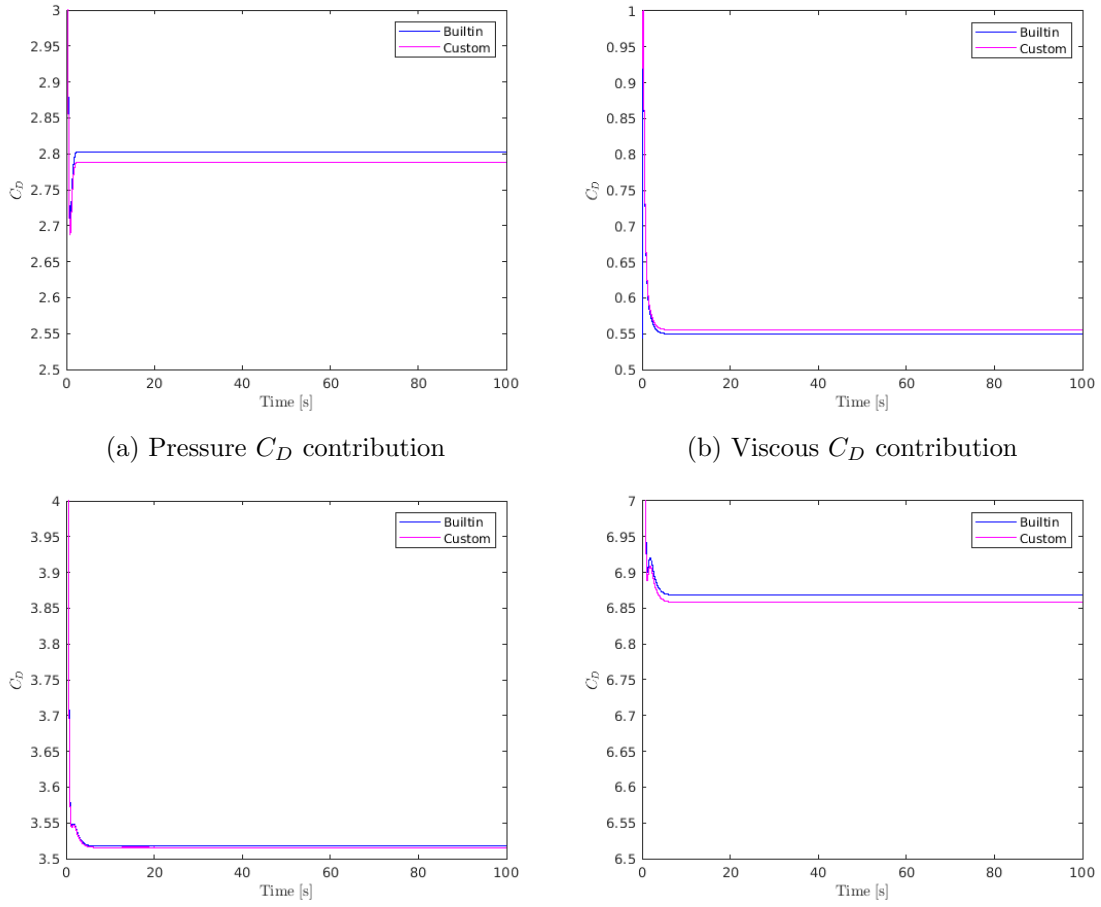
Property	Type	Description
`patches`	List	List of patch names used to calculate pressure and viscous contributions - patch surface normals must point inside porous body
`rho`	Float	Density value to use in forces calculation
`p`	Name	Pressure field (volScalarField) name
`U`	Name	Velocity field (volVectorField) name
`K_`	Name	Permeability field (volScalarField) name
`pRef`	Float	reference pressure to use in forces calculation
`porosity`	Bool	Include porous force contributions true/false (default is false)
`porousZone`	Name	Name of the cellZone used to calculate porous contributions

Table 3.1: forcesConventional functionObject properties

Source code for the forcesConventional functionObject can be downloaded from: https://github.com/c-bruce/forcesConventional.

To verify the implementation of the forcesConventional functionObject a two-dimensional porous square cylinder case was set up and run using: a) pisoFoam solver using laminar turbulence model (see pisoFoam) with OpenFOAM's built-in Darcy-Forchheimer porous media model (see DarcyForchheimer) and built-in forces functionObject; and b) custom porousIcoFoam solver and custom forcesConventional functionObject. These approaches will herein be refered to as "built-in" and "custom" respectively. The aim of this verification case is to demonstrate that the forcesConventional function object has been correctly

implemented. Note that in the built-in approach the porous media model only modifies the Navier Stokes equations by adding sink terms to the RHS of the governing equation (via fvOptions in OpenFOAM) and does not include the porosity terms that appear in equation 1.2 which modify the local acceleration, convective and viscous terms in the equation. Hence to match this set up in the custom approach porosity is set to unity everywhere. The porous square cylinder is centered on the origin and has edge length, $L = 1\ m$. A fixed velocity inlet condition ($u_\infty = 1\ ms^{-1}$) is applied $2L$ upstream of the origin, a fixed pressure outlet condition ($p^* = 0\ Pakg^{-1}m^3$) is applied $6L$ downstream of the origin and, slip boundary conditions are applied at $\pm 2L$ spanwise. Kinematic viscosity, ν, is set to $0.033333\ m^2s^{-1}$. This corresponds to a Reynolds number, $Re = 30$. Permeability, K, is set to $1 \times 10^{-4}\ m^2$. The Darcy number, Da is defined as $Da = K/L^2$ therefore, $Da = 1 \times 10^{-4}$. The form drag constant, c_F, is set to zero neglecting the Forchheimer term.



(a) Pressure C_D contribution

(b) Viscous C_D contribution

(c) Porous (Darcy) C_D contribution

(d) Total C_D

Figure 3.1: Contributions to total C_D for the two-dimensional porous square cylinder verification case calculated using the built-in and custom approaches.

Figure 3.1 shows the development, over time, of individual force contributions and total force displayed in non-dimensional form as the drag coefficient, C_D, for the built-in and custom approaches described above. The solution becomes steady after $\sim 10s$. Table 3.2 shows the steady state values for the built-in and custom approaches. Pressure force con-

C_D Contribution	Built-in	Custom	Difference (%)
Pressure	2.8021	2.7880	-0.50
Viscous	0.5495	0.5553	1.06
Porous (Darcy)	3.5176	3.5158	-0.05
Total	6.8692	6.8591	-0.15

Table 3.2: forcesConventional verification results.

tributes $\sim 41\%$, viscous force $\sim 8\%$ and porous force $\sim 51\%$ to the total drag force. Results in figure 3.1 and table 3.2 show that forces calculated using the built-in and custom approaches do not agree exactly. The largest percentage difference appears in the viscous contribution ($\sim 1\%$) while the pressure (0.5%) and porous (0.05%) contributions display less significant percentage differences. The overall percentage difference between the two approaches for the total value of C_D is 0.15%. The two approaches solve the govenring equations in distinct ways: in the custom approach the porous terms are included directly in the porousIcoFoam `UEqu` while the built-in approach solves the Navier Stokes equations and includes the porous terms at run time using fvOptions. The size of the difference in results is small considering the differences in approaches outlined above and gives confidence that the forcesConventional functionObject has been correctly implemented.

3.2 Vorticity-moment theorem (impulse theory) forces and moments calculation

An alternative method for calculating the forces and moments acting on a body is to use the vorticity-moment theorem, or impulse theory. This method has it's origins in the general formulas relating aerodynamic forces and moments to rates of change of vorticity moments originally presented by Wu [5]. The aerodynamic force acting on a solid body immersed in a fluid region extending to infinity, R_f, with a reference frame fixed with the body, is given by:

$$\mathbf{F} = -\frac{\rho}{2}\frac{d}{dt}\iiint_{R_f} \mathbf{r} \times \boldsymbol{\omega} dR \tag{3.18}$$

The aerodynamic moment acting on a solid body immersed in R_f, with a reference frame fixed with the body, is given by:

$$\mathbf{M} = \frac{\rho}{2}\frac{d}{dt}\iiint_{R_f} |\mathbf{r}|^2 \boldsymbol{\omega} dR \tag{3.19}$$

Equations 3.18 and 3.19 are taken from Elements of Vorticity Aerodynamics [6] which discusses in detail the vorticity-moment theorem and presents versions of 3.18 and 3.19 where the body is moving and rotating relative to the fluid.

3.2.1 forcesImpulseTheory functionObject

Bibliography

[1] C. Hsu and P. Cheng, "Thermal dispersion in a porous medium," *International Journal of Heat and Mass Transfer*, vol. 33, pp. 1587–1597, aug 1990.

[2] C. M. Rhie, "Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation," vol. 21, no. 11, pp. 1525–1532, 1983.

[3] G. Neale and W. Nader, "Practical significance of brinkman's extension of darcy's law: Coupled parallel flows within a channel and a bounding porous medium," *The Canadian Journal of Chemical Engineering*, vol. 52, no. 4, pp. 475–478, 1974.

[4] K. Vafai and S. J. Kim, "On the limitations of the Brinkman-Forchheimer-extended Darcy equation," *International Journal of Heat and Fluid Flow*, vol. 16, no. 1, pp. 11–15, 1995.

[5] J. C. Wu, "Theory for Aerodynamic Force and Moment in Viscous Flows," vol. 19, no. 4, pp. 432–441, 1981.

[6] J. C. Wu, *Elements of Vorticity Aerodynamics*. 2018.