PIFISS Potential (Incompressible) Flow Iterative Solution Software guide

Silvester, David J. and Powell, Catherine E.

2007

MIMS EPrint: **2007.14**

ISSN 1749-9097

PIFISS
Potential (Incompressible) Flow & Iterative Solution Software Guide [*]

David J. Silvester
School of Mathematics, University of Manchester
`d.silvester@manchester.ac.uk`.

Catherine E. Powell
School of Mathematics, University of Manchester
`c.powell@manchester.ac.uk`.

Version 1.0, released 7 February 2007

Contents

[*] This software was developed with support from the UK Engineering and Physical Sciences Research Council and the British Council under the British–German Academic Research Collaboration (ARC) scheme.

1. Background

The acronym PIFISS stands for Potential Incompressible Flow and Iterative Solution Software. The contents and scope of the package are outlined in the following sections and also illustrated with examples. PIFISS supplements the previously released IFISS software package that is described in

- Howard Elman, Alison Ramage & David Silvester, *IFISS: a Matlab toolbox for modelling incompressible flow,* MIMS Eprint 2005.45, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, December 2005,
`http://www.manchester.ac.uk/ifiss/`

and which is associated with the reference book [ESW, 2005],

- Howard Elman, David Silvester & Andrew Wathen, *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*, Oxford University Press, 2005.

Broadly speaking, PIFISS extends the functionality of IFISS to steady-state diffusion problems with discontinuous and anisotropic coefficients, and to potential flow problems (the so-called mixed formulation of steady-state diffusion problems) that arise in modelling groundwater flow. The package offers users the possibility to investigate different iterative solution techniques for the symmetric positive definite problems that arise in the first case and for the saddle-point systems that arise in the second. In particular, we include state of the art preconditioning techniques based on AMG (algebraic multigrid). The preconditioners for the saddle-point systems were developed and discussed in the following references,

- C.E. Powell, *Optimal Preconditioning for Mixed Finite Element Formulations of Second-Order Elliptic Problems*, PhD thesis, UMIST, 2003.
- C.E. Powell & D.Silvester, *Optimal Preconditioning for Raviart-Thomas Mixed Formulation of Second-Order Elliptic Problems*, SIAM J. Matrix Anal. Appl., **25**, pp. 718–738, 2003.

The AMG code that is included in the toolbox was written by J. Boyle at the University of Manchester as part of the EPSRC project EP/C000528/1.

The PIFISS software is "open-source" and is written in MATLAB. To run it you also require access to the functionality of FEMLAB or its successor, COMSOL multiphysics.[1] You can redistribute PIFISS and/or modify it under the terms of the GNU Lesser General Public License as published by the Free

[1] This software is marketed by COMSOL, see `http://www.comsol.com`.

Software Foundation; either version 2.1 of the License, or any later version—for precise details, see the file `readme.m`. The software is distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU Lesser General Public License for a definitive statement.

The software can be run under Windows, Unix and Mac architectures. The current version, PIFISS 1.0, was developed using the syntax of MATLAB 7.0 (Release 13), and FEMLAB 3.1[2] but it has also been extensively tested using MATLAB 7.1 and 7.2 (Release 14) and COMSOL 3.3. (If you are not sure which version of MATLAB or COMSOL you have running, just type `ver` at the system prompt.)

2. Installation

PIFISS is downloaded as the zipped `tar` file `pifiss.tar.gz`, which contains all the individual files bundled together. After unpacking the `tar` file two steps are required to set the package up:

1. The file `gohome.m` identifies the 'home' directory of the package via a command of the form

   ```
   cd('<local directory information>/pifiss')
   ```

   (where the name `pifiss` has the appropriate version number appended). You must edit this file to point to the user's local directory. The simplest way to do this is to type `pwd` in the MATLAB command window when PIFISS is the current directory and copy and paste the returned path name into the file `gohome.m`

2. After starting "FEMLAB–with–MATLAB" or, (depending on the version available) "COMSOL–with–MATLAB", the user must run one of the functions `install_unix` or `install_pc` in the PIFISS home directory. These commands initialize all Unix-dependent and Windows-dependent files, respectively. After this has been done, PIFISS is set to run in a Unix or Windows environment without additional user intervention.

Once PIFISS is installed (by performing steps 1 and 2 above), for all subsequent uses, the MATLAB path must include the PIFISS home directory. This requirement can be enforced by simply typing `setpath` in response to the MATLAB prompt each time you begin a new PIFISS session. Having run `setpath` at the MATLAB prompt, simply type `helpme` to get started.

[2] Note that because the default mesh generator that is built into COMSOL's software is version dependent, slightly different solutions and iteration counts are to be expected when running the test problems described below using later releases of the COMSOL software.

3. A Model Steady-State Diffusion Problem

The focus of the PIFISS package is on efficiently solving two-dimensional steady-state diffusion problems, with discontinuous and/or anisotropic coefficients using the so-called primal formulation (see below); or the mixed formulation (see section 4).

The first type of problem that is addressed is: compute a scalar potential p satisfying

$$-\nabla \cdot \mathcal{A}\nabla p = f,$$

at every point $\vec{x}$ in a two-dimensional region Ω, where $f : \Omega \to \mathbb{R}$ is a given "source" function and $\mathcal{A}$ is a symmetric positive definite 2×2 matrix whose coefficients are also given functions of $\vec{x}$. When supplemented with suitable boundary conditions, for example, if p is specified at every point on the boundary $\partial\Omega$, then we have a well posed problem and a unique solution.

This PDE problem arises in many applications in science and engineering, including electromagnetic problems and in fluid flow modelling. The coefficient term $\mathcal{A}$ is a material parameter, specific to the problem at hand, and usually represents a 'permeability' or 'permittivity' of the medium being considered. For the types of application we have in mind, this parameter can exhibit discontinuities and/or anisotropies within Ω.

Having specified a region Ω, a discretized problem is assembled by PIFISS using linear finite element approximation on a triangular subdivision (mesh) that is generated using FEMLAB. For example, suppose we wish to solve the steady-state diffusion problem on a circular disk with unit radius, with three distinct regions of permeability. The coarse mesh that is "automatically generated" using PIFISS is that illustrated in Figure 3.1. Different values of $\mathcal{A}$ are then easily assigned to each subdomain, which are individually meshed.

Our primary interest is in developing fast iterative solvers for the linear equation systems that arise from the finite element discretization. To facilitate this development a set of five predefined reference problems is built into PIFISS (see the Appendix for complete details of each one). Running the driver `diff_testproblem` enables the user to choose the desired test problem and the degree of refinement of the spatial discretization. In addition, the resulting linear system is solved using the MATLAB "backslash" solver and an estimate of the energy error is computed using the residual error estimator described in Section 1.5.2 of [ESW, 2005].

To illustrate this process, a sample PIFISS session showing the solution of one of these problems (**D5**, see the Appendix) is reproduced below. The mesh that is used in this case is a uniform refinement of the subdivision in Figure 3.1 (it is generated by subdividing every triangle into four) and the computed finite element solution is shown in Figure 3.2. To reproduce the session, just type `diff_testproblem` and select the options shown.

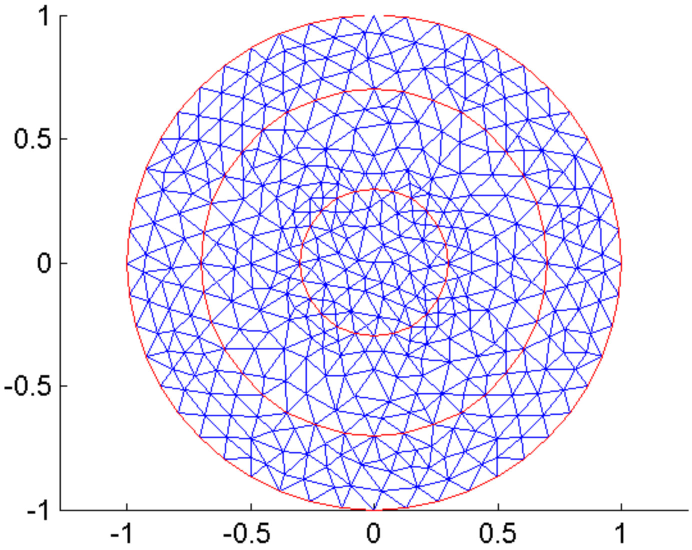


Figure 3.1. FEMLAB *mesh associated with reference problem* ***D5****.*

```
>> diff_testproblem
specification of reference diffusion problem.
choose specific example
     1  Square domain, unit coefficients
     2  Square domain, anisotropic coefficients
     3  Square domain, discontinuous anisotropic coefficients
     4  punched ticket domain, unit coefficients
     5  circle domain, discontinuous coefficients, analytic solution
 : 5
number of refinements? (default is 2) : 1
setting up P1 diffusion matrices...done
system saved in femlab_adiff.mat ...
solving linear system ...  done
Galerkin system solved in  4.822e-002 seconds
plotting solution... done
computing P1 variable diffusion flux jumps...  done
computing P1 interior residuals...  done
computing residual error estimator...
estimated global error (in energy): 6.248754e-002
```

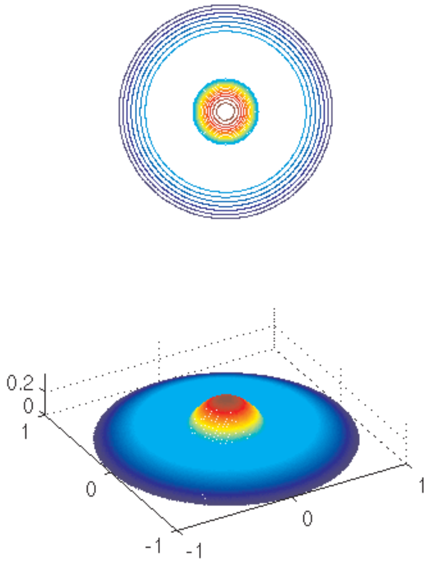


Figure 3.2. *Solution to the variable diffusion problem* ***D5***.

Once a discrete problem has been set up and solved in this way, the performance of alternative preconditioned iterative solvers may be explored using the drivers `it_solve` and `amg_solve`. Issuing the first command allows the user to re-solve the system, iteratively, using Krylov subspace methods and choose from a set of 'standard' preconditioners. Issuing the second command allows the user to re-solve the system using an algebraic multigrid preconditioner.

In the following session, the chosen iterative method is MINRES and the preconditioner is the zero fill-in incomplete Cholesky routine built into MATLAB. This solution method converges in 59 iterations, and gives the convergence curve represented by the blue line in Figure 3.3. (Using an incomplete factorization with the default drop tolerance gives much faster convergence!)

```
>> it_solve
discrete diffusion system ...
PCG/MINRES? 1/2 (default PCG) : 2
tolerance? (default 1e-6) :
maximum number of iterations? (default 100) :
preconditioner:
   0  none
   1  diagonal
   2  incomplete cholesky (no fill in)
```

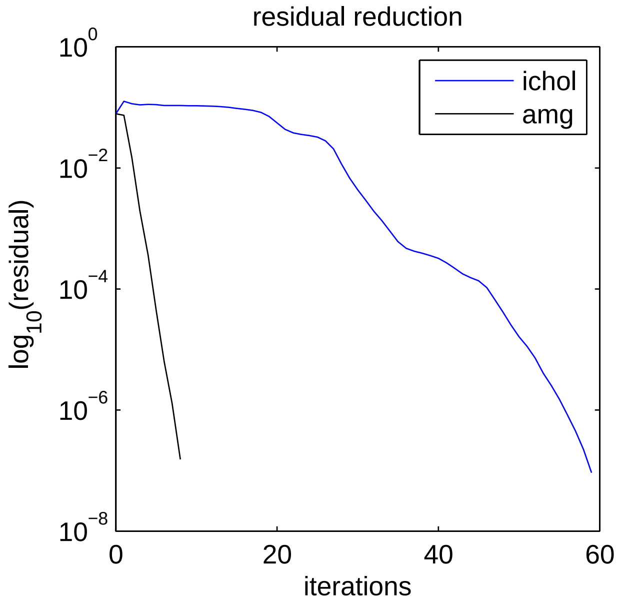


Figure 3.3. *Convergence curves using preconditioned* MINRES.

```
   3  incomplete cholesky with drop tolerance
default is standard incomplete cholesky :
MINRES iteration ...
convergence in  59 iterations
    k  log10(||r_k||/||r_0||)
    0           0.0000
    1           0.2041
    2           0.1633
    3           0.1437
                   .
                   .
   57          -5.2465
   58          -5.5512
   59          -5.9321
Bingo!

  4.4401e-001 seconds

use new (enter figno) or existing (0) figure, default is 0 : 11
```

```
colour (b,g,r,c,m,y,k): enter 1--7 (default 1) :
```

The second (black) convergence curve in Figure 3.3 can be generated by solving the problem a second time, using amg_solve.

```
>> amg_solve
discrete diffusion system... PCG/MINRES? 1/2 (default PCG) : 2
tolerance? (default 1e-6) :
maximum number of iterations? (default 100) :
compute / load AMG data? 1/2 (default 1) :

Level 2 step 1 coarsening started
 C points after 1st pass: 524
 C points after 2nd pass: 714
 C points after coarsening: 714
Weak positive connections exist
Strong positive connections exist
Coarsening completed (714 Coarse, 927 Fine and 96 Unconnected
points)
 ...
Level 9 step 1 coarsening started
 C points after 1st pass: 1
 C points after 2nd pass: 1
 C points after coarsening: 1

 Coarsening completed (1 Coarse, 1 Fine and 0 Unconnected points)
 setup done.
 plot AMG grid sequence? yes/no 1/2 (default no) :
 PDJ/PGS smoother? 1/2 (point Gauss-Seidel) : 1
 point damped Jacobi smoothing ..
 MINRES iteration... convergence in   8 iterations
    k  log10(||r_k||/||r_0||)
    0           0.0000
    1          -0.0262
    2          -0.7292
    3          -1.6080
    4          -2.3378
    5          -3.2432
    6          -4.0928
    7          -4.7913
    8          -5.7120
Bingo!
      2.3632e-001 seconds
```

4. Potential Flow Problems (Groundwater flow)

The problem addressed in Section 3 is often referred to as a *primal problem*. We can generate a *mixed formulation* of the same problem by explicitly introducing the vector $\vec{u} = -\mathcal{A}\nabla p$. This gives the reformulated problem: find the pair $(\vec{u}, p)$ satisfying

$$\begin{aligned} \mathcal{A}^{-1}\vec{u} + \nabla p &= \vec{0}\,; \\ \nabla \cdot \vec{u} &= f\,. \end{aligned} \tag{1}$$

This problem represents the fundamental model of potential flow. The variable $\vec{u}$ is a vector-valued function representing the velocity of the fluid, and the scalar function p represents the pressure (or more precisely 'head'). The first equation above is called Darcy's law and the second equation enforces conservation of mass. Note that in groundwater flow modelling, it is $\vec{u}$, and not the scalar variable p, that is of primary interest. The mixed formulation is the preferred formulation in this case as it is more robust when the permeability coefficient $\mathcal{A}$ is discontinuous.

The boundary value problem considered is (1) posed on a two-dimensional region Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

$$p = g \text{ on } \partial\Omega_D, \quad \frac{\partial\vec{u}}{\partial n} = 0 \text{ on } \partial\Omega_N, \tag{2}$$

where $\vec{n}$ is the outward-pointing normal to the boundary, and $\frac{\partial\vec{u}}{\partial n}$ denotes the directional derivative in the normal direction. The condition on $\partial\Omega_N$ is called a *no penetration* condition. In order to ensure a unique pressure solution, we need to assume that the imposed pressure part of the boundary is non-trivial, that is $\int_{\partial\Omega_D} \mathrm{ds} \neq 0$.

Having specified a region Ω, a discretized problem is assembled by PIFISS using the lowest order Raviart-Thomas approximation. This approximates the x-component of the velocity vector by a linear function in x and the y-component of the velocity vector by a linear function in y in such away that the normal component of $\vec{u}$ is continuous across the edges of the triangulation. The pressure variable is approximated by a constant in each triangle. Further details of this particular finite element approximation scheme can be found in the following references.

- Franco Brezzi & Michel Fortin, *Mixed and Hybrid Finite Element Methods*, Springer-Verlag, 1991. (See, in particular, Section III.3.)

- J.E. Roberts & J.M. Thomas, *Mixed and hybrid methods, in Handbook of Numerical Analysis, Vol II: Finite Element Methods (Part 1),* P.G. Ciarlet & J.L. Lions, Eds, North-Holland, pp. 523–633, 1991.

There are four reference potential flow problems built into pifiss (see Appendix for a complete list). These are accessed by running the driver `flow_testproblem`. A session which explores the specific problem **P4** is outlined below. To reproduce it, simply type `flow_testproblem` and select the options listed. The flow domain in this case is a porous medium composed of five different zones with distinct permeability coefficients (see Figure 4.1). We specify a different constant for $\mathcal{A}$ in each one and compute the resulting flow. The coarse mesh that is generated by FEMLAB in pifiss is shown in Figure 4.2 and the computed flow solution is visualized in Figure 4.3.

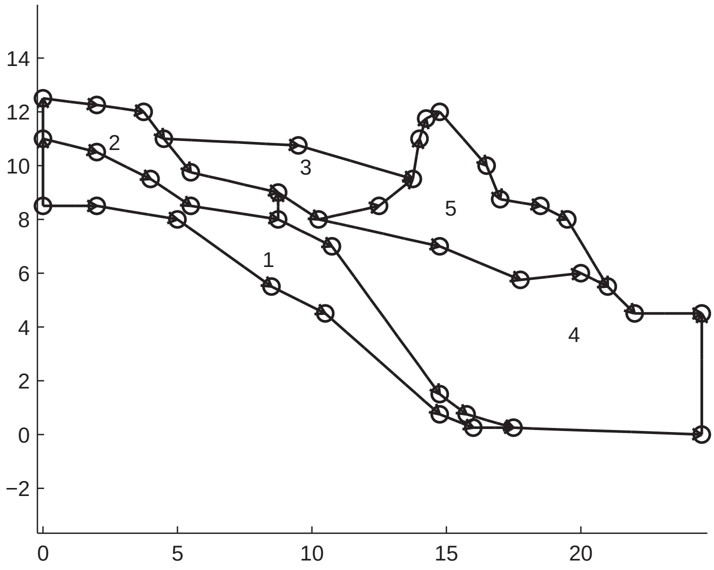


Figure 4.1. *Geology of reference problem **P4**.*

```
>> flow_testproblem

specification of reference groundwater flow problem.
choose specific example
     1  square domain
     2  circle domain
     3  punched ticket domain
     4  texas lsite domain
 : 4
number of refinements? (default is 2) :
```

```
Choose type of permeability coefficients
     1  unit coefficients (A=1) default choice
     2  single-valued piecewise constant coefficients
     3  piecewise constant diagonal matrix coefficients
 : 2

Enter value of A in each subdomain in order

 A1  : 1.04
 A2  : 17.2
 A3  : 0.31
 A4  : 2.60
 A5  : 17.2

Setting up Raviart-Thomas matrices and RHS vectors ... done
Imposing essential boundary condition ... done
Plotting solutions ... done
```

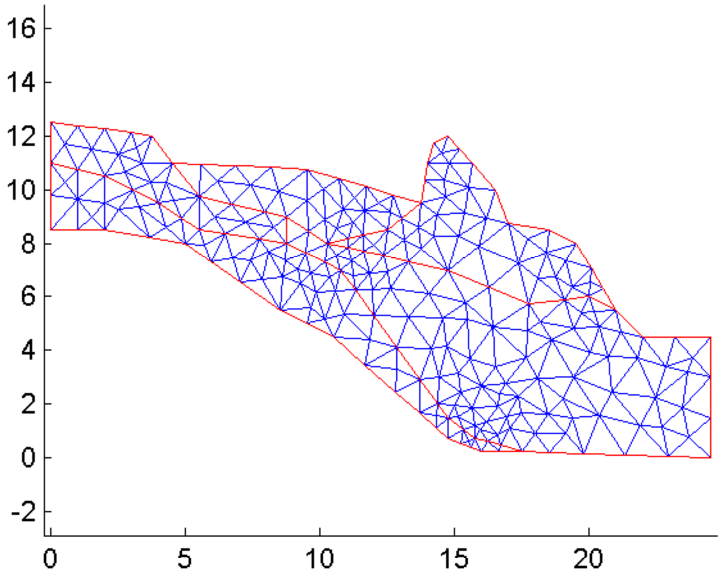
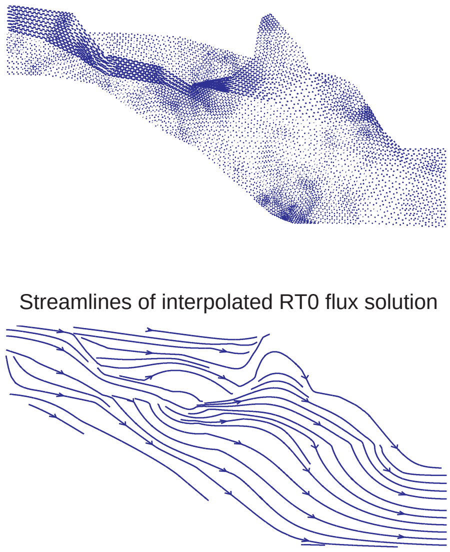


Figure 4.2. FEMLAB *mesh associated with reference problem* ***P4***.



Streamlines of interpolated RT0 flux solution

Figure 4.3. *Solution to the flow problem **P4**.*

The driver `flow_testproblem` solves the resulting discrete system using a MATLAB backslash solve. Once the problem has been set up and solved in this way, the performance of an alternative preconditioned iterative solver may be explored using the drivers `it_solve` and `amg_solve`. Issuing either of these commands allows the user to re-solve the system, iteratively, using preconditioned MINRES. The default preconditioner is a block-diagonal matrix, consisting of a diagonal matrix in the 1-1 block, and a sparse Schur-complement matrix in the 2-2 block. If the command `it_solve` is issued, the preconditioner is implemented 'ideally.' That is, the blocks of the preconditioner are inverted using a backslash solve. Alternatively, by issuing the command `amg_solve`, the action of the inverse of the 2-2 block of the preconditioner is approximated using algebraic multigrid. (Consult the references given in section one for technical details.)

Sample output for both solution schemes is reproduced below. If the ideal preconditioner is used (`it_solve`), 29 iterations are required, whereas if the AMG based preconditioner is used (`amg_solve`), 33 iterations are required. The iteration count rises hardly at all!

```
>> it_solve
discrete potential flow system...
 iterative solution with preconditioned MINRES
 tolerance? (default 1e-6) :
```

```
 maximum number of iterations? (default 100) :

MINRES iteration phase convergence in  29 iterations

    k  log10(||r_k||/||r_0||)
    0           0.0000
    1          -0.1458
    2          -0.5270
    3          -1.0600
                  .
                  .
   26          -5.6916
   27          -5.9107
   28          -5.9151
   29          -6.1757
Bingo!
  1.1251e+000 seconds

 >> amg_solve
discrete potential flow system...
iterative solution with preconditioned MINRES
tolerance? (default 1e-6) :
maximum number of iterations? (default 100) :

AMG setup phase ...

Level 2 step 1 coarsening started
 C points after 1st pass: 3209
 C points after 2nd pass: 3515
 C points after coarsening: 3515
Coarsening completed (3515 Coarse, 3157 Fine and 0 Unconnected
points)
   ....
Level 10 step 1 coarsening started
 C points after 1st pass: 1
 C points after 2nd pass: 1
 C points after coarsening: 1
Coarsening completed (1 Coarse, 3 Fine and 0 Unconnected points)

setup phase took   3.34 seconds

PDJ/PGS smoother? 1/2 (point damped Jacobi) :
point damped Jacobi smoothing ..
```

```
MINRES iteration phase convergence in  33 iterations
    k  log10(||r_k||/||r_0||)
    0           0.0000
    1          -0.1545
    2          -0.4860
    3          -1.1380
                  .
                  .
   31          -5.8741
   32          -5.8841
   33          -6.0205
Bingo!

  4.3924e+000 seconds
```

To circumvent the need to repeatedly type in coefficients PIFISS also provides a `batchmode` facility via which data may be input from a pre-prepared file rather than directly from the terminal. Sample input files for each of the model problems are provided (located in the appropriate `test_problems` subdirectory); these can be easily modified by the user for a particular run. The names of these input files must have the form "`*_batch.m`" where "`*`" begins with either "`diff`" or "`flow`" for the variable diffusion or potential flow problem respectively. For example, typing the command

```
batchmode('flowlsite')
```

uses the file `flowlsite_batch.m` to generate and solve the above reference problem without interactive input. The results of the run are stored in the file `batchrun` in the `datafiles` subdirectory (see below).

5. Directory structure

PIFISS comprises functions which generate finite element approximations of the steady-state diffusion problems and the potential flow problems. The discretization aspects are associated with the functions in the following directories

- `/pifiss/diffusion/`
- `/pifiss/potential_flow/`

Each of these directories has a subdirectory `/test_problems/`. These contain the boundary and coefficient function files associated with the reference problems described in the Appendix. The functions associated with the domain geometry and grid generation are independent of the problem being solved. These functions are thus located in a separate directory:

- `/pifiss/grids/`

The computed solutions are visualized using the 3-D plotting functions that are built into MATLAB and FEMLAB (or COMSOL). The functions that call these built-in functions to generate this visual output are also located in a separate directory:

- `/pifiss/graphs/`

For both types of discrete problem we provide specialized fast iterative solvers. The associated functions are contained in the following directory:

- `/pifiss/solvers/`

Finally, there are three directories that are used for storing intermediate data (for example finite element matrices and multigrid data) and plot files:

- MATLAB data (`.mat`) files : directory `/pifiss/datafiles/`
- FEMLAB structures (`.mat`) files : directory `/pifiss/femfiles/`
- plot (`.ps`) files : directory `/pifiss/plotfiles/`

6. Help facility and PIFISS function glossary

Help for the package is integrated into the MATLAB help facility. The command `help pifiss` gives a pointer to the IFISS general help command `helpme`. Typing `help` ⟨`directory name`⟩ lists the files in that directory that users may want to look at more closely. In MATLAB version 7, the function names are 'clickable' to give additional information.

The PIFISS package consists of over 70 MATLAB functions and script files, of which the high level ones are listed below. Simply type `help <file-name>` for further information on any of these. For a complete list of functions and scripts in a specific directory type `help <directory-name>`.

pifiss1.0	
gohome	positions command prompt at top level directory
helpme	pifiss interactive help facility
pifiss	returns pifiss version number
install_pc	sets up pifiss on non-UNIX computer
install_unix	sets up pifiss on UNIX computer
readme	distribution information file
setpath	sets pifiss search path
activemode	turns off batch processing for pifiss
batchmode	enables batch processing for pifiss testproblem

grids	
femlab_psfem	convert FEMLAB mesh to pifiss format
imeshplot	triangular mesh verification
p1grid	linear ($\boldsymbol{P}_1$) element grid generator
square_domain	square domain $\boldsymbol{Q}_2$ grid generator
subint	geometrically stretched subdivision generator

diffusion	
circle_adiff	solve anisotropic problem in circular domain
cylinder_adiff	solve anisotropic problem in exterior domain
femlab_adiff	solve anisotropic problem in general domain
femp1_adiff	vectorized linear anisotropic diffusion matrix
helpme_diff	variable diffusion problem interactive help
nonzerobc	imposes Dirichlet boundary condition
p1fluxjmps_adiff	variable diffusion flux jumps for triangular $\boldsymbol{P}_1$ grid
p1res_diff	interior residuals for triangular $\boldsymbol{P}_1$ grid
specific_adiff	(current) problem diffusion coefficients
specific_bc	(current) problem boundary condition
specific_rhs	(current) problem forcing function
square_adiff	solve anisotropic problem in unit square domain
tdiffpost_res	$\boldsymbol{P}_1$ element residual error estimator

POTENTIAL_FLOW	
RT0_mesh_info	generate RT0 mesh from FEMLAB structure
circle_groundwaterflow	solve flow problem in circular domain
cylinder_groundwaterflow	solve flow problem in exterior cylinder domain
femRT0	assemble RT0 matrices and rhs vectors
femlab_groundwaterflow	solve flow problem in general domain
get_RT0_boundary_labels	sets up data for natural boundary conditions
helpme_flow	potential flow problem interactive help
impose_hom_neuman_bc	imposes essential Neumann boundary conditions
impose_sign	fixes generic normal direction on edges
lsite_groundwaterflow	solve flow problem in lsite domain
piola_transform	evaluates RT0 basis functions via Piola transform
specific_f	(current) problem forcing function
specific_flowbc	(current) problem flow boundary conditions
specific_g	(current) problem nonzero pressure boundary
square_groundwaterflow	solve flow problem in square domain

SOLVERS	
a_nst	matrix-vector product for saddle-point system
amg_smoother	performs AMG smoothing step
amg_smoother_setup	generates data for AMG smoothing step
amg_solve	driver for AMG iterative solution for predefined problem
helpme_it	iterative solvers interactive help
it_solve	driver for iterative solution of predefined problem
m_ps	ideal Powell-Silvester preconditioner
m_ps_amg	AMG iterated Powell-Silvester preconditioner
pminres	preconditioned MINRES algorithm
resplot	plot residuals computed by iterative solvers

GRAPHS	
bookplot	saves PIFISS figure as postscript file
errplot_t	plots solution and error estimate on square-shaped domain
htmlplot	saves PIFISS figure as html file
plot_RT0_flux	plots RT0 velocity solution
plot_RT0_pressure	plots RT0 pressure solution
solplot	plots nodal data on square-shaped domain
solplotc	plots nodal data on long thin domain
solsurf	plots solution surface on square-shaped domain

7. Appendix: List of Test Problems

In this appendix, we give a brief description of each of the test problems in the PIFISS library, in terms of: the geometry of the domain Ω; the boundary data; the PDE coefficients $\mathcal{A}$; and the source term f. There are five 'diffusion' test problems labelled **D1–D5** (as discussed in section 3) and four 'potential flow' test problems labelled **P1–P4** (as discussed in section 4). Note that FEMLAB is not needed if one only wants to run the problems **D1–D3**.

Steady-state diffusion test problems

D1 Unit coefficients, square domain, unit source

This problem is posed on a unit square domain $[0, 1] \times [0, 1]$ with unit coefficients, i.e. $\mathcal{A} = I$, where I is the 2×2 identity matrix, with source term $f = 1$ and a homogenous Dirichlet boundary condition ($p = 0$ everywhere on the boundary). A sample solution is plotted in Figure 7.4.

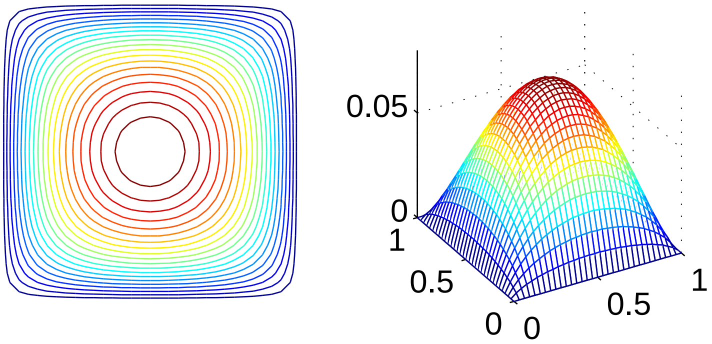


Figure 7.4. *Contours and mesh plot of solution to diffusion test problem **D1**.*

D2 Anisotropic coefficients, square domain, unit source

This problem is also posed on a unit square domain. The diffusion coefficient in this case is anisotropic. Specifically we have:

$$\mathcal{A} = \begin{pmatrix} 1 & 0 \\ 0 & 100 \end{pmatrix},$$

over the entire domain, with source term $f = 1$ and a homogenous Dirichlet boundary condition. Users may select either a uniform or stretched finite element mesh. A sample solution is plotted in Figure 7.5. The anisotropy coefficient (100 in this case) can be changed by editing the file `variable_adiff.m` .

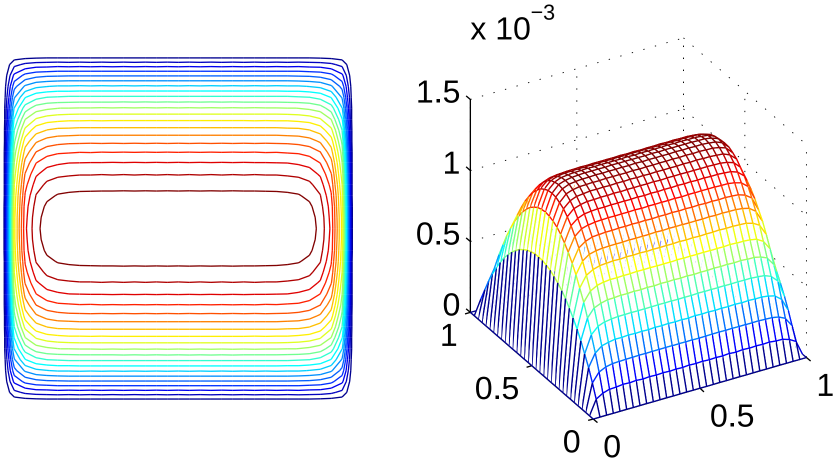


Figure 7.5. *Contours and mesh plot of solution to diffusion test problem* ***D2****.*

D3 Discontinuous anisotropic coefficients, square domain, unit source

This problem is posed on the square domain, $[-1, 1] \times [-1, 1]$ with the source term $f = 1$ and a homogenous Dirichlet boundary condition. The diffusion coefficient in this case however, is both discontinuous and anisotropic. It is given by

$$\mathcal{A} = \begin{pmatrix} A_x & 0 \\ 0 & A_y \end{pmatrix},$$

where the diagonal entries of this 2×2 matrix are

$$A_x = \begin{cases} 10^{-2} & \texttt{abs(y)} < 0.5 \\ 1 & \text{otherwise} \end{cases}, \qquad A_y = \begin{cases} 10^{-2} & \texttt{abs(x)} < 0.5 \\ 1 & \text{otherwise} \end{cases}.$$

This results in nine distinct subdomains with jumping coefficients across all the interfaces. A sample solution is plotted in Figure 7.6. The anisotropy parameter (10^{-2} in this case) can easily be changed by editing the file `ref_adiff.m` .

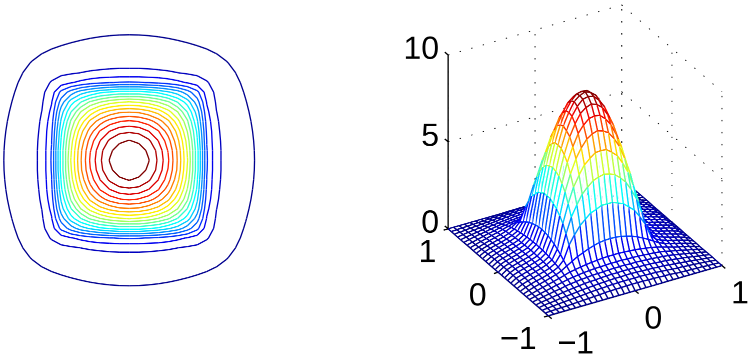


Figure 7.6. *Contours and mesh plot of solution to diffusion test problem **D3**.*

D4 Rectangular domain with interior circular obstacle, unit coefficients, unit source

This problem is posed on the domain consisting of the rectangle $[0, 2.2] \times [0, 0.41]$, with a small circle, of radius 0.05, centered at $(0.2, 0.2)$ removed. The source term is $f = 1$. A homogeneous Dirichlet boundary condition is prescribed on the circle and on the top, bottom and left exterior boundary. The natural condition $\partial p/\partial x = 0$ applies on the right exterior boundary. A sample solution is plotted in Figure 7.7.

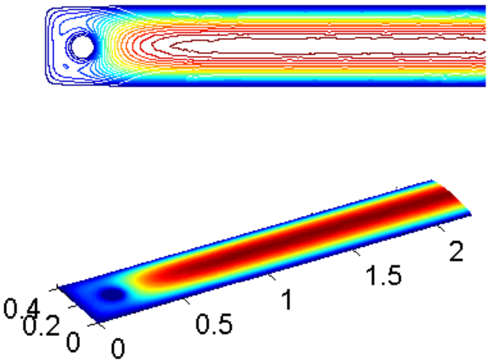


Figure 7.7. *Contours and mesh plot of solution to diffusion test problem **D4**.*

D5 Circular domain, discontinuous coefficients, unit source

This problem is posed on a unit circle domain, centered at the origin. The source term is $f = 1$ and a homogeneous Dirichlet boundary condition is prescribed on the exterior boundary. The coefficient $\mathcal{A}$ is piecewise constant and is a function of the radial distance r from the origin. We have $\mathcal{A} = \beta I$ and

$$\beta = \begin{cases} \frac{1}{10} & \text{for} \quad 0 \le r < 0.3 \\ 100 & \text{for} \quad 0.3 \le r < 0.7 \\ 1 & \text{for} \quad 0.7 \le r \le 1. \end{cases}$$

A sample solution is plotted in Figure 3.2. This problem has an analytic solution with p radially symmetric so that

$$p = \alpha + \frac{1}{4\beta}(1 - r^2),$$

where α is determined in each region to ensure continuity of p across the interface, and satisfaction of the boundary condition when $r = 1$.

This concludes the range of test problems for the primal formulation of the steady-state diffusion problem. We now describe the PIFISS library of potential flow (or 'groundwater flow') test problems.

Potential flow test problems

In each of the following four test problems, three options are available to prescribe the PDE coefficients:

```
1  unit coefficients (A=1) default choice
2  single-valued piecewise constant coefficients
3  piecewise constant diagonal matrix coefficients
```

For option 2, the user is prompted to enter a single value for $\mathcal{A}$ in each subdomain of the geometry. This means that $\mathcal{A}$ is a diagonal matrix with the same value for both diagonal entries in each subdomain. For option 3, the user is prompted to enter, for each subdomain in order, the values A_x and A_y where

$$\mathcal{A} = \begin{pmatrix} A_x & 0 \\ 0 & A_y \end{pmatrix}.$$

All three test problems below can thus be set up with unit coefficients, or with discontinuous and/or anisotropic coefficients.

P1 Square domain, zero source, mixed boundary conditions

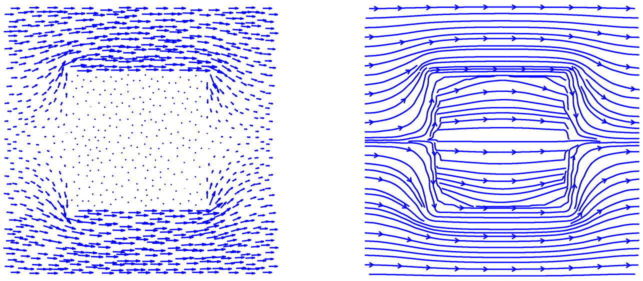


Figure 7.8. *Quiver plot and streamline plot of flux solution to potential flow test problem **P1**, $\mathcal{A} = 1$ in the outer subdomain, $\mathcal{A} = 1/1000$ in the inner subdomain.*

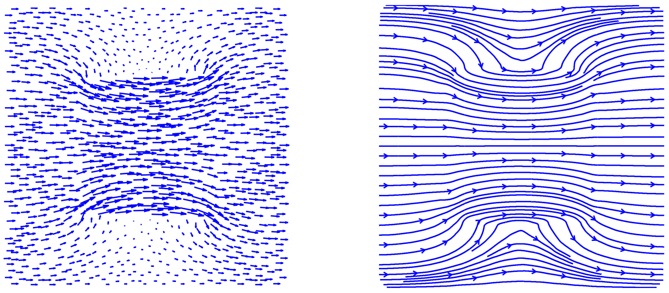


Figure 7.9. *Quiver plot and streamline plot of flux solution to potential flow test problem **P1**, $\mathcal{A} = 1$ in the outer subdomain, $\mathcal{A} = 1000$ in the inner subdomain.*

The first flow problem is posed on the square $[0, 1] \times [0, 1]$ with the option to prescribe an inner-subdomain $[0.25, 0.75] \times [0.25, 0.75]$. The source term is $f = 0$ and mixed boundary conditions are applied. Specifically, a no-flow condition $\frac{\partial \vec{u}}{\partial n} = 0$ is imposed at the upper and lower horizontal walls. In addition, we prescribe the conditions $p = 1$ at the left hand vertical wall and $p = 0$ at the right hand vertical wall so that the flow is forced from left to right across the domain. If the option to use an embedded square domain is used, then the entries of the coefficient matrix $\mathcal{A}$ can be chosen in each subdomain, independently. Sample solutions with discontinuous coefficients are plotted in Figures 7.8 and 7.9.

P2 Circular domain, unit source, zero boundary conditions

This problem is the mixed formulation of the steady-state diffusion problem **D5**. That is, we solve the potential flow problem on a circular disk of radius one. The source term is $f = 1$ and we apply the homogeneous condition $p = 0$ on the boundary. The circular disk is composed of three circular subdomains (see Figure 3.1) and the user may now prescribe the PDE coefficients in the manner discussed above.

P3 Rectangular domain with interior circular obstacle, zero source, mixed boundary conditions

This is a classical flow problem of computing the potential flow around a cylinder. The domain is described in problem **D4**. A no-penetration condition is imposed on the inner circle. The source term is $f = 0$ and the mixed boundary conditions from problem **P1** are prescribed to generate a flow from left to right. A sample solution is plotted in Figure 7.10.

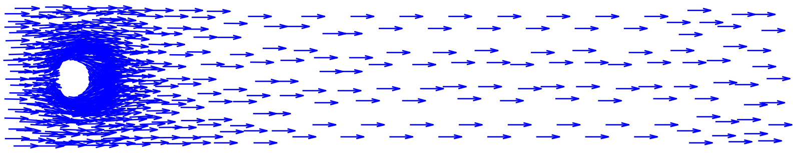


Figure 7.10. *Quiver plot of flux solution to potential flow test problem **P3**, $\mathcal{A} = I$.*

P4 L-site domain, zero source, mixed boundary conditions

This test problem is a model of a geological site that is located in the south-eastern United States. The L-site consists of a large fly ash disposal pond that is adjacent to a river. A cross section is given in Figure 4.1 and is composed of five different types of rock. Our specification of the problem is taken from the following article [RWB, 1999].

- Beatrice Riviere, Mary Wheeler & Carlos Baumann, *Part II: Discontinuous Galerkin method applied to a single phase flow in porous media,* TICAM Report 99-10, Austin, 1999. `utexas.edu/reports/1999/9910.ps.gz`

The source term is $f = 0$ and mixed boundary conditions similar to those used in problem **P1** are prescribed. That is, to obtain a flow from left to right, we set $p = 1$ on the left-hand vertical boundary and $p = 0$ on the right-hand boundary. A no penetration condition is imposed on all other boundary segments. The sample flow solution shown in Figure 4.3 is obtained by specifying the material permeability values reported in [RWB, 1999].