

Fastflo

Flexible finite element software for the numerical solution of PDEs

Technical information for prospective users

Technical information about *Fastflo*

Fastflo is a finite element package for the numerical solution of partial differential equations (PDEs) in two- and three-dimensional regions.

Fastflo is very flexible because the finite element methodology can handle domains with complex shapes, and because it incorporates a high level language, *Fasttalk*, in which a mathematical notation is used to specify and solve a wide range of PDEs. Users interact with *Fastflo* via a Graphical User Interface which offers editing, file management, graphics and hands-on control of the computations.

Fastflo is available for PCs and UNIX systems. A Fluids ToolBox, released with Version 3 of *Fastflo*, provides easy access to advanced Computational Fluid Dynamics (CFD) algorithms. *Fastflo* has a comprehensive Tutorial Guide and an on-line Reference Manual.

Benefits of *Fastflo*

As a general PDE solver, *Fastflo*'s main advantage is its flexibility in specifying models in two and three dimensions and algorithms to solve them. The PDEs can be well known (such as fluid flow, linear elasticity, electromagnetism, eigenvalue problems) or non-standard equations encountered in scientific or industrial applications. Users are free to specify what equation to solve, to design the algorithm used for the solution and to control the computations as desired.

Fastflo allows users to create or modify a finite element environment for their own needs or for sharing with a team. There is no need for time-consuming low-level programming in languages like FORTRAN or C. In *Fasttalk*, the problem description is concise, still allowing as much detail as is needed.

Fastflo's main features

- an internal triangular mesh generator for 2D problems, plus ability to import data from external mesh generators
- a range of isoparametric element types: triangles, quadrilaterals, tetrahedra, hexahedra; all these are available with linear or quadratic interpolation and can be modified by users
- the use of *Fasttalk* to specify PDEs and the algorithms to solve them; users are not limited to a fixed menu of techniques and applications
- the ability to solve systems of PDEs with several unknowns, with multiple domains and fixed or moving boundaries
- a selection of sparse matrix solvers, both direct and iterative with pre-conditioning
- a Graphical User Interface to specify problems, direct the solution procedures, and display the results in 2D and 3D
- availability on PCs and UNIX systems
- Tutorial Guide and on-line Reference Manual
- a Fluids ToolBox providing advanced CFD algorithms for laminar and turbulent flow.
- developed since 1991 by mathematicians and engineers from CSIRO, Australia's largest R&D organisation

Applications

The following examples from the Tutorial Guide can serve as starting points for a range of PDEs:

- steady state elliptic PDEs in one variable (Laplace, Poisson, Helmholtz) in 2D geometries with polygonal or curved boundaries
- coupled elliptic PDEs; linear elasticity, thermoelasticity
- transient problems such as the diffusion equation solved by various timestepping algorithms

- computations in domains with multiple sub-regions; axisymmetric computations
- porous media flow
- transient advection-diffusion problems in cylindrical coordinates (solute dispersion)
- eigenvalue analyses
- acoustic scattering using the Sommerfeld radiation condition
- moving boundary problems by the ALE method; deformable meshes
- 2D and 3D flow problems using various solvers (penalty method, augmented Lagrangian, streamline-vorticity, operator-splitting)
- options pricing by the Black-Scholes equation

The following Case Studies described in the Tutorial Guide illustrate the use of *Fastflo* on advanced topics:

- viscoelastic flow around a sphere in a cylindrical tube
- convective cooling of heated plates
- turbulent flow in an axisymmetric narrowing U-bend
- 3D turbulent flow in a tundish
- swirling turbulent flame inside a burner
- free surface flow in a double roll coater
- turbulent air flow around a Daimler-Benz model car
- two-phase flow in a mixing tank stirred by a turbine
- options pricing using the Black-Scholes equations
- pricing continuous and discrete Asian options
- saturated porous media flow with heat and chemical reaction

Users present their PDE problems to *Fastflo* via two files, one for the mesh and one for the problem specification. *Fastflo* uses unstructured meshes, and problems can be solved in complex geometrical shapes. If the PDE is time-dependent, the user may develop an algorithm for timestepping. If the PDE is nonlinear, an appropriate iterative strategy can be implemented. After any necessary timestepping and iterative algorithms have been implemented, the user will have a set of linear PDEs to be solved at each timestep or iteration. In *Fastflo*, these PDEs can have partial derivatives up to second order and vector or tensor coefficients. A complete set of 38 derivative expressions is available.

Dependence of coefficients on spatial and other variables is managed by a mathematical expression capability with a wide range of operators. A global vector stack is used to store and manipulate the field variables.

Fastflo can handle systems of PDEs. For example, a benchmark problem solved using *Fastflo* was simulation of 3D turbulent air flow around an automobile. This problem, discussed at the 1996 meeting of the World User Association for Computational Fluid Dynamics, had six variables, namely three velocity components, pressure, turbulent kinetic energy and dissipation. *Fastflo*'s results compared well with experimental measurements for drag and pressure on the upper surface. Other important features are *Fastflo*'s moving mesh capability, and capability to specify problems in multiple sub-domains.

***Fasttalk* code for an example**

Fasttalk code is compact and describes PDE problems in an intuitive and mathematical way. To illustrate, we present computations for two linked problems – 2D flow around a cylinder in a duct, and dispersion of a tracer substance in the flow. The flow problem is solved by the augmented Lagrangian method, whilst the tracer dispersion problem is solved by a Crank-Nicolson procedure.

For this 2D flow problem, there are two Navier–Stokes equations and a continuity equation

$$\mathbf{v} \cdot \nabla \mathbf{v} = -\nabla p + \text{Re}^{-1} \nabla^2 \mathbf{v}, \quad \nabla \cdot \mathbf{v} = 0$$

These are solved by the iterative procedure

$$\mathbf{v}_{n-1} \cdot \nabla \mathbf{v}_n = -\nabla p_{n-1} + \text{Pen} \nabla (\nabla \cdot \mathbf{v}_n) + \text{Re}^{-1} \nabla^2 \mathbf{v}_n, \quad \Delta p + \text{Pen} \nabla \cdot \mathbf{v}_n = 0$$

The nonlinear term is linearised by a Picard iteration. $\Delta p = p_n - p_{n-1}$ is the difference between successive iterations for the pressure p . The augmented Lagrangian method, if it converges, does not introduce any further approximation to the Navier-Stokes equations and provides an answer for the pressure.

The calculation takes place on an unstructured mesh with 800 corner nodes (see next page). The output, also shown, is an arrow plot of the velocity. The files that completely specify the problem are given below (together with comments following the % signs). This example illustrates the compactness of the *Fasttalk* code.

```

800 0 4      % unstructured triangular mesh
1 0 0 0 0    % 800 corner nodes, quad approx
3.0 1.5      % concentrate mesh near (3.0,1.5)
List of vertices
1.0 0.5      % vertices 1 and 6-8 specify
3.0 1.25     % the duct; vertices 2-5
2.75 1.5     % specify the cylinder
3.0 1.75
3.25 1.5
6.0 0.5
6.0 2.5
1.0 2.5
List of boundary tags
1 0          % bridge from duct to cylinder
2 4          % tag 4 is for cylinder
3 4
4 4
5 4
2 0          % bridge from cylinder to duct
1 1          % duct wall
6 5          % outlet
7 1          % duct wall
8 3          % inlet
End of boundary list

```

```

P Rinv 1/40      % assign parameter Rinv
P Pen 20.        % assign parameter Pen
P iter 0         % assign parameter iter
P nstep 10       % assign parameter nstep
A EQNmtm        % problem for velocity
e {V100}_jD_jU1_i = - D_i{V201} + D_i{Pen}D_jU1_j \
    + D_j{Rinv}D_jU1_i + D_j{Rinv}D_iU1_j
b 1 U1 = {0,0}   % boundary condition at duct wall
b 3 U1 = {1,0}   % boundary condition at inlet
b 4 U1 = {0,0}   % boundary condition at cylinder
b 4 curvature=-0.2071 % enforces curvature for cylinder
A [reduced] EQNpress % problem for pressure increment
e U1={Pen}D_j{-V100}_j
< runVel
  while iter < nstep
    EQNmtm % assemble problem for velocity
    solve % solve problem for velocity
    EQNpress % assemble problem for pressure increment
    solve % solve problem for pressure increment
    expand 1 % interpolate to full mesh
    V401=V401+V101 % calculate incremented pressure
    popp % adjust vector stack
    V200=V100 % assign new velocity field
    black
    arrow
    popp % adjust vector stack
    iter = iter + 1
  endwhile
>

```

The species transport equation is

$$\frac{\partial C}{\partial t} + \mathbf{v} \cdot \nabla C - k \nabla^2 C = 0$$

where k is the molecular diffusion coefficient for the species, \mathbf{v} is the velocity vector and C is the concentration of the species. This equation is here solved using an implicit scheme. The initial value problem is advanced over a timestep (δt) using the algorithm

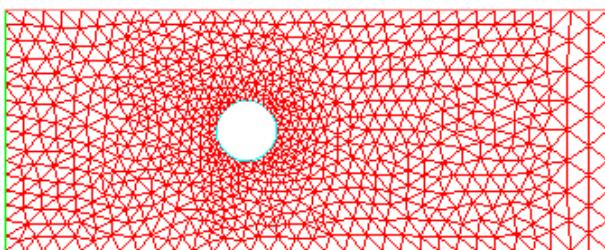
$$\frac{(\delta C)}{(\delta t)} + \mathbf{v} \cdot \nabla (\delta C) - k \nabla^2 (\delta C) = - \mathbf{v} \cdot \nabla C^n + k \nabla^2 C^n$$

where $C^{n+1} = C^n + \delta C$.

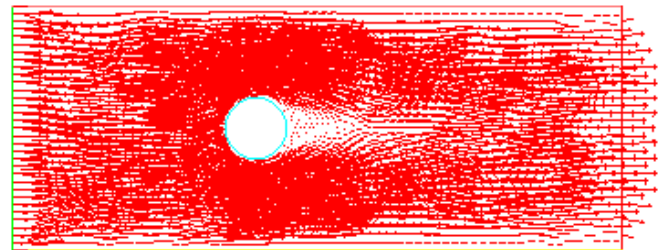
```

A tracer
e {1/dt}U1 - D_j{k}D_jU1 + {V100}_jD_jU1 \
    = D_j{k}D_j{V201} - {V100}_jD_j{V201}
< runConc
  V201 = (X1>2.0) & (X1<2.2) % assign initial concentration
  t1 = 0.9 % end time
  dt = 0.1 % set timestep size
  k = Rinv % set value for k
  t = 0 % starting time
  while t < t1
    tracer % assemble problem for tracer
    solve % solve
    V301 = V101 + V301 % solution at new timestep
    popp % adjust vector stack
    t = t + dt
    black
    shade 201
  endwhile
>

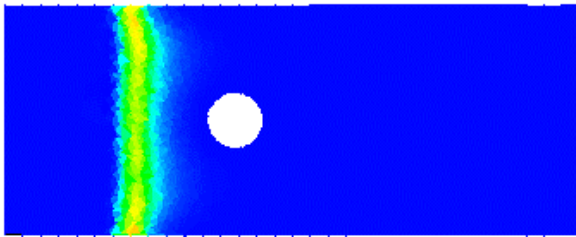
```



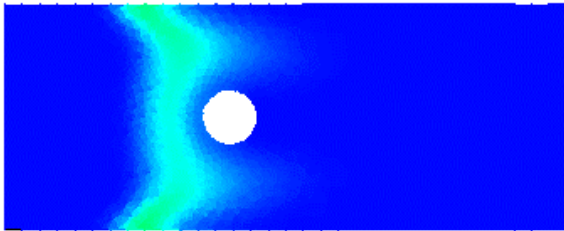
Mesh for the cylinder-in-duct problem



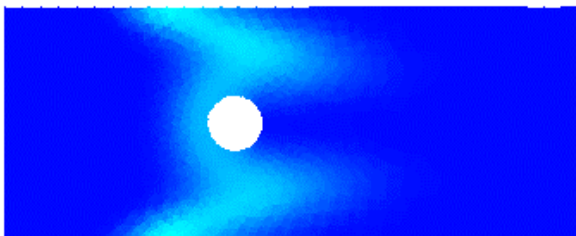
Arrow plot of velocity field



Tracer concentration at t = 0.1.



Tracer concentration at t = 0.5.



Tracer concentration at t = 0.9.

Several features of the above example deserve comment. Note that the flow problem does not have an explicit boundary condition at the outlet. Similarly the tracer problem does not have explicit boundary conditions. In cases like these, a natural boundary condition is automatically applied – this is equivalent to a zero stress condition in the flow problem, and a zero concentration gradient condition in the tracer problem. The term $D_j\{R_{inv}\}D_iU_{1_j}$ is zero on the interior for incompressible flows, but is included because it generates a non-zero boundary integral, which is a component of the boundary stress, and is needed in natural boundary conditions. For more details, see Section 14.2 of the *Fastflo* Tutorial Guide.

Note that advanced algorithms for CFD are available to users via a Fluids ToolBox, released with Version 3 of *Fastflo*.

FURTHER INFORMATION
 CSIRO Mathematical and Information Sciences
 Private Bag 33, Clayton, VIC 3169, Australia
www.csiro.au/Fastflo
 Telephone 61 3 9545 8005
 Fax: 61 3 9545 8080
 E-mail: Fastflo@cmis.csiro.au

Table of operators available in *Fastflo*

No	Notation	Interpretation
1	$D_j A D_j U_1$	$\nabla \cdot (a \nabla u)$
2	$A U_1$	au
3	$A_j D_j U_1$	$\mathbf{a} \cdot \nabla u$
4	$D_j A_j U_1$	$\nabla \cdot (\mathbf{a} u)$
5	$D_j A_{jk} D_k U_1$	$\nabla \cdot (\mathbf{A} \nabla u)$
6	$D_j A U_{1_j}$	$\text{div}(a \mathbf{u})$
7	$A D_j U_{1_j}$	$a \nabla \cdot \mathbf{u}$
8	$A_j U_{1_j}$	$\mathbf{a} \cdot \mathbf{u}$
9	$D_j A_k D_k U_{1_j}$	$\text{div}(\mathbf{a} \cdot \nabla u)$
10	$D_j A_j D_k U_{1_k}$	$\text{div}(\mathbf{a} \nabla \cdot \mathbf{u})$
11	$D_j A_{jk} U_{1_k}$	$\text{div}(\mathbf{A} \mathbf{u})$
12	$A_{jk} D_j U_{1_k}$	
13	$D_i A U_1$	$\nabla (au)$
14	$A D_i U_1$	$a \nabla u$
15	$A_i U_1$	$\mathbf{a} u$
16	$D_i A_j D_j U_1$	$\nabla (\mathbf{a} \cdot \nabla u)$
17	$D_j A_j D_i U_1$	$\nabla \cdot (\mathbf{a} \nabla u)$
18	$D_j A_{ji} U_1$	$\nabla \cdot (\mathbf{A} u)$
19	$A_{ij} D_j U_1$	$\mathbf{A} \nabla u$
20	$A U_{1_i}$	$\mathbf{a} u$
21	$A_j D_j U_{1_i}$	$\mathbf{a} \cdot \nabla u$
22	$D_j A_j U_{1_i}$	$\mathbf{a} \cdot \nabla u + u \nabla \cdot \mathbf{a}$
23	$D_j A D_j U_{1_i}$	$\nabla \cdot (a \nabla u)$
24	$D_j A_{jk} D_k U_{1_i}$	$\nabla \cdot (\mathbf{A} \nabla u)$
25	$D_i A D_j U_{1_j}$	$\nabla (a \nabla \cdot \mathbf{u})$
26	$D_i A_j U_{1_j}$	$\nabla (\mathbf{a} \cdot \mathbf{u})$
27	$D_j A D_i U_{1_j}$	
28	$A_j D_i U_{1_j}$	$\mathbf{a} \cdot (\nabla u)$
29	$D_j A_i U_{1_j}$	$\mathbf{a} (\nabla \cdot \mathbf{u}) + \mathbf{u} \cdot \nabla a$
30	$A_i D_j U_{1_j}$	$\mathbf{a} (\nabla \cdot \mathbf{u})$
31	$A_{ij} U_{1_j}$	$\mathbf{A} u$
32	$D_i A_{jk} D_j U_{1_k}$	$\nabla (\mathbf{A} \cdot \nabla u)$
33	$D_j A_{jk} D_i U_{1_k}$	
34	$D_j A_{ik} D_k U_{1_j}$	
35	$D_j A_{ij} D_k U_{1_k}$	
36	$D_j A_{ik} D_j U_{1_k}$	
37	$D_j A_k D_j U_{1_k}$	$\text{div } \mathbf{a} \nabla u$
38	$D_j A_i D_j U_1$	$\text{div } \mathbf{a} \nabla u$

Lower case bold type denotes a vector, and upper case denotes a matrix (2nd order tensor). The summation convention applies to suffices. D_j denotes $\partial/\partial x_j$