Postprocessing of FEATFLOW Data with the Particle Tracing Tool GMVPT (Version 1.2)

J.F.Acker, S.Turek
Angewandte Mathematik und Numerik (LS III), Universität Dortmund
Im Vogelpothsweg 87, 44221 Dortmund, Germany
Email: featflow@featflow.de
Homepage: http://www.featflow.de

25th June 2002

Contents

1 How to start ?

1.1 Installation

First you have to copy the source files `gmvpt.c` and `movbc.inc` (which are downloadable from our homepage) into the same directory where e.g. the FEATFLOW simulation tools are located.

Then you have to compile the source code with your preferred C-compiler, e.g. gcc:

```
gcc -O3 -lm -o gmvpt gmvpt.c
```

If you use bouss, that means the modified version of pp2d including temperature, you have to do the following variant:

```
gcc -O3 -lm -DBOUSS -o gmvpt gmvpt.c
```

To process the field "ny" given by the **POWERLAW** variants, you have to specify additionally "**-DPOWERLAW**". Because tracers can be deleted, the numeration of our tracers is not constant in GMV. To correct this, add the switch "**-DTIN**" (only useable for GMV 2.1 or higher!). If you are interested, how old your particles are, you can get that information in an additional datafield called "Alter" by specifying the switch "**-DAGE**". We added also a support for 3D data, that can be activated by the switch "-DPART3D". If you work with moving boundaries (`cc2d_movbc` or `cc3d_movbc`) you have to add the switch "**-DMOVBC**" and edit the file `movbc.inc`, too.

> **Remark: If you have modified the output-routines in FEATFLOW or if you want to apply GMVPT to GMV data not created by FEATFLOW, make the corresponding modifications of `gmvpt.c` !**

Caution! This Manual describes how to use version 1.2 of GMVPT. The usage of GMVPT can change in later versions!

1.2 Directory structure

You need the following directories and files to successfully run GMVPT:

1. A coarse mesh `#gmv/coarse.gmv` and refined meshes `#gmv/tria*.gmv`, e.g. generated by invoking `trigen2d` (see [1]), with the finest mesh exactly the same as the mesh written into your simulation data files. These are needed to visualize the domain and to initialize the hierarchical search structures. GMVPT adds an reference to the file `coarse.gmv` to every written tracer file. So you can change the mesh that GMV uses later in the visualization of your tracers by copying one of your `tria*.gmv` files to `coarse.gmv`
2. Data files `#gmv/u.*.gmv`, e.g. generated by `bouss`, `cc2d`, `cc3d`, `pp2d` or `pp3d`. GMVPT scans automatically for its input files, starting the search at `#gmv/u.1.gmv`
3. A configuration file `#data/gmvpt.dat`

1.3 Syntax of the configuration file `gmvpt.dat`

The configuration file is only a bunch of integer and floating point values. Linebreaks and blanks between those numbers are ignored. It's up to you to make it readable by inserting some linebreaks and blanks.
Unsigned integer values are preceeded by '#'. Everything else is a **floating point** value.

> <#Number of time steps> <Duration of a single time step for particle tracing> <#Read next input file after this number of time steps> <#Write next tracer file after this number of time steps>
>
> <#Number of Blocks>

Each MxN[xO] block of (starting) tracers with lower left corner (x1,y1[,z1]) and upper right corner (x2,y2[,z2]) is given by:

> <x1> <y1> [<z1>] <x2> <y2> [<z2>] <#M> <#N> [<#O>] <#Insert this block after this number of time steps (frequency)> <#Subtract this offset to the number of time steps (phase change)> <#First time step> <#Last time step> <Value of data field 'Farbe'> <Remove particles after this time (lifetime)>

1.4 A short examplary configuration file (2D)

```
1000 0.05 2 2
2
0.0 0.0 0.1 0.1 10 10 4 0 0 999 0.0 25.0
0.0 0.0 0.1 0.1 10 10 4 1 0 999 1.0 25.0
```

This sample file makes 1000 micro time steps with duration 0.05. Every second time step it reads a new input file and writes a new tracer file (be sure to have enough input files!). This implicitly requires that the time step in the simulation process to output the data was 0.1. There are two blocks located at (0.0,0.0)-(0.1,0.1) with a 10x10 grid of tracers. Each tracer exists for 25 seconds. Both blocks are inserted each fourth (micro) time step, but the second has a phase change. The effect is that tracers with color 0.0 are inserted at time steps 0,4,8,12,... and tracers with color 1.0 are inserted at time steps 1,5,9,13,... (if (i-1) mod 4 = 0 is true for time step i). To assign real colors to these values you have to modify the data limits of tracer field `Farbe` and to adjust the colormap to your needs. It will generate output files named `t.0.gmv`, `t.1.gmv`, ..., `t.500.gmv` (prefix 't' by default). For making particle tracing with stationary data (GMVPT looks for the first `u.i.gmv` file!) set the third value (Read next input file after this number of time steps) higher than the first value (Number of time steps).

1.5 Moving boundaries

Moving boundaries are implemented analytically by calling an user defined test function with node or tracer coordinates. If a tracer hits the moving boundary, it is deleted. For each cell completely inside this moving boundary (all nodes inside), polygons describing that cell are included in our tracer files (hidden polygons are actually not removed in 3D!). This test function is located inside the source file `movbc.inc`.

1.6 Example include files

By default, the test function `Test_MovBC` returns an integer value 0.

```
int Test_MovBC(Daten_Typ K[DIM]) { return 0; }
```

This results in no moving boundary at all. The location of the point, which should be tested, is stored in the array parameter here called `K`. The next include file gives the more advanced case of a rotating cross-shape boundary located in the origin.

```
#include <math.h>
int Test_MovBC(Daten_Typ K[DIM])
{
const Daten_Typ drad=0.25;
const Daten_Typ deps=0.05;
const Daten_Typ dvel=M_PI;
Daten_Typ phi1,phi2,phi3,phi4,phi5,phi6=0.0,phi7=0.0;
Daten_Typ phi1d,phi2d,dist1,dist2,drad2;

phi1=Global_Time*dvel;
phi2=phi1+M_PI*0.5;
phi3=phi2+M_PI*0.5;
phi4=phi3+M_PI*0.5;
drad2=sqrt(K[0]*K[0]+K[1]*K[1]);
if (drad2<=drad)
{
phi5=acos(K[0]/drad2);
if (K[1]<0.0) phi5=2.0*M_PI-phi5;
while(phi5<phi1) phi5+=2.0*M_PI;
if ((phi5>=phi1)&&(phi5<=phi2))
{
phi6=phi1;
phi7=phi2;
}
if ((phi5>=phi2)&&(phi5<=phi3))
{
phi6=phi2;
phi7=phi3;
}
if ((phi5>=phi3)&&(phi5<=phi4))
{
phi6=phi3;
phi7=phi4;
}
if (phi5>=phi4)
{
phi6=phi4;
phi7=phi1;
}
phi1d=phi5-phi6;
phi2d=phi7-phi5;
dist1=drad2*sin(phi1d);
dist2=drad2*sin(phi2d);
if ((fabs(dist1)<=deps)||(fabs(dist2)<=deps)) return 1;
}
return 0;
}
```

1.7 Usage

Go to the directory, where the subdirectories `#gmv` and `#data` are located and start GMVPT with

gmvpt [<prefix>]

The optional parameter <prefix> is per default 't' and gives the leading string of the filenames of your generated tracer files. These names are constructed by '<prefix>.<#filenumber>.gmv'.

2 Visualisation of particles using GMV

2.1 Preparations

After starting GMV (see also [5]), open the cells menu at `Display/Cells`. Then deactivate `Faces` and activate `Edges`. Adjust also magnification, line color and line width to get a proper display of the underlying coarse mesh.

2.2 The tracers menu

Open the tracers menu at `Display/Tracers`. There you can activate different representations of tracers. Normally choose `Regular Points`. You can also choose between different data fields which determine, together with the color map and actual data limits, the color of each tracer/particle (see Fig. 1). Open the specific data limits menu at `Controls-1/Data Limits/Tracers`. There

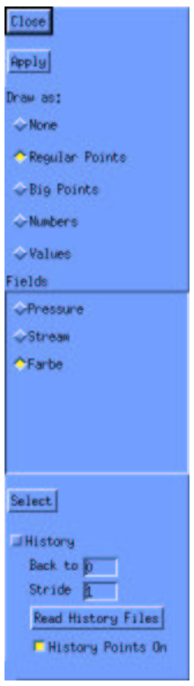


Figure 1: Tracers Menu

you can explicitly specify data limits for each data field. If you activate `Auto Reset` or `Auto Reset All`, for each single input file and for selected or all data fields, GMV calculates actual data limits. Turn off the color bar if you want to make an animation, since GMV would normally change in every frame such that it becomes unreadable (see Fig. 2).

2.3 The polygons menu

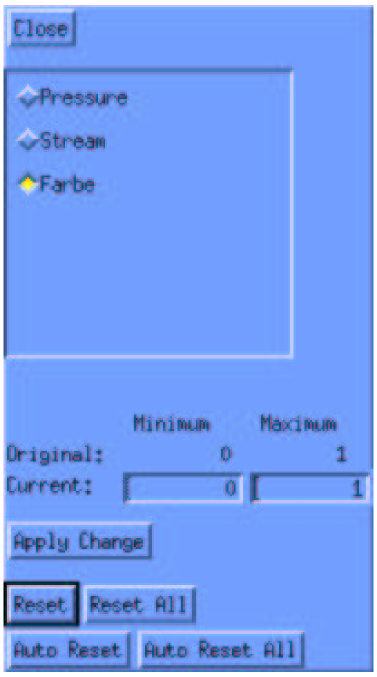
Open the polygons menu at `Display/Polygons` (see Fig. 3). With `shaded` and `lines`, you can toggle the display of the interior and the outline of all included polygons in our tracer files, marking the nodes inside of our moving boundary. To change the color of our polygons, you have to edit the color of material number 2 at `Controls-1/Coloredit/Materials, Isosurfaces, Isovolume.`

2.4 The timestamp

GMVPT inserts the global time of your particle tracing as timestamp in the tracer files. By default, this time is displayed in the right upper corner of the visualization. You can toggle the display of that timestamp at `Controls-2/Time`.

2.5 Generating MPEG videos with GMV

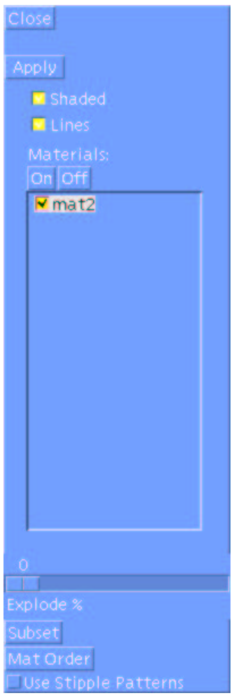
First load one of your generated `t.*.gmv` files. Adjust GMV to get the desired visualisation of your tracers and create an attribute file `test.attr` with `Files/Put Attributes`. Then further steps depend on which tools you want to use (e.g. image size 640x480, number of frames 250, name of MPEG will be `test.mpeg`):



[✓ Farbe]

	Minimum	Maximum
Original:	0	1
Current:	0	1

Figure 2: Set Tracers Limits Menu



[✓ Shaded]
[✓ Lines]
[✓ mat2]
0

Figure 3: The Polygons Menu

- MKYUV/MKMPEG (see [2]):

  1. Replace every 'u.' by 't.' in the file `mkyuv.c` and recompile `mkyuv`
  2. Invoke MKYUV with `mkyuv 640 480 test.attr test 1 250`
  3. Invoke MKMPEG with `mkmpeg 640 480 test 250`

- GMVMPEG (see [4]):

  Invoke GMVMPEG with `'gmvmpeg -a test.attr -x 640 -y 480 -fls 1,250,1 -o test'`. If you have installed the virtual framebuffer Xvfb, you can add the option '-I' to avoid annoying popups by GMV. Invoke GMVMPEG without any parameter, to get a complete list of available parameters.

3 Examples (2D)

3.1 Flow around a cylinder

The problem here is to visualize the vortex shedding behind a cylinder. We continuously insert a rectangular area of tracers with a constant frequency to simulate a constant stream of particles. Between each frame we make ten interpolating time steps (if each input file has time difference 0.025, you have to ensure that time step is 0.025/10=0.0025, to be synchronous with the simulation). An examplary configuration file looks like the following (see Fig. 4 for its output).

```
2500 0.0025 10 10
1
0.0 0.175 0.1 0.225 10 5 27 0 0 2499 0.0 2.5
```

3.2 Leapfrogging of smoke rings ('Puff-Puff' problem)

Here the problem is to visualize two smoke rings chasing each other (**Remark:** It's almost impossible to visualize this flow configuration without particle tracing!). We insert a squared 100x100 block of 10,000 particles located at (4.3,4.4)-(4.5,4.6) at times 0.0 and 120*0.03333333334=4.0 with different colors (0.0 represents blue, 1.0 represents red in the default color map, if we use a value range of 0.0-1.0 for data field 'Farbe'). Each input file corresponds to a time step of 0.1. Thus we make three interpolating time steps between the input files and due to the 500 input files, we have to use a time step of about 0.1/3=0.033333... and we make 500*3=1500 time steps. An examplary configuration file is shown below in this text (see Fig. 5 for its output).

```
1500 0.03333333334 3 3
2
4.3 4.4 4.5 4.6 100 100 1 0 0 0 0.0 50.1
4.3 4.4 4.5 4.6 100 100 1 0 120 120 1.0 50.1
```

3.3 Moving boundary example ('Rotor' problem)

Here is the problem to visualize a rotating cross-shaped boundary inside a quadratic box with inflow at the middle of the right boundary and outflow at the middle of the top, left and bottom boundary. Each input file corresponds to a timestep of 0.01. The timestep between each tracer file is 0.1. To visualize the flow, each 0.02 sec. are 201 new particles placed along a streak line at the outer right boundary. An examplary configuration file is shown below in this text (see Fig. 6 for its output). See the 'Virtual Album' (*http://www.featflow.de*) for more details and the complete MPEG-videos!

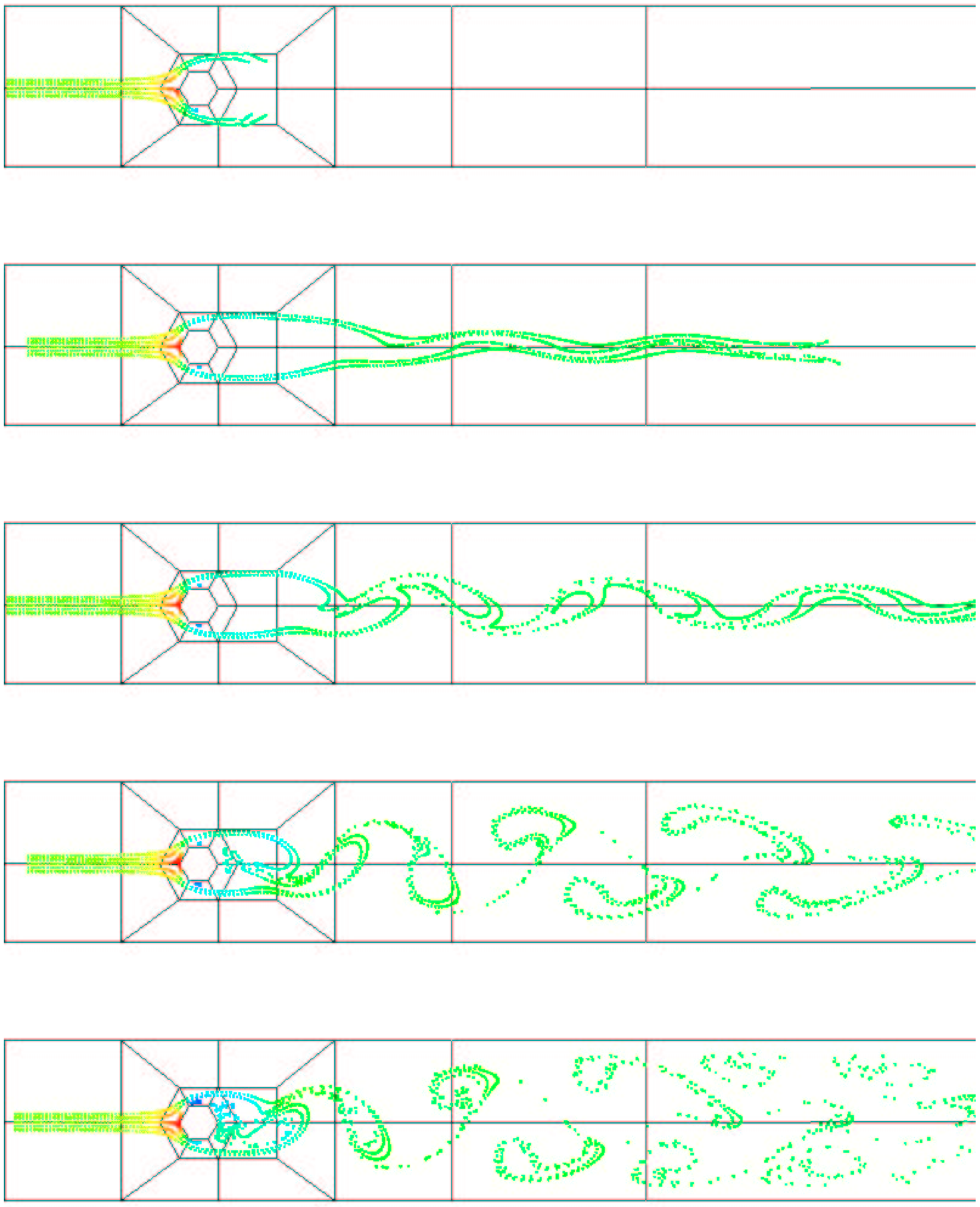


Figure 4: Vortex shedding behind a cylinder visualized by a particle stream

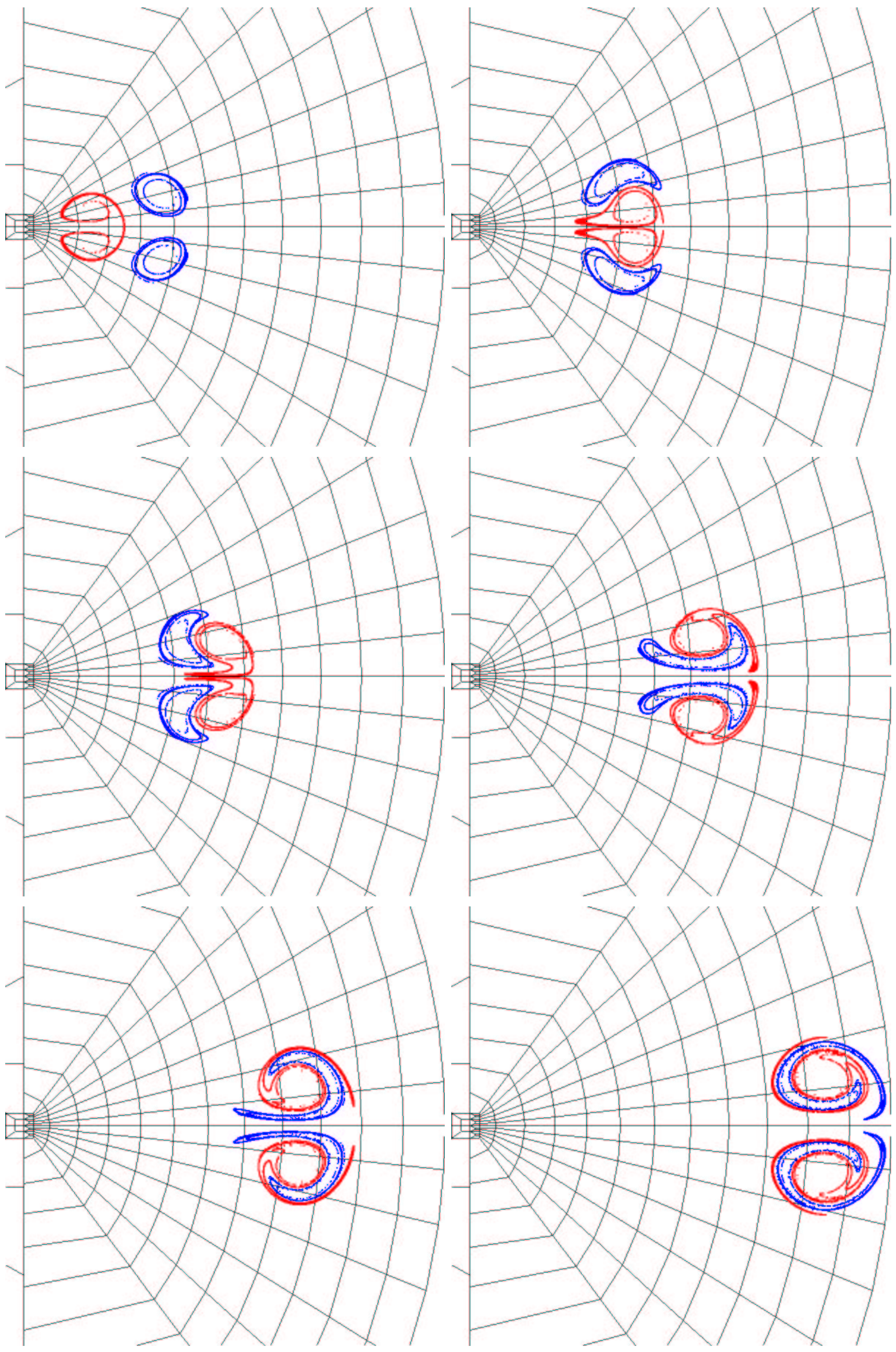


Figure 5: Two smoke rings chasing each other

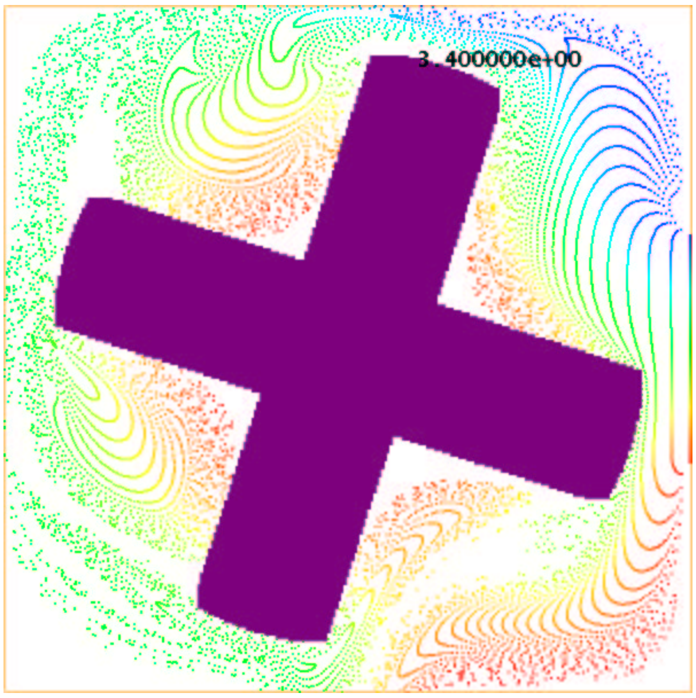


Figure 6: Flow around a moving rotor with streak lines

```
160 0.005 2 20
1
0.299 -0.1 0.299 0.1 1 201 4 0 0  2000 0.0 100.1
```

4 Examples (3D)

4.1 Flow behind a cylinder

It's the same problem from section 3.1 extended into 3D. But here we insert a small square of particles *behind* the cylinder. See Figures 7 for the output. The content of the configuration file is listed below.

```
10000 0.001 100 25

1

0.7 0.1909 0.195 0.7 0.2109 0.215 1 5 5 25 0 0 100000000 0.0 3.0
```

4.2 Moving boundary example ('Rotor' problem)

It's a similar problem as in section 3.3. Here we have a channel flow colliding with a moving rotor and outflows at the side and the end of that channel. The rotor is described inside the file *movbc.inc*:

```
int Test_MovBC(Daten_Typ K[DIM])

{

  const Daten_Typ DVel=M_PI_4;

  const Daten_Typ DRad=0.226;
```

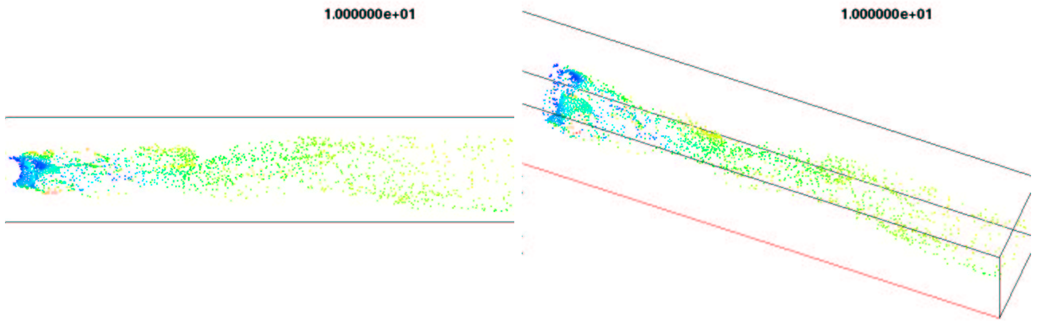


Figure 7: Flow behind a cylinder in 3D visualized with particle tracing

```
const Daten_Typ DEps=0.05;
Daten_Typ Phi1,Phi2,Phi3,Phi4,Phi5,Phi6,Phi7;
Daten_Typ Dist1,Dist2,DRad2;

if(fabs(K[3])<=0.226) {
  Phi1=Global_Time*DVel;
  Phi2=Phi1+M_PI_2;
  Phi3=Phi2+M_PI_2;
  Phi4=Phi3+M_PI_2;
  DRad2=sqrt(K[0]*K[0]+K[1]*K[1]);
  if(DRad2<=DRad) {
    if(DRad2<=1E-6) return 1; /**/
    Phi5=acos(K[0]/DRad2);
    if(K[1]<0.0) Phi5=2.0*M_PI-Phi5;
    while(Phi5<Phi1) Phi5+=2.0*M_PI;
    if((Phi5>=Phi1)&&(Phi5<=Phi2)) {
      Phi6=Phi1;
      Phi7=Phi2;
    } else if((Phi5>=Phi2)&&(Phi5<=Phi3)){
      Phi6=Phi2;
      Phi7=Phi3;
    } else if((Phi5>=Phi3)&&(Phi5<=Phi4)){
      Phi6=Phi3;
      Phi7=Phi4;
    } else if(Phi5>=Phi4){
      Phi6=Phi4;
      Phi7=Phi1;
    } else {
      fprintf(stderr,"Schrott!!!!!!  %.8f, %.8f\n",Phi5,Phi1);
```

```
          return 0;/**/
        }
        Dist1=DRad2*sin(Phi5-Phi6);
        Dist2=DRad2*sin(Phi7-Phi5);
        if((fabs(Dist1)<=DEps)||(fabs(Dist2)<=DEps)) return 1;
      }
    }
    return 0;
  }
```

The content of the configuration file is listed below. See Figure 8 for the output.

```
119000 1.33333338E-4 500 250
1
-0.9 0.09 -0.09 -0.9 0.00 0.09 1 15 30 250 0 0 100000000 0.0 16.0
```

5 Outlook

GMVPT 1.2 is the recent version with a rich variety of configurations for particle tracing as postprocessing tool. Additionally, the following improvements for the next version are planned:

1. Better format of the configuration file.
2. A GUI environment for GMVPT.
3. Better visualization of moving boundaries in 3D.
4. Adaptive time steps.
5. Ability to use adaptive meshes in time and space.

References

[1] ***FEATFLOW***. *Finite element software for the incompressible Navier-Stokes equations: User Manual. Release 1.2*, 1999

[2] J. F. Acker, S. Turek: *Arbeiten mit* ***GMV*** *unter* ***FEATFLOW***, Preprint 98–50, U Heidelberg, SFB 359, 1998

[3] J. F. Acker, S. Turek: *3D presentation of* ***FEATFLOW*** *data with* ***GMV***, Preprint 99–19, U Heidelberg, SFB 359, 1999

[4] S. Buijssen, S. Turek: *Batch-oriented* ***MPEG*** *generation with* ***GMV*** *in background mode*, Preprint 99–29, U Heidelberg, SFB 359, 1999

[5] ***GMV*** *(**G**eneral **M**esh **V**iewer) user manual. Release 2.2*, 2000 (see also: *http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html*)

See also *http://www.featflow.de* for downloads of our documentations and preprints.

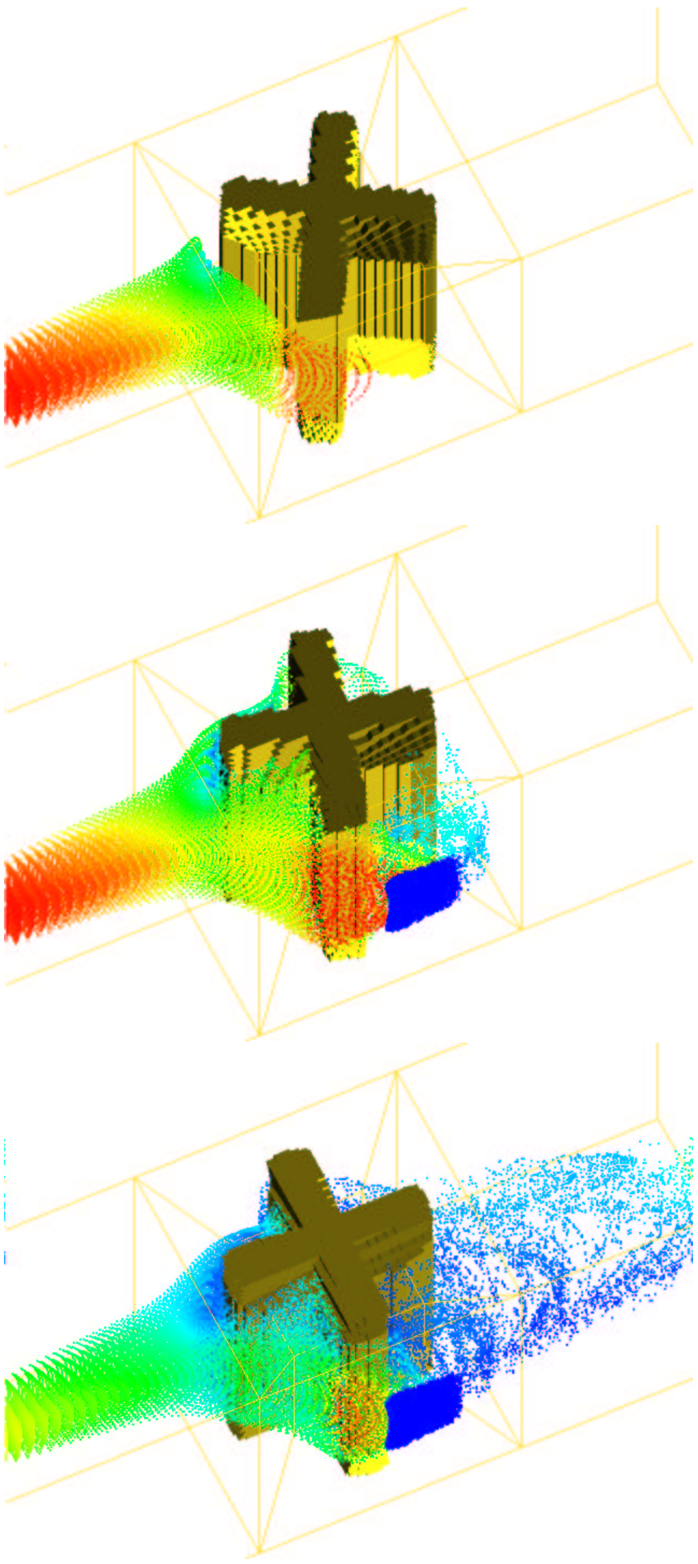


Figure 8: Flow around a moving rotor in 3D. Particles colored by norm of velocities.