Institute for Applied Mathematics
Heidelberg University
Prof. Dr. Thomas Richter

# Problem Set 1 – Gascoigne Workshop
# Summer Term 2013

## Preliminaries

This workshop is based on a tutorial by Thomas Richter. The script is available for download under

http://numerik.uni-hd.de/~lehre/SS13/gascoigne/pdf-files/gascoignescript.pdf

For initialization of Gascoigne, we need to copy one configuration file into the home directory. Open a terminal window and type:

```
cd ~
cp /usr/share/doc/gascoigne/gascoignerc .gascoignerc
```

Next, we create a new folder in the home directory and move there by the following commands:

```
  mkdir MyGas
  cd MyGas
```

There is a template program for every problem set. Each set must be copied to the above created directory, e.g.:

```
  cp -r /srv/share/gascoigne/2013/ps1 .
```

This first introductory example gathers all required functions to solve a simple PDE with Gascoigne. The two basic ingredients of every program are the classes `Problem` and the `Loop` (see section *“A minimal example solving a partial differential equation“* in the script for detailed information).

## Problem 1.1:

**Compilation** In order to compile the program, proceed as follows:

```
  cd ps1/bin
  cmake ../src
  make
```

For detailed informations see the chapter *“Installation“* in the script.

**Start** The first program solves the Laplace equation on the unit square using different levels of mesh refinement. To start it type:

```
  cd ../src
  ../bin/PS1
```

**Visualization** To visualize the solution use the following line:

```
visusimple Results/u.00001.vtk
```

Try to visualize different refinement levels by changing the number in the last command (e.g., `Results/u.00003.vtk`).
Try different options in Visusimple:

(a) Click `Actor->Carpet` and then `Parameter->Carpet->Scale->0.50` in the menu bar for a better visualization.

(b) Click `Actor->ScalarBar` and then `DetailedParameter->ScalarBar` to try different positions and options for the legend.

(c) Click `File->Export As->Tiff` to save the visualization of the solution as an image (e.g., `image.tif`). You can open the saved image with the following line:

```
gwenview Results/image.tif
```

How does the program behave if you change the number of iterations in the parameter file? To this end, open the file

```
kate run.param
```

What happens for larger values of `niter` (this is the number of iterations) and `prerefine` (this is the number of mesh refinement steps for the first iteration)? Be careful not to increase `prerefine+niter` to a larger value than 10! The memory of the computer is easily exceeded. Check the memory usage of Gascoigne by typing `top` in another terminal window.

### Problem 1.2:

Now change `niter` and `prerefine` back to 4 in the parameter file.

(a) What happens if you skip the pointers to `Equation`, `RightHandSide` or `DirichletData` in `Problem::BasicInit()` in the file `problem.h`?

(b) What happens, if you remove one or more of the `ReInit` lines in `Loop::run(...)` in the file `loop.cc`?

### Problem 1.3:

Open the file `ps1/src/run.param`. Gascoigne uses Newton's method both for solving nonlinear and linear equations.

Adding a block `//Block MultiLevelSolver` to the parameter file, we can adjust several parameters to control the behaviour of the Newton-method:

```
//Block MultiLevelSolver
 nonlinear_tol       1.e-6
 nonlinear_globaltol 1.e-12
 nonlinear_maxiter   10
 nonlinear_miniter   5
```

With the parameter `nonlinear_tol` we set the tolerance of the Newton solver, it is the factor by which the initial error is to be reduced. With `nonlinear_globaltol` a tolerance for the absolute value of the residual can be given. Gascoigne stops if one of the two conditions is fulfilled. The parameter `nonlinear_maxiter` indicates how many Newton iterations are allowed. By `nonlinear_miniter` a minimum number of iterations is prescribed. Note that Gascoigne assigns a standard value to the variables if they are not present in the `MultiLevelSolver` block.

How far can you reduce the Newton error? Look carefully at the output of Gascoigne and read section *“Output of Gascoigne“* in the script.

Each step of the Newton method requires the solution of a linear problem. We can also set similar parameters for the linear solver in the `MultiLevelSolver` block.

```
linear_tol       1.e-4
linear_globaltol 1.e-12
linear_maxiter   10
linear_miniter   5
```

Try to change these parameters and look how the Newton convergence is affected.

**Problem 1.4:**

Reset the modifications of **Problem 1.3** in the file `run.param`.

Our next goal is to refine the mesh several times in each refinement step:

(a) Add a new parameter `GlobalRefine` to the parameter file in a new block `//Block MyBlock`.

(b) Open the file `loop.cc`. The new parameter is to be read in the method `run(...)`. Reading parameters from the parameter-file is described in chapter *“The parameter file“* of the script. The name of the parameter file is accessible in the function `run(...)` by the object `_paramfile`.

(c) Now, find the line where the global refinement is done and adjust it.

**Hint**: If Gascoigne does not compile after your changes, you might miss the correct `include`-files. Probably you have to include

```
#include "paramfile.h"
#include "filescanner.h"
```

Be careful and do not set too large values for your new parameter `GlobalRefine`!