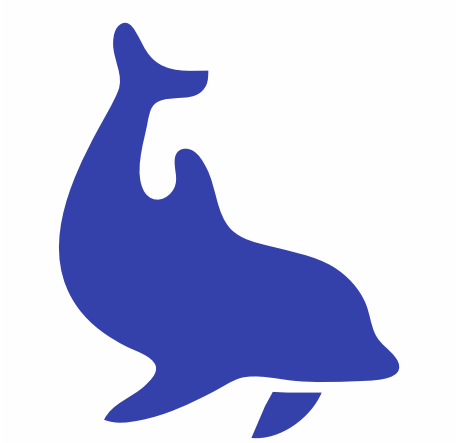
DOLFIN User Manual

September 19, 2003



Hoffman, Logg, et al.

Contents

1 Introduction

This is a first draft for a DOLFIN manual. Contributions are most welcome.

2 Installation

In preparation.

3 Linear algebra

In preparation.

4 Grid management

In preparation.

5 The log system

The purpose of the log system is to provide a simple and clean interface for logging messages, including warnings and errors.

The following functions / macros are provided for logging:

```
dolfin_info();
dolfin_debug();
dolfin_warning();
dolfin_error();
dolfin_assert();
```

Examples of usage:

```
dolfin_info("Created vector of size %d.", x.size());
dolfin_debug("Opened file");
dolfin_warning("Unknown cell type.");
dolfin_error("Out of memory.");
dolfin_assert(i < m);
```

Note that in order to pass additional arguments to the last three functions (which are really macros, in order to automatically print information about file names, line numbers and function names), the variations `dolfin_debug1()`, `dolfin_debug2()` and so on, must be used.

As an alternative to `dolfin_info()`, C++ style output to cout (`dolfin::cout`, and not `std::cout`) can be used. These messages will be delivered to the same destination as messages by use of the function `dolfin_info()`.

Examples of usage:

```
cout << "Assembling matrix: " << A << endl;
cout << "Refining grid: " << grid << endl;
```

The `dolfin_assert()` macro should be used for simple tests that may occur often, such as checking indexing in vectors. The check is turned on only if DEBUG is defined.

To notify progress by a progress session, use the class Progress.

Examples of usage:

```
Progress p("Assembling", grid.noCells());

for (CellIterator c(grid); !c.end(); ++c) {
  ...
  p++;
}
```

Progress also supports the following usage:

```
p = i;    // Specify step number
p = 0.5;  // Specify percentage
p.update(t/T, "Time is t = %f", t);
```

6 Handling parameters

In preparation.

7 Writing a new module / solver

In preparation.