# Efficient Representation of Computational Meshes

Anders Logg
logg@simula.no

Simula Research Laboratory

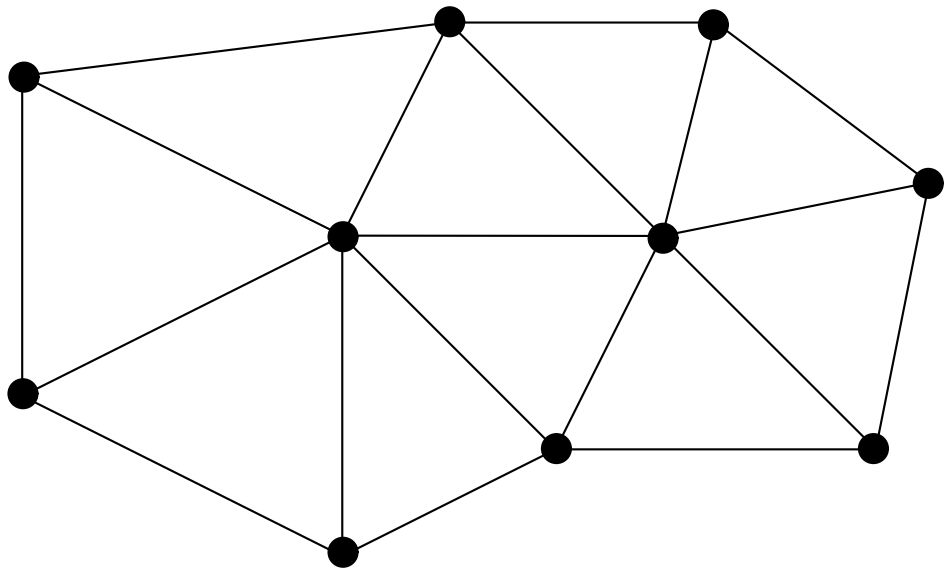
MekIT'07, Trondheim May 23–24 2007

# Computational meshes

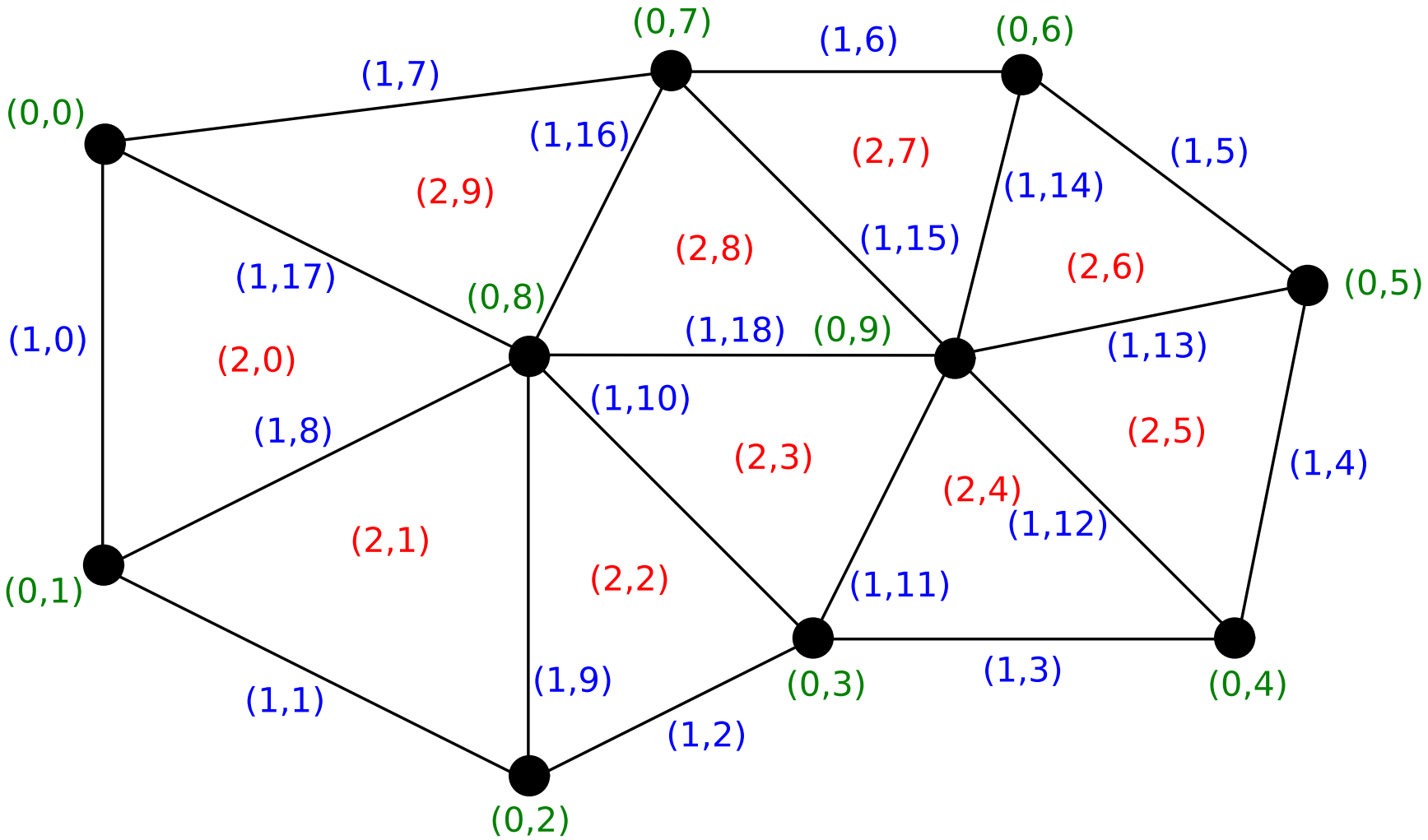
- A central component of any (mesh based) PDE solver
- Needs to be efficient
  - Fast access to mesh data
  - Minimal storage, minimal overhead
- Needs to be generic
  - 1D, 2D, 3D, $n$D
  - Simplices, quadrilaterals, hexahedra, ...
  - $n$-dimensional meshes embedded in $\mathbb{R}^d$, $d \geq n$
  - Common data structures, common user interface
- Needs to be simple
  - Specialized user interfaces
  - Simple concepts, small amount of code, easy to maintain

# Mesh abstractions

- Mesh $=$ (Topology, Geometry)
- Topology $=$ ($\{$Mesh entities$\}$, Connectivity)
- Mesh entity $=$ $(d, i)$
- Connectivity $=$ $\{$Incidence relations $d - d'\}$
- Mesh function, $f : (d, \mathbb{N}) \rightarrow V$



# Mesh entities



# Named mesh entities

| Entity | Dimension | Codimension |
|---|---|---|
| Vertex | $0$ | $D$ |
| Edge | $1$ | $D-1$ |
| Face | $2$ | $D-2$ |
| Facet | $D-1$ | $1$ |
| Cell | $D$ | $0$ |

- Mesh entity defined by a pair $(d, i)$
- Named mesh entities: Vertex, Edge, Face, Facet, Cell

# Implementation

- Implemented in C++, about 3,000 lines (core library)
- Freely available in **FEniCS**/**DOLFIN**: `www.fenics.org`
- SWIG-generated Python interface in **PyDOLFIN**/**PyCC**

- `Mesh`
- `MeshTopology`, `MeshGeometry`
- `MeshEntity`, `MeshEntityIterator`
- `Vertex`, `Edge`, `Face`, `Facet`, `Cell`
- `MeshFunction`
- `MeshEditor`
- `UnitSquare`, `UnitCube`

# Conversion to DOLFIN XML format

- Use `dolfin-convert` to convert to the DOLFIN XML format
- Conversion from Medit (tetgen) and Gmsh
- Conversion from the old DOLFIN XML format:

```
dolfin-convert --input old-xml old.xml new.xml
```

- Convert all meshes in current directory:

```
convertall
```

This script can be found in `src/utils/xml`

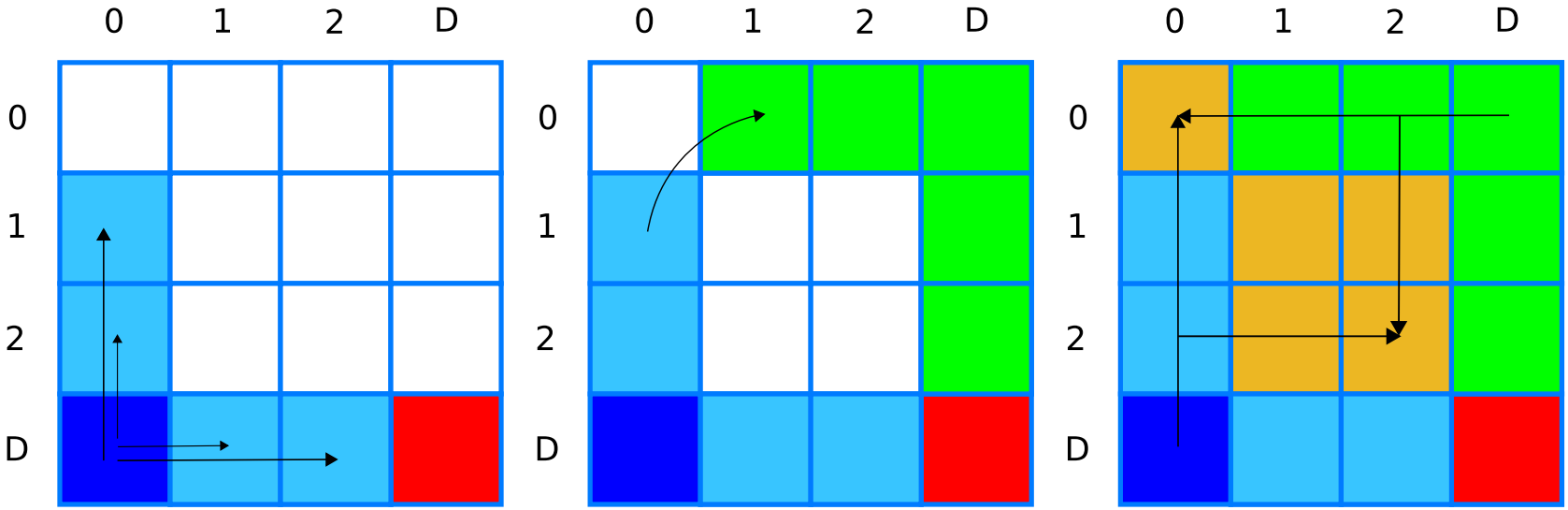
# Mesh connectivity

- Mesh of topological dimension $D$
- Connectivity $D - 0$ given (cells $-$ vertices)
- Compute the connectivity $d - d'$ for $0 \leq d, d' \leq D$
- Compute only as needed

| | 0 | 1 | 2 | | | $D$ |
|---|---|---|---|---|---|---|
| 0 | | | | | | |
| 1 | | | | | | |
| 2 | | | | | | |
| | | | | | | |
| | | | | | | |
| $D$ | X | | | | | |

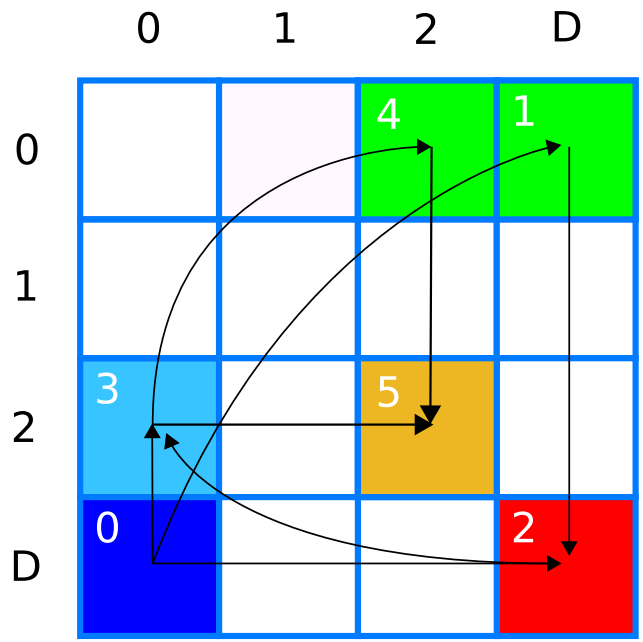
# Computing mesh connectivity

- Build $D - d$ and $d - 0$ from $D - 0$ and $D - D$ for $0 < d < D$
- Compute $d - d'$ from $d' - d$ for $d < d'$ (transpose)
- Compute $d - d'$ from $d - d''$ and $d'' - d'$ (intersection)
- All algorithms are $\mathcal{O}(n^p)\mathcal{O}(N)$ for small $n$ and $p$



# Example: computing $2 - 2$ (faces $-$ faces)

```
for f0 in in faces(mesh):
    for f1 in faces(f0):
        # Iterators automatically initialize 2 - 2
```

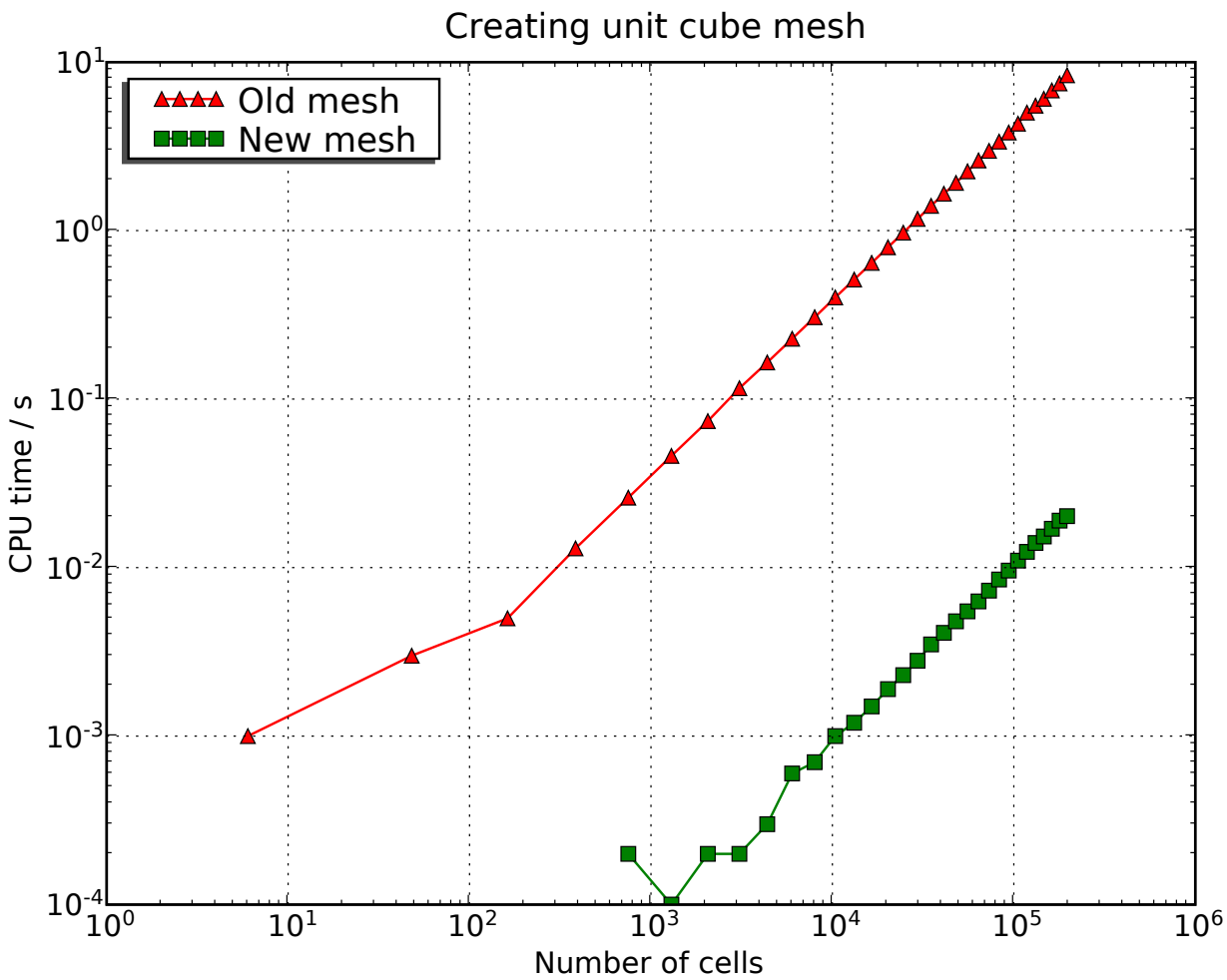


# Simple benchmarks (against DOLFIN 0.6.2)

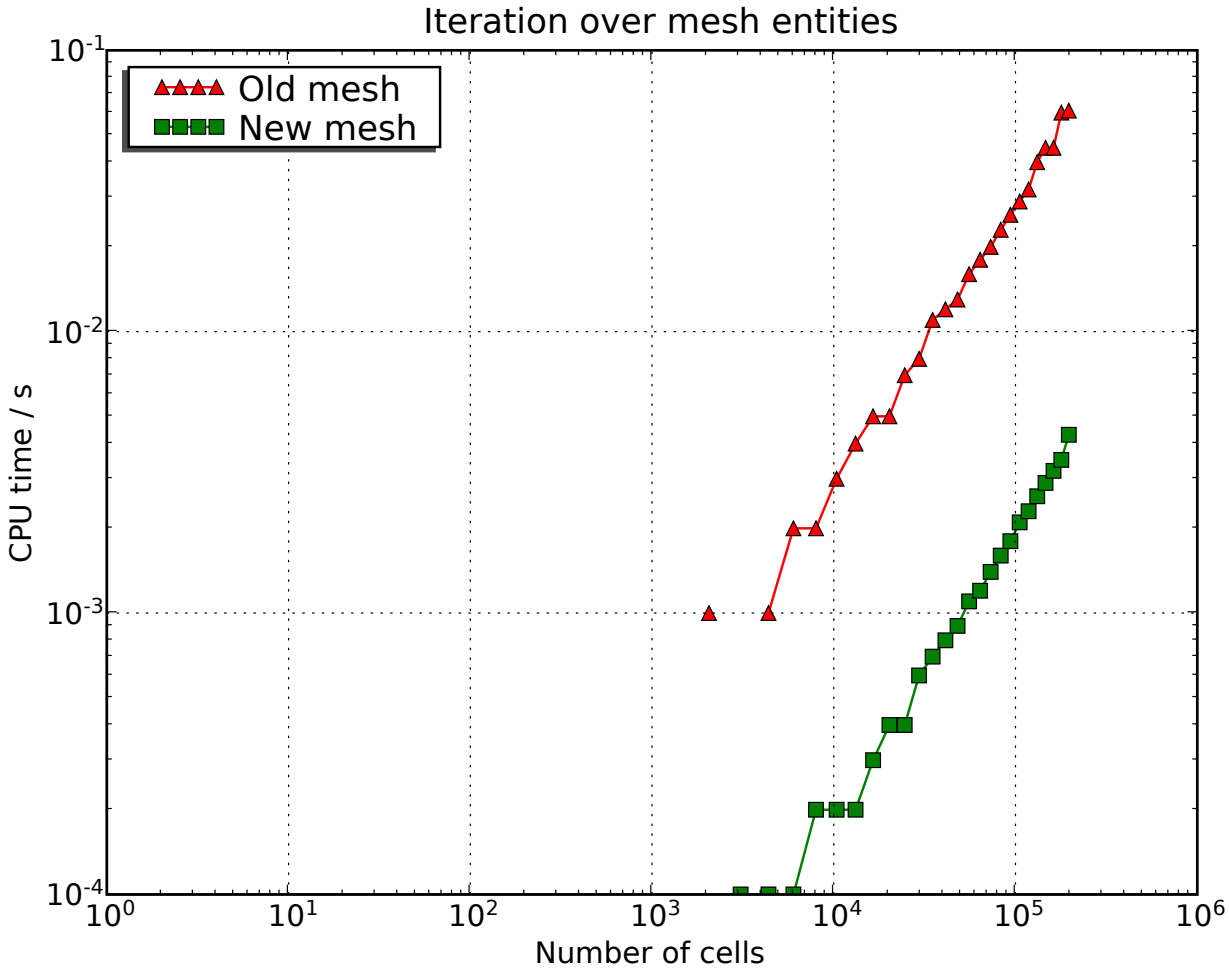
- Creating unit cube mesh
- Iteration over mesh entities
- Uniform mesh refinement
- Memory usage

- Speedup: a factor 10–100
- Reduced memory usage: a factor 10

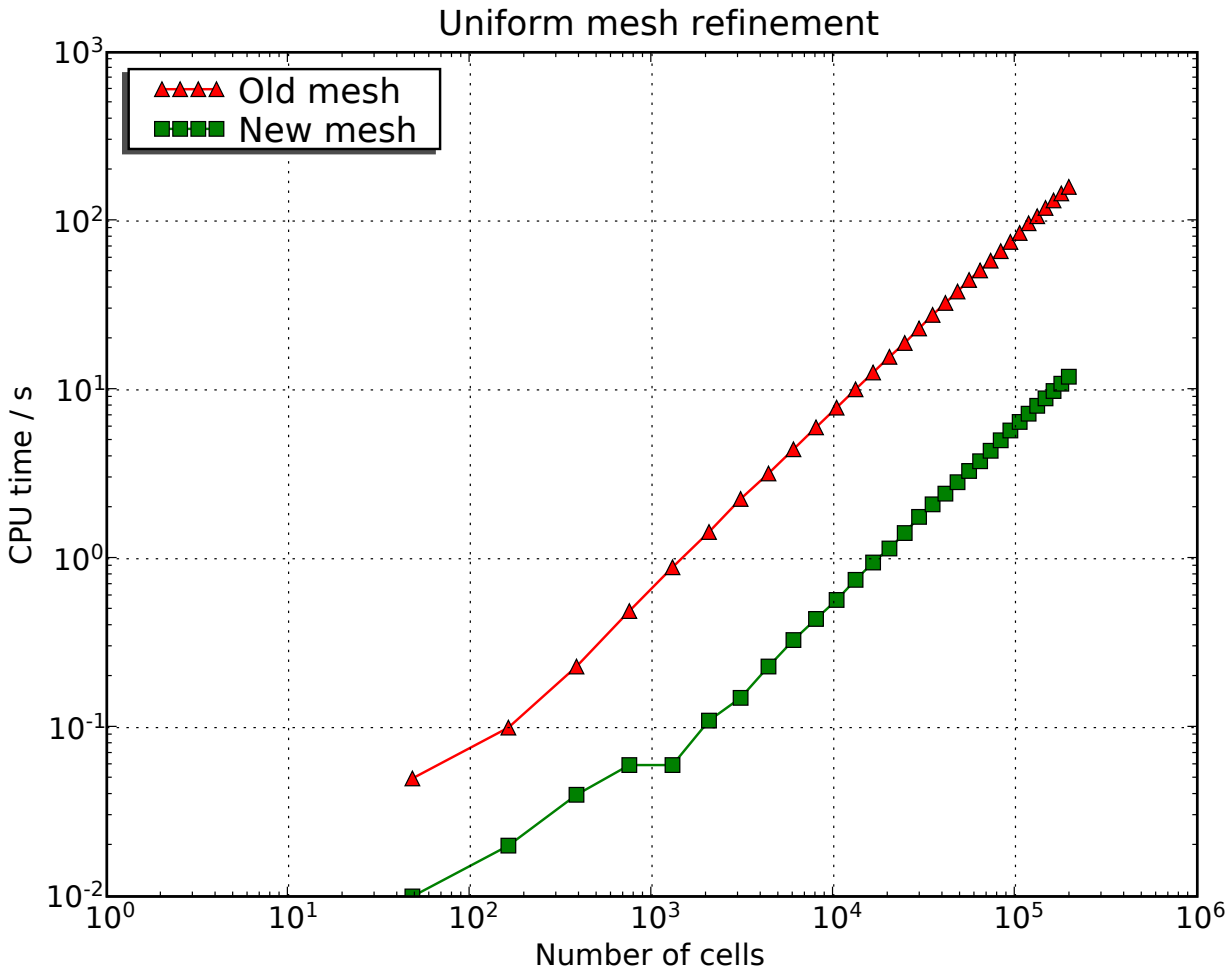
# Creating unit cube mesh



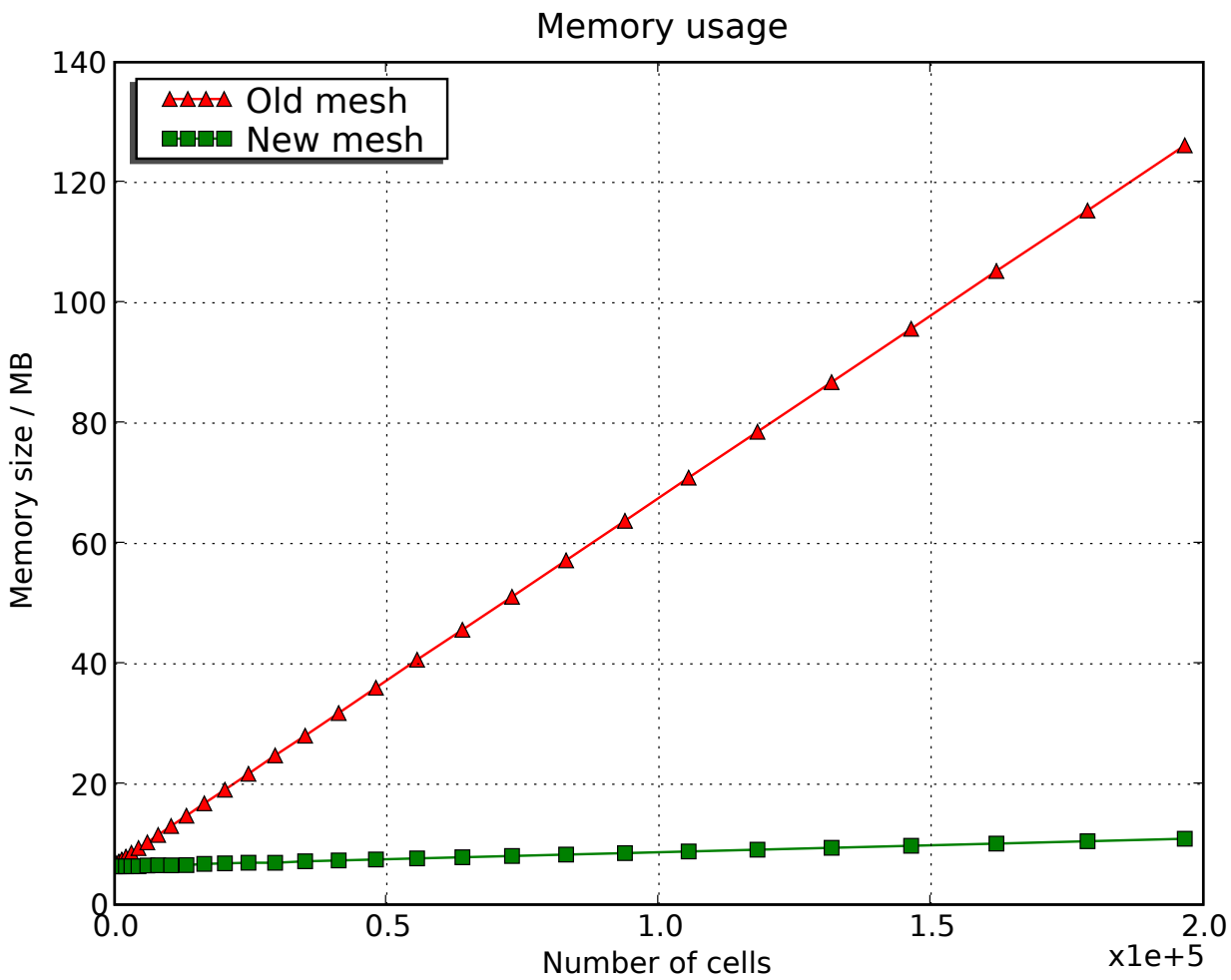
# Iteration over mesh entities (in C++)



# Uniform mesh refinement



# Memory usage



# Live demo

- Run demo*!*

# Downloading

Available as part of DOLFIN from

`http://www.fenics.org/`

Next stable release expected in **June 2007**.

Current development version available by

```
hg clone http://www.fenics.org/hg/dolfin
```