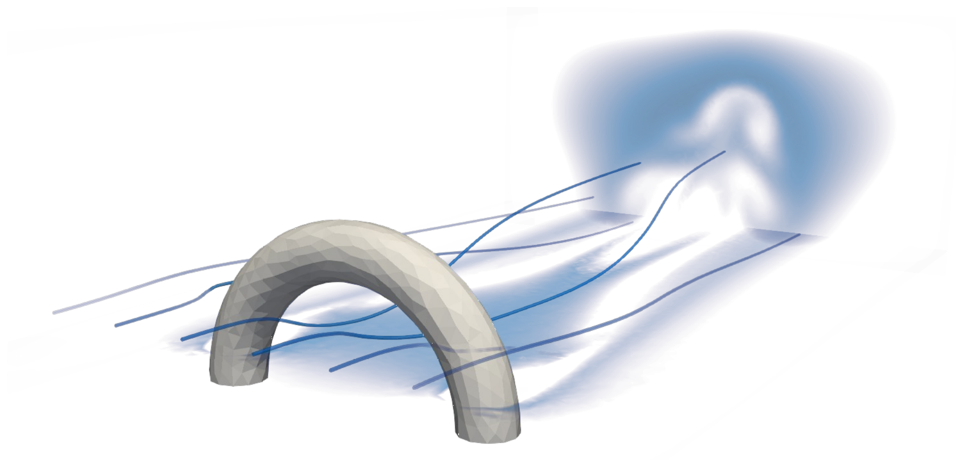
Sparselizard: a general purpose multiphysics FEM c++ library

sparselizard.org



Some history

- 2014-2016: first line of precursor Matlab FEM code @ University of Liege
- 2017: first sparselizard c++ line, starting from scratch @ University of Liege
- 2018-2019: free-time development @ IMEC (nanoelectronics, MEMS research centre)
- 2020-2024: academy of Finland grant @ Tampere University
  → particle accelerator magnet design in collaboration with CERN

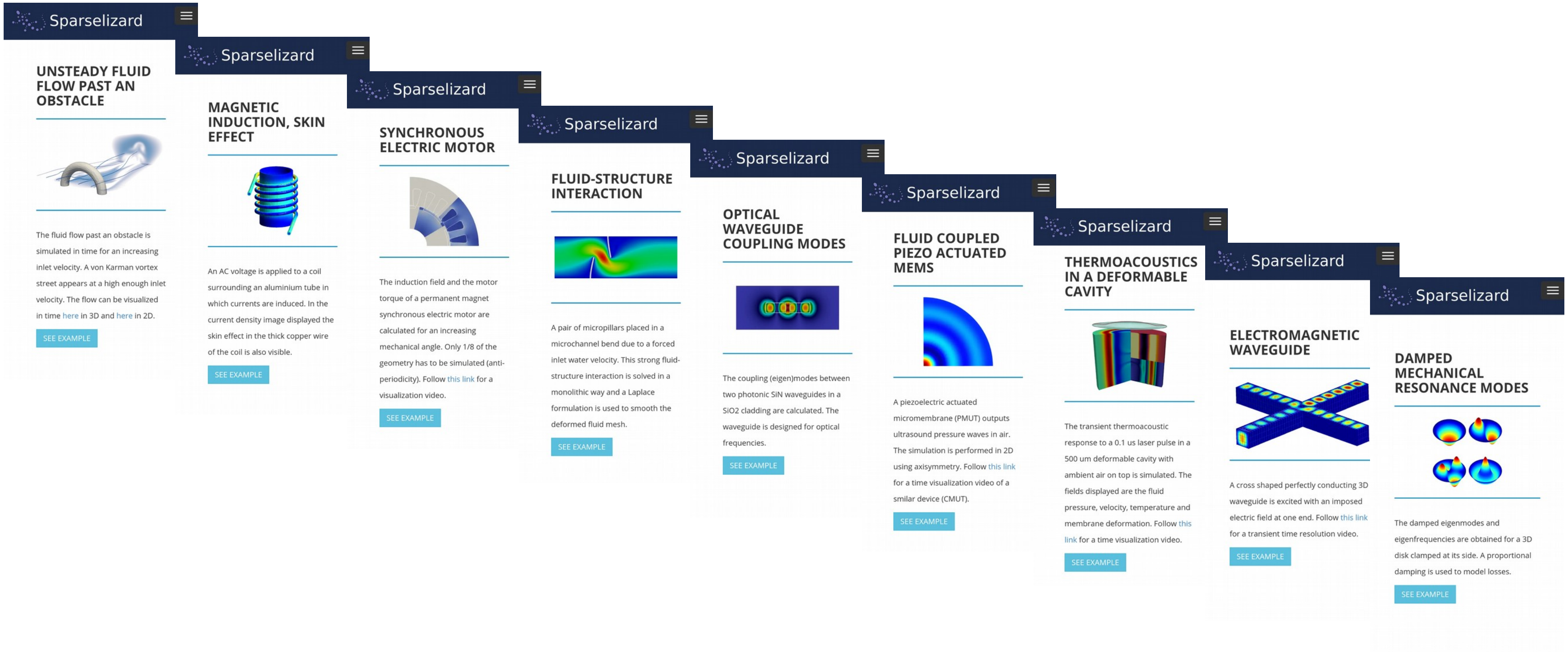
What's special to sparselizard?

- Natively designed to be highly multiphysics
- Large demonstrated capabilities
- Support of advanced FEM and extra tools
- Concise and user-friendly (even for a c++ library)
- Carefully validated and debugged
- Clearly documented
- Efficient, highly multi-threaded
- Rapidly expanding

What’s special to sparselizard?

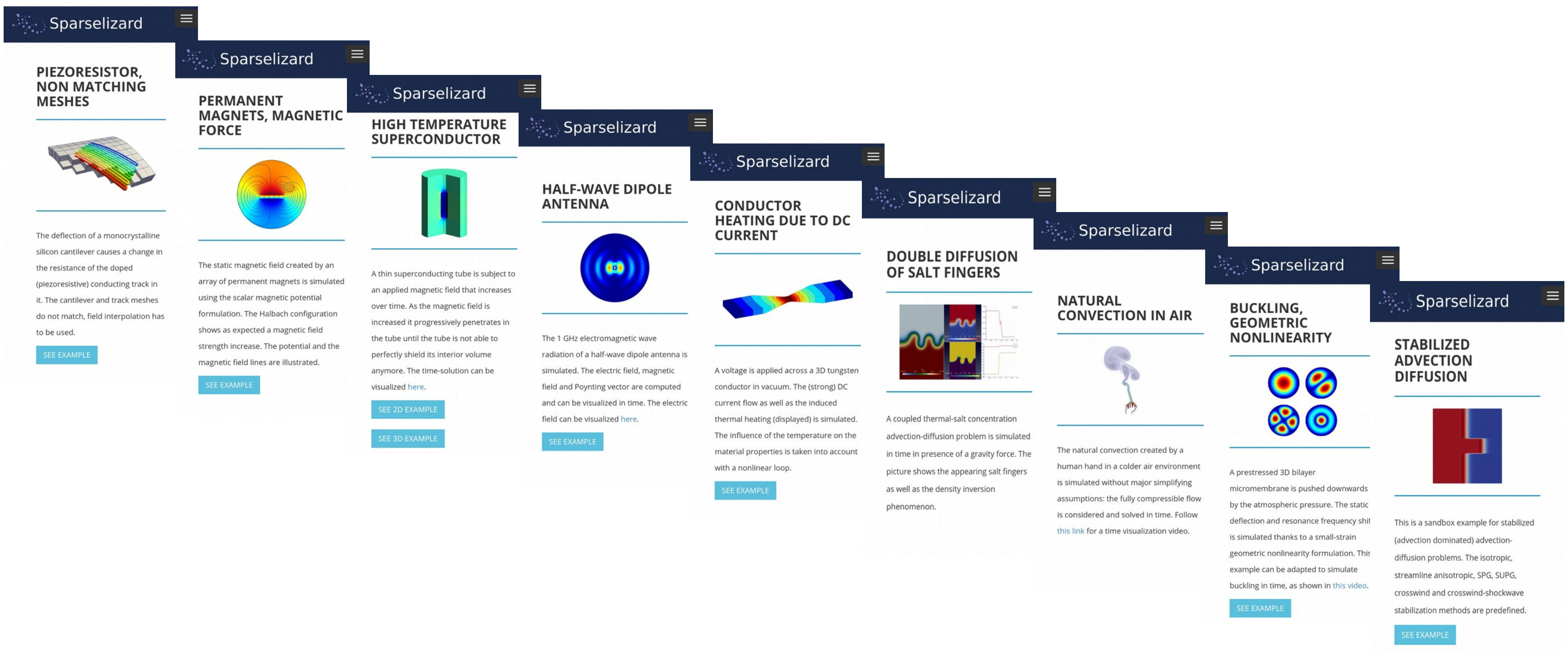
- Natively designed to be highly multiphysics
- Large demonstrated capabilities
- Support of advanced FEM and extra tools
- Concise and user-friendly (even for a c++ library)
- Carefully validated and debugged
- Clearly documented
- Efficient, highly multi-threaded
- Rapidly expanding

Natively designed to be highly multiphysics.



Fluids | Magnetics | Electricity | Mechanics | Rotating machines | Acoustics | Thermal

Demonstrated capabilities through validated examples.



Transient | Harmonic | Harmonic balance | Eigenmodes | Predefined physics

What’s special to sparselizard?

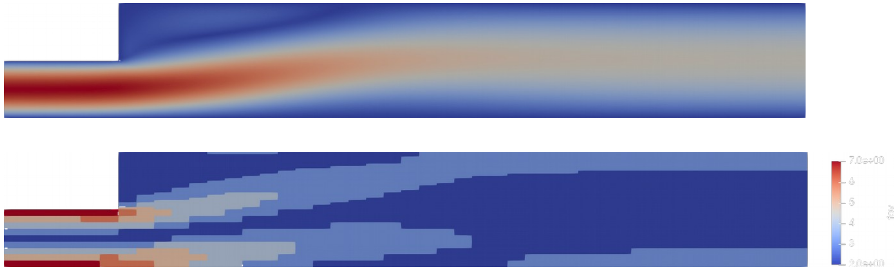
- Natively designed to be highly multiphysics
- Large demonstrated capabilities
- Support of advanced FEM and extra tools
- Concise and user-friendly (even for a c++ library)
- Carefully validated and debugged
- Clearly documented
- Efficient, highly multi-threaded
- Rapidly expanding

Support of advanced FEM and extra tools.

- Native support of harmonic balance FEM → nonlinear harmonic analysis
- Fast 3D unstructured mesh to mesh interpolation
- General 3D mortar FEM
- True p-adaptivity pFEM (to be officially released next week)
- Ultra compact .slz file format (x4 to x8 more compact than usual formats)
- One-liner probes, max, average,... functionalities
- ParaView output format
- Gmsh, Nastran,... mesh input format
- Curved meshes allowed

https://youtu.be/rdKATF6aBSw

P-adaptivity in pipe flow (order 2→ 7):



What's special to sparselizard?

- Natively designed to be highly multiphysics
- Large demonstrated capabilities
- Support of advanced FEM and extra tools
- Concise and user-friendly (even for a c++ library)
- Carefully validated and debugged
- Clearly documented
- Efficient, highly multi-threaded
- Rapidly expanding

Concise. User-friendly.

- No advanced knowledge of c++ required
- No hacks
- High readability

Working 3D electrostatic example:

```
int vol = 1, sur = 2;              // Disk volume and boundary as set in 'disk.geo'
mesh mymesh("disk.msh");

field v("h1");                     // Nodal shape functions for the electric potential
v.setorder(vol, 2);                // Interpolation order 2 on the whole domain
v.setconstraint(sur, 0);           // Force 0 V on the disk boundary

formulation elec;                  // Electrostatics with 1 nC/m^3 charge density
elec += integral(vol, -8.85e-12 * grad(dof(v)) * grad(tf(v)) + 1e-9 * tf(v));

solve(elec);                       // Generate, solve and save solution to field v

(-grad(v)).write(vol, "E.vtk");    // Write the electric field to ParaView format
```

Documented. For humans.

- All functions/predefined physics are clearly documented
- A working example gets you started

Functions are illustrated with simple working examples.

```
double integrate(int physreg, int integrationorder)
```

```
mesh mymesh("disk.msh");
int vol = 1;
expression myexpression = 12.0;
// Mesh with curved elements at order 3 to accurately
// capture the cylinder geometry and get value 3.7699!
double integralvalue = myexpression.integrate(vol, 4);
```

This integrates the expression over the geometric region 1. The integration is exact for up to 4th order polynomials.

---

sparselizard.org

github.com/halbux/sparselizard