A Compiler for Variational Forms

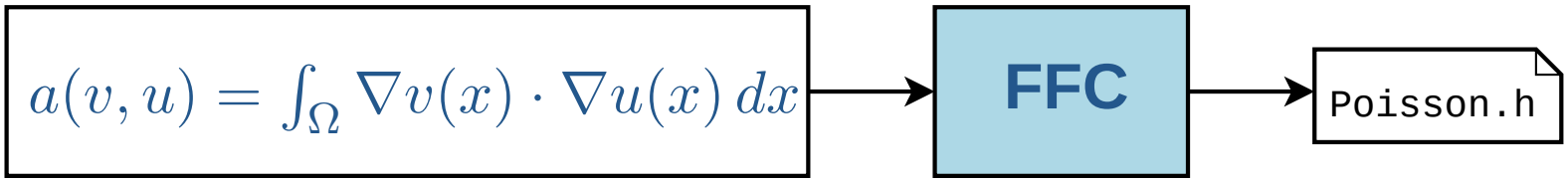
Anders Logg

logg@tti-c.org

Toyota Technological Institute at Chicago

FFC: the FEniCS Form Compiler

- Automates a key step in the implementation of finite element methods for partial differential equations
- Input: a variational form and a finite element
- Output: optimal C/C++ code




```
>> ffc [-l language] poisson.form
```

Design goals

Primary goals:

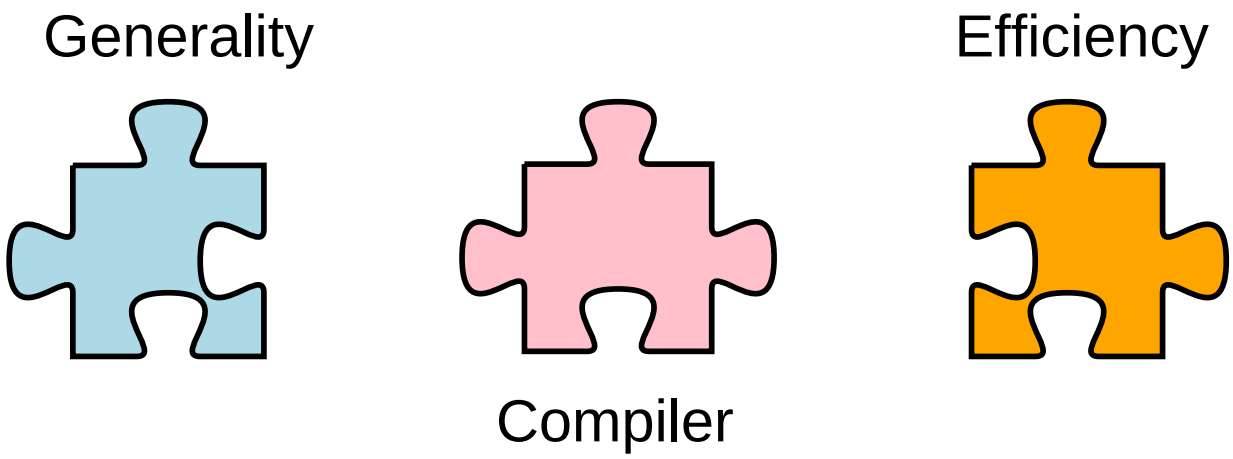
- Any form
- Any element
- Maximum efficiency

Secondary goals:

- Create a new standard in form evaluation
- Integration with other **FEniCS** projects (Robert Kirby, UofC)
- Integration with PETSc tools (Matthew Knepley, Argonne)

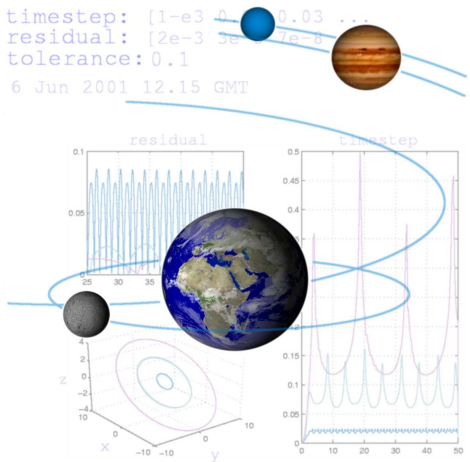
Why a compiler?

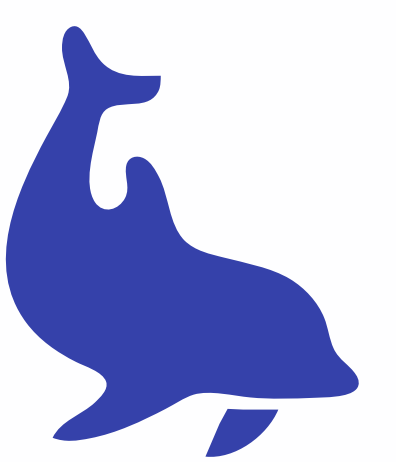
- Add a new level of automation
  - Don’t need to write the code anymore
  - Less chance of making mistakes
- Add a new level of generality (any form, any element)
- Add a new level of efficiency
  - New techniques of form evaluation
  - New techniques of optimization

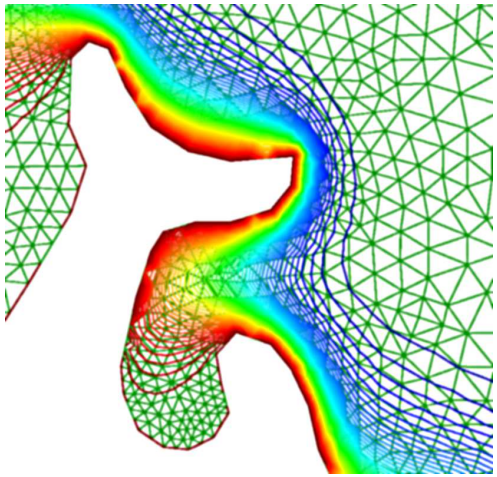


Outline

- Background
- Form evaluation
- Implementation
- Demonstration
- The **FEniCS** project







Background

Main goal: the Automation of CMM

- Input: Model $Au = f$ and tolerance $\mathrm{TOL} > 0$
- Output: Solution $U \approx u$ satisfying $\|U - u\| \leq \mathrm{TOL}$

- Produce a solution U satisfying a given accuracy requirement, using a minimal amount of work

A key step: the automation of discretization

- Input: Model $Au = f$ and discrete representation $(V, \hat{V})$
- Output: Discrete system $F(x) = 0$

- Produce a discrete system $F(x) = 0$ corresponding to the model $Au = f$ for the given discrete representation $(V, \hat{V})$

Basic algorithm: the (Galerkin) finite element method

- Input: Variational problem $a(v,u) = L(v)$ for all v and discrete representation $(V, \hat{V})$
- Output: Discrete system $F(x) = 0$

Basic example: Poisson's equation

- Strong form: Find $u \in \mathcal{C}^2(\overline{\Omega})$ with $u = 0$ on $\partial\Omega$ such that

$$-\Delta u = f \quad \text{in } \Omega$$

- Weak form: Find $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla v(x) \cdot \nabla u(x)\, dx = \int_\Omega v(x) f(x)\, dx \quad \text{for all } v \in H_0^1(\Omega)$$

- Standard notation: Find $u \in V$ such that

$$a(v, u) = L(v) \quad \text{for all } v \in \hat{V}$$

with $a : V \times \hat{V} \to \mathbb{R}$ a *bilinear form* and $L : \hat{V} \to \mathbb{R}$ a *linear form* (functional)

Obtaining the discrete system

Let V and $\hat{V}$ be discrete function spaces. Then

$$a(v, U) = L(v) \quad \text{for all } v \in \hat{V}$$

is a discrete linear system for the approximate solution $U \approx u$. With $V = \text{span}\{\phi_i\}_{i=1}^{M}$ and $\hat{V} = \text{span}\{\hat{\phi}_i\}_{i=1}^{M}$, we obtain the linear system

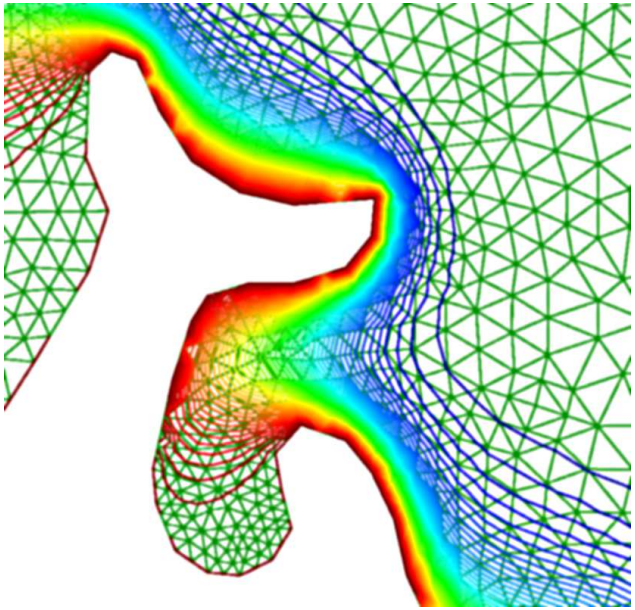
$$Ax = b$$

for the degrees of freedom $x = (x_i)$ of $U = \sum_{i=1}^{M} x_i \phi_i$, where

$$A_{ij} = a(\hat{\phi}_i, \phi_j)$$

$$b_i = L(\hat{\phi}_i)$$

Computing the linear system: assembly



Noting that $a(v,u) = \sum_{K \in \mathcal{T}} a_K(v,u)$, the matrix A can be assembled by

$$A = 0$$
$$\texttt{for all elements } K \in \mathcal{T}$$
$$\quad A \texttt{ += } A^K$$

The *element matrix* A^K is defined by

$$A_{ij}^K = a_K(\hat{\phi}_i, \phi_j)$$

for all local basis functions $\hat{\phi}_i$ and ϕ_j on K

Form evaluation

Multi-linear forms

Consider a multi-linear form

$$a : V_1 \times V_2 \times \cdots \times V_n \to \mathbb{R}$$

with $V_1, V_2, \ldots, V_n$ function spaces on the domain Ω

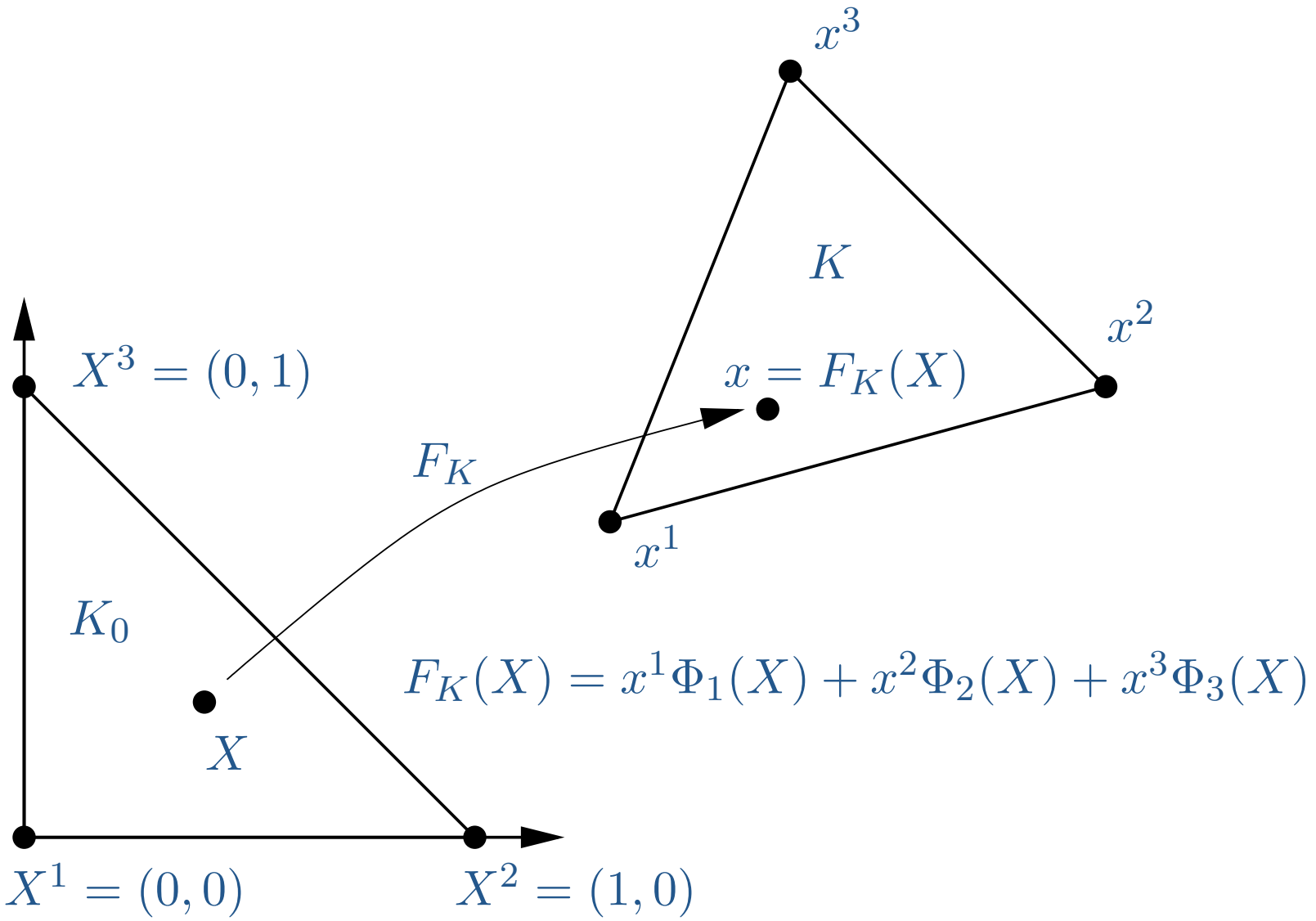
- Typically, $n = 1$ (linear form) or $n = 2$ (bilinear form)
- Assume $V_1 = V_2 = \cdots = V_n = V$ for ease of notation

Want to compute the rank n *element tensor* A^K defined by

$$A_i^K = a_K(\phi_{i_1}, \phi_{i_2}, \ldots, \phi_{i_n})$$

with $\{\phi_i\}_{i=1}^d$ the local basis on K and multi-index $i = (i_1, i_2, \ldots, i_n)$

The (affine) map $F_K : K_0 \rightarrow K$



Example 1: the mass matrix

- Form:

$$a(v,u) = \int_\Omega v(x)u(x)\,dx$$

- Evaluation:

$$A_i^K = \int_K \phi_{i_1}\phi_{i_2}\,dx$$

$$= \det F_K' \int_{K_0} \Phi_{i_1}(X)\Phi_{i_2}(X)\,dX = A_i^0 G_K$$

with $A_i^0 = \int_{K_0} \Phi_{i_1}(X)\Phi_{i_2}(X)\,dX$ and $G_K = \det F_K'$

Example 2: Poisson

- Form:

$$a(v,u) = \int_\Omega \nabla v(x) \cdot \nabla u(x)\, dx$$

- Evaluation:

$$A_i^K = \int_K \nabla \phi_{i_1}(x) \cdot \nabla \phi_{i_2}(x)\, dx$$

$$= \det F_K' \frac{\partial X_{\alpha_1}}{\partial x_\beta} \frac{\partial X_{\alpha_2}}{\partial x_\beta} \int_{K_0} \frac{\partial \Phi_{i_1}}{\partial X_{\alpha_1}} \frac{\partial \Phi_{i_2}}{\partial X_{\alpha_2}}\, dX = A_{i\alpha}^0 G_K^\alpha$$

with $A_{i\alpha}^0 = \int_{K_0} \frac{\partial \Phi_{i_1}}{\partial X_{\alpha_1}} \frac{\partial \Phi_{i_2}}{\partial X_{\alpha_2}}\, dX$ and $G_K^\alpha = \det F_K' \frac{\partial X_{\alpha_1}}{\partial x_\beta} \frac{\partial X_{\alpha_2}}{\partial x_\beta}$

Example 3: Navier–Stokes

- Form:

$$a(v,u) = \int_\Omega v \cdot (w \cdot \nabla) u \, dx$$

- Evaluation:

$$A_i^K = \int_K \phi_{i_1} \cdot (w \cdot \nabla) \phi_{i_2} \, dx$$

$$= \det F_K' \frac{\partial X_{\alpha_3}}{\partial x_{\alpha_1}} w_{\alpha_2} \int_{K_0} \Phi_{i_1}[\beta] \Phi_{\alpha_2}[\alpha_1] \frac{\partial \Phi_{i_2}[\beta]}{\partial X_{\alpha_3}} \, dX = A_{i\alpha}^0 G_K^\alpha$$

with $A_{i\alpha}^0 = \int_{K_0} \Phi_{i_1}[\beta] \Phi_{\alpha_2}[\alpha_1] \frac{\partial \Phi_{i_2}[\beta]}{\partial X_{\alpha_3}} \, dX$ and

$G_K^\alpha = \det F_K' \frac{\partial X_{\alpha_3}}{\partial x_{\alpha_1}} w_{\alpha_2}$

Tensor representation

In general, the element tensor A^K can be represented as the product of a *reference tensor* A^0 and a *geometry tensor* G_K:

$$A_i^K = A_{i\alpha}^0 G_K^\alpha$$

- A^0: a tensor of rank $|i| + |\alpha| = n + |\alpha|$
- G_K: a tensor of rank $|\alpha|$

Basic idea:

- Precompute A^0 at compile-time
- Generate optimal code for run-time evaluation of G_K and the product $A_{i\alpha}^0 G_K^\alpha$

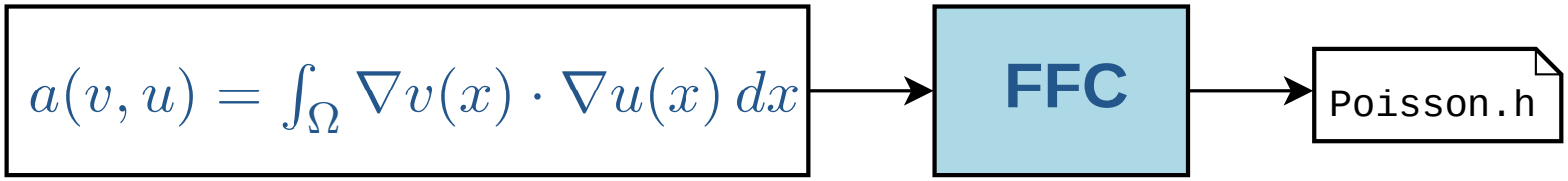
Example: Poisson with $\mathcal{P}^2$ elements in 2D

3 3 3 3	1 0 1 0	0 1 0 1	0 0 0 0	0 -4 0 -4	-4 0 -4 0
1 1 0 0	3 0 0 0	0 -1 0 0	0 4 0 0	0 0 0 0	-4 -4 0 0
0 0 1 1	0 0 -1 0	0 0 0 3	0 0 4 0	0 0 -4 -4	0 0 0 0
0 0 0 0	0 0 4 0	0 4 0 0	8 4 4 8	-8 -4 -4 0	0 -4 -4 -8
0 0 -4 -4	0 0 0 0	0 -4 0 -4	-8 -4 -4 0	8 4 4 8	0 4 4 0
-4 -4 0 0	-4 0 -4 0	0 0 0 0	0 -4 -4 -8	0 4 4 0	8 4 4 8

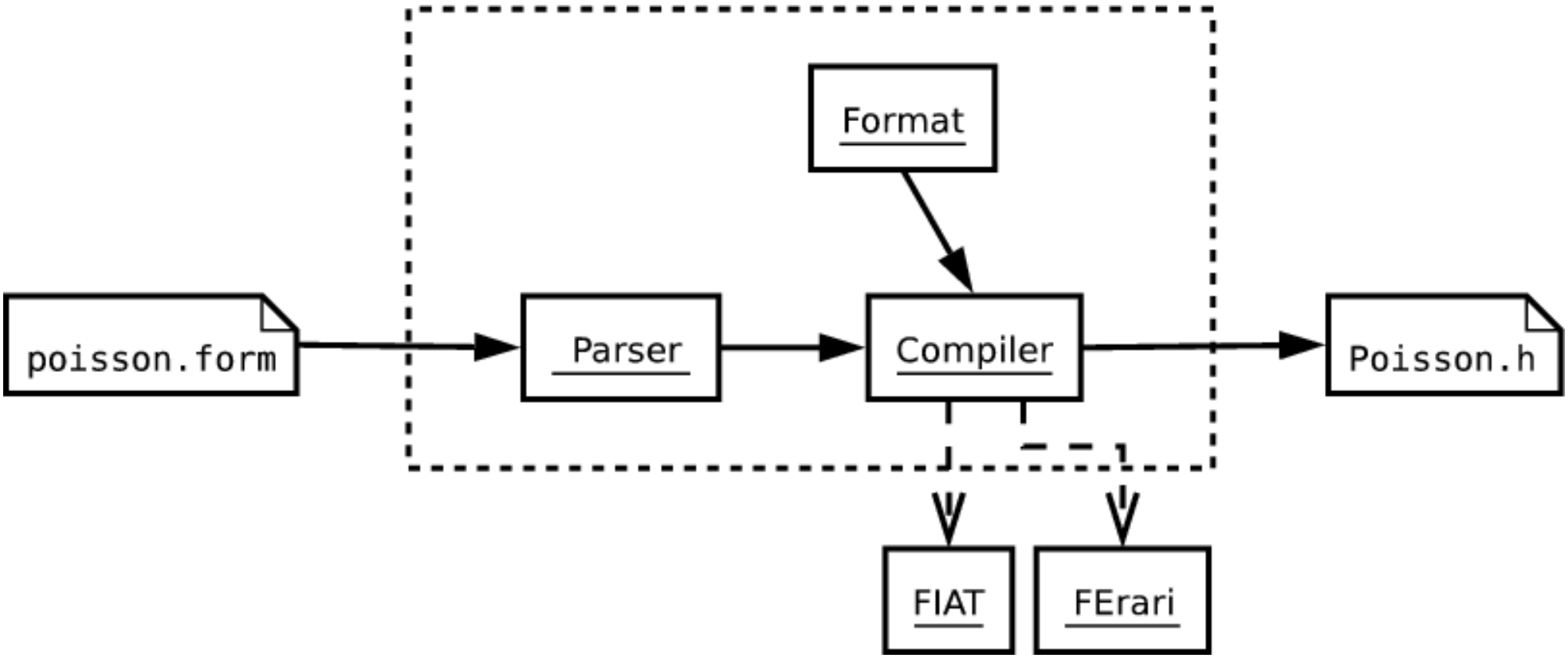
Implementation

FFC

- Implemented in Python
- Access: Python module or command-line interface
- **FIAT** used as the finite element backend
- Optimization through **FErari**
- Supports multiple output languages (formats)



Components



Representation of forms

- Need to build a data structure to represent forms:

$$V_B = \{\frac{\partial^{|\cdot|}\phi_{(\cdot)}}{\partial x_{(\cdot)}} : \quad \phi : \mathbb{N} \to V\}$$

$$V_P = \{c\prod v : \quad v \in V_B, \quad c \in \mathbb{R}\}$$

$$V_S = \{\sum v : \quad v \in V_P\}$$

  Note: $V_B \subset V_P \subset V_S \subset \{v : \mathbb{N}^r \times \Omega \to \mathbb{R}\}$

- V_S is an algebra (a vector space with multiplication):

$$v, w \in V_S \Rightarrow v + w \in V_S$$

$$v, w \in V_S \Rightarrow vw \in V_S$$

Evaluation of forms

For any $v = \sum c \prod \partial^{|\cdot|}\phi_{(\cdot)}/\partial x_{(\cdot)}$, we have

$$
\begin{aligned}
A_i^K = a_K(\phi_{i_1}, \phi_{i_2}, \ldots, \phi_{i_n}) &= \int_K v_i \, dx \\
&= \sum \left(\int_K c \prod \partial^{|\cdot|}\phi_{(\cdot)}/\partial x_{(\cdot)} \, dx \right)_i \\
&= \sum c'_\alpha \left(\int_{K_0} \prod \partial^{|\cdot|}\Phi_{(\cdot)}/\partial X_{(\cdot)} \, dX \right)_{i\alpha} \\
&= \sum A^0_{i\alpha} G^\alpha_K,
\end{aligned}
$$

where $A^0_{i\alpha} = \left(\int_{K_0} \prod \partial^{|\cdot|}\Phi_{(\cdot)}/\partial X_{(\cdot)} \, dX \right)_{i\alpha}$ and $G^\alpha_K = c'_\alpha$

Implementation

- Build a class hierarchy: `BasisFunction`, `Product`, `Sum` corresponding to the spaces $V_B \subset V_P \subset V_S$
- Overload operators `+`, `-`, `*`, `[·]`, `.dx(·)` :

$$\begin{cases} \texttt{+} : V_B \times V_B \to V_S \\ \texttt{+} : V_B \times V_P \to V_S \\ \quad \ldots \\ \texttt{+} : V_S \times V_S \to V_S \end{cases} \qquad \begin{cases} \texttt{*} : V_B \times V_B \to V_P \\ \texttt{*} : V_B \times V_P \to V_P \\ \quad \ldots \\ \texttt{*} : V_S \times V_S \to V_S \end{cases}$$

- Examples:

```
a = v*u*dx
a = v.dx(i)*u.dx(i)*dx
a = v[i]*w[j]*u[i].dx(j)*dx
L = v*f*dx + v*10*ds
```

Demonstration

Examples

- $a(v,u) = \int_\Omega vu\,dx$
- $a(v,u) = \int_\Omega \nabla v \cdot \nabla u\,dx = \int_\Omega \frac{\partial v}{\partial x_i}\frac{\partial u}{\partial x_i}\,dx$
- $a(v,u) = \int_\Omega v \cdot (w \cdot \nabla)u\,dx = \int_\Omega v_i w_j \partial u_i/\partial x_j\,dx$

- Run demonstration

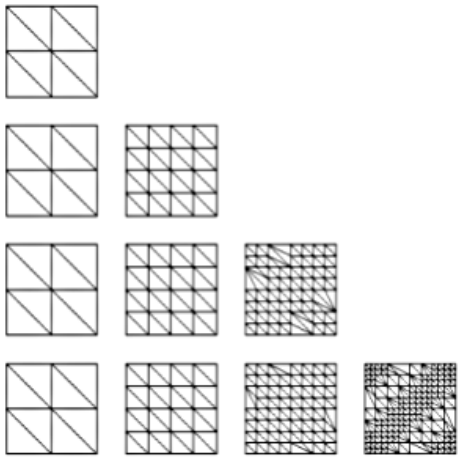
The FEniCS project

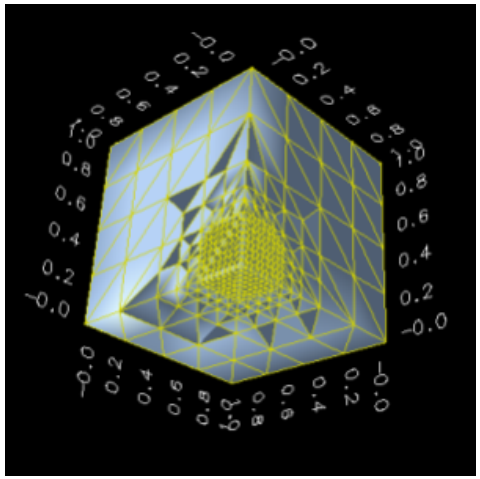
The **FEniCS** project

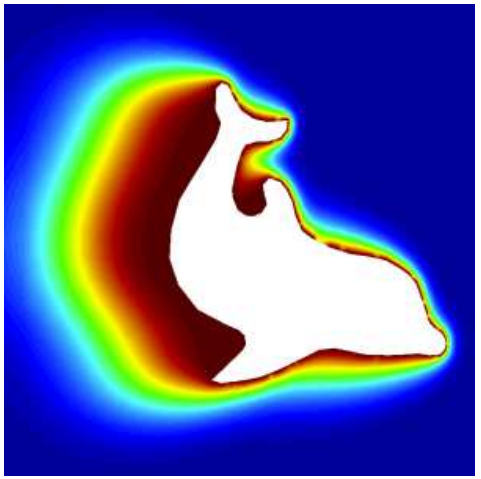
FEniCS is free software for the Automation of Computational Mathematical Modeling (ACMM), initiated in cooperation between

- The Toyota Technological Institute at Chicago
- The University of Chicago
- Chalmers University of Technology (Göteborg, Sweden)

More information at `http://www.fenics.org`







People

- Todd Dupont, University of Chicago
- Johan Hoffman, Chalmers University of Technology
- Johan Jansson, Chalmers University of Technology
- Claes Johnson, Chalmers University of Technology
- Matthew Knepley, Argonne National Laboratory
- Robert C. Kirby, University of Chicago
- Mats G. Larson, Umeå University
- Anders Logg, Toyota Technological Institute at Chicago
- Ridgway Scott, University of Chicago
- F. Bengzon, N. Ericsson, G. Foufas, D. Heintz, R. Hemph, P. Ingelström, K. Kraft, A. Krusper, A. Mark, A. Nilsson, E. Svensson, J. Tilander, T. Svedberg, H. Svensson, W. Villanueva

Projects

- **DOLFIN**, the C++ interface of FEniCS
  - Hoffman, Jansson, Logg, et al.
- **FErari**, optimized form evaluation
  - Kirby, Knepley, Scott
- **FFC**, the FEniCS Form Compiler
  - Logg
- **FIAT**, automatic generation of finite elements
  - Kirby, Knepley
- **Ko**, simulation of mechanical systems
  - Jansson
- **Puffin**, light-weight version for Octave/MATLAB
  - Hoffman, Logg

TODO list

- Handle boundary terms correctly
- Integrate **FFC** with **DOLFIN**
- Integrate **FFC** with **FErari**
- Add a (non-trivial) parser
- Manual, installation, man page, ...

- Further information:

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

Additional slides

The Automation of CMM