UFL - The Unified Form Language

Easy Finite Element Discretization of Variational Forms Based on Tensor Algebra and Automatic Differentiation

Martin Sandve Alnæs
martinal@simula.no

[simula . research laboratory]

June 19, 2009

UFL is part of the FEniCS project

Long term FEniCS[1] goal

Automation of Mathematical Modeling

Contributions from UFL[2]

- Improvements to *Automation of Discretization*.
- Add *Automation of Linearization*.

[1] http://www.fenics.org
[2] http://www.fenics.org/ufl/

Motivational example

Expressing PDEs close to mathematical notation!

Poissons equation

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\Omega} f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Typical workflow

Or how to combine UFL with a DOLFIN based C++ program

- Write forms in `.ufl` file:

```
a = dot(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds
```

Typical workflow

Or how to combine UFL with a DOLFIN based C++ program

- Write forms in `.ufl` file:

```
a = dot(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds
```

- Run form compiler to produce C++ code:

```
sfc Poisson.ufl
```

Typical workflow

Or how to combine UFL with a DOLFIN based C++ program

- Write forms in `.ufl` file:

```
a = dot(grad(u), grad(v))*dx
L = f*v*dx + g*v*ds
```

- Run form compiler to produce C++ code:

```
sfc Poisson.ufl
```

- Include a generated header file in your C++ code

```
#include "Poisson.h"
int main() {
    Poisson::Form_a a(...);
    Poisson::Form_L L(...);
}
```

A complete UFL file for a projection

```
V1 = FiniteElement("Lagrange", triangle, 1)
V2 = FiniteElement("Lagrange", triangle, 2)

v = TestFunction(V1)
u = TrialFunction(V1)
f = Function(V2)

a = u*v*dx
L = f*v*dx
```

$$u \in V_1, \quad f \in V_2,$$
$$(u, v) = (f, v), \quad \forall v \in V_1$$

Running a form compiler to produce C++ code

```
sfc -w1 Projection.ufl
```

Produces `Projection.h` and some other C++ files.

Include one generated header file in your C++ code

```
#include <dolfin.h>
#include "Projection.h"
int main() {
    dolfin::UnitSquare mesh(10,10);
    ...
}
```

Defining function spaces on a mesh in C++

```
#include <dolfin.h>
#include "Projection.h"
int main() {
    ...
    Projection::Form_a::TrialSpace V1(mesh);
    Projection::CoefficientSpace_f V2(mesh);
    ...
}
```

Projection.ufl →
Projection.h →
Projection::

Defining forms on a function space

```
#include <dolfin.h>
#include "Projection.h"
int main() {
    ...
    Projection::Form_a a(V1, V1);
    Projection::Form_L L(V1);
    ...
}
```

a = u*v*dx $\longrightarrow$ Form_a

Attaching functions to forms

```
#include <dolfin.h>
#include "Projection.h"
int main() {
    ...
    MyFunction f(V2);
    L.f = f;
    ...
}
```

L = f*v*dx $\longrightarrow$ L.f

Solving a variational problem in DOLFIN

```
#include <dolfin.h>
#include "Projection.h"
int main() {
    ...
    dolfin::VariationalProblem problem(a, L);
    dolfin::Function u(V1);
    problem.solve(u);
}
```

Example revisited

Form arguments

Poissons equation

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\Omega} f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

A standard definition of a finite element

A finite element is a triplet $(K, \mathcal{P}_K, \{\ell_i\})$:

- a polygon (or cell) K,
- a polynomial space $\mathcal{P}_K$ on K,
- a set of degrees of freedom, $\{\ell_i : \mathcal{P}_K \to \mathcal{R},\ i = 1, 2, \ldots, n_K\}$.

In UFL, cells are defined by shape and degree

Examples of valid cells in UFL are

- cell = Cell(shape, degree)
- cell = Cell("triangle", 1)
- cell = Cell("triangle", 2)
- cell = triangle
- interval, triangle, tetrahedron, quadrilateral, hexahedron

Polynomial spaces are defined by family and degree

A basic element

element = FiniteElement(family, cell, degree)

Polynomial spaces are defined by family and degree

A basic element

element = FiniteElement(family, cell, degree)

Some of the valid families

- "Lagrange" or "CG"
- "Discontinuous Lagrange" or "DG"
- "Brezzi-Douglas-Marini" or "BDM"

Polynomial spaces are defined by family and degree

A basic element

element = FiniteElement(family, cell, degree)

Some of the valid families

- "Lagrange" or "CG"
- "Discontinuous Lagrange" or "DG"
- "Brezzi-Douglas-Marini" or "BDM"

Note that

$\{$ `family` , `degree` $\} \leftrightarrow \{ \mathcal{P}_K , \ell_i \}$

Basic finite elements can be combined

A vector element

element = VectorElement(family, cell, degree)
element = VectorElement(family, cell, degree, dim)

A tensor element

element = TensorElement(family, cell, degree)
element = TensorElement(family, cell, degree, symmetry=True)

Mixed elements can be arbitrarily nested

A mixed element (Taylor-Hood)

V = VectorElement("Lagrange", cell, 2)
P = FiniteElement("Lagrange", cell, 1)
TH = V + P

Mixed elements can be arbitrarily nested

A mixed element (Taylor-Hood)

V = VectorElement("Lagrange", cell, 2)
P = FiniteElement("Lagrange", cell, 1)
TH = V + P

Another one

T = TensorElement("Lagrange", cell, 1, symmetry=True)
V = VectorElement("Lagrange", cell, 2)
P = FiniteElement("Lagrange", cell, 1)
TH = MixedElement(T, V, P)

There are two classes of form arguments (1/2)

Basis Functions

v = TestFunction(element)
u = TrialFunction(element)
w = BasisFunction(element)

Each `BasisFunction` represents *any and all* $\phi \in V_h$, and corresponds to an axis in the element tensor

$$A_{ij} = a(v_i, u_j).$$

There are two classes of form arguments (2/2)

Functions

f = Function(element)
c = Constant(cell)
c = VectorConstant(cell[, dim])
c = TensorConstant(cell[, shape, symmetry])

Each `Function` represents a fixed function

$$f(x) = \sum_i f_i \phi_i(x) \in V_h,$$

with f_i a set of unknown coefficients.

Other terminal/atomic values

- Scalars: 1.23, 3.14159, 2
- Identity matrix: I = Identity(cell.d)
- Spatial coordinates: x = cell.x
- Facet normal: n = cell.n

Example revisited

Index notation

Poissons equation

$$\int_\Omega u_{,i} v_{,i} \, dx - \int_\Omega f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = Dx(u, i)*Dx(v, i)*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Basic indexing of tensor valued expressions

Scalar components

```
v2 = v[2]
vi = v[i]
Aij = A[i,j]
Ai0 = A[i,0]
```

Slicing

```
Arow0 = A[0,:]
Acol1 = A[:,1]
```

Defining indices for index notation

Predefined indices

i, j, k, l and p, q, r, s.

Creating new indices

```
i = Index()
i, j, k = indices(3)
```

Building vectors from scalars

1. $\mathbf{v} = \sum_k u_k \mathbf{i}_k$
2. $\mathbf{v} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}$

Building vectors from scalars

1. $\mathbf{v} = \sum_k u_k \mathbf{i}_k$
2. $\mathbf{v} = \begin{bmatrix} 1.0 \\ 2.0 \\ 3.0 \end{bmatrix}$

```
1 v = as_vector(u[k], k)
2 v = as_vector([1.0, 2.0, 3.0])
```

Building tensors from scalars

1. $\mathbf{A} = \sum_i \sum_j B_{ij} \mathbf{i}_i \mathbf{i}_j$
2. $\mathbf{A} = \left[\begin{array}{cc} 1.0 & 2.0 \\ 3.0 & 4.0 \end{array} \right]$

```
1 A = as_tensor(Bij, (i,j))
2 A = as_tensor([[1.0, 2.0],
                 [3.0, 4.0]])
```

Any UFL expression v has some basic properties

- Its value shape v.shape()
  - () for a scalar
  - (2,) for a 2D vector, e.g., triangle.n
  - (3,3) for a 3 by 3 matrix

Any UFL expression v has some basic properties

- Its value shape v.shape()
  - () for a scalar
  - (2,) for a 2D vector, e.g., triangle.n
  - (3,3) for a 3 by 3 matrix
- Its free indices v.free_indices()
  - (i,) for v[i]
  - (i,j) for A[i,j]
  - () for something without free indices, e.g., `v[1]` or `v[i]*v[i]`

Example revisited

Algebra operators

Poissons equation

$$\int_\Omega \nabla u \cdot \nabla v \, dx - \int_\Omega f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Basic algebra operators

Operand requirements for the basic operators:

`a + b` Same shape, same indices.

`a - b` Same shape, same indices.

`a * b` `a*b`, `a*C`, `C*a`, `A*v`, `A*B`

`a / b` Scalar denominator only.

Here `a` and `b` are scalar expressions, `v` is a vector, `A` is a matrix, and `C` is a tensor of arbitrary rank.

Basic tensor algebraic operators

`dot(a, b)` $\mathbf{i}_i \cdot \mathbf{i}_j = \delta_{ij}, \quad \mathbf{v} \cdot \mathbf{u} = v_k u_k, \quad \mathbf{A} \cdot \mathbf{B} = A_{ik} B_{kj} \mathbf{i}_i \mathbf{i}_j$

`inner(a, b)` $\mathbf{i}_i \mathbf{i}_j : \mathbf{i}_k \mathbf{i}_l = \delta_{ik} \delta_{jl}, \quad \mathbf{A} : \mathbf{B} = A_{ij} B_{ij}$

`outer(a, b)` $\mathbf{i}_i \otimes \mathbf{i}_j = \mathbf{i}_i \mathbf{i}_j, \quad \mathbf{A} \otimes \mathbf{B} = A_{ij} B_{kl} \mathbf{i}_i \mathbf{i}_j \mathbf{i}_k \mathbf{i}_l,$

`cross(a, b)` $\mathbf{v} \times \mathbf{u}$

More tensor algebraic operators

In all the following, $\mathbf{A}$ is a rank two tensor (matrix) with no free indices.

`tr(A)`	$tr\,\mathbf{A} = A_{ii} \equiv \sum_i A_{ii}$
`det(A)`	$det\,\mathbf{A} = \lvert\mathbf{A}\rvert$
`inv(A)`	$inv\,\mathbf{A} = \mathbf{A}^{-1}$
`cofac(A)`	$cofac\,\mathbf{A} = \lvert\mathbf{A}\rvert\mathbf{A}^{-1}$
`sym(A)`	$sym\,\mathbf{A} = \frac{1}{2}(\mathbf{A} + \mathbf{A}^T)$
`skew(A)`	$skew\,\mathbf{A} = \frac{1}{2}(\mathbf{A} - \mathbf{A}^T)$

Some other operators

Some nonlinear operators, all operating on scalar expressions with no free indices.

- `pow(f, g)` same as `f**g`
- `sqrt(f)` same as `f**0.5`
- `exp(f)`
- `ln(f)`
- `sin(f)`
- `cos(f)`

Discontinuous Galerkin (DG) operators

Restriction of an expression to one side of a facet:

`v('+')`	v_+
`v('-')`	v_-
`avg(v)`	$\frac{1}{2}(v_+ + v_-)$
`jump(v)`	$v_+ - v_-$
`jump(v,n)`	$v_+ \cdot n - v_- \cdot n$

Conditional expressions

Conditions

`lt(a,b)` $a < b$

`gt(a,b)` $a > b$

`le(a,b)` $a \leq b$

`ge(a,b)` $a \geq b$

`eq(a,b)` $a = b$

`ne(a,b)` $a \neq b$

Conditional

`g = conditional(c,t,f)`

$$g = \begin{cases} t, & \text{if} \quad t, \\ f, & \text{otherwise} \end{cases}$$

Example

`conditional(lt(g0,g1),g0,g1)`

$$\min\{g_0, g_1\}$$

Example revisited

Differential operators

Poissons equation

$$\int_\Omega \nabla u \cdot \nabla v \, dx - \int_\Omega f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Basic differential operators

For any expression f,

$$\frac{\partial f}{\partial x_i}$$

can be written in two ways:

- `Dx(v, i)`
- `v.dx(i)`

where `i` can be either an `Index` or an integer.

Compound differential operators

Defining $\nabla = \frac{\partial}{\partial x_k}\mathbf{i}_k$:

`div(A)`	$\nabla \cdot \mathbf{A} = A_{ij,i}\mathbf{i}_j$
`grad(v)`	$\nabla \mathbf{v} = \nabla \otimes \mathbf{v} = v_{j,i}\mathbf{i}_i\mathbf{i}_j$
`curl(v)`	$\nabla \times \mathbf{v}$
`rot(v)`	$\nabla \times \mathbf{v}$ FIXME

Differentiation w.r.t. a user defined variable

- Define an expression the variable should take on the value of

```
y = exp(g.dx(0)**2))
```

- Declare a variable

```
z = variable(y)
```

- Define an expression as a function of z

```
f = sin(z**4/2)
```

- Differentiate!

```
df = diff(f, z)
```

Example revisited
Integrals

Poissons equation

$$\int_{\Omega} \nabla u \cdot \nabla v \, dx - \int_{\Omega} f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Example revisited
Form transformations

Poissons equation

$$\int_\Omega \nabla u \cdot \nabla v \, dx - \int_\Omega f v \, dx - \int_{\partial\Omega} g v \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```

Transforming forms to form new forms

a = lhs(pde)	Extract left and right hand side terms of PDE.
L = rhs(pde)	Extract left and right hand side terms of PDE.
L = action(a, f)	Compute action of a bilinear form on a function.
M = action(L, f)	Compute “action” of a linear form on a function.
ap = adjoint(a)	Compute adjoint of a bilinear form.

Differentiating forms

L = derivative(M, f) Compute residual equation from functional.

a = derivative(L, f) Linearize residual equation.

This closes the language part of the talk

Questions?

Poissons equation

$$\int_\Omega \nabla u \cdot \nabla v \, dx - \int_\Omega fv \, dx - \int_{\partial\Omega} gv \, ds = 0$$

UFL notation

```
pde = dot(grad(u), grad(v))*dx - f*v*dx - g*v*ds
a, L = lhs(pde), rhs(pde)
```