Symbolic numerical computing

Lawrence Mitchell[1,*]

C.J. Cotter, D.A. Ham, M. Homolya, T. Kärnä, P.H.J. Kelly, R.C. Kirby, E.H. Müller, ...

25th January 2018

[1]Departments of Computing and Mathematics, Imperial College London

*lawrence.mitchell@imperial.ac.uk

EPSRC
Engineering and Physical Sciences
Research Council

Imperial College
London

Outline

- Overview of some recent research
    - Fast solvers for numerical weather prediction
    - Automated finite elements
- In depth
    - Compiling finite element problems
- Future directions

Recent highlights

Numerical weather prediction

$$\begin{pmatrix} M_2 & -\frac{\Delta t}{2} D^T & -\frac{\Delta t}{2} Q \\ \frac{\Delta t}{2} c^2 D & M_3 & 0 \\ \frac{\Delta t}{2} N^2 Q^T & 0 & M_b \end{pmatrix} \begin{pmatrix} \boldsymbol{U} \\ \boldsymbol{P} \\ \boldsymbol{B} \end{pmatrix} = \begin{pmatrix} M_2 \boldsymbol{R}_u \\ M_3 \boldsymbol{R}_p \\ M_b \boldsymbol{R}_b \end{pmatrix}$$

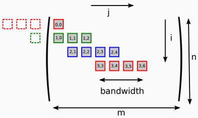
Challenge

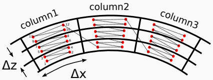
Invert elliptic operator fast enough for operational forecasting.

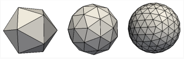
High aspect ratio: black-box numerical solvers struggle.

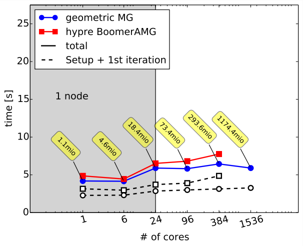
Need scalable, custom multigrid solver.

Multigrid approach









Developed custom tensor-product multigrid scheme, showed robustness and scalability.

Research impact

Approach is used by UK Met Office in their next generation dynamical core.

Providing a weather forecast to you in 2025.

Automated finite elements

Challenge

- Simulation software needs to exploit *fine-grained* parallelism.
- Most code intimately intertwines the numerical algorithm with its *implementation*.
- To apply program transformations, we have to unpick, understand, and reimplement.
- *Every time* the hardware changes.
- Most researchers do not have the skills necessary to be application and HPC experts.

Firedrake `www.firedrakeproject.org`

> [...] *an automated system for the solution of partial differential equations using the finite element method.*

- Written in Python.
- Finite element problems specified with embedded domain specific language.
- Domain-specific optimising compiler.
- Runtime compilation to low-level (C) code.
- Transparently parallel.

F. Rathgeber, D.A. Ham, **LM**, M. Lange, F. Luporini, A.T.T. McRae, G.-T. Bercea, G.R. Markall, P.H.J. Kelly. ACM Transactions on Mathematical Software, 2016.

`arXiv: 1501.01809 [cs.MS]`

Highlights

- Dramatically simplifies numerical model development. Often from months/years to days/weeks.
- Delivers “better than most humans” computational performance.
- New code separates mathematics from implementation. More portable to future architectures.
- *Enables* productive interdisciplinary collaboration.

Future research direction

- Code transformations for efficient execution on GPUs

Impact & use

Academic

- Used by research groups at Imperial, Baylor, Kiel, Exeter, Oxford, Leeds, Waterloo, Buffalo, Washington, …
- Teaching tool at Waterloo, Imperial.

Industrial

- Guides design of computational and numerical schemes in UK Met Office's next forecasting system.
- Optimisation of tidal turbine array placement.

Compiling finite element problems

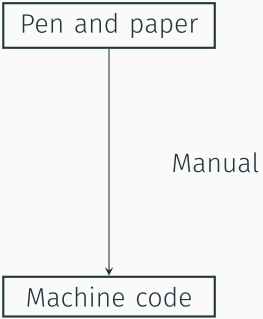
Exploit mathematical abstractions

Compute $y \leftarrow \nabla^2 x$ using finite differences.

$$y_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1} - 4x_{i,j}$$

Before 1953

```
...
faddp   %st, %st(1)
movl    -8(%ebp), %edx
movl    %edx, %eax
sall    $2, %eax
addl    %edx, %eax
leal    0(,%eax,4), %edx
addl    %edx, %eax
sall    $2, %eax
movl    %eax, %edx
movl    -4(%ebp), %eax
addl    %edx, %eax
subl    $101, %eax
flds    x.3305(,%eax,4)
flds    .LC0
fmulp   %st, %st(1)
faddp   %st, %st(1)
fstps   y.3307(,%ecx,4)
...
```



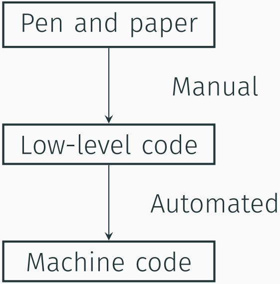
Exploit mathematical abstractions

Compute $y \leftarrow \nabla^2 x$ using finite differences.

$$y_{i,j} = x_{i-1,j} + x_{i+1,j} + x_{i,j-1} + x_{i,j+1} - 4x_{i,j}$$

1953–present: Formula Translation

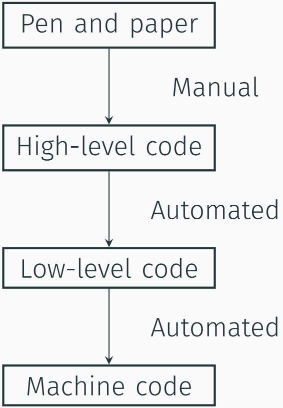
```
      PROGRAM MAIN
      PARAMETER (N=100)
      REAL X(N,N), Y(N,N)
      [...]
      DO 10 J=2,N-1
         DO 20 I=2,N-1
            Y(I,J)=X(I-1,J)+X(I+1,J)+
     $           X(I,J-1)+X(I,J+1)+4*X(I,J)
20       CONTINUE
10    CONTINUE
      [...]
      END
```



Fit to the mathematics

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, dx \quad \forall v \in V$$

```
V = FiniteElement("Lagrange",
                  triangle, 1)
u = TrialFunction(V)
v = TestFunction(V)
F = dot(grad(u), grad(v))*dx
```



TSFC: an optimising compiler for finite elements

Translate UFL into low-level code for performing an element integral.

M. Homolya, **LM**, F. Luporini, D.A. Ham. `arXiv:1705.03667 [cs.MS]`

- Element integral

$$\int_e \nabla u \cdot \nabla v \,\mathrm{d}x$$

```
V = FiniteElement("Lagrange", triangle, 1)
u = TrialFunction(V)
v = TestFunction(V)
a = dot(grad(u), grad(v))*dx
```

- Is transformed to a tensor algebra expression

$$\sum_q w_q \, |d| \sum_{i_5} \left(\sum_{i_3} K_{i_3,i_5} \begin{bmatrix} E^{(1)}_{q,k} & E^{(2)}_{q,k} \end{bmatrix}_{i_3} \right) \left(\sum_{i_4} K_{i_4,i_5} \begin{bmatrix} E^{(1)}_{q,j} & E^{(2)}_{q,j} \end{bmatrix}_{i_4} \right)$$

- Multiple optimisation passes aim to minimise FLOPs required to evaluate this expression.

TSFC: an optimising compiler for finite elements

Translate UFL into low-level code for performing an element integral.

M. Homolya, **LM**, F. Luporini, D.A. Ham. arXiv:1705.03667 [cs.MS]

```
void cell_integral(double A[3][3],
                   double coords[3][2]) {
  static const double t10[3] = {...};
  static const double t12[3] = {...};
  double t13[3];
  double t14[3];
  double t0 = (-1 * coords[0][1]);
  double t1 = (t0 + coords[1][1]);
  double t2 = (-1 * coords[0][0]);
  double t3 = (t2 + coords[1][0]);
  double t4 = (t0 + coords[2][1]);
  double t5 = (t2 + coords[2][0]);
  double t6 = ((t3 * t4) + (-1 * (t5 * t1)));
  double t7 = ((-1 * t1) / t6);
  double t8 = (t4 / t6);
  double t9 = (t3 / t6);
  double t11 = ((-1 * t5) / t6);
  for (int k0 = 0; k0 < 3; k0++) {
    t13[k0] = (t11 * t12[k0]) + (t9 * t10[k0]);
    t14[k0] = (t8 * t12[k0]) + (t7 * t10[k0]);
  }
  double t15 = (0.5 * fabs(t6));
  for (int j0 = 0; j0 < 3; j0++) {
    double t16 = ((t11 * t12[j0])
                 + (t9 * t10[j0]));
    double t17 = ((t8 * t12[j0])
                 + (t7 * t10[j0]));
    for (int k0 = 0; k0 < 3; k0++) {
      A[j0][k0] += t15 * ((t17 * t14[k0])
                         + (t16 * t13[k0]));
    }
  }
}
```

TSFC: compiler passes

Vectorisation

Align and pad data structures, then use intrinsics or rely on C compiler.

Loop transformations & flop reduction

Solve ILP problem to drive factorisation, code motion, and common subexpression elimination.

Sum factorisation

Some finite elements use *tensor product* basis functions

$$\phi_{i,q} := \phi_{(j,k),(p,r)} = \varphi_{j,p}\varphi_{k,r}$$

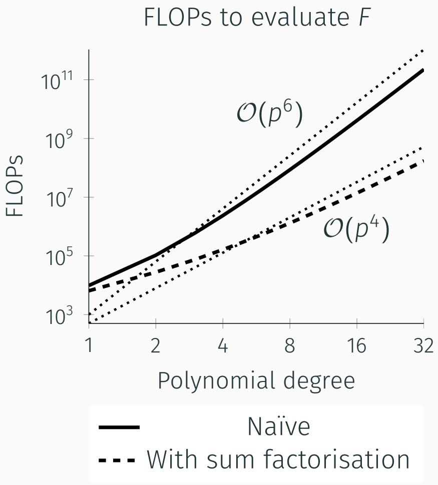
These permit low-complexity algorithms for evaluation of integrals.

Automated, not just for toy problems

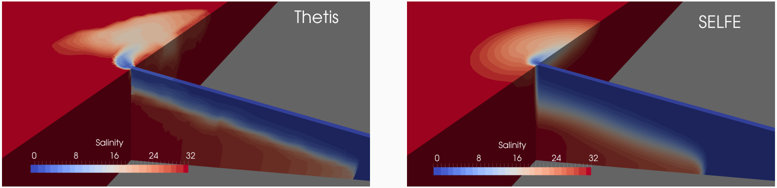
Find $u \in V \subset H(\mathrm{curl})$ s.t.

$$\int \mathrm{curl}\, u \cdot \mathrm{curl}\, v \, \mathrm{dx} = \int B \cdot v \, \mathrm{dx} \quad \forall v \in V.$$

```
NCE = FiniteElement("NCE", hexahedron, degree)
Q = VectorElement("Q", hexahedron, degree)
u = Coefficient(NCE) # Solution in H(curl)
B = Coefficient(Q)   # Coefficient in H^1
v = TestFunction(NCE)
F = (dot(curl(u), curl(v)) - dot(B, v))*dx
```



Application: coastal ocean modelling



New model

- Better solutions than previous model in group.
- 4-8x faster than models with comparative quality of results.
- Is differentiable: can use for PDE-constrained optimisation.
- Only 1.5 person years.

Future research

High performance solvers

- In most cases, after discretising a PDE, we need to *solve* a (non)linear problem.
- Designing robust, scalable solvers is a vast area of research in applied mathematics.
- Papers often only present (serial) proof of concept.

Idea

- Mathematics is the language used to derive optimal solvers.
- It *should be* the language we use to describe their implementation.

Challenges

- How to capture the mathematical building blocks in computer code?
- What does the compiler for this look like?
- How can it be portable across new computer hardware?

Rewards

- Bring state-of-the-art numerical solvers to the masses.
- Cross-fertilisation with programming language design.
- Enable more exploratory, and creative, mathematics.

Summary

- Broad interdisciplinary research agenda spanning:
    - Programming languages
    - Compiler design
    - Numerical methods & (non)linear algebra
- Application areas:
    - coastal ocean & freshwater outflow
    - renewable energy
    - numerical weather prediction

Lecture excerpt

Definition

recursion *noun*

see: recursion.

Building from the bottom

- Many problems in computing lend themselves to a recursive formulation
- Enumerate a few *base* cases, and a general rule

Building from the bottom

- Many problems in computing lend themselves to a recursive formulation
- Enumerate a few *base* cases, and a general rule

The Fibonacci sequence: F_n

$$0, 1, 1, 2, 3, 5, 8, 11, \ldots$$

Building from the bottom

- Many problems in computing lend themselves to a recursive formulation
- Enumerate a few *base* cases, and a general rule

The Fibonacci sequence: F_n

$$0, 1, 1, 2, 3, 5, 8, 11, \ldots$$

$$F_0 = 0$$

Building from the bottom

- Many problems in computing lend themselves to a recursive formulation
- Enumerate a few *base* cases, and a general rule

The Fibonacci sequence: F_n

$$0, 1, 1, 2, 3, 5, 8, 11, \ldots$$

$$F_0 = 0$$
$$F_1 = 1$$

Building from the bottom

- Many problems in computing lend themselves to a recursive formulation
- Enumerate a few *base* cases, and a general rule

The Fibonacci sequence: F_n

$$0, 1, 1, 2, 3, 5, 8, 11, \ldots$$

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \quad n \geq 2 \end{aligned}$$

Computing F_n

Require: $n \geq 0$ and n integer

```
function FIBONACCI(n)                    ▷ The n^th Fibonacci number
    if n = 0 then
        return 0
    else if n = 1 then
        return 1
    else
        return FIBONACCI(n − 1) + FIBONACCI(n − 2)
    end if
end function
```

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2} \quad n \geq 2$$

Would you use this approach?

- To compute F_1?

Would you use this approach?

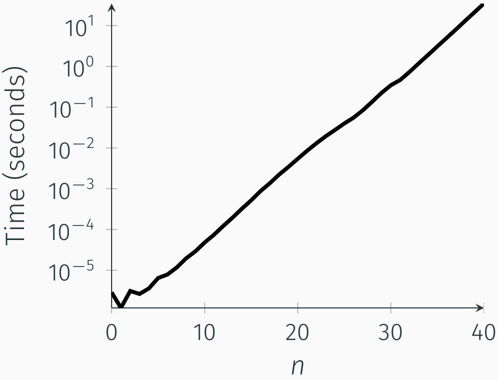
- To compute F_1?
- How about F_{10}?

Would you use this approach?

- To compute F_1?
- How about F_{10}?
- or F_{50}?

Would you use this approach?

- To compute F_1?
- How about F_{10}?
- or F_{50}?



Memoise, memoise, memoise

- We do far too much work!

Memoise, memoise, memoise

- We do far too much work!
- Let's draw the call tree for F_5

Memoise, memoise, memoise

- We do far too much work!
- Let's draw the call tree for F_5
- At each level, we split in two, and recurse

Memoise, memoise, memoise

- We do far too much work!
- Let's draw the call tree for F_5
- At each level, we split in two, and recurse
- There are approximately n levels.

Memoise, memoise, memoise

- We do far too much work!
- Let's draw the call tree for F_5
- At each level, we split in two, and recurse
- There are approximately n levels.
- So this does around 2^n calculations

Memoise, memoise, memoise

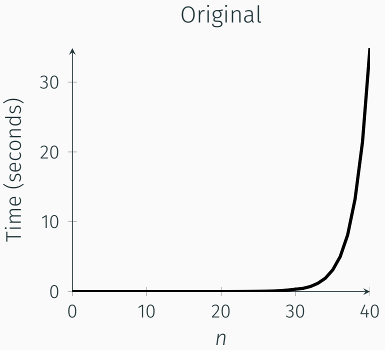
- We do far too much work!
- Let's draw the call tree for F_5
- At each level, we split in two, and recurse
- There are approximately n levels.
- So this does around 2^n calculations
- But many of them are the same, so why not remember them?

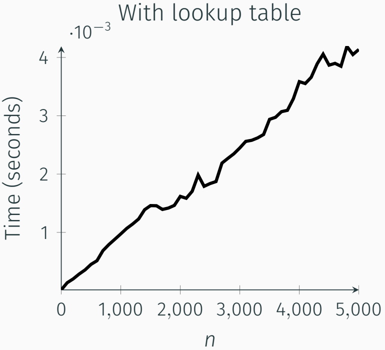
An improved algorithm

Require: $n \geq 0$ and n integer

```
function FIBONACCI(n, table)                ▷ The n^th Fibonacci number
    if n < 2 then
        return n
    else if n ∈ table then
        return table[n]
    else
        F ← FIBONACCI(n − 1, table) + FIBONACCI(n − 2, table)
        table[n] ← F
        return F
    end if
end function
```

An improved algorithm





- Now we only calculate each F_n once.
- So runtime is now proportional to n. At the cost of storing n values.
- Challenge: is this the best you can do?

What’s in a name?

- This trick, replacing repeated computation by a lookup of the result, is called *dynamic programming*.
- Coined by Richard Bellman in the 1950s

  dynamic [...] *has a very interesting property as an adjective,* [...] *it’s impossible to use the word dynamic in a pejorative sense.*