So you can prove theorems - but can you code?

Knut–Andreas Lie

SINTEF ICT, Dept. Applied Mathematics, Oslo, Norway /
Norwegian University of Science and Technology, Trondheim, Norway

Math Meets Industry, 22–23 September 2016

Math meets industry. . . ?

Math meets industry. . . ?

Storyline:
In a far-off land, the wise mathematician sits in his ivory tower looking at the world below through his telescope. Seeing how the people struggle with their lives, he decides to descend . . .

Math meets industry. . . ?

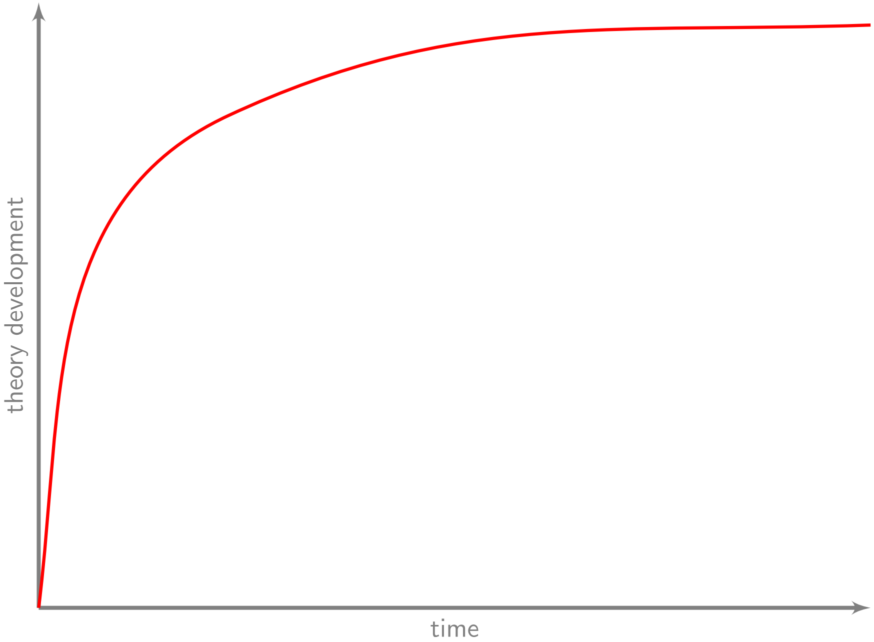
Storyline:
In a far-off land, the wise mathematician sits in his ivory tower looking at the world below through his telescope. Seeing how the people struggle with their lives, he decides to descend . . .

However, reality is different:
Apparently, too few people realize that the 'high technology' that is celebrated today is essentially a mathematical technology.

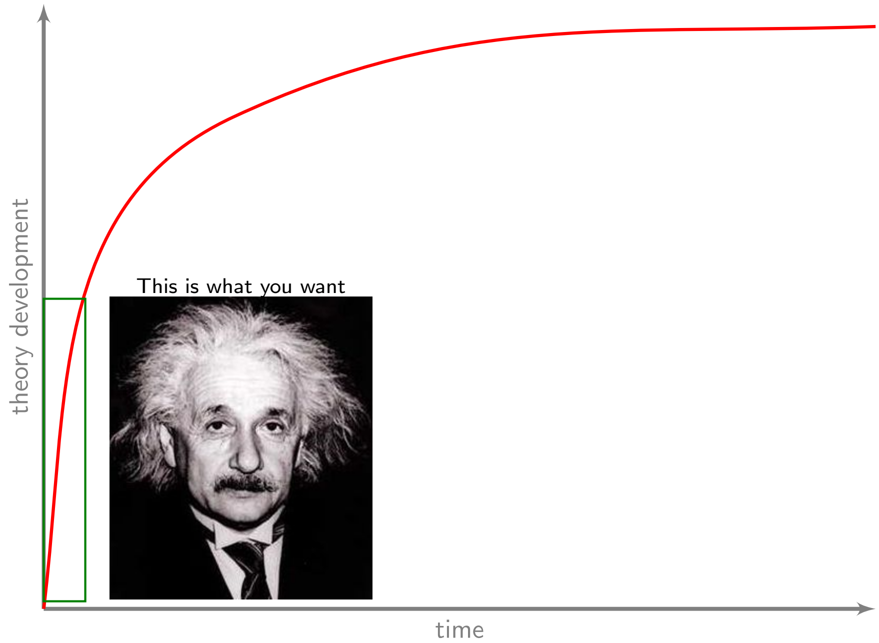
For mathematics to be renewed, mathematics must be perceived as it historically has been: as an essential contributor to science and technology

– Edward E. David, President of Exxon R&D

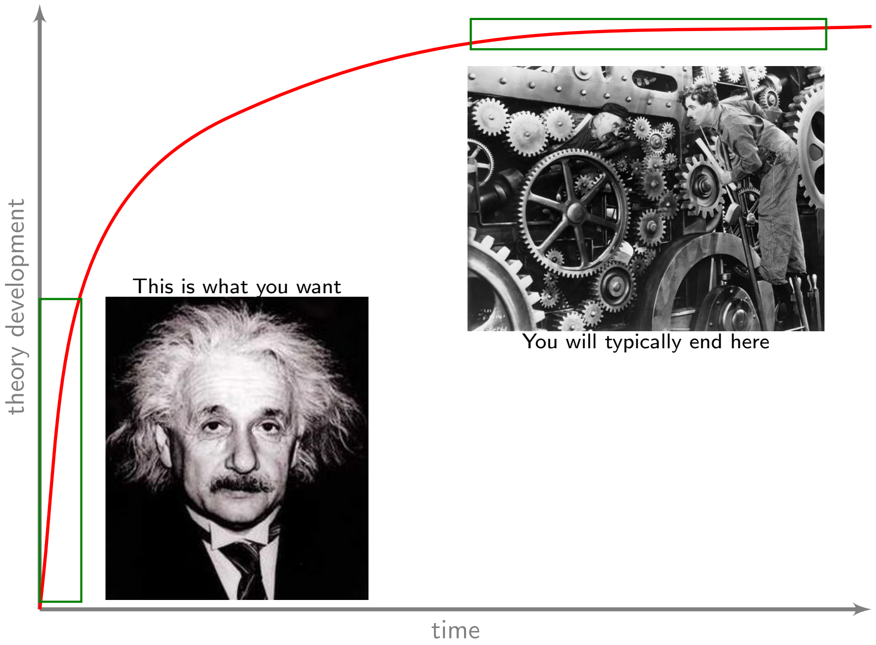
This has some implications. . .



This has some implications...



This has some implications. . .



My background

Education:

- Siv.ing. from Department of Mathematical Sciences, NTH, 1988–1993
  Numerical solution of boundary value problems using probabilistic methods
- Dr.ing. from NTNU, 1994–1998
  Front tracking and operator splitting for convection dominated problems

Work:

- Post doc, Department of informatics, UiO, 1998–1999
- SINTEF Applied Mathematics, 1999, Research director
  ⋮
  SINTEF ICT, Chief scientist / research manager

Also part-time positions

IFI/UiO and Simula Research Laboratory, 1999–2005

Center of Mathematics for Applications, 2005–2012

Professor II, Department of Mathematics, UiB, 2007–2014

Professor II, Dept. Mathematical Sciences, NTNU, 2015–

My job: running a contract research group

Computational Geosciences group:

- one of five groups in Department of Applied Mathematics
- twelve researchers/PhD students + 6–7 master students
- offices in Oslo, Norway
- performs a mixture of basic and applied research
- strong focus on *scientific publications* (highest rate in SINTEF)
- internationally oriented
- main clients: Statoil, ExxonMobil, Schlumberger, Research Council of Norway, . . .

André R. Brodtkorb

Atgeirr F. Rasmussen

Bård Skaflestad

Halvor M. Nilsen

Kai Bao

Knut-Andreas Lie

Odd A. Andersen

Olav Møyner

Rebecca Allen

Sindre T. Hilden

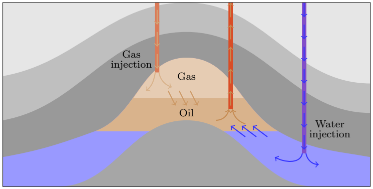
Stein Krogstad

Xavier Raynaud

What is our research focus?

We develop efficient, robust, and accurate simulation methods for partial differential equations applied to simulate:

- enhanced and improved oil recovery
- geological storage of CO_2
- surface water



To this end, we do a lot of experimental programming and develop open-source software.

Most known for: the Matlab Reservoir Simulation Toolbox (MRST), work on multiscale methods, unstructured grids, methods for fast simulation

If we let $\hat{u}_{\Delta t}$ denote the piecewise constant function

$$\hat{u}_{\Delta t}(\,\cdot\,,t) = \sum_n u^{n+1}(\,\cdot\,)\chi_{I^n}(t),$$

then (for $T = N\Delta t$)

$$\begin{aligned} \|\hat{u}_{\Delta t} - \tilde{u}_{\Delta t}\|_{L^1(\Pi_T)} &= \sum_{n=0}^{N-1} \int_{t_n}^{t^{n+1}} \int_{\mathbb{R}} |\tilde{u}_{\Delta t}(x,t) - \tilde{u}_{\Delta t}(x,t_{n+1})|\; dx \\ &= \sum_{n=0}^{N-1} \int_{t^n}^{t_{n+1}} \mathrm{Const}\sqrt{t_{n+1}-t}\,dt \\ &\le \mathrm{Const}\sum_{n=0}^{N-1} (\Delta t)^{3/2} \le \mathrm{Const}_T\sqrt{\Delta t}. \end{aligned}$$

Therefore $A(\hat{u}_{\Delta t}) \to A(u)$ in $L^p(\Pi_T)$, $1 \le p < \infty$, as $\Delta t \to 0$. Furthermore,

$$\begin{aligned} J(y)\Delta x^2 &= j_y\Delta x\big((\Delta x - h)(j_y - 1) + h\,(j_y+1)\big) \\ &= j_y^2\Delta x^2 + 2j_y h\Delta x \\ &\le j_y^2\Delta x^2 + 2j_y h\Delta x + h^2 \\ &= (j_y\Delta x + h)^2 = y^2. \end{aligned}$$

Multiplying (4.38) with Δt and summing over n, we get

$$\begin{aligned} &\int_0^T \int_{\mathbb{R}} \big(A\big(\hat{u}_{\Delta t}(x+y,t)\big) - A\big(\hat{u}_{\Delta t}(x,t)\big)\big)^2\; dx\,dt \\ &\qquad \le y^2\,\|A\|_{\mathrm{Lip}} \left[\int_{\mathbb{R}} u_0^2(x)\,dx + \|\tilde{u}_{\Delta t}\|_{L^\infty(\Pi_T)}\,\big\|f^\delta\big\|_{\mathrm{Lip}}\,\mathrm{T.V.}\,(u_0)\,T\right], \end{aligned}$$

assuming that (we have set things up so that) $\|f^\delta\|_{\mathrm{Lip}} \le 2\,\|f\|_{\mathrm{Lip}}$ and the initial data $\|u^0\|_{L^2(\mathbb{R})} \le 2\,\|u_0\|_{L^2(\mathbb{R})}$. Thus, letting $\Delta t \to 0$ above, we get

$$\int_0^T\!\!\int_{\mathbb{R}} \Big(A\big(u(x+y,t)\big) - A\big(u(x,t)\big)\Big)^2\, dx\,dt \le \mathrm{Const}_{A,f,u_0,T}\,y^2. \tag{4.39}$$

Finally, we can let $y \to 0$ to conclude that $\partial_x A(u) \in L^2(\Pi_T)$.

Summing up, we have derived the following result:

Theorem 4.4. *Assume that $u_0 \in L^1(\mathbb{R}) \cap \mathrm{BV}(\mathbb{R})$, and that f and A are locally Lipschitz continuous with $A' \ge 0$ and $A(0) = 0$. Define a sequence of functions $\{u_{\Delta t}\}_{\Delta t\ge 0}$ by (4.18). There exists a subsequence of $\{\Delta t_j\}_{j\in\mathbb{Z}} \subset \{\Delta t\}$ such that*

$$u_{\Delta t_j} \to u \quad \textit{in } L^1_{\mathrm{loc}}(\Pi_T) \textit{ as } \Delta t_j \to 0,$$

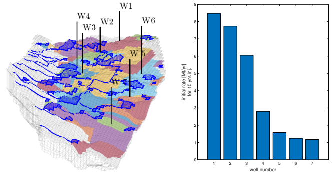


Figure 8 *Northward view of the northern part of the Utsira formation with seven wells placed at the base of spill paths that spill towards the west of the formation (left). The right plot shows the rates suggested so that a volume equal the upslope trap volume is injected in ten years.*

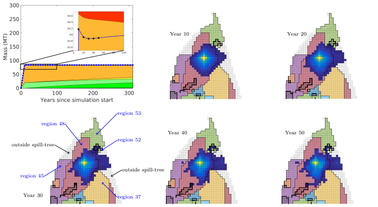


Figure 9 *Simulation of north parts of Utsira where forecast curve exhibits non-monotonic behavior immediately following the injection period.* Upper left: *Trapping inventory with forecast curve in dashed-blue line.* Year 10–50: *Snap-shots of CO_2 saturations around one injection well.*

Table 1 *Mass inventory for Utsira North during years immediately following the injection period. Total injected mass by year 10 is 84.738 Mt. Inventory is given in terms of Mt and percentage of total injected.*

Region	**year 10**		**year 20**		**year 30**		**year 40**		**year 50**	
Within trees	84.029	99.16%	83.726	98.81%	83.665	98.73%	83.658	98.73%	83.664	98.73%
- region 37	1.488	1.76%	2.026	2.39%	2.165	2.56%	2.222	2.62%	2.261	2.67%
- region 45	0.041	0.05%	0.043	0.05%	0.043	0.05%	0.043	0.05%	0.043	0.05%
- region 48	82.384	97.22%	81.511	96.19%	81.309	95.95%	81.247	95.88%	81.216	95.84%
- region 53	0.112	0.13%	0.140	0.17%	0.141	0.17%	0.139	0.16%	0.137	0.16%
- region 52	0.003	0.00%	0.004	0.01%	0.005	0.01%	0.005	0.01%	0.005	0.01%
Outside trees	0.633	0.75%	0.823	0.97%	0.843	1.00%	0.829	0.98%	0.807	0.95%

Difference in focus: academic vs industry research

Academic mathematicians:

- work on complex methods for simple problems
- develop theories and prove theorems that last until eternity

Difference in focus: academic vs industry research

Academic mathematicians:

- work on complex methods for simple problems
- develop theories and prove theorems that last until eternity

In industry research, you are expected to:

- Provide insight to improve business and products
- Develop algorithms and software
- Provide a sufficient solution on time rather than the promise of a perfect solution sometime in the future
- Provide specific answers that can be used to make decisions

Most often, you will work on simple methods for complex problems

Science is what we understand well enough to explain to a computer. Art is everything else we do.

– Donald Knuth, Foreword to the book A=B (1996)

SEVEN GUIDELINES FOR SCIENTIFIC COMPUTING

HELGE HOLDEN

Abstract. It is called for an improvement in the standards of present day scientific computing and the way computer simulations are performed and presented. Seven guidelines are proposed to this effect.

In giving advice seek to help, not to please your friends.
— SOLON (7th–6th c. BC)

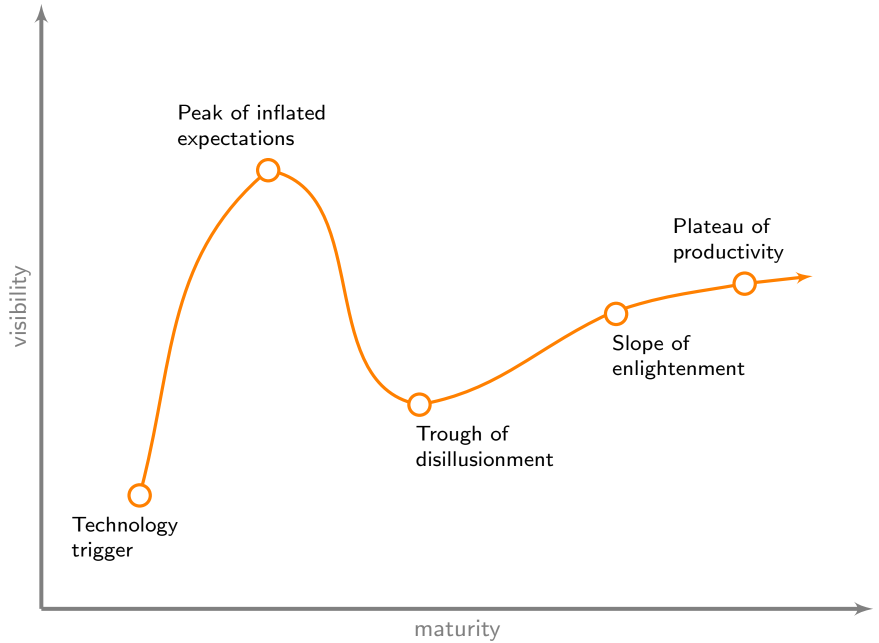
1. Your results should always be reproducible
2. Test the stability of your method with respect to variation of parameters
3. Compare your method to other methods
4. Report cases where your method fails
5. When possible, compare the computer simulations to real experiments
6. Establish standard test cases in your field
7. Make your code available to your colleagues

Reproducible computational science

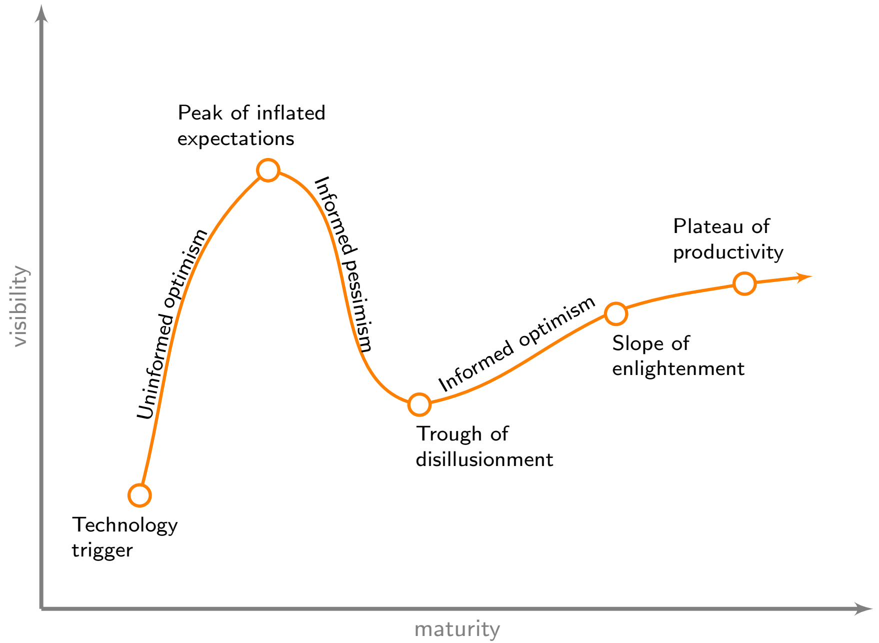
Within the world of science, computation is now rightly seen as a third vertex of a triangle complementing experiment and theory. However, as it is now often practiced, one can make a good case that computing is the last refuge of the scientific scoundrel [...] Where else in science can one get away with publishing observations that are claimed to prove a theory or illustrate the success of a technique without having to give a careful description of the methods used, in sufficient detail that others can attempt to repeat the experiment? [...] Scientific and mathematical journals are filled with pretty pictures these days of computational experiments that the reader has no hope of repeating. Even brilliant and well intentioned computational scientists often do a poor job of presenting their work in a reproducible manner. The methods are often very vaguely defined, and even if they are carefully defined, they would normally have to be implemented from scratch by the reader in order to test them

– Randy LeVeque, Proc. Internat. Congress Math., Madrid, Spain, 2006

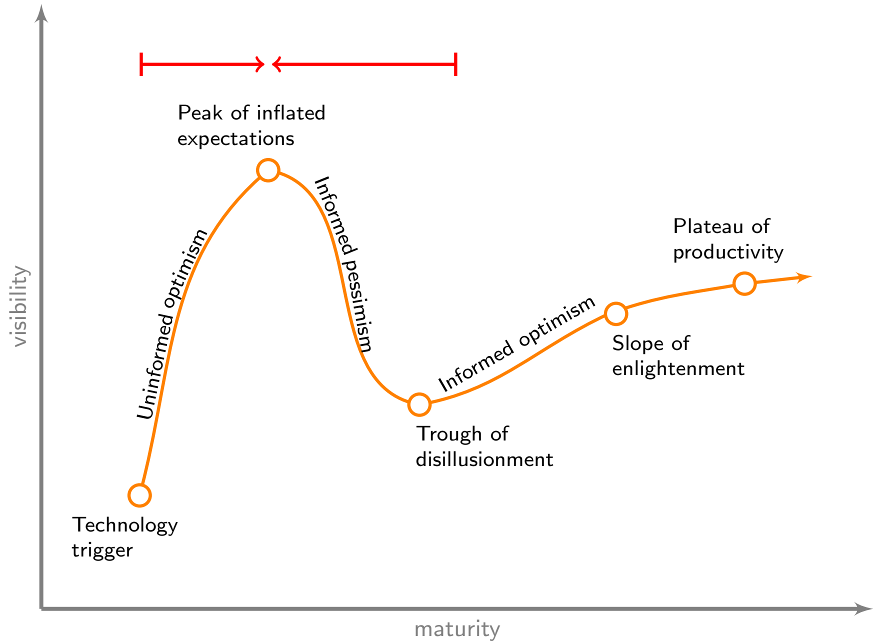
Gartner's hype cycle



Gartner's hype cycle



Gartner's hype cycle



Accelerating time to informed optimism

Get good test problems to drive research in the right directions

Accelerating time to informed optimism

Get good test problems to drive research in the right directions

Practice the art of simplification

- Simplify to get rid of effects and details that are not important.
  Occam's razor principle:

  > *Among competing hypotheses, the one with the fewest assumptions should be selected.*

- Don't simplify because you cannot handle important mechanisms by your favorite theory, computational method, or software

Accelerating time to informed optimism

Get good test problems to drive research in the right directions

Practice the art of simplification

- Simplify to get rid of effects and details that are not important.
  Occam's razor principle:

  > *Among competing hypotheses, the one with the fewest assumptions should be selected.*

- Don't simplify because you cannot handle important mechanisms by your favorite theory, computational method, or software

Use or develop software for rapid prototyping and validation/verification of new models and computational methods on problems *relevant to industry or other scientific disciplines*

How to prototype new ideas?

Should you use commercial tools or community codes, or should you develop your own home brew?

- Know what is current state-of-the-art in commercial tools
- To understand limitations of existing tools, it is important that you have experience developing your own tools
- However, remember that the life-span of most research codes $\approx$ the employment period of the developer (i.e., 3–4 years. . .)
- Don't underestimate the value of using other peoples' software (leverage years of development and testing)

My recommendation: use and contribute to existing community codes, and if you have developed a good method – share it with others!

Bottlenecks to experimental programming

Factors that contribute to slow down the development cycle in a third-generation compiled language

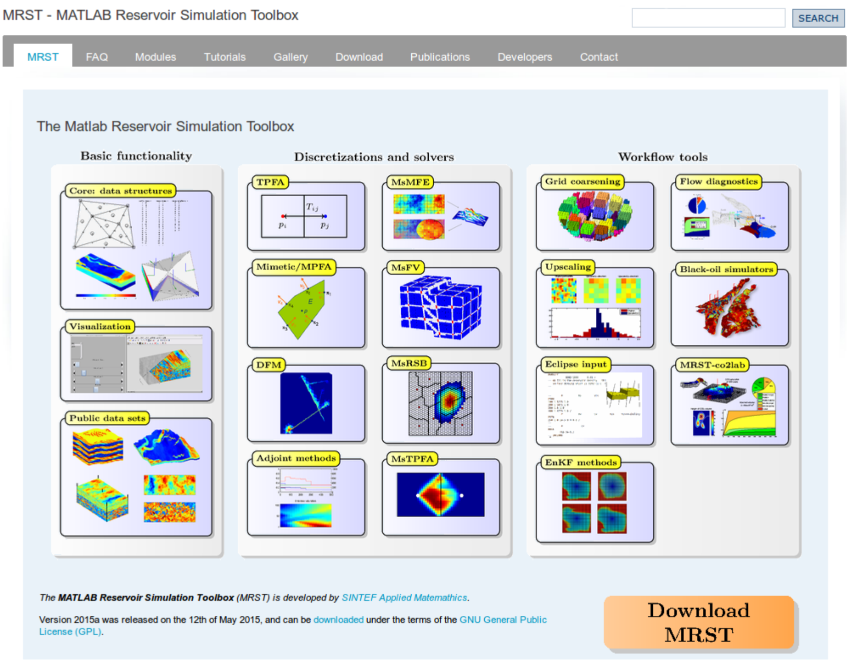
	3rd generation Fortran, C, C++	4th generation Matlab, Python
Syntax	complicated	intuitive
Cross-platform	challenging	✓
Build process	✓	✗
Linking of external libraries	✓	✗
Type-checking	static	dynamic
Mathematical abstractions	user-defined	built-in
Numerical computations	libraries	built-in
Data analysis and visualization	libraries/external	built-in
Debugger, profiling, etc	external/IDE	built-in
Traversing data structures	loops, iterators	vectorization†

† also: indirection maps and logical indices

Different development process

- Use of abstractions enables the user to express ideas in a form close to the underlying mathematics
- Using interactive environment to build new program:
  - try out each operation and build program as you go
  - wide range of built-in functions for numerical computations
  - powerful data analysis, graphical user interface, and visualization
- Prototyping while testing existing program:
  - run code line by line, inspect and change variables at any point
  - step back and rerun parts of code with changed parameters
  - add new behavior and data members while executing program

Also: use scripting language as a wrapper when developing solvers in compiled languages



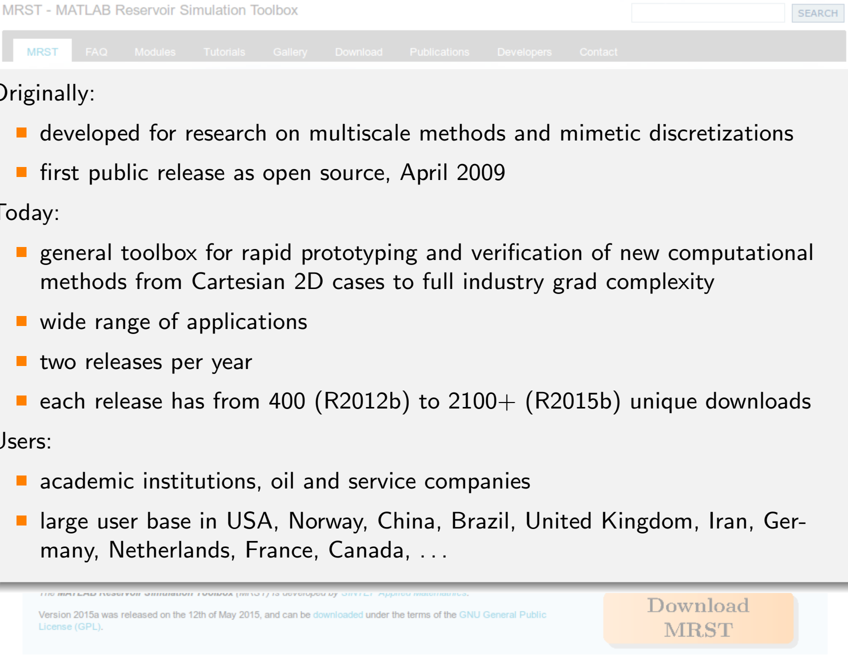
MRST - MATLAB Reservoir Simulation Toolbox

The Matlab Reservoir Simulation Toolbox

*The **MATLAB Reservoir Simulation Toolbox** (MRST) is developed by SINTEF Applied Matemathics.*

Version 2015a was released on the 12th of May 2015, and can be downloaded under the terms of the GNU General Public License (GPL).

http://www.sintef.no/MRST


MRST - MATLAB Reservoir Simulation Toolbox

Originally:

- developed for research on multiscale methods and mimetic discretizations
- first public release as open source, April 2009

Today:

- general toolbox for rapid prototyping and verification of new computational methods from Cartesian 2D cases to full industry grad complexity
- wide range of applications
- two releases per year
- each release has from 400 (R2012b) to 2100+ (R2015b) unique downloads

Users:

- academic institutions, oil and service companies
- large user base in USA, Norway, China, Brazil, United Kingdom, Iran, Germany, Netherlands, France, Canada, . . .

The MATLAB Reservoir Simulation Toolbox (MRST) is developed by SINTEF Applied Mathematics.

Version 2015a was released on the 12th of May 2015, and can be downloaded under the terms of the GNU General Public License (GPL).

`http://www.sintef.no/MRST`

Key ideas for rapid prototyping

Focus on clean and simple implementation of equations, close to the mathematics $\longrightarrow$ less error-prone coding, simpler to maintain and extend

- **unstructured grid format** – algorithms can be implemented without knowing the specifics of the grid
- **vectorize** – no visible loops and few indices, i.e., 1-to-1 correspondence between continuous and discrete variables
- **discrete operators**, mappings, and forms – not tied to specific flow equations and can be precomputed independently

$$\frac{1}{\Delta t}\left[(\phi\rho)^{n+1} - (\phi\rho)^{n}\right] + \mathrm{div}(\rho v)^{n+1} = 0$$

```
R = ( pv(p).*rho(p) - pv(p0).*rho(p0) )/dt
                   + div( avg(rho(p)).*v(p) );
```

- **automatic differentiation** – no need to derive gradients and Jacobians analytically

Automatic differentiation

Idea: avoid having to explicitly differentiate complex functions. Use object consisting of a value (vector) and a *list* of derivatives (Jacobian matrix):

$$x = (\boldsymbol{x}, \{\mathbf{I}, 0\}), \qquad y = (\boldsymbol{y}, \{0, \mathbf{I}\})$$

Any code can be broken down to a limited set of arithmetic operations and elementary functions (`sin`, `exp`, `power`, ...).

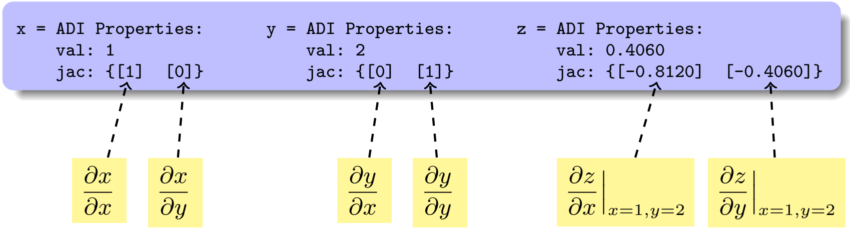
Operator overloading is used to implement common derivative rules

$$(f + g) = (\mathbf{f} + \mathbf{g}, \{\mathbf{f}_x + \mathbf{g}_x, \mathbf{f}_y + \mathbf{g}_y\})$$
$$(f * g) = (\mathbf{f} * \mathbf{g}, \{\mathbf{f}_x \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{g}_x, \mathbf{f}_y \cdot \mathbf{g} + \mathbf{f} \cdot \mathbf{g}_y\})$$

Then: you write equations and constitutive laws, the software determines the correct linearization

Automatic differentiation: in MRST

```
[x,y] = initVariablesADI(1,2);
z = 3*exp(-x*y)
```



```
x = ADI Properties:          y = ADI Properties:          z = ADI Properties:
    val: 1                       val: 2                       val: 0.4060
    jac: {[1]   [0]}             jac: {[0]   [1]}             jac: {[-0.8120]   [-0.4060]}
```

$\frac{\partial x}{\partial x}$
$\frac{\partial x}{\partial y}$
$\frac{\partial y}{\partial x}$
$\frac{\partial y}{\partial y}$
$\frac{\partial z}{\partial x}\Big|_{x=1,y=2}$
$\frac{\partial z}{\partial y}\Big|_{x=1,y=2}$

MRST implementation tailored to reservoir simulation and MATLAB:

- designed to be efficient for vector variables
- works with sub-Jacobians rather than full Jacobians to simplify subsequent manipulation

Key ideas of basic framework

The MRST framework is developed to:

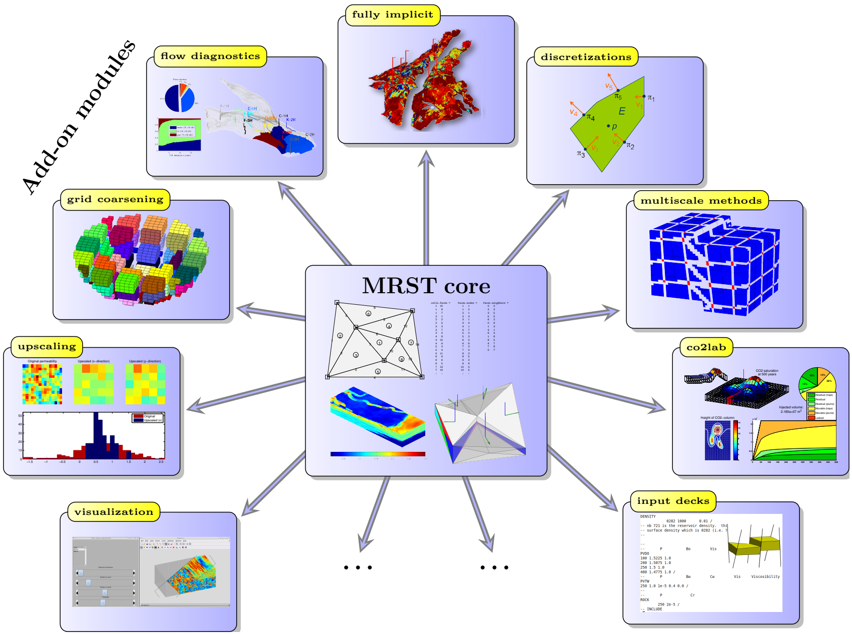
- mainly support low-order finite-volume methods
- be as flexible as possible (i.e., willing to sacrifice efficiency)
- only use standard Matlab (no expensive toolboxes)
- support internal/external collaboration
- preserve know-how
- encourage reproducible research and leveraging previous research

Key ideas of basic framework

A **modular design** has proved to be essential:

- **small core** with mature and well-tested functionality used in *many* programs or modules
- built-in support for typical data and basic numerical operations
- **semi-independent modules** that extend/override core functionality
- code well documented according to Matlab standard
- all modules must have code examples and/or tutorials
- separate software repositories, (automated testing)

Key ideas of basic framework



```
DENSITY
            0282 1000      0.01 /
-- nb 721 is the reservoir density.  thi
-- surface density which is 0282 (i.e. 7
--


--
--       P           Bo         Vis
PVDO
100 1.5225 1.0
200 1.5075 1.0
250 1.5 1.0
400 1.4775 1.0 /
--       P           Bw         Cw         Vis     Viscosibility
PVTW
250 1.0 1e-5 0.4 0.0 /
--
--       P            Cr
ROCK
        250 2e-5 /
-- INCLUDE
```

Key ideas of basic framework

Recently, object-oriented extension to separate:

- physical models
- discretizations and discrete operators
- nonlinear solver and time-stepping
- assembly and solution of the linear system

and only expose needed details and enable more reuse of functionality

The result is a prototyping tool capable of industry-standard complexity

How to be a good software developer

Don't show off! Try to aid others rather than alienating them

How to be a good software developer

Don't show off! Try to aid others rather than alienating them

Keep it simple stupid!
If you feel particularly brilliant after having written a piece of code, chances are high that you will be the only one who understands it

How to be a good software developer

Don't show off! Try to aid others rather than alienating them

Keep it simple stupid!
If you feel particularly brilliant after having written a piece of code, chances are high that you will be the only one who understands it

Document your code! If nothing else, so that you a few years into the future can understand your own code

How to be a good software developer

Don't show off! Try to aid others rather than alienating them

Keep it simple stupid!
If you feel particularly brilliant after having written a piece of code, chances are high that you will be the only one who understands it

Document your code! If nothing else, so that you a few years into the future can understand your own code

Avoid being an onion-maker, i.e., making layers upon layers of abstractions (with no core) in an attempt to be generic

How to be a good software developer

Don't show off! Try to aid others rather than alienating them

Keep it simple stupid!
If you feel particularly brilliant after having written a piece of code, chances are high that you will be the only one who understands it

Document your code! If nothing else, so that you a few years into the future can understand your own code

Avoid being an onion-maker, i.e., making layers upon layers of abstractions (with no core) in an attempt to be generic

Community projects attract nerds/geeks – don't become a 'religious' fanatic who quarrels over standardization and minor issues

People's Front of Judea

Popular Front of Judea

The only people we hate more than the Romans are the Judean People's Front