The semantics and epistemology of accuracy + the epistemology of computational error

HLRS Summer School Trust and ML

Nicolas Fillion
Simon Fraser University
nfillion@sfu.ca
www.nfillion.com

Machine learning, conceived as a **computational discipline**, is a kind of **scientific computing**.

How much of scientific computing is machine learning?
Well, how much of world population is in Germany?

Since the early days of computer-based scientific computing in the 1940s, the attitude has been:

Trust, but verify

There is **eight decades** of **practical and theory wisdom** about what this entails.

Since the early days of computer-based scientific computing in the 1940s, the attitude has been:

Trust, but verify

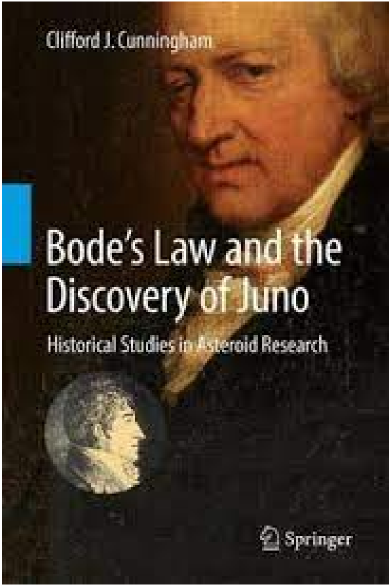
There is **eight decades** of **practical and theory wisdom** about what this entails.

Numerical analysts have been conceptualizing how to **negociate what computational methods to trust** for all this time.

Yet, the literature on the **philosophy** of machine learning seems quite disconnected. . .

An hommage to Hegel?

An hommage to Hegel?



After Hegel's death, the asteroid ephemeris calculator Heinrich Christian Schumacher (1780–1850; Fig. 1.14) felt compelled to comment. In a letter to Gauss, he noted that Hegel's Dissertation had been included in a publication of his collected works. He **expressed his disgust in Biblical terms***: " 'Among Noah's sons there was at least one who covered up his father's shame, but the Hegelians pulled off the cloak which time and forgetfulness had spread over the shame of their master.' Gauss replied that the comparison limped badly, for* **Noah got drunk only once, while Hegel's *insania* was pure wisdom compared to what he wrote later!***"*

An hommage to Hegel?

Let's go on a hypothetical trip
to the end of the solar system with Hegel!

(See the simple Python experiment. . .)

A hard landing!

What is happening?

How should we diagnose the problem?

What are the important takehome messages about how science works, more broadly?

My proposal is to discuss a **non-existent book**:

For our purposes, the **history of computer arithmetic** can be a wealth of information!

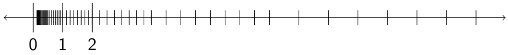
> *There is no controversy here; it can hardly arise in the context of exact integer arithmetic, so long as there is general agreement on what integer the correct result should be. However,* **as soon as approximate arithmetic enters the picture, so does controversy, as if one person's "negligible" must be another's "everything."**
>
> *Computer Organization and Design, 2013*

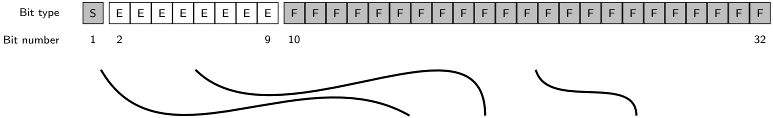
We can understand some of the **non-trivial differences** between contexts involving approximations and those that don't just by looking at **conceptually intricate** questions about this **mathematically simple** theory.

Floating-point arithmetic (FPA)

What are floating-point numbers all about?

Operations on $\mathbb{R}, \mathbb{C}$ $\xrightarrow{\text{rounding}}$ floating-point arithmetic $\mathbb{F}$

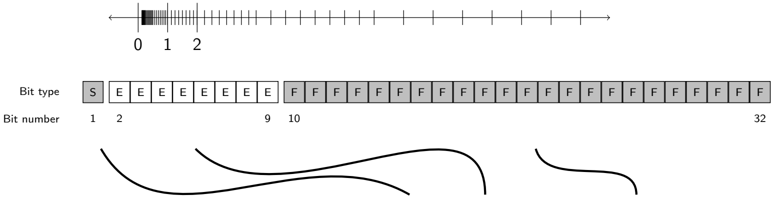




Scientific notation: $+2.99792458 \times 10^{8}$

What are floating-point numbers all about?

Operations on $\mathbb{R}, \mathbb{C}$ $\xrightarrow{\text{rounding}}$ floating-point arithmetic $\mathbb{F}$



Scientific notation: $+2.99792458 \times 10^8$

"95% of the folks out there are completely clueless about floating-point." (James Gosling, 1998)

FPA can lead to **surprising errors**. On your “pocket calculator,” chances are those all return different values:

$$
\begin{aligned}
s_1 &= 10^{20} + 17 - 10 + 130 - 10^{20} \\
s_2 &= 10^{20} - 10 + 130 - 10^{20} + 17 \\
s_3 &= 10^{20} + 17 - 10^{20} - 10 + 130 \\
s_4 &= 10^{20} - 10 - 10^{20} + 130 + 17 \\
s_5 &= 10^{20} - 10^{20} + 17 - 10 + 130 \\
s_6 &= 10^{20} + 17 + 130 - 10^{20} - 10
\end{aligned}
$$

FPA can lead to **surprising errors**. On your "pocket calculator," chances are those all return different values:

$$s_1 = 10^{20} + 17 - 10 + 130 - 10^{20}$$
$$s_2 = 10^{20} - 10 + 130 - 10^{20} + 17$$
$$s_3 = 10^{20} + 17 - 10^{20} - 10 + 130$$
$$s_4 = 10^{20} - 10 - 10^{20} + 130 + 17$$
$$s_5 = 10^{20} - 10^{20} + 17 - 10 + 130$$
$$s_6 = 10^{20} + 17 + 130 - 10^{20} - 10$$

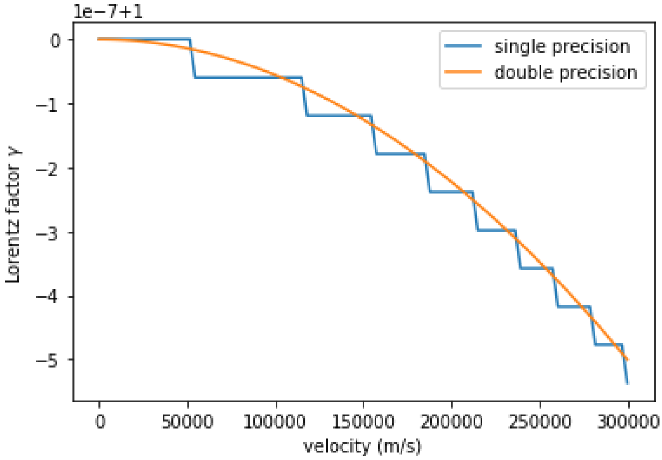
The answers will probably be $0, 17, 120, 147, 137$ and -10.

(You may need to add '.0' to the numbers to enforce `float` types.)

Suppose we want to calculate

$$\gamma = \sqrt{1 - v^2/c^2}$$

in a Lorentz transform in SR, but in `float32`. **Step function!**

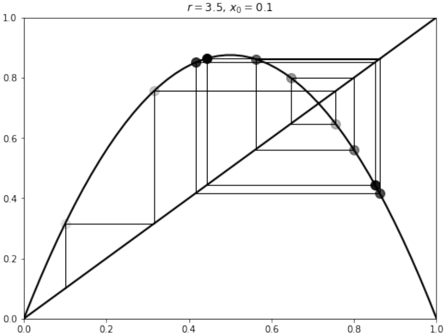


There are tons of interesting funny things about floating-points!

For example, take the discrete logistic map:

$$x_{k+1} = \mu x_k (1 - x_k)$$

What would happen **in the long run** if we simulate this system?

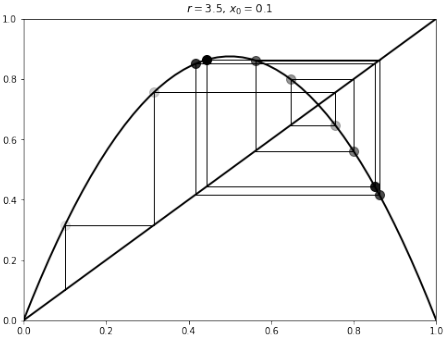


There are tons of interesting funny things about floating-points!

For example, take the discrete logistic map:

$$x_{k+1} = \mu x_k(1 - x_k)$$

What would happen **in the long run** if we simulate this system?



A: For any precision, **all** discrete dynamical systems are **periodic**!

In each case, there's an important sense in which the computation does **not** provide the **correct answer**.

A couple observations:

- Double-precision seems to be giving better answers than single-precision.

In each case, there's an important sense in which the computation does **not** provide the **correct answer**.

A couple observations:

- Double-precision seems to be giving better answers than single-precision.
- In such context, we may be tempted to make the following associations:

  more precision, more accuracy, less error, better result

In each case, there's an important sense in which the computation does **not** provide the **correct answer**.

A couple observations:

- Double-precision seems to be giving better answers than single-precision.
- In such context, we may be tempted to make the following associations:

  more precision, more accuracy, less error, better result

Should we always be **more satisfied with the more precise answer**?

Often, such computations are **just fine**, though there's always some error. How should we think about such situations?

This suggests **two importantly different questions**:

Often, such computations are **just fine**, though there's always some error. How should we think about such situations?

This suggests **two importantly different questions**:

1. Is this answer closer to the truth than this other answer? (conceptually easy)
2. **Is the answer accurate enough**? (conceptually trickier)

This second question is **context-dependent in a sense that needs clarification**. Can we illustrate its relevant features within the confines of arithmetic?

Floating-point arithmetic is only **one of the interesting paradigms**!

In case you forgot the idea behind **significant-figures arithmetic**. . .

Bazooka Joe is showing a friend a fossilized bone. The friend asks how old it is and Bazooka Joe responds that it is one hundred million and three years old. "How do you know that?" asks the friend. Bazooka Joe responds "The museum expert told me it was a hundred million years old and that was three years ago."

The joke is based on the idea that

In case you forgot the idea behind **significant-figures arithmetic**. . .

Bazooka Joe is showing a friend a fossilized bone. The friend asks how old it is and Bazooka Joe responds that it is one hundred million and three years old. “How do you know that?” asks the friend. Bazooka Joe responds “The museum expert told me it was a hundred million years old and that was three years ago.”

The joke is based on the idea that

- $1.0 \cdot 10^8 + 3 = 1.00000003$ is **not right**.
- $1.0 \cdot 10^8 + 3 = 1.0 \cdot 10^8$ is **right**.

This is different than the FPA paradigm.

In case you forgot the idea behind **significant-figures arithmetic**. . .

Bazooka Joe is showing a friend a fossilized bone. The friend asks how old it is and Bazooka Joe responds that it is one hundred million and three years old. "How do you know that?" asks the friend. Bazooka Joe responds "The museum expert told me it was a hundred million years old and that was three years ago."

The joke is based on the idea that

- $1.0 \cdot 10^8 + 3 = 1.00000003$ is **not right**.
- $1.0 \cdot 10^8 + 3 = 1.0 \cdot 10^8$ is **right**.

This is different than the FPA paradigm.

Here, whether an answer is **good (enough)** is assessed **with respect to an epistemic context** that characterizes **uncertainty**.

Again, equivalent propositions are not so straightforward:

$$f(x) = x(\sqrt{x+1} - \sqrt{x}) \qquad g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

$f(x)$ and $g(x)$ are identically equal, so the two equations have the same truth-conditions.

Again, equivalent propositions are not so straightforward:

$$f(x) = x(\sqrt{x+1} - \sqrt{x}) \qquad g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

$f(x)$ and $g(x)$ are identically equal, so the two equations have the same truth-conditions.

However, if we perform arithmetic operations in **significant figures arithmetic**, or with any other **finite-precision arithmetic**, things change... Take $x = 5.000 \cdot 10^2$ (i.e., 4 sig figs):

$$f(500) = 10.00 \qquad \text{and} \qquad g(500) = 11.18$$

If everything were **exact**, we'd have $\approx 11.17476\cdots$.

Again, equivalent propositions are not so straightforward:

$$f(x) = x(\sqrt{x+1} - \sqrt{x}) \qquad g(x) = \frac{x}{\sqrt{x+1} + \sqrt{x}}$$

$f(x)$ and $g(x)$ are identically equal, so the two equations have the same truth-conditions.

However, if we perform arithmetic operations in **significant figures arithmetic**, or with any other **finite-precision arithmetic**, things change... Take $x = 5.000 \cdot 10^2$ (i.e., 4 sig figs):

$$f(500) = 10.00 \qquad \text{and} \qquad g(500) = 11.18$$

If everything were **exact**, we'd have $\approx 11.17476 \cdots$.

Good scientific programming requires sensitivity to such matters! It it absolutely crucial!

Comparing floating-point and significant-figure arithmetic has revealed **two distinct standards of accuracy**, i.e., two standards for **assessing matters of approximation**.

So the situation is **not** **simply** one where:

Okay, we know what is **epistemically better**,
and we just want **more of it**.

Comparing floating-point and significant-figure arithmetic has revealed **two distinct standards of accuracy**, i.e., two standards for **assessing matters of approximation**.

So the situation is **not** **simply** one where:

Okay, we know what is **epistemically better**,
and we just want **more of it**.

Moreover, the **second standard** introduces something quite interesting:

- The inference's quality depends on what we know (**and ignore!**) about the premises.
- So, there's no sharp cut between matters of truth-of-premises and inferential strenght.

And this is just the tip of the iceberg!

In 1972, MIT mathematician Gilbert Strang introduced the term **variational crime** to describe a theoretical problem with FEM.

As it turns out, **articulating more standards of accuracy** is essential to understanding what's happening.

Preliminary conclusions:

- Computers can easily give you answers that are **way off**.
- There are different standards of accuracy in scientific practice.
- It's not just about minimizing error (relative to a given norm).
- It's about **interpreting whether the error is small enough in an epistemic context**.

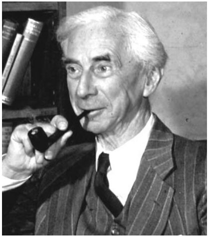
From here, I'd like to emphasize this last idea, as (I think) it **remains marginally known** to many philosophers & practitionners.

Let's start with a brute fact about science:

Our theories, models, hypotheses, and what have you, are typically **not strictly true**.
(Technically, they are not *satisfied* by the universe.)

Let's start with a brute fact about science:

Our theories, models, hypotheses, and what have you, are typically **not strictly true**.
(Technically, they are not *satisfied* by the universe.)



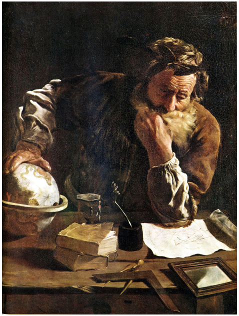
"Although this may seem a paradox, all exact science is dominated by the idea of approximation."

Let's start with a brute fact about science:

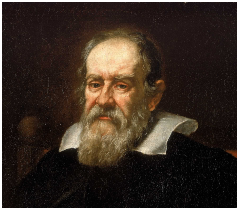
Our theories, models, hypotheses, and what have you, are typically **not strictly true**.
(Technically, they are not *satisfied* by the universe.)

"I can't get no satisfaction. [. . .] He's tellin' me more and more about some useless information."

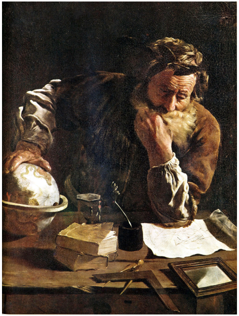
But that's OK. **Reasoning with false premises is a sign of greatness!**



To justify his use of idealizations in physics, Galileo claimed that he was following the example of Archimedes.

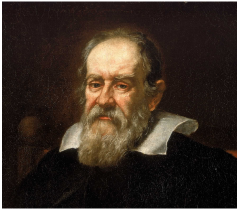


But that's OK. **Reasoning with false premises is a sign of greatness!**



To justify his use of idealizations in physics, Galileo claimed that he was following the example of Archimedes.

Archimedes had made the same false assumptions "**perhaps to show that he was so far ahead of others that he could draw true conclusions even from false assumptions**."



In real scientific practice, we need to rely on **approximate truth** (equivalently, accuracy) and related concepts.

In real scientific practice, we need to rely on **approximate truth** (equivalently, accuracy) and related concepts.

Some philosophers speak as if it were a small trivial thing, with **no deep consequence** concerning **what science fundamentally is**.

In real scientific practice, we need to rely on **approximate truth** (equivalently, accuracy) and related concepts.

Some philosophers speak as if it were a small trivial thing, with **no deep consequence** concerning **what science fundamentally is**. It couldn't be more false! . . . but if it were true, many of the good people working in applied maths would end up unemployed!

In real scientific practice, we need to rely on **approximate truth** (equivalently, accuracy) and related concepts.

Some philosophers speak as if it were a small trivial thing, with **no deep consequence** concerning **what science fundamentally is**. It couldn't be more false! . . . but if it were true, many of the good people working in applied maths would end up unemployed!

In addition to mischaracterizing scientific practice, focussing on truth-conditions generates many problems (e.g., about **idealizations**) that are quite hard to figure out from that perspective.

Clifford Truesdell very well explained why we're not seeking **the whole truth and nothing but the truth**:

"One good theory extracts and exaggerates some facets of the truth. Another good theory may idealize other facets. A theory cannot duplicate nature, for if it did so in all respects, it would be isomorphic to nature itself and hence useless, a mere repetition of all complexity which nature presents to us, that very complexity we frame theories to penetrate and set aside."

My view is that we avoid many difficulties by just changing our angle on these matters:

The aim of science is not primarily to generate accurate representations, but to accurately answer questions about real systems about which much is unknown.

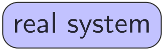
My view is that we avoid many difficulties by just changing our angle on these matters:

The aim of science is not primarily to generate accurate representations, but to accurately answer questions about real systems about which much is unknown.

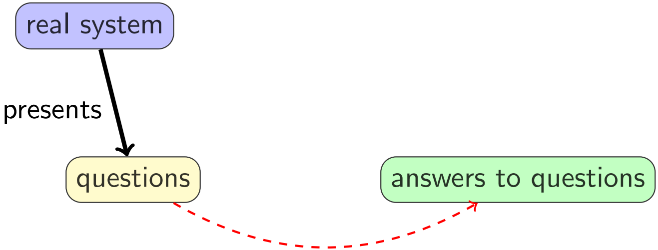
Thus, **mathematical modeling is a question-driven endeavour.**

Modeling as a question-driven endeavour

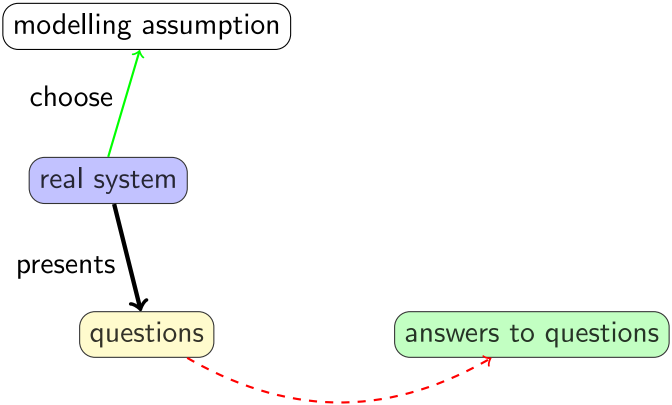
Here's the general picture:



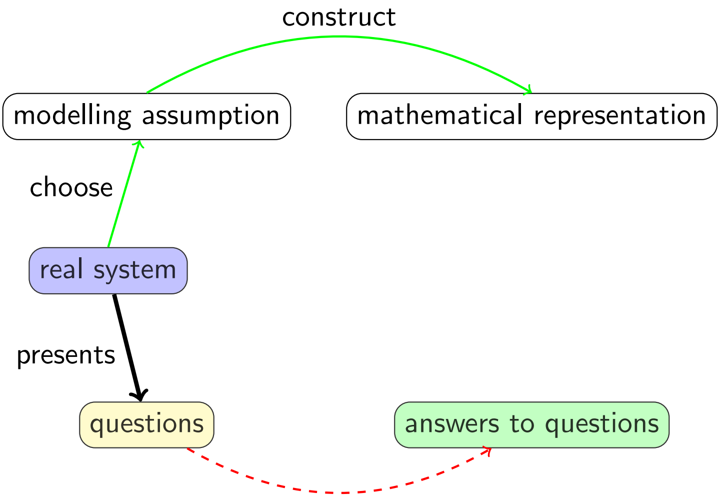
Here's the general picture:



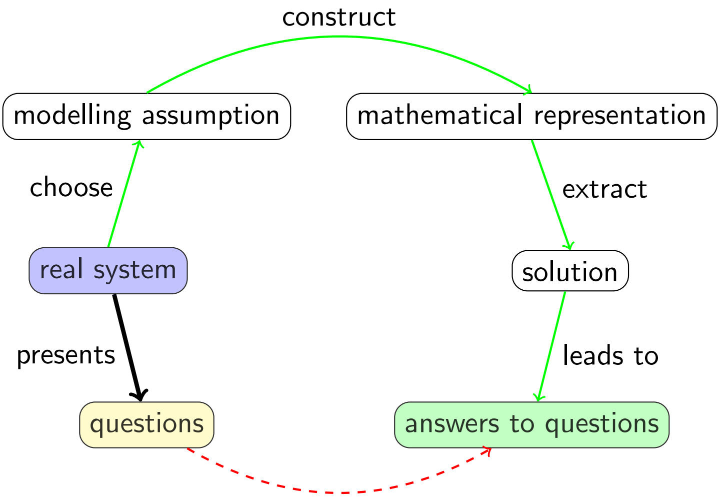
Here's the general picture:



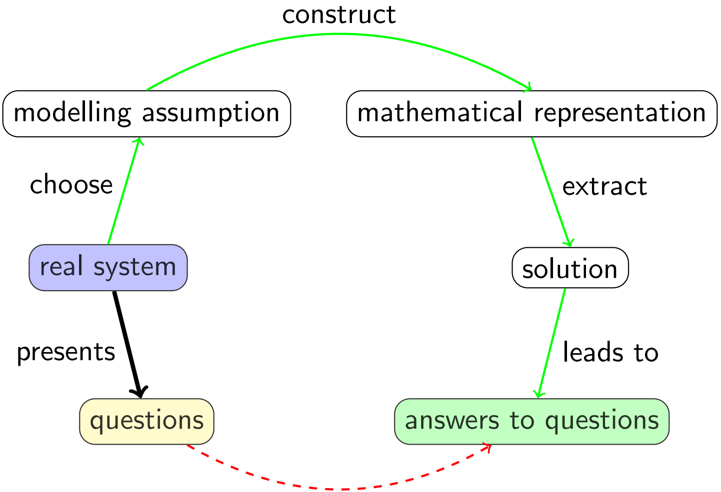
Here's the general picture:



Here's the general picture:



Which steps typically contain errors? Every key step!



This has three obvious consequences:

1. a good model doesn't have to capture all aspects of a system correctly, but only those we're interested with (behaviour of interest). I call this '**selective accuracy**'.

This has three obvious consequences:

1. a good model doesn't have to capture all aspects of a system correctly, but only those we're interested with (behaviour of interest). I call this '**selective accuracy**'.
2. a representation does not have to be true or even accurate in order to lead us to satisfactory answers.

This has three obvious consequences:

1. a good model doesn't have to capture all aspects of a system correctly, but only those we're interested with (behaviour of interest). I call this '**selective accuracy**'.
2. a representation does not have to be true or even accurate in order to lead us to satisfactory answers.
3. **It's more important for a model to be informative than true (or accurate).**

This has three obvious consequences:

1. a good model doesn't have to capture all aspects of a system correctly, but only those we're interested with (behaviour of interest). I call this '**selective accuracy**'.
2. a representation does not have to be true or even accurate in order to lead us to satisfactory answers.
3. **It's more important for a model to be informative than true (or accurate).**

Selectively accurate representations are only a **means to an end**. Everything else being equal, the more accurate the better, but everything else is rarely equal in actual scientific practice.

So, the key question now is:

Are the errors accumulated in this whole process leading us to **mischaracterize the behaviour of interest**?

So, the key question now is:

Are the errors accumulated in this whole process leading us to **mischaracterize the behaviour of interest**?

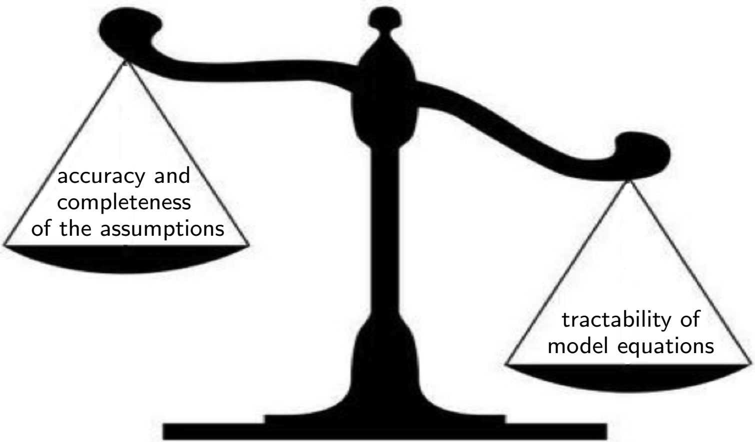
The tricky part is that we cannot just pretend that we can use the "**exact model** that gets all the details right" **as a benchmark**, since once we add all those details to get our modelling assumptions perfectly right (supposing we could do that), then the resulting model equations would likely be completely **intractable**

So, the key question now is:

Are the errors accumulated in this whole process leading us to **mischaracterize the behaviour of interest**?

The tricky part is that we cannot just pretend that we can use the "**exact model** that gets all the details right" **as a benchmark**, since once we add all those details to get our modelling assumptions perfectly right (supposing we could do that), then the resulting model equations would likely be completely **intractable** (in which case: no description, no prediction, no explanation, **no trifecta!**).

Assessing Models Requires a Delicate Balancing Act



In other words, some models are **too true to be good**.

The criteria for assessing the consequences of errors are thus the following:

- If we introduce error concerning a dominant factor, the representation will be invalidated.

The criteria for assessing the consequences of errors are thus the following:

- If we introduce error concerning a dominant factor, the representation will be invalidated.
- If we introduce error concerning a non-dominant factor, the representation will be selectively accurate.

The criteria for assessing the consequences of errors are thus the following:

- If we introduce error concerning a dominant factor, the representation will be invalidated.
- If we introduce error concerning a non-dominant factor, the representation will be selectively accurate.

From this point of view, the **epistemological burden** is to **determine the impact of a factor**.

The criteria for assessing the consequences of errors are thus the following:

- If we introduce error concerning a dominant factor, the representation will be invalidated.
- If we introduce error concerning a non-dominant factor, the representation will be selectively accurate.

From this point of view, the **epistemological burden** is to **determine the impact of a factor**.

The general method to determine this is **perturbation analysis**.

Perturbation analysis examines the **effects of small changes** of an aspect of a representation (**tweaking a parameter or a functional term**).

Some systems are sensitive to changes in some aspects, others are **robust under perturbation**.

Assessing Models Requires a Delicate Balancing Act

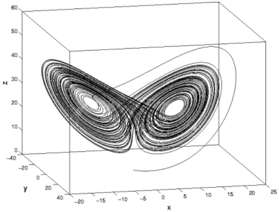
Perturbation analysis examines the **effects of small changes** of an aspect of a representation (**tweaking a parameter or a functional term**).

Some systems are sensitive to changes in some aspects, others are **robust under perturbation**.

Assessing Models Requires a Delicate Balancing Act

Perturbation analysis examines the **effects of small changes** of an aspect of a representation (**tweaking a parameter or a functional term**).

Some systems are sensitive to changes in some aspects, others are **robust under perturbation**.



Moreover, the robustness can change in functions of some parameters.

In a qualitative analysis, we can find **bifurcation points** that will allow us to determine the situation's sensitivity.

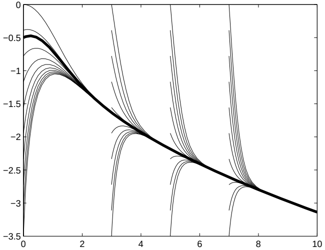
Assessing Models Requires a Delicate Balancing Act

Moreover, the robustness can change in functions of some parameters.

In a qualitative analysis, we can find **bifurcation points** that will allow us to determine the situation's sensitivity.

Consider the qualitative change in the solution of

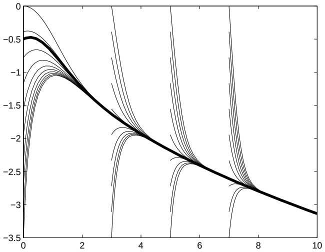
$$\frac{dx}{dt} = x^2 - t$$



Assessing Models Requires a Delicate Balancing Act

Moreover, the robustness can change in functions of some parameters.

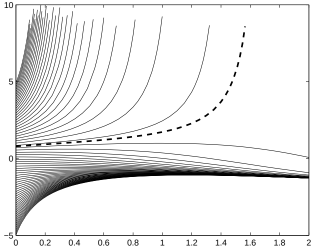
In a qualitative analysis, we can find **bifurcation points** that will allow us to determine the situation's sensitivity.



Consider the qualitative change in the solution of

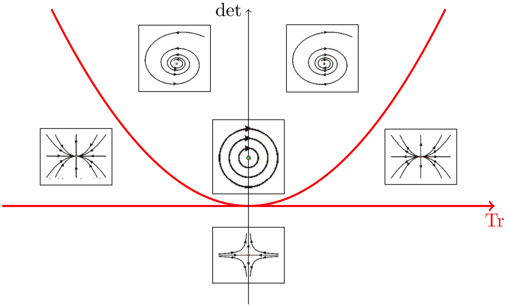
$$\frac{dx}{dt} = x^2 - t$$

near $x(0) = \sqrt[3]{1/2}$ (which is the dotted line).



Assessing Models Requires a Delicate Balancing Act

Classification of all 2D linear diff. equations $d\mathbf{x}/dt = \mathbf{A}\mathbf{x}$ by two parameters.



Knowing critical and bifurcation points is **typically easier** than working with a "perfectly accurate and complete model," so there is an **epistemic gain**.

My suggestion is that, to the extent that the **success of mathematics remains mysterious or miraculous-looking**, it is because we have **failed to understand how science can prosper while pervaded with error and uncertainty**.

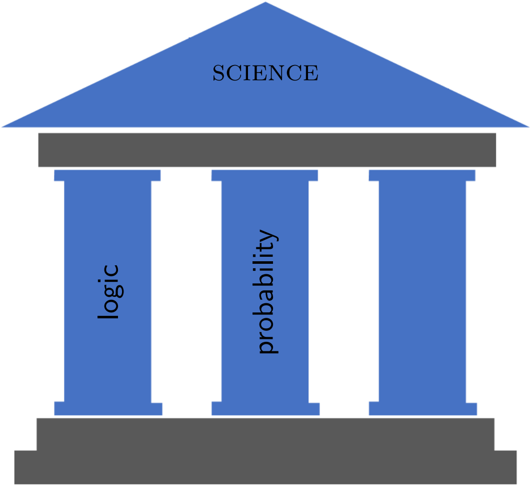
This is a **failure** to appreciate **how to establish trust** in mathematical methods.

My suggestion is that, to the extent that the **success of mathematics remains mysterious or miraculous-looking**, it is because we have **failed to understand how science can prosper while pervaded with error and uncertainty**.

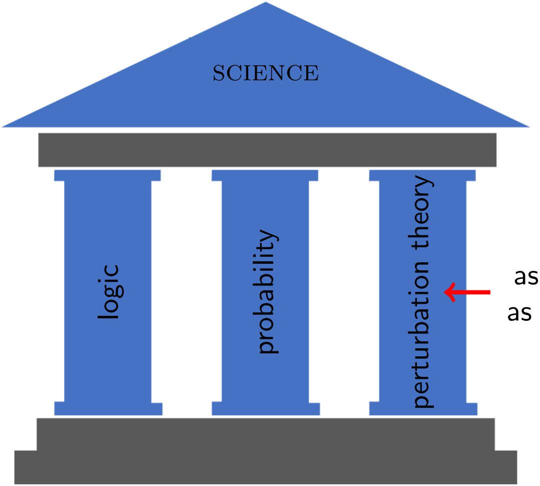
This is a **failure** to appreciate **how to establish trust** in mathematical methods.

Moreover, to the extent that there is such a failure, it is because we have an **insufficiently rich set of rational reconstruction tools** in our philosophical toolbox.

Assessing Models Requires a Delicate Balancing Act

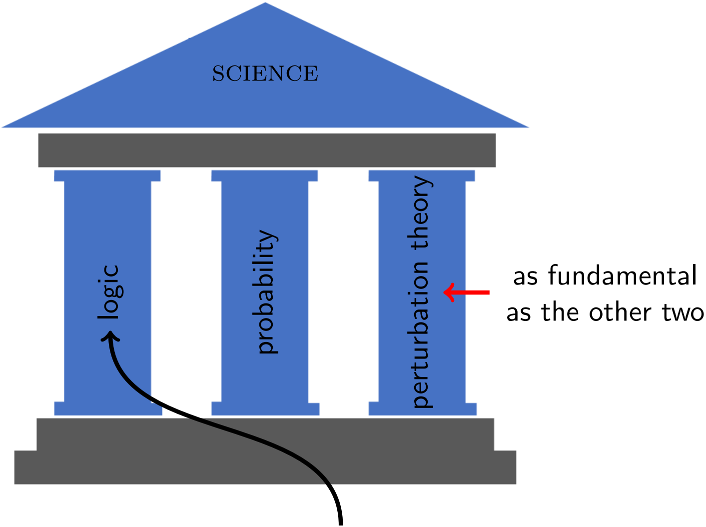


Assessing Models Requires a Delicate Balancing Act



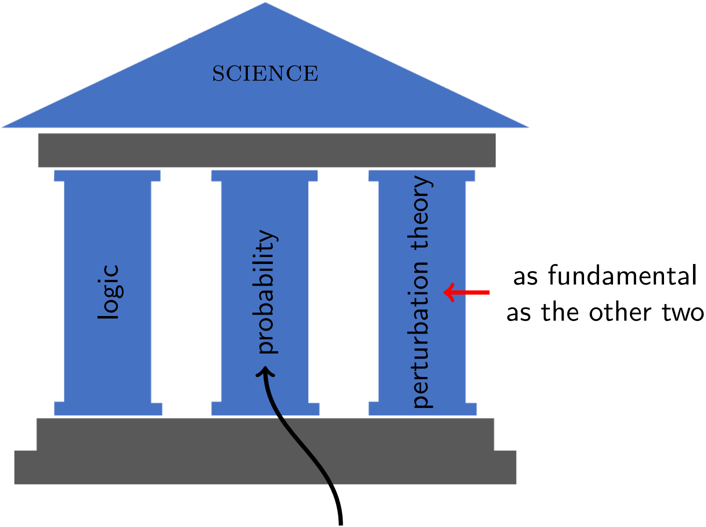
as fundamental
as the other two

Assessing Models Requires a Delicate Balancing Act



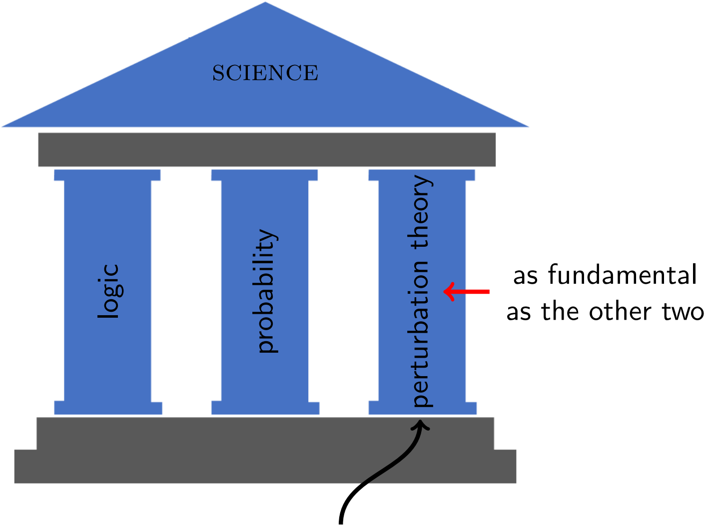
Is P true? Does P follow from Σ?

Assessing Models Requires a Delicate Balancing Act



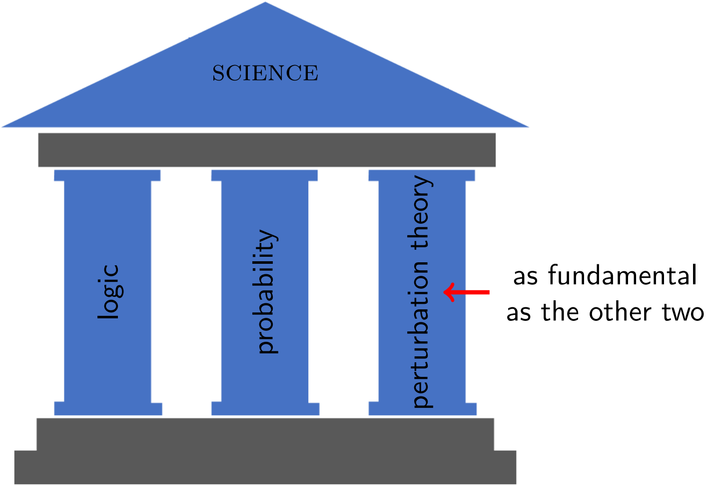
How probable is P given Σ? How confident am I in P?

Assessing Models Requires a Delicate Balancing Act



What are the consequences of tweaking paramaters in P?

Assessing Models Requires a Delicate Balancing Act



I call them the **three pillars of scientific rationality**.

But more generally, what is the **basic strategy** deployed in perturbative reasoning?

But **more generally**, what is the **basic strategy** deployed in perturbative reasoning?

Start with an **analogy**:
s'habiller comme un oignon

A flexible way to dress for a wide variety of temparatures, exercise levels, etc.
You're never **perfectly comfortable**, but you're always **pretty much alright**.

But **more generally**, what is the **basic strategy** deployed in perturbative reasoning?

Start with an **analogy**:
s'habiller comme un oignon

A flexible way to dress for a wide variety of temparatures, exercise levels, etc.
You're never **perfectly comfortable**, but you're always **pretty much alright**.

In honor of this practice, I call the strategy **semantic layering**.

For lack of a precise technical characterization of what **semantic layering** is (working on it!), I'll give three simple examples widespread in scientific practice:

1. An example from linear algebra (SVD processing).
2. An example from ODEs (marching methods).
3. An example from multidimensional interpolation (matching boundary conditions).

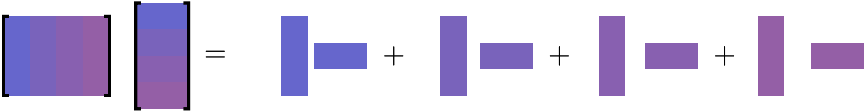
Consider an $m \times n$ matrix A (any m, n will do).

What can be interesting about an array of numbers? Well, lots!

First, re-think how you think of **matrix multiplication**. Go from thinking about $A \times B$ as a matrix with elements

$$c_{ij} = \sum a_{ik} b_{jk}$$

to the **outer product** view:



That's a sum of rank-1 matrices (layers)!

Linear algebra: image processing using SVDs

Images are matrices!
The classic Mandrill example
of destructive compression
uses just this approach!

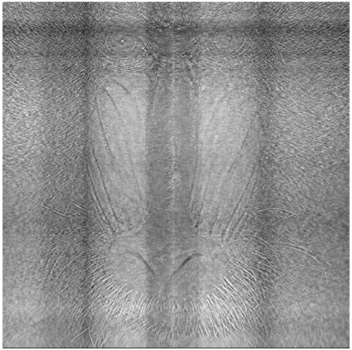
Linear algebra: image processing using SVDs

Images are matrices!
The classic Mandrill example of destructive compression uses just this approach!

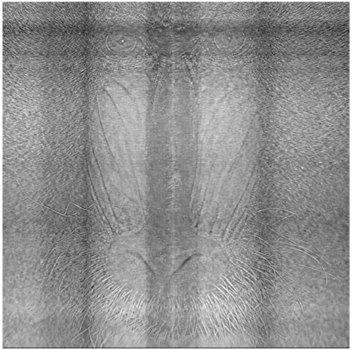
Now, here's one with 90% of the layers.

What's gone **wrong**?

rank=462, i.e 90% of information



Images are matrices!
The classic Mandrill example of destructive compression uses just this approach!

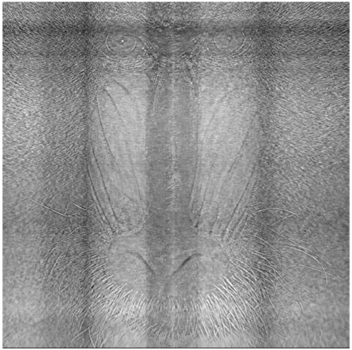


Now, here's one with 90% of the layers.

What's gone **wrong**?

I've used a bad **layering method**.

Images are matrices!
The classic Mandrill example of destructive compression uses just this approach!



Now, here's one with 90% of the layers.

What's gone **wrong**?

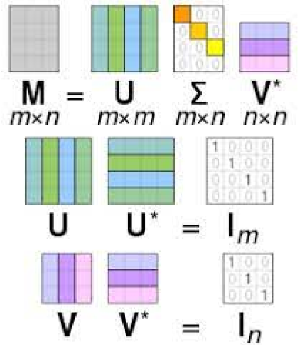
I've used a bad **layering method**.
A **good layering method** is one that decomposes layers in order of importance to **create an impression of monotonicity**.

I haven't cherry-picked my image.

It works with images of other primates as well!

(only 5.2% of the layers)

Last minute addition: Had I known we'd do linear regressions yesterday, I would have expanded more on this technique.
It's based on the the **holy grail of numerical linear algebra**, i.e., the **singular value decomposition** (pic lifted from Wiki):



Destructive image compression, linear regressions, factor analysis, Lyapunov eponents, etc. are just SVD!

Solution of ODEs: marching methods

Take a differential equation $\frac{dx}{dt} = f(x,t)$. Write its solution $x(t)$ as an asymptotic power series about t_0:

$$x(t) = x(t_0) + x'(t_0)(t - t_0) + \frac{x''(t_0)}{2}(t - t_0)^2 + \frac{x'''(t_0)}{6}(t - t_0)^3 + \cdots$$

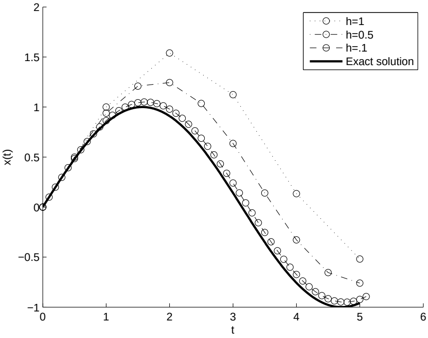
Take a differential equation $\frac{dx}{dt} = f(x,t)$. Write its solution $x(t)$ as an asymptotic power series about t_0:

$$x(t) = x(t_0) + x'(t_0)(t - t_0) + \frac{x''(t_0)}{2}(t - t_0)^2 + \frac{x'''(t_0)}{6}(t - t_0)^3 + \cdots$$

Each term in the Taylor series is a layer; the method **creates an impression of monotonicity**.

Take a differential equation $\frac{dx}{dt} = f(x,t)$. Write its solution $x(t)$ as an asymptotic power series about t_0:

$$\mathbf{x(t) = x(t_0) + x'(t_0)(t - t_0)} + \frac{x''(t_0)}{2}(t-t_0)^2 + \frac{x'''(t_0)}{6}(t-t_0)^3 + \cdots$$



Each term in the Taylor series is a layer; the method **creates an impression of monotonicity**.

Truncate after **first order term** and use this as a "marching method" through the vector field.

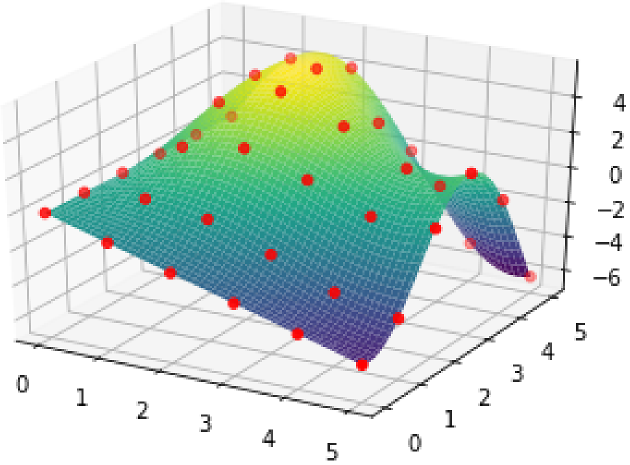
The same idea can be deployed to approximate **multivariate functions** $f(x_1, x_2, \ldots, x_n)$:
this time, we use multivariate gauge functions in our asymptotic series.

The formula for multivariate Taylor series looks more messy, but it's **conceptually as simple** as the former case:

It representents $f(x_1, x_2, \ldots, x_n)$ as an infinite superposition of layers that **create an impression of monotonicity**.

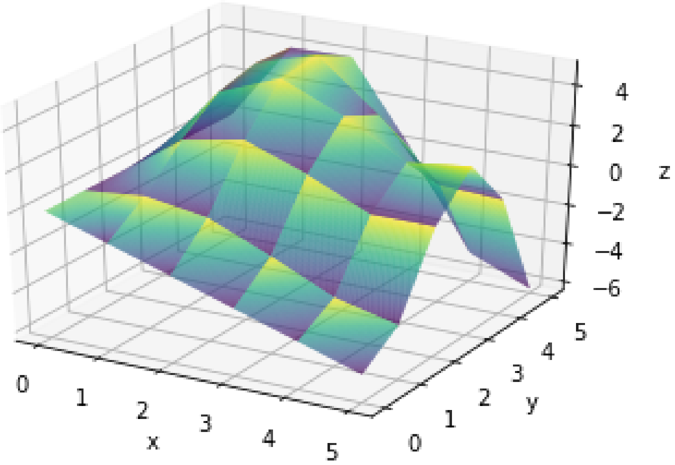
Multidimensional interpolation

Example of increasingly higher-order approximations:



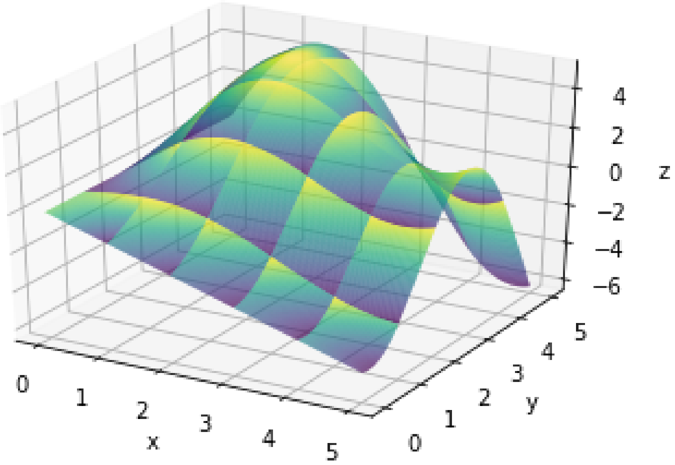
Example of a surface to approximate.

Example of increasingly higher-order approximations:



A so-called **bilinear interpolant**.

Example of increasingly higher-order approximations:



A so-called **quadratic interpolant**.

In the next sections, I'll present a few views on **computing** that deploys those ideas in a systematic framework. My aim is to convince you of the following thesis:

You should trust your computer's solutions precisely when constructing a simplified (or idealized), layered model would be justifiable (in some sense to be determined).

Two perspectives on computing

There are at least two mainstreams of views about computing:

1. Computation theory (Turing machine paradigm).

2. Scientific computing (Numerical analysis paradigm).

Two perspectives on computing

There are at least two mainstreams of views about computing:

1. Computation theory (Turing machine paradigm).
   Paradigmatic algorithm is Euclidean GCD algorithm.
   Exact integer-valued computation.
2. Scientific computing (Numerical analysis paradigm).

Two perspectives on computing

There are at least two mainstreams of views about computing:

1. Computation theory (Turing machine paradigm).
   Paradigmatic algorithm is Euclidean GCD algorithm.
   Exact integer-valued computation.
2. Scientific computing (Numerical analysis paradigm).
   Paradigmatic algorithm is Newton's method for rootfinding or Euler's method for solving ODEs.
   Inexact real/complex-valued computation.

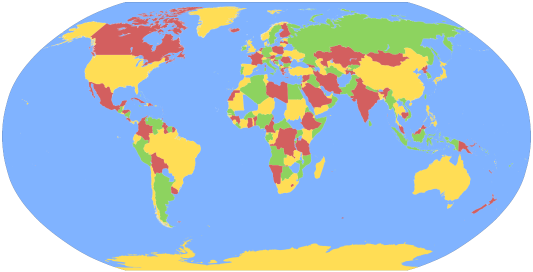
There are at least two mainstreams of views about computing:

1. Computation theory (Turing machine paradigm).
   Paradigmatic algorithm is Euclidean GCD algorithm.
   Exact integer-valued computation.
2. Scientific computing (Numerical analysis paradigm).
   Paradigmatic algorithm is Newton's method for rootfinding or Euler's method for solving ODEs.
   Inexact real/complex-valued computation.

Most real-world problems require the second perspective (e.g., most scientific simulations are based on the second paradigm)—but is it an **essentially different** view of computing? **Yes**.

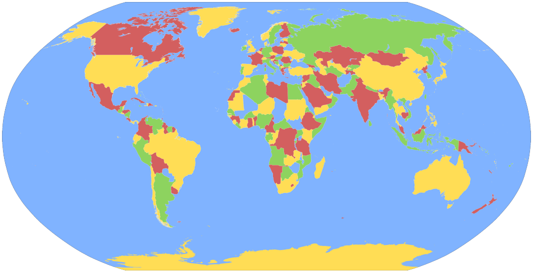
Sure, computation is based on "**algorithms**", but in **practice** this can mean very different things...

A lot of attention devoted to computing in philosophy of mathematics has to do with things like the 4-color theorem:



Sure, computation is based on "**algorithms**", but in **practice** this can mean very different things. . .

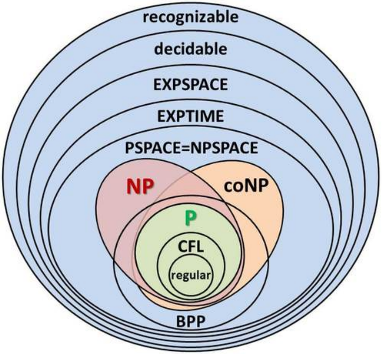
A lot of attention devoted to computing in philosophy of mathematics has to do with things like the 4-color theorem:



Here computers contribute by surveying massive problem involving **case-by-case brute force discrete computation**.

Sure, computation is based on "**algorithms**", but in **practice** this can mean very different things. . .

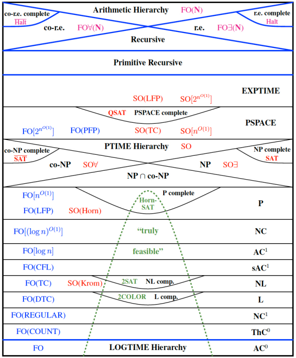
That essentially **combinatorial** idea is what is articulated in **complexity theory**:



Two perspectives on computing

Sure, computation is based on "**algorithms**", but in **practice** this can mean very different things...

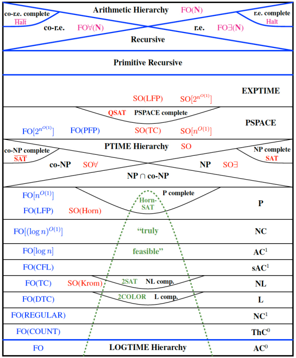
That essentially **combinatorial** idea is what is articulated in **complexity theory**:



Two perspectives on computing

Sure, computation is based on "**algorithms**", but in **practice** this can mean very different things...

That essentially **combinatorial** idea is what is articulated in **complexity theory**:



This approach is inspired by **meta-mathematics** and **theoretical computer science**.

$P = NP$? Who cares? For SVDs and PDEs, $O(n^3)$ is already pushing the limits!

Let's say a bit more about orders of complexity...

Defining **computational complexity** demands a bit more work. For instance, take the problem of finding the determinant of a matrix $A \in \mathbb{R}^{n \times n}$ using two methods:

Defining **computational complexity** demands a bit more work. For instance, take the problem of finding the determinant of a matrix $A \in \mathbb{R}^{n\times n}$ using two methods:

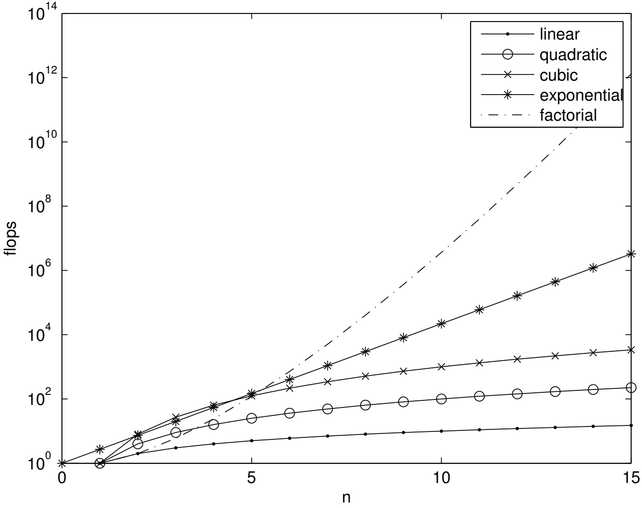
1. Lapacian expansion by minors:

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} M_{ij} \ . \tag{1}$$

   This recursion has a cost of $O(n \cdot (n-1) \cdots 2 \cdot 1) = O(n!)$.
2. Finding trace of matrix diagonalized by Gaussian elimination. The computational cost is only $O(n^3)$ operations.

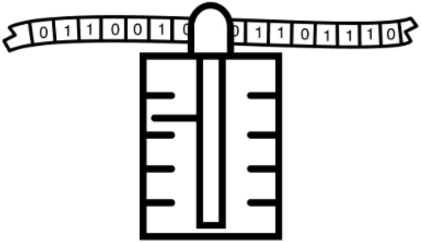
They have **very** different **orders of computational complexity**.

Logarithmic scale plot of orders of computational cost.

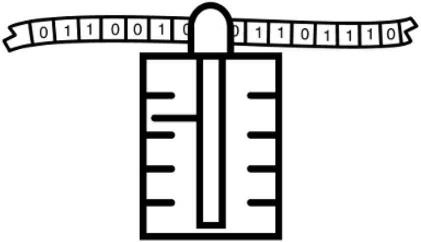


Turing computation

The **Turing machine** model of computation is perfect to understand this concept of complexity.

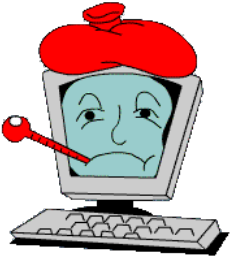


The **Turing machine** model of computation is perfect to understand this concept of complexity.



It elaborates a notion of computation based on **effective computability**, and an idea of what is **truly feasible** by further adding **constraints on time and memory** for given implementations on digital computers.

The **Turing machine** model of computation is perfect to understand this concept of complexity.



It elaborates a notion of computation based on **effective computability**, and an idea of what is **truly feasible** by further adding **constraints on time and memory** for given implementations on digital computers.

However, this is misleading—is does **not** give a good image of what is **truly feasible** in practice.

However, this is misleading—is does **not** give a good image of what is **truly feasible** in practice.

We can see it by considering this intriguing quote from Nick Trefethen (a Jedi master of Num.An.):

> *[...the numerical analysts'] central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.*

However, this is misleading—is does **not** give a good image of what is **truly feasible** in practice.

We can see it by considering this intriguing quote from Nick Trefethen (a Jedi master of Num.An.):

> *[. . . the numerical analysts'] central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.*

Computing something incomputable? At lighting speed?
Has the Master gone mad?

No, he's using the root 'compute' in two different senses.

Neither modeling nor simulating are error-free:

1. Systemic Error
2. Experimental Error

3. **Discretization** Error
4. **Rounding** Error

Sources of error in mathematical modelling

Neither modeling nor simulating are error-free:

1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error

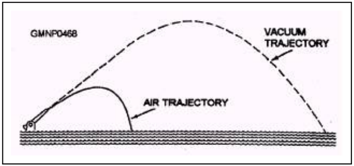
Sources of error in mathematical modelling

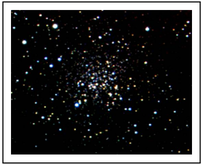
Neither modeling nor simulating are error-free:

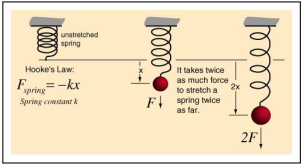
1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error







Neither modeling nor simulating are error-free:

1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error

} **Computational** Error

Sources of error in mathematical modelling

Neither modeling nor simulating are error-free:

1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error

} **Computational** Error

fcts, integrals, etc. $\xrightarrow{\text{truncate}}$ truncated asymptotic series

$f(x)$, $\int g(x)$, etc. $\quad$ $y(x, \varepsilon) = \sum_{k=0}^{N} y_k(x)\phi_k(\varepsilon)$

Neither modeling nor simulating are error-free:

1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error

} **Computational** Error

flow
$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}(t); \mu)$

$\xrightarrow{\text{discretization}}$

discrete functions (maps)
$x_{k+1} = \Phi(t_k, x_k, \ldots, x_0, h, \mathbf{f})$

Neither modeling nor simulating are error-free:

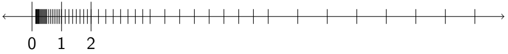
1. Systemic Error
2. Experimental Error

} Modelling Error
incl. ideal./simpl./omission/etc

3. **Discretization** Error
4. **Rounding** Error

} **Computational** Error

Operations on $\mathbb{R}, \mathbb{C}$ $\xrightarrow{\text{rounding}}$ floating-point arithmetic $\mathbb{F}$



Bit type	S	E	E	E	E	E	E	E	E	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
Bit number	1	2							9	10																						32

Here, the **crucial epistemological question** is:

When we don't know the exact solution of a model, how do we determine if our "approximate" solution is **sufficiently accurate**?

Here, the **crucial epistemological question** is:

When we don't know the exact solution of a model, how do we determine if our "approximate" solution is **sufficiently accurate**?

Even if it might seem counter-intuitive, **it is generally easier to determine whether we're close enough to the truth than to know what the truth is**!

Here, the **crucial epistemological question** is:

When we don't know the exact solution of a model, how do we determine if our "approximate" solution is **sufficiently accurate**?

Even if it might seem counter-intuitive, **it is generally easier to determine whether we're close enough to the truth than to know what the truth is**!

I will further argue that the question **makes no sense** if we don't consider a **specific** (collection of) **modelling context**(s) — so I argue for a variant of the sig. fig. approach.

Backward Error Analysis

A fruiful perspective on error in scientific computing is **backward error analysis**. Let me sketch it. . .

A fruiful perspective on error in scientific computing is **backward error analysis**. Let me sketch it. . .

We represent a mathematical problem by an operator φ, that has an **input** (data) space $\mathscr{I}$ as its domain and an **output** (result, solution) space $\mathscr{O}$ as its codomain:

$$\varphi : \mathscr{I} \to \mathscr{O},$$

and we write $y = \varphi(x)$. (φ can be a function or some other operator.)

A fruiful perspective on error in scientific computing is **backward error analysis**. Let me sketch it. . .

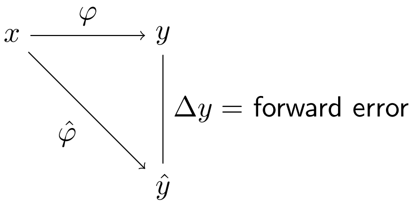
We represent a mathematical problem by an operator φ, that has an **input** (data) space $\mathscr{I}$ as its domain and an **output** (result, solution) space $\mathscr{O}$ as its codomain:

$$\varphi : \mathscr{I} \to \mathscr{O},$$

and we write $y = \varphi(x)$. (φ can be a function or some other operator.)

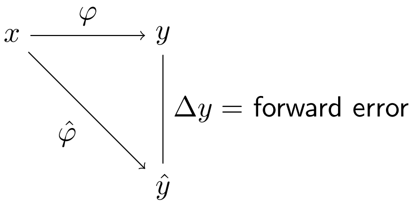
As we said, when the **problem stems from a realistic modelling context**, it typically **can't be solved directly**.

Accordingly, we introduce the notion of an **engineered problem** $\hat{\varphi}$ (which is by design computable):



We also call φ the **reference problem**.

Accordingly, we introduce the notion of an **engineered problem** $\hat{\varphi}$ (which is by design computable):



We also call φ the **reference problem**.
"Wrong" question: Is Δy small enough?

Instead, we write $\hat{y} = \hat{\varphi}(x)$. Then, instead of saying that $\hat{y}$ is the **approximate solution to** φ (the reference problem), we say that it is the **exact solution to** $\hat{\varphi}$ (the engineered problem).

But we can go further, and "reflect back" the forward error:

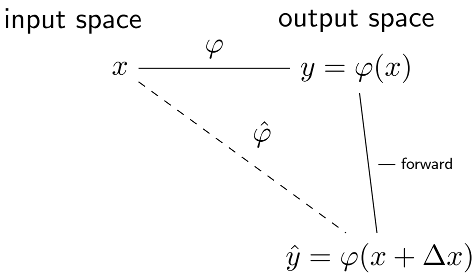


Figure: Backward error analysis: The general picture.

But we can go further, and “reflect back” the forward error:

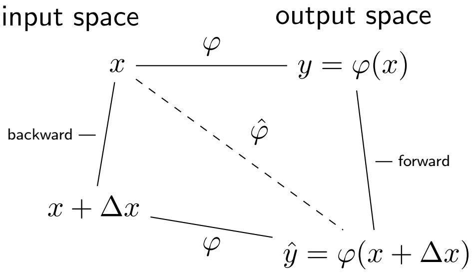


Figure: Backward error analysis: The general picture.

But we can go further, and “reflect back” the forward error:

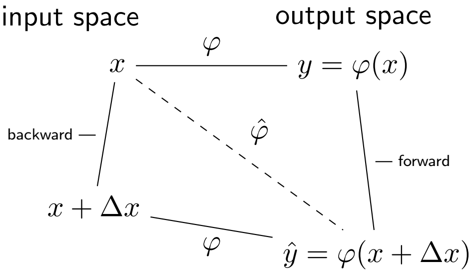


Figure: Backward error analysis: The general picture.

The smallest such Δx is what is called the **backward error**.

Perhaps more nicely, when problems working with a ring of formal power series, we can rigorously define "approximately" commuting diagram in which we can replace '≈' by the order to which the approximation holds.

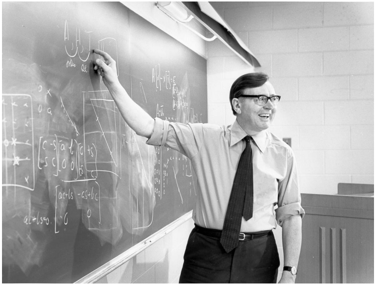
$$
\begin{array}{ccc}
x & \longrightarrow & y = \varphi(x) \\
\le \eta \Big| & \approx & \Big| \le \varepsilon \\
x + \Delta x & \longrightarrow & \hat{y} = \varphi(x + \Delta x)
\end{array}
$$

That gives us three different but interrelated kinds of errors:

1. forward error
2. backward error
3. residual

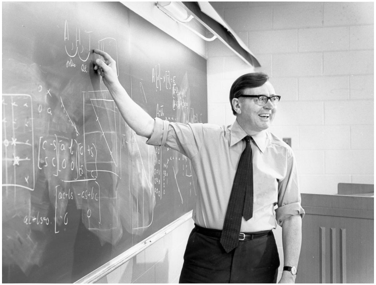
They are used in a number of ways, and measured in a number of ways, resulting in **different standards of accuracy**.

Here's one of the first historical case of Backward Error Analysis.



BEA has become broadly influential in the 1990s and systematized for the first time in the late 60s with the works on Wilkinson on linear algebra and algebraic equations.

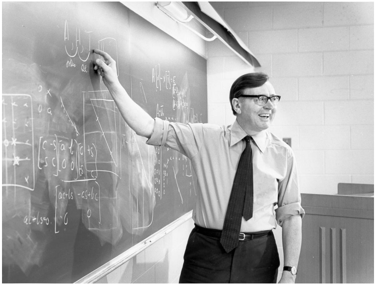
Here's one of the first historical case of Backward Error Analysis.



BEA has become broadly influential in the 1990s and systematized for the first time in the late 60s with the works on Wilkinson on linear algebra and algebraic equations.

Suppose you want to solve $\mathbf{Ax} = \mathbf{b}$. You (unwisely) choose to use Gaussian elimination without pivoting to find an approximate $\hat{\mathbf{x}}$.

Here's one of the first historical case of Backward Error Analysis.



BEA has become broadly influential in the 1990s and systematized for the first time in the late 60s with the works on Wilkinson on linear algebra and algebraic equations.

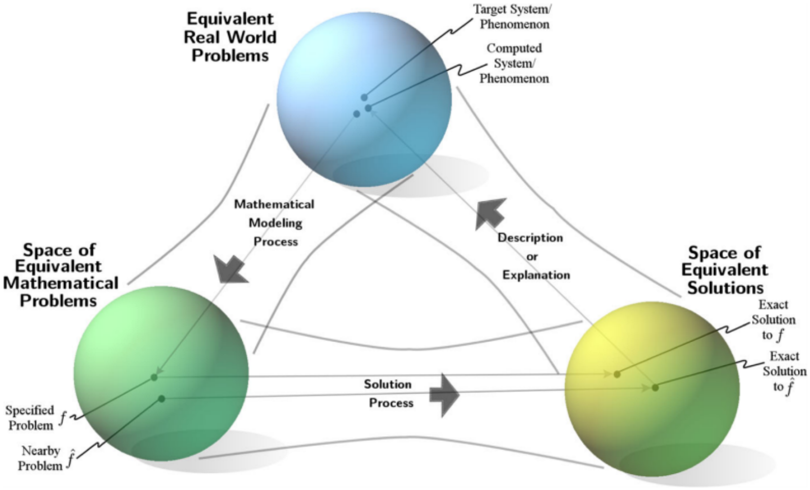
Suppose you want to solve $\mathbf{Ax} = \mathbf{b}$. You (unwisely) choose to use Gaussian elimination without pivoting to find an approximate $\hat{\mathbf{x}}$. Wilkinson showed that there exists a matrix $\mathbf{E}$ with "relatively small" entries such that $(\mathbf{A} + \mathbf{E})\hat{\mathbf{x}} = \mathbf{b}$. That is, the method exactly solved a slightly different problem.

Then, the situation is this:

⇒ If solving the problem $\hat{\varphi}(x)$ **amounts to having solved the problem $\varphi(x + \Delta x)$ for a Δx smaller than the perturbations inherent in the modeling context** (specifying estimates of error and uncertainty), then our solution $\hat{y}$ must be considered **completely satisfactory**.

The focus has shifted from small forward error to small perturbation of the input.

Backward-Error Analysis in a picture:



Let's take stock.

Again, consider Nick Trefethen's quote:

> *[. . . the numerical analysts'] central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.*

What are the two senses of 'computable'?

Let's take stock.

Again, consider Nick Trefethen's quote:

> *[...the numerical analysts'] central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.*

What are the two senses of 'computable'?

1. A problem is computable if you can find an algorithm that exactly computes it in finite time.
2. A problem is computable if there's an easy-to-compute nearby problem that you can compute instead.

Let's take stock.

Again, consider Nick Trefethen's quote:

> *[. . . the numerical analysts'] central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.*

What are the two senses of 'computable'?

1. A problem is computable if you can find an algorithm that exactly computes it in finite time.
2. A problem is computable if there's an easy-to-compute nearby problem that you can compute instead.

The latter gives a **very** different perspective on computab**ility**.

Now, the next question is: *what is the relationship between the forward and the backward error*?

The relationship we seek lies in a **problem-specific** coefficient of magnification, *i.e.*, the sensitivity of the solution to perturbations in the data, that we call the **conditioning of the problem**.

Now, the next question is: *what is the relationship between the forward and the backward error*?

The relationship we seek lies in a **problem-specific** coefficient of magnification, *i.e.*, the sensitivity of the solution to perturbations in the data, that we call the **conditioning of the problem**.

The **normwise relative condition number** κ is the maximum of the ratio of the relative change in the solution to the relative change in input, which is expressed by

$$\kappa_{rel} = \sup_x \frac{\|\delta y\|}{\|\delta x\|} = \sup_x \frac{\|\frac{\Delta y}{y}\|}{\|\frac{\Delta x}{x}\|} = \sup_x \frac{\|\frac{(\varphi(\hat{x}) - \varphi(x))}{\varphi(x)}\|}{\|\frac{\hat{x}-x}{x}\|}$$

for some norm $\|\cdot\|$.

Now, the next question is: *what is the relationship between the forward and the backward error*?

The relationship we seek lies in a **problem-specific** coefficient of magnification, *i.e.*, the sensitivity of the solution to perturbations in the data, that we call the **conditioning of the problem**.

The **normwise relative condition number** κ is the maximum of the ratio of the relative change in the solution to the relative change in input, which is expressed by

$$\kappa_{rel} = \sup_x \frac{\|\delta y\|}{\|\delta x\|} = \sup_x \frac{\|\frac{\Delta y}{y}\|}{\|\frac{\Delta x}{x}\|} = \sup_x \frac{\|\frac{(\varphi(\hat{x}) - \varphi(x))}{\varphi(x)}\|}{\|\frac{\hat{x}-x}{x}\|}$$

for some norm $\|\cdot\|$. Note: this is just the **sensitivity** measure from perturbation theory; it really introduces nothing fundamentally new, but it's **more quantitatively precise**.

As a result, we obtain the relation

$$\|\delta y\| \leq \kappa_{rel} \|\delta x\| \tag{2}$$

between the forward and the backward error. Knowing the backward error and the conditioning thus gives us an **upper bound** on the forward error.

As a result, we obtain the relation

$$\|\delta y\| \leq \kappa_{rel} \|\delta x\| \tag{2}$$

between the forward and the backward error. Knowing the backward error and the conditioning thus gives us an **upper bound** on the forward error.

If κ has a moderate size, we say that the problem is **well-conditioned**. Otherwise, we say that the problem is **ill-conditioned**.

As a result, we obtain the relation

$$\|\delta y\| \leq \kappa_{rel} \|\delta x\| \tag{2}$$

between the forward and the backward error. Knowing the backward error and the conditioning thus gives us an **upper bound** on the forward error.

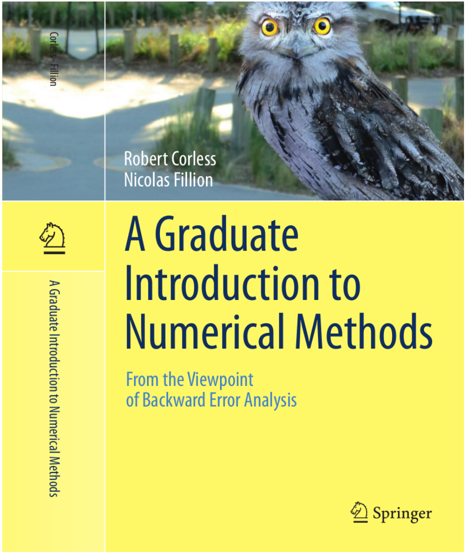
If κ has a moderate size, we say that the problem is **well-conditioned**. Otherwise, we say that the problem is **ill-conditioned**.

Note: even for a very good algorithm, the approximate solution to an ill-conditioned problem may have a large forward error.

Words of wisdom from Rob Corless:

"[. . .] most people will have to deal eventually with the fact that mathematical problems encountered in science and engineering are usually merely one representative out of an infinite class of mathematical models for the phenomenon in question, and further that the input data to the model will usually be of low accuracy compared to the precision available on most computers or calculators. In such cases, fanatical obsession with accurately solving the specified model problem is neither necessary nor appropriate, while analysis of the effect of perturbations of the input data and/or the model is essential." Corless (1993)

Shameless self-promotion!



For a given problem φ, **the image y can have many forms**. *E.g.*, if the reference problem φ consists in finding the roots of the equation $\xi^2 + x\xi + 2 = 0$, then for each value of x the object y will be a set containing two numbers satisfying $\xi^2 + x\xi + 2 = 0$, *i.e.*,

$$y = \left\{ \xi \;\middle|\; \xi^2 + x\xi + 2 = 0 \right\}. \tag{3}$$

For a given problem φ, **the image y can have many forms**. *E.g.*, if the reference problem φ consists in finding the roots of the equation $\xi^2 + x\xi + 2 = 0$, then for each value of x the object y will be a set containing two numbers satisfying $\xi^2 + x\xi + 2 = 0$, *i.e.*,

$$y = \left\{ \xi \;\middle|\; \xi^2 + x\xi + 2 = 0 \right\}. \tag{3}$$

In general, we can then define a problem to be a map

$$x \xrightarrow{\quad \varphi \quad} \left\{ \xi \mid \phi(x, \xi) = 0 \right\}, \tag{4}$$

where $\phi(x, \xi)$ is some function of the input x and the output ξ.

For a given problem φ, **the image y can have many forms**. *E.g.*, if the reference problem φ consists in finding the roots of the equation $\xi^2 + x\xi + 2 = 0$, then for each value of x the object y will be a set containing two numbers satisfying $\xi^2 + x\xi + 2 = 0$, *i.e.*,

$$y = \left\{ \xi \;\middle|\; \xi^2 + x\xi + 2 = 0 \right\}. \tag{3}$$

In general, we can then define a problem to be a map

$$x \xrightarrow{\quad \varphi \quad} \left\{ \xi \;\middle|\; \phi(x, \xi) = 0 \right\}, \tag{4}$$

where $\phi(x, \xi)$ is some function of the input x and the output ξ. The function $\phi(x, \xi)$ is called the **defining function**

For a given problem φ, **the image y can have many forms**. *E.g.*, if the reference problem φ consists in finding the roots of the equation $\xi^2 + x\xi + 2 = 0$, then for each value of x the object y will be a set containing two numbers satisfying $\xi^2 + x\xi + 2 = 0$, *i.e.*,

$$y = \left\{ \xi \;\middle|\; \xi^2 + x\xi + 2 = 0 \right\}. \tag{3}$$

In general, we can then define a problem to be a map

$$x \xrightarrow{\quad \varphi \quad} \left\{ \xi \mid \phi(x, \xi) = 0 \right\}, \tag{4}$$

where $\phi(x, \xi)$ is some function of the input x and the output ξ. The function $\phi(x, \xi)$ is called the **defining function** and the equation $\phi(x, \xi) = 0$ is called the **defining equation** of the problem.

We can now give a general definition of residual.

Given the reference problem φ—whose value at x is a y such that the defining equation $\phi(x,y)=0$ is satisfied—and an engineered problem $\hat{\varphi}$, the residual r is defined by

$$r = \phi(x, \hat{y}). \tag{5}$$

We can now give a general definition of residual.

Given the reference problem φ—whose value at x is a y such that the defining equation $\phi(x, y) = 0$ is satisfied—and an engineered problem $\hat{\varphi}$, the residual r is defined by

$$r = \phi(x, \hat{y}). \tag{5}$$

As we see, we obtain the residual by substituting the computed value $\hat{y}$ (*i.e.*, the exact solution of the engineered problem) for y as the second argument of the defining function.

We can now give a general definition of residual.

Given the reference problem φ—whose value at x is a y such that the defining equation $\phi(x,y)=0$ is satisfied—and an engineered problem $\hat{\varphi}$, the residual r is defined by

$$r=\phi(x,\hat{y}). \tag{5}$$

As we see, we obtain the residual by substituting the computed value $\hat{y}$ (*i.e.*, the exact solution of the engineered problem) for y as the second argument of the defining function.

The residual is always computable if the defining equation is closed-form.

Residual-based *a posteriori* backward error analysis then proceeds as follows:

1. For the problem φ, use an engineered version of the problem to compute the value $\hat{y} = \hat{\varphi}(x)$.

Residual-based *a posteriori* backward error analysis then proceeds as follows:

1. For the problem φ, use an engineered version of the problem to compute the value $\hat{y} = \hat{\varphi}(x)$.
2. Compute the residual.

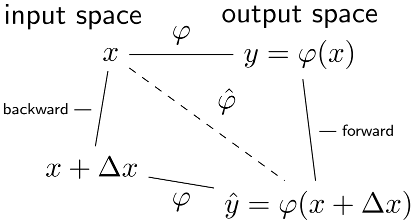
Residual-based *a posteriori* backward error analysis then proceeds as follows:

1. For the problem φ, use an engineered version of the problem to compute the value $\hat{y} = \hat{\varphi}(x)$.
2. Compute the residual.
3. Use the computed value of the residual to obtain an estimate of the backward error (i.e., reflect the residual back as a perturbation of the input data).

Residual-based *a posteriori* backward error analysis then proceeds as follows:

1. For the problem φ, use an engineered version of the problem to compute the value $\hat{y} = \hat{\varphi}(x)$.
2. Compute the residual.
3. Use the computed value of the residual to obtain an estimate of the backward error (i.e., reflect the residual back as a perturbation of the input data).
4. How satisfactory is the solution? Compare the backward error to the modelling error and uncertainty.
5. Finally, examine the conditioning (sensitivity) of the problem. If the problem is well-conditioned and the computed solution amounts to a small backward error, then conclude that your solution is satisfactory.

For **many kinds of problems**, there is a **disarmingly easy** way to find such a Δx based on the **(always computable) residual**.



Calculate $y = \varphi(x;p)$.

Rewrite this as:
$y - \varphi(x;p) = 0$.

Suppose $\hat{y}$ is an inexact solution (so, $\neq y$).

Then, $\hat{y} - \varphi(x;p) = r(x;p)$.
(approximate solution $\Rightarrow$ non-zero residual).

Equivalently: $\hat{y} = \varphi(x;p) + r(x;p) =_{df} \hat{\varphi}(x;p)$.
So: approximate solution to $\varphi \Rightarrow$ exact solution to a perturbed problem $\hat{\varphi}$.

Let's see how all this applies to initial value problems:

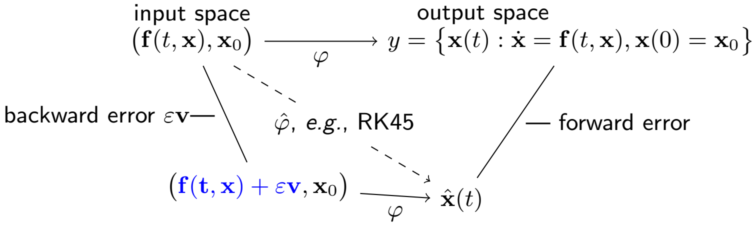


Figure: Commutative diagram for the backward error analysis of initial value problems. Note that we can also perturb $\mathbf{x}_0$, or both $\mathbf{x}_0$ and $\mathbf{f}$. In some cases, this diagram will be implicitly replaced by an "almost commutative diagram."

We have exactly solved this **modified problem** (which we call the **reversed-engineered problem**):

$$x' = f(t,x) + \varepsilon v(t)$$

For the practitioners, here's how simple it is in Matlab:

```
sol = ode45(@myodefun,tspan,x0,options);
mesh = linspace( ti, tf, numpoints );
[xhat,dotxhat] = deval( sol, mesh );
Residual = dotxhat - myodefun(xhat,mesh);
```

It's so easy, it almost feels like cheating!

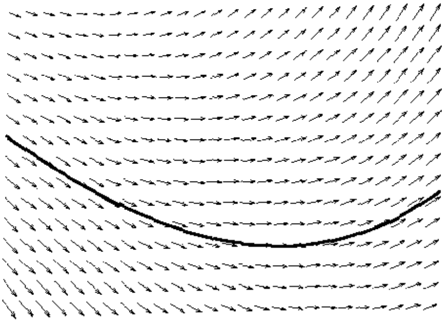
Note: Only possible with modern continuous methods, such as the continuous Runge-Kutta methods.

If we consider perturbations of the functional $\mathbf{f}$ from the p.o.v. of **dynamical systems**, the analysis allows us to find **to which perturbed vector field our computed solution is tangent**!

$$\Delta(t) = \dot{\hat{\mathbf{x}}}(t) - \mathbf{f}(t, \hat{\mathbf{x}}(t)) \quad \Rightarrow \quad \dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(t, \hat{\mathbf{x}}(t)) + \Delta(t)$$

If we consider perturbations of the functional $\mathbf{f}$ from the p.o.v. of **dynamical systems**, the analysis allows us to find **to which perturbed vector field our computed solution is tangent**!

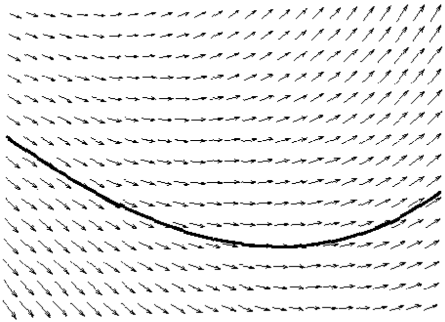
$$\Delta(t) = \dot{\hat{\mathbf{x}}}(t) - \mathbf{f}(t, \hat{\mathbf{x}}(t)) \quad \Rightarrow \quad \dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(t, \hat{\mathbf{x}}(t)) + \Delta(t)$$



Δt can be understood as asserting modelling assumptions!

If we consider perturbations of the functional $\mathbf{f}$ from the p.o.v. of **dynamical systems**, the analysis allows us to find **to which perturbed vector field our computed solution is tangent**!

$$\Delta(t) = \dot{\hat{\mathbf{x}}}(t) - \mathbf{f}(t, \hat{\mathbf{x}}(t)) \quad \Rightarrow \quad \dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(t, \hat{\mathbf{x}}(t)) + \Delta(t)$$



Δt can be understood as asserting modelling assumptions!

Assessing computational error is thereby **reduced to assessing modelling error** in a **completely metric-independent** way.

Then, the situation is this:

- If solving the problem $\hat{\varphi}(x)$ **amounts to having solved the problem $\varphi(x + \Delta x)$ for a Δx smaller than the perturbations inherent in the modeling context** (specifying estimates of error and uncertainty), then our solution $\hat{y}$ must be considered **completely satisfactory**.
- The algorithm found a solution **as good the modeling context deserves**.
- For all we known, the computed solution might be the exact description of the system modeled.

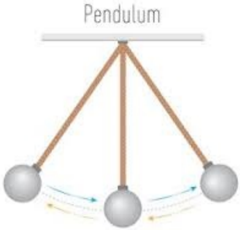
This BEA framework sheds an interesting light on the **dual nature of perturbations**.

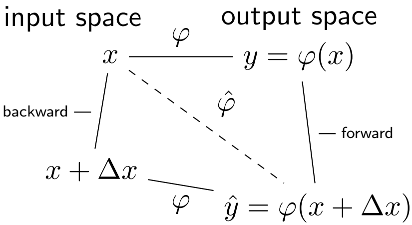
Both philosophers and scientists use the phrase “perturbation theory” with two distinct ideas in mind:

- **approximation** (how different approximate–or perturbative–solutions to a problem relate)
- physical **disturbance** (how solutions to different problems–or perturbed equations–relate to each other)

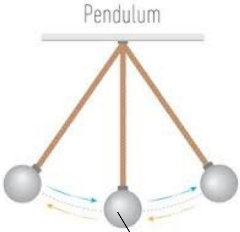
One of the insight of this approach is that **approximations and disturbances are the flip side of the same coin**. **Semantically speaking**, there's no difference between the two.

A cool connection shedding light on perturbations





A cool connection shedding light on perturbations



inexact equation of motion

exact equation of motion

integrate

force field, mass, IC — input space

output space

φ

x — $y = \varphi(x)$

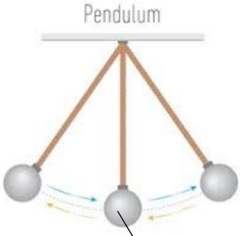
approx. integrate with $\theta \approx 0$ — $\hat{\varphi}$

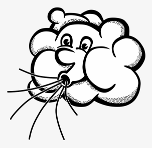
backward —

— forward

$x + \Delta x$ — φ — $\hat{y} = \varphi(x + \Delta x)$

A cool connection shedding light on perturbations





inexact equation of motion

exact equation of motion

integrate

force field, mass, IC

input space

output space

φ

x — $y = \varphi(x)$

approx. integrate with $\theta \approx 0$

$\hat{\varphi}$

backward —

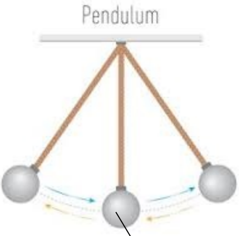
— forward

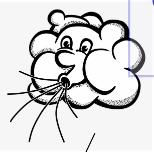
$x + \Delta x$

φ

$\hat{y} = \varphi(x + \Delta x)$

A cool connection shedding light on perturbations





exact eqs. of perturbed system

inexact equation of motion

exact equation of motion

integrate

force field, mass, IC — input space

output space

$x \xrightarrow{\varphi} y = \varphi(x)$

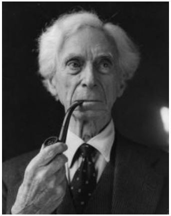
approx. integrate with $\theta \approx 0$ — $\hat{\varphi}$

backward — forward

perturbed force field, etc — $x + \Delta x \xrightarrow{\varphi} \hat{y} = \varphi(x + \Delta x)$

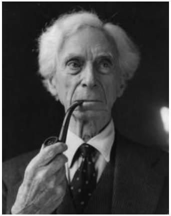
In conclusion, let's return to Russell's nice quote:

"Although this may seem a paradox, all exact science is dominated by the idea of approximation."



In conclusion, let's return to Russell's nice quote:

"Although this may seem a paradox, all exact science is dominated by the idea of approximation."



Accordingly, **the point of the epistemology of sciences is not to try to understand how science would be without errors and uncertainty, but rather the point is to understand how we can live with them**.

Thank you!

nfillion@sfu.ca