EngSci 721

Inverse Problems and Learning From Data

Oliver Maclaren (oliver.maclaren@auckland.ac.nz)

1. *Basic concepts* [5 lectures + 1 Tutorial]
   Forward vs inverse problems. Well-posed vs ill-posed problems. Algebra and calculus of inverse problems (left and right inverses, generalised and pseudo inverses, resolution operators, matrix calculus). Representing higher dimensional problems (image data etc).
2. *Instability and regularisation in linear and nonlinear problems* [6 lectures + 1 Tutorial]
   Instability and related issues for generalised inverses. Introduction to regularisation and trade-offs. Tikhonov regularisation. Higher-order Tikhonov regularisation. Sparsity and regularisation using different norms. Truncated singular value decomposition. Iterative regularisation, including stochastic/mini-batch gradient descent.
3. *Further topics* [3 lectures + 1 Tutorial]
   Regularisation parameter choice, including statistical and machine learning views of regularisation. Confidence sets for linear and nonlinear models. Physics-informed machine learning and neural networks.

Module overview

Inverse Problems and Learning From Data (*Oliver Maclaren*)
[~14 lectures/3 tutorials]

Lecture 1: Overview

Topics:

- Forward vs inverse problems and learning from data
- Examples
- Well-posed vs ill-posed problems
- Illustrations of ill-posedness

EngSci 721: Inverse Problems & 'Learning from data'

Lecture 1: Overview.

- Inverse problems, UQ, data etc
- Forward vs inverse problems
- Examples of inverse problems
- Well-posed & ill-posed problems
- Illustrations of ill-posedness

Prelude:

Inverse Problems, UQ, Data-centric engineering, scientific machine learning...

The imperative of physics-based modeling and inverse theory in computational science

To best learn from data about large-scale complex systems, physics-based models representing the laws of nature must be integrated into the learning process. Inverse theory provides a crucial perspective for addressing the challenges of ill-posedness, uncertainty, nonlinearity and under-sampling.

Karen E. Willcox, Omar Ghattas and Patrick Heimbach

The notions of 'artificial intelligence (AI) for science' and 'scientific machine learning' (SciML) are gaining widespread attention in the scientific community. These initiatives target development and adoption of AI approaches in scientific and engineering fields with the goal of accelerating research and development breakthroughs in energy, basic science, engineering, medicine and national security. For the past six decades, these fields have been advanced through the synergistic and principled use of theory, experiments and physics-based simulations. Our increased ability to sense and acquire data is clearly a game-changer in these endeavors. Yet, in our excitement to define a new generation of data-centric approaches, we must be careful not to chart our course based entirely on the successes of data science and machine learning in the vastly different domains of social media, online entertainment, online retail, image recognition, machine translation and natural language processing — domains for which data are plentiful and physics-based models do not exist. In contrast, many of today's scientific grand challenges suffer from the lack of adequate sampling of the processes underlying the complex, large-scale systems. Yet, for many of these systems, a great deal is known regarding the underlying physical principles or governing equations; we must continue to appeal to computational science to unleash this information. As Coveney et al. argue elegantly[1], big data need big theory — and big physics-based simulation models — too.

The unreasonable effectiveness of physics-based models

But what are physics-based models and why are they indispensable? A physics-based model is a representation of the governing laws of nature that innately embeds the concepts of time, space, causality and generalizability. These laws of nature define how physical, chemical, biological and geological processes evolve. Physics-based models typically encode knowledge in the form of conservation and constitutive laws, often based on decades if not centuries of theoretical development and experimental validation. These laws often manifest as systems of differential equations that are solved numerically with high-performance computing (HPC).

In his famous 1960 article, Eugene Wigner wrote about 'The unreasonable effectiveness of mathematics in the natural sciences'[2], pointing to "the 'laws of nature' being of almost fantastic accuracy but of strictly limited scope." As Wigner discusses, physics-based modeling is powerful and effective because it gives us a predictive window into the future based on understanding. It achieves this because any particular model limits its scope to a particular class of physical systems or processes, building a universal representation within that class. Armed with that universal representation, physics-based modeling is a way to simulate 'what if' scenarios and to issue predictions that have explanatory power or projections with quantified uncertainties that go beyond the current state and available data. For example, in our modern world, physics-based models are used to issue predictions about the future evolution of a cancer patient's tumor, or about the loads that a yet to be built aircraft may find itself experiencing under different operating conditions. They enable predictions of weather over the next five to ten days, or scenario-based projections about the future state of the Earth's climate in the decades to come.

The role of inverse theory in learning from data

As attention turns from simulation to learning from data (that is, from the forward problem to the inverse problem), we must bring these learned lessons — the big theory and the big physics-based simulation models — with us. Without physical constraints, purely data-driven approaches are unlikely to be predictive, no matter how expressive the underlying representation. Even when physical models are not well-established (such as for many biological processes, in constitutive laws for complex materials, or in subgrid scale models for unresolved physics), we know that certain universal properties and relationships must hold, such as conservation properties, material frame indifference, objectivity, symmetries, or other invariants. The learning-from-data problem is fundamentally an inverse problem that merges the partial knowledge reservoir of data with that of physics-based models in a systematic and rigorous way, and in a way that exploits the complementary and mutually reinforcing aspects of both data and models.

Data and models invariably come with uncertainties. Data are often noisy, sparsely and heterogeneously sampled, and representative of disparate observables. Experiments and data gathering are costly, time-consuming, and sometimes dangerous or impossible. Often data are hardest to acquire and are thus sparsest in the most decision-critical regions (for example, failure, instability, extreme environments). Even if it is possible to generate more data (for example, via simulation) a fundamental challenge remains: due to information loss in the forward problem and resulting ill-posedness of the inverse problem, data often contain only low-dimensional information about the physics, even when the data are large-scale[3].

In turn, physics-based models are typically characterized by uncertain parameters, which may include initial and boundary conditions, sources, material properties, geometry and model structure, all of which can be heterogeneous in space or time. In this setting, rather than ignore known physics, we must employ them to define the maps from parameters to observables, and invert them to project

2021

Statistics? (Frequentist? Bayesian? ...)

Inverse problems as statistics

Steven N Evans[1] and Philip B Stark[2]

[1] Department of Statistics and Department of Mathematics, University of California, Berkeley, CA 94720-3860, USA
[2] Department of Statistics, Space Sciences Laboratory, and Center for Theoretical Astrophysics, University of California, Berkeley, CA 94720-3860, USA

E-mail: evans@stat.berkeley.edu and stark@stat.berkeley.edu

Received 27 September 2001, in final form 26 February 2002
Published 20 May 2002
Online at stacks.iop.org/IP/18/R55

Abstract
What mathematicians, scientists, engineers and statisticians mean by 'inverse problem' differs. For a statistician, an inverse problem is an inference or estimation problem. The data are finite in number and contain errors, as they do in classical estimation or inference problems, and the unknown typically is infinite dimensional, as it is in nonparametric regression. The additional complication in an inverse problem is that the data are only indirectly related to the unknown. Canonical abstract formulations of statistical estimation problems subsume this complication by allowing probability distributions to be indexed in more-or-less arbitrary ways by parameters, which can be infinite dimensional. Standard statistical concepts, questions and considerations such as bias, variance, mean-squared error, identifiability, consistency, efficiency and various forms of optimality apply to inverse problems. This paper discusses inverse problems as statistical estimation and inference problems, and points to the literature for a variety of techniques and results. It shows how statistical measures of performance apply to techniques used in practical inverse problems, such as regularization, maximum penalized likelihood, Bayes estimation and the Backus–Gilbert method. The paper generalizes results of Backus and Gilbert characterizing parameters in inverse problems that can be estimated with finite bias. It also establishes general conditions under which parameters in inverse problems can be estimated consistently.

Motivation/Working principle:

→ The mathematics underlying 'classical' inverse problems sheds light on statistics, machine learning etc as well as vice-versa

} we'll look at both!

statistics : mappings from data to parameter
ML : mappings from data to data
} eg

| Inverse problems : mappings between general spaces |

Forward vs Inverse Problems

Typical modelling/simulation setup:

parameters →
initial conditions →
boundary conditions →
etc
(inputs)

→ Model/simulator → solution $y(x,t)$ etc.
(output)

'Forward' →

& Typical 'forward' models
- ODE
- PDE
- ABM (Agent-based model)
- etc
- (Many things!)

Problem: usually measure (noisy) |outputs| (data!) & want |inputs|

ie input → $\boxed{\text{Model}}$ → output

input ↑ want ← 'inverse' problem. ← output ↑ measured/known 'data'

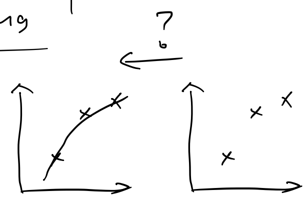
Examples

Many important problems can be considered 'inverse' problems

Obvious example:

parameter estimation, fitting, parameterisation, learning

eg 'fit' an ODE, PDE, polynomial, neural net etc to given data.



But the general concept is very broad & essentially about (math) 'inverting' given functions, or (physics) determining 'causes' from 'effects'

---

eg $y = f(x)$, say $y = x^2$

$x \longmapsto x^2$

observe $y = 4$

what was x?

Note: no unique solⁿ!

↳ solution set: $\{2, -2\}$

(note: $f^{-1}\{4\}$, not $f^{-1}(4)$!)

Examples (see readings for details)

Some common 'physical' examples include

- Tomography — CT (computerised), EIT (electrical impedance) } etc

  ↳ 'imaging by sections'

  → given slices, reconstruct object

- Deconvolution (deblurring)

  ↳ given a blurred photograph, reconstruct unblurred version

- Geophysics — determine the origin of an earthquake, given seismographs measuring arrival times

- Geothermal engineering — determine the density or permeability etc of underground rocks given surface or well measurements of temperature

Examples (see Aster et al L1 reading for details)

More physical/mathematical examples:

- Determine the origin of a groundwater contaminant after it has been transported (by advection/diffusion) downstream to measurement locations

- Given a function, determine its derivative!
  - HUH? See later →

- Given crime scene clues, determine the perpetrator...

→ many more!

What's the difference between 'forward' & 'inverse' though?

Keller (1976): two problems are inverse to each other if the formulation of one is naturally 'opposite' to the other

→ eg differentiation & integration
→ intuitive rather than fully-formal

However, often one problem is more naturally considered the direct or forward problem & the other more naturally the inverse problem

Example pairs :

Forward	Inverse
multiplication to find product	division to find factors
given present/past predict future	given present/future predict past
integration	differentiation

features

Forward:

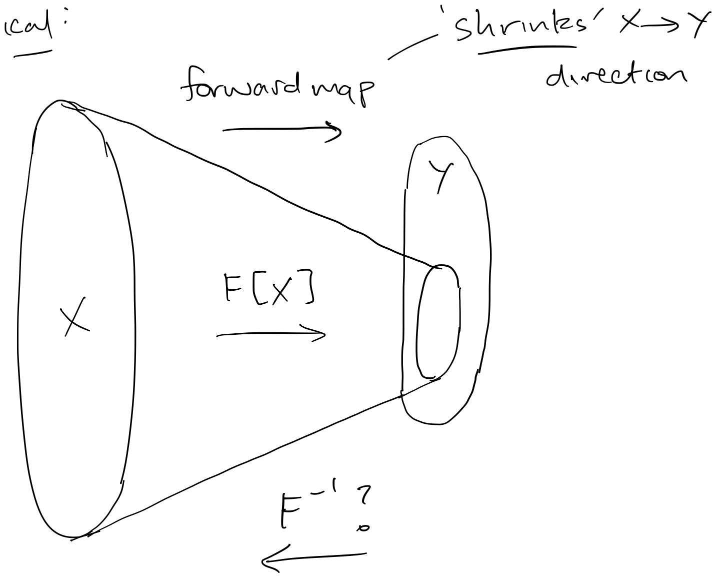
- 'easier'
- well-defined
- tend to 'smooth', 'combine' or 'reduce' inputs to simpler output
- respect causality/flow of time
- stable

Inverse:

- 'harder'
- can be ill-defined eg $\frac{1}{0}$
- often require recovering 'lost' info or 'uncombining' to get inputs from combined output
- amplify noise
- acausal/backwards in time?
- unstable ?

Mathematical Picture

Typical:

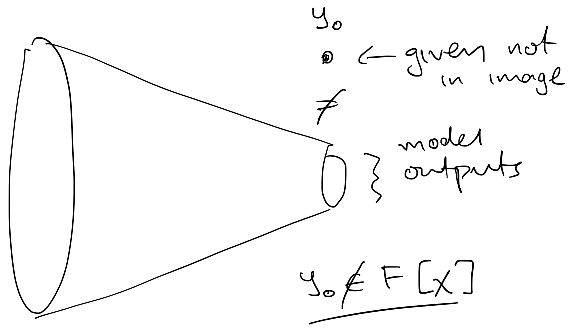


Think:

size {inputs/domain space} > size {outputs/range/image space}

---

also:



Well-posed vs ill-posed Problems

Hadamard (1902) call a problem well-posed if a solution:

- Exists (possible to satisfy)
- Is unique
- Is stable wrt 'small' changes in the givens (continuity)

Problems that are not well-posed are said to be

$\boxed{\text{ill-posed}}$

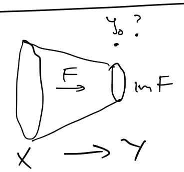
Key: forward problems → well-posed
inverse problems → ill-posed
(generally speaking)

Well-posed vs ill-posed Problems

Math: given model $F: X \to Y$ & data $y_0 \in Y$

we are essentially looking for the inverse F^{-1}: when does it exist? when unique? when continuous/stable

1. $\boxed{\text{Existence}}$: $y_0 \in \operatorname{Im} F$ (F is onto)



$\operatorname{Im} F$ = 'image of F'
$= F[X]$
$= \{F(x) \mid x \in X\}$

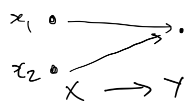
note: image of a set is a set.

2. $\boxed{\text{Uniqueness}}$: (F is 1-1)

$F(x_1) = F(x_2) \Rightarrow x_1 = x_2$

ie

$x_1 \neq x_2 \Rightarrow F(x_1) \neq F(x_2)$



No! Want: distinct → distinct.

3. $\boxed{\text{Stability}}$: F^{-1} (which exists if 1&2 sat.) is (sufficiently) continuous:

if $\underbrace{y_\delta \approx y}_{\text{small data perturb}}$ then $\underbrace{F^{-1}(y_\delta) \approx F^{-1}(y)}_{\text{small sol}^n\text{ perturb.}} = x$

Stability/continuity can be more formally defined in various ways.

- Sequential continuity: for any sequence $y_n \to y$ & $x = F^{-1}y$ we have $F^{-1}(y_n) \to F^{-1}(y) = x$

(F^{-1}(converging sequence of 'input' data) = converging sequence of 'output' sol^{n}s)

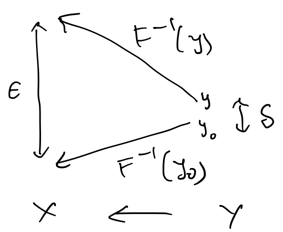
- ϵ-δ def^{n}s

$$(\forall \epsilon)(\exists \delta(\epsilon))\left[\,\| y - y_0 \| < \delta \Rightarrow \| F^{-1}(y) - F^{-1}(y_0) \| < \epsilon\,\right]$$

[Given a tolerance ϵ, find a δ for that tolerance, s.t. if 'inputs' of F^{-1} differ by $< \delta$ then 'outputs' of F^{-1} differ less than ϵ

practical/sufficient: can't take δ lower than some 'resolution'

control via δ →

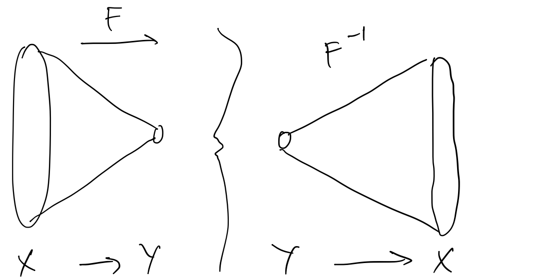


Well-posed vs Ill-posed Problems

Intuition:

- Forward 'shrinks' (eg 'compact operators')
- Inverse needs to 'expand'

→ risks instability



small change → large change in inverse.

Key Message:

Even when inverse exists in principle it may be unstable

(might be 1-1 but 'close to' non 1-1)

Illustrations : instability even when inverse exists

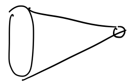
Integration : 'sums up' $\rightarrow$ reduces

eg $1+2+3 = 6$

$3+3 = 6$

$2+4 = 6$

&



differentiation : 'breaks down' $\rightarrow$ expands ?

$\hookrightarrow$ Numerical Example.

- Given a $\boxed{\text{signal 'vector'}}$ $\boxed{x \in \mathbb{R}^n}$,

- we can sum the first k elements by taking dot product with

$$w_k = (\underbrace{1 \; 1 \; 1 \; \cdots \; 1}_{k} \; 0 \; 0 \cdots 0)^T = (1_k^T \; 0 \; 0 \cdots 0)^T$$

$$1_k^T = (\underbrace{1, 1, \ldots 1}_{k})$$

(Note we will often think of 'samples'/'signals' as finite dimensional vectors $(x_1, x_2, \ldots)$ approximating infinite-dimensional random vars $X(\Omega), \Omega \in \Omega$, & operations like summation etc as finite versions of integration).

Illustrations

Note : (assume all vectors are column by default)

Eg $x = (1 \;\; 2 \;\; 3 \;\; 0 \;\; 0)^T$

$w_1 = (1 \;\; 0 \;\; 0 \;\; 0 \;\; 0)^T$

$w_2 = (1 \;\; 1 \;\; 0 \;\; 0 \;\; 0)^T$

$\vdots$

$$\text{cumsum}(k, x) = w_k^T x$$

$$\text{eg} \quad cs(2, x) = (1 \;\; 1 \;\; 0 \;\; 0 \;\; 0) \begin{pmatrix} 1 \\ 2 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

$$= 1 + 2 = 3$$

Return vector of all cumulative sums

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ \vdots & & & & \\ 1 & 1 & \cdots & \cdots & 1 \end{pmatrix} \leftarrow \text{forward map } X \rightarrow Y$$

$$y = Ax$$

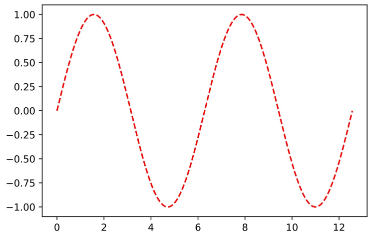
Illustrations

(Import numpy as np)

Python (see Canvas)

Input signal. $x(t)$

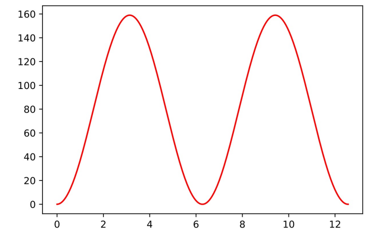
```
t = np.linspace(0,4*np.pi,1000)
x = np.sin(t)
plt.plot(t,x,'r--')
plt.show()
```



numpy's 'cumsum'

'Integral' $y = \int_0^t x(\xi)\,d\xi$

```
#built-in 'integration'
plt.plot(t,np.cumsum(x),'r')
plt.show()
```



Roll our own as matrix mapping

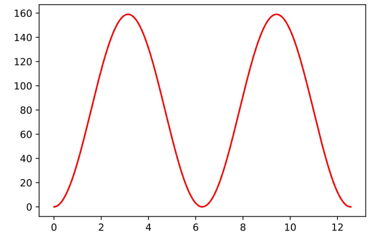
```
#forward mapping for integration (cumulative sum)
def create_fmap_int(d=1):
    A = np.zeros([d,d])
    for i in range(0,d):
        A[i,:i+1] = 1
    return A

#create forward mapping
A = create_fmap_int(len(x))
print(A)
```

```
[[ 1.  0.  0. ...,  0.  0.  0.]
 [ 1.  1.  0. ...,  0.  0.  0.]
 [ 1.  1.  1. ...,  0.  0.  0.]
 ...,
 [ 1.  1.  1. ...,  1.  0.  0.]
 [ 1.  1.  1. ...,  1.  1.  0.]
 [ 1.  1.  1. ...,  1.  1.  1.]]
```

Check A correct

indistinguishable output

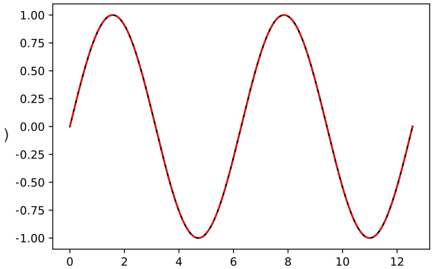
```
#calculate output of forward mapping
y = np.dot(A,x)

#compare our forward mapping to built-in
plt.plot(t,np.cumsum(x),'r--')
plt.plot(t,y,'r')
plt.show()
```



Invert using A^{-1} & compare to original output

indistinguishable input

```
#invert noise-free case
plt.plot(t,np.dot(np.linalg.inv(A),y),'k')
plt.plot(t,x,'r--')
plt.show()
```



Note: A^{-1} exists (1-1 & onto)

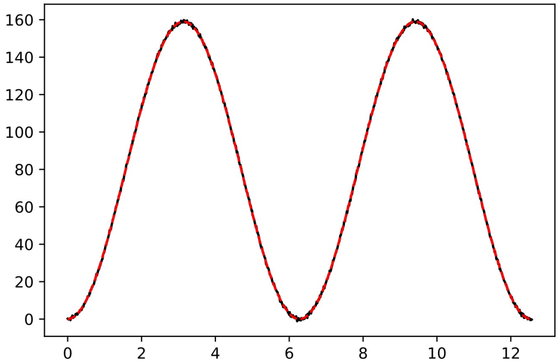
→ A is square
& all rows/cols are linearly independent

stability?

Add small amount of noise to output

ie $y_{observed} = Ax + \epsilon = y + \epsilon$

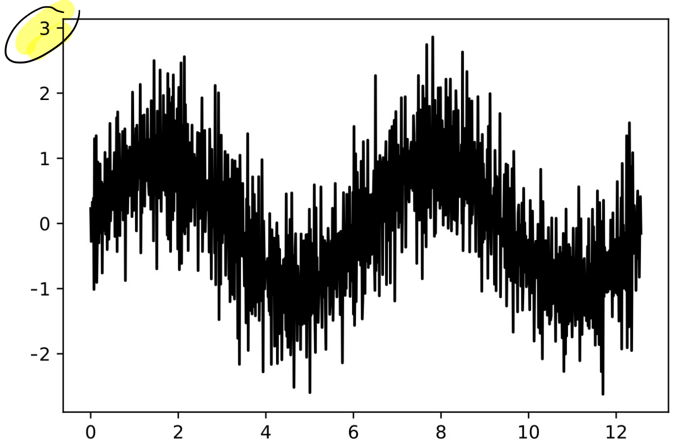
```
#add almost undetectable noise
y_noisy = y+np.random.normal(0,0.5,size=len(y))
plt.plot(t,y_noisy,'k')
plt.plot(t,y,'r--')
plt.show()
```



Question: is A^{-1} still 'good'?

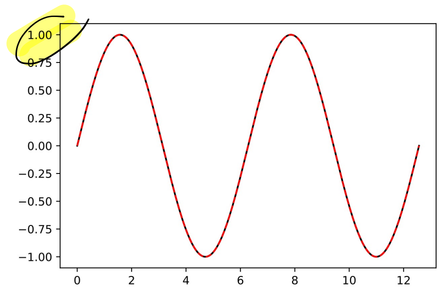
Instability of differentiation

```
#invert noisy
plt.plot(t,x,'r--')
plt.plot(t,np.dot(np.linalg.inv(A),y_noisy),'k')
plt.show()
```



Very 'irregular'! Large deviations

(Recall noise-free:



)

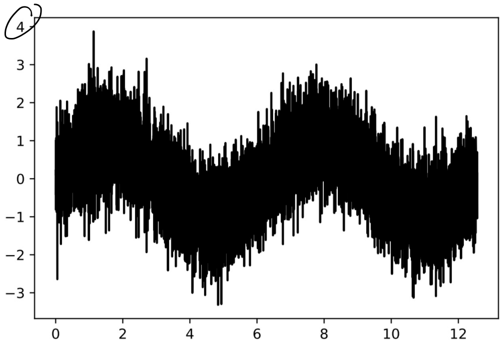
Can we just make grid finer?

→ Makes worse!

Finer grid: x10 finer } like finer discretis.

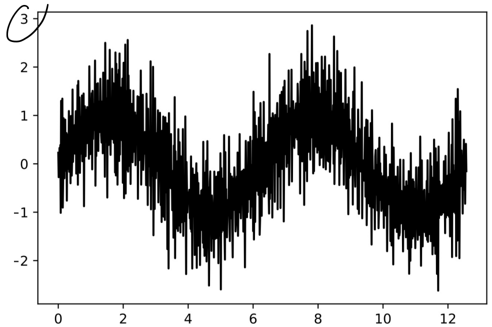
```
t = np.linspace(0,4*np.pi,10000)
x = np.sin(t)
plt.plot(t,x,'r--')
plt.show()
```

Same process gives:



noise ↑ even more!

compare:



Instability

- Integration 'smooths'
- Differentiation 'coarsens'

Mathematically, differentiation is an unbounded (discontinuous) operator [see functional analysis]

Numerically, differentiation is ill-conditioned

↳ not quite unbounded since discrete/finite, but practically unbounded.

discretise ↓ continuous : ill-posed
discrete : ill-conditioned

see [Engel et al L1 reading]

& →

Differentiation as discontinuous

Sequential continuity version.

consider sequence $y_n = \frac{1}{n}\sin(nx)$.

Now:

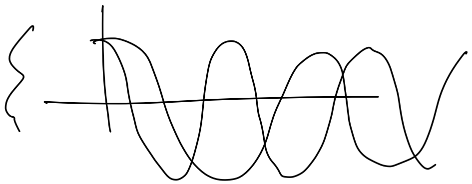
$n \to \infty$, $y_n \to 0$ (why?)

And clearly $\frac{d}{dx}(0) = 0$

but $\frac{d}{dx}\left(\frac{1}{n}\sin(nx)\right) = \cos nx$

$\not\to 0$

since amplitude fixed as $n \to \infty$



So not the case that for all y_n

$y_n \to y$ implies $F^{-1}(y_n) \to F^{-1}(y)$

(not 'A implies B' $\equiv$ A and not B)

Illustration : Local averaging (smoothing)

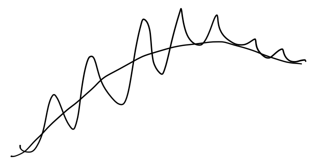
Given $\boxed{\text{'signal'}}$ or $\boxed{\text{'input'}}$ $\boxed{x \in \mathbb{R}^n}$

Local average over neighbourhood $w(i)$ of i is :

$$\frac{1}{|W|} \sum_{i \in W(i)} x(i)$$

$$\cdots\ x_{j-1}\ x_j\ x_{j+1}\ \cdots \qquad \underbrace{\longleftrightarrow}_{W(j)}$$

Smoothing : moving average



Inverse Problem:

- given a smoothed (or 'blurred') image, determine original
  - → eg photography

same idea as integration/differentiation

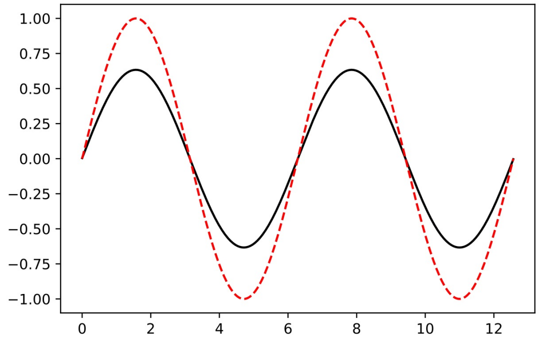
→ smoothing forward map

→ inverse amplifies any noise in output when trying to recover

$$y_0 = y + \epsilon = \underbrace{Ax}_{\text{smoothing}} + \underbrace{\epsilon}_{\text{noise}}$$

A^{-1} unstable!

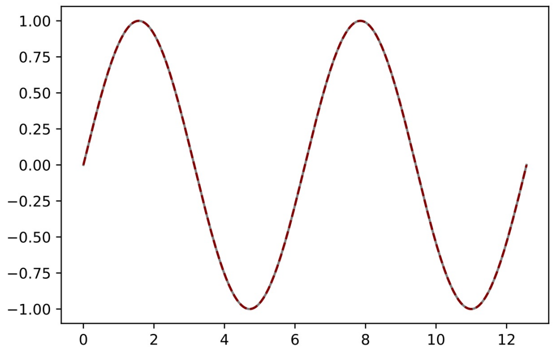
Smoothing map:



red: original
black: local av.

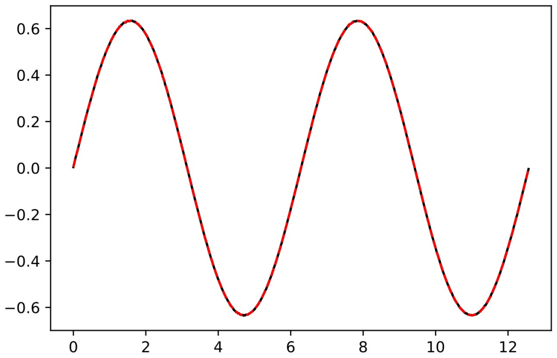
Can implement as linear forward map, ie matrix A (exercise?)

Invert noise free gives:



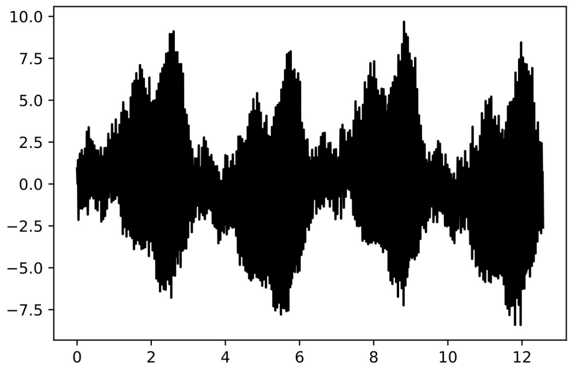
Instability of 'deblurring':

Add small noise:



practically indectable!

Invert noisy using A^{-1}



Terrible!

Should we give up?

No! (Well....)

We can stabilise the inversion procedure

"Regularisation"

→ A key theme of the course!

We will first look at the existence & uniqueness of 'inverses', however, then at their stability

We will also develop matrix calculus along the way

↳ key to machine learning too!