

AI Tools in Scientific Workflows

Ulrich Degenhardt¹

15/01/2026

¹Max-Planck-Institut for Dynamics and Self-Organization
Göttingen

Introduction

Github Repository

<https://github.com/udegenh1/genAI-Talk-January-2026>

- Notes
- Slides
- Prompts
- Transcripts of chats

Why talk about LLMs in science?

- LLMs are widely used in practice, but rarely discussed openly → Emerging taboo?
- The dominant narrative is: “LLMs are just hallucinating stochastic parrots”. Using LLMs is evil.
- But LLMs are here to stay. If we want it or not, there is no going back.
- Goal: Identify *high-value* uses and *high-risk* failure modes.

Note

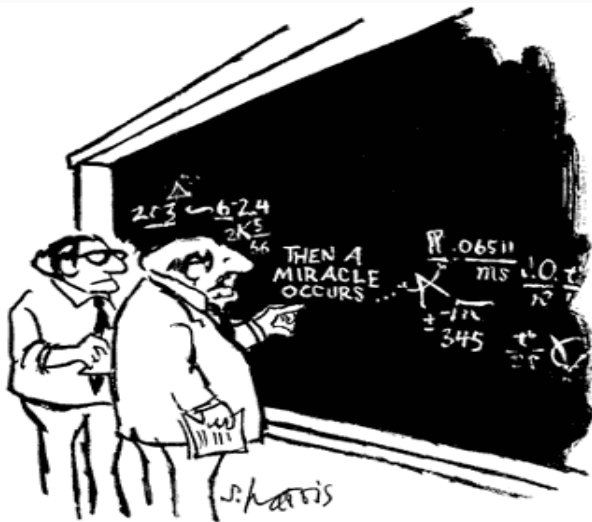
“Once a new technology rolls over you, if you’re not part of the steamroller you’re part of the road.”

Anonymous

Intro example 1: When keyword search fails

- Vague memory: a cartoon with two guys standing at a blackboard, mathematical equations and maybe the word “magic”.
- Google fails because there are no reliable keywords.
- Try this: Give an LLM a verbal description of your vague memories and let it look for an answer

Intro example 1: When keyword search fails



"I think you should be more explicit here in step two."

Intro example 2: Solving undergrad math exercises

- Provide a screenshot/PDF snippet of a math exercise and request a solution.
- The LLM produces a plausible proof-like response written in \LaTeX .
- Key point: LLMs seem to simulate understanding nontrivial math problems and are able to compose a plausible looking answer in \LaTeX . WTF? The mind boggles!

Intro example 3: Help writing slides for a talk

- Upload your notes to the LLM.
- Then ask the LLM to create slides based on the uploaded file.
- Ask the LLM to format the slides in \LaTeX for use with the Beamer class, then ask it to save the result as a file and to create a download link (works with ChatGPT).
- Use the result as scaffolding and revise the slides.
- Iterate until you are happy with the solution.

Note

"The best way to get the right answer on the internet is not to ask a question; it's to post the wrong answer."

Cunningham's law

Intro: Brandolini's law (It's becoming important now...)

"The amount of energy needed to refute bullshit is an order of magnitude bigger than that needed to produce it."

Alberto Brandolini

- This is known as **Brandolini's Law** aka the *Bullshit Asymmetry Principle* on the debunking of misinformation
- LLMs are talkative and generate lots of text. Reading and vetting LLM output may well exceed the cognitive resources of a human.
- LLMs generate plausible statements. In science, that is far more dangerous than obvious nonsense.

Intro: Preliminaries before using LLMs

- To be useful in science LLM sessions need documentation. Use a note taking system (e.g. Obsidian) as an “LLM lab book”.
- Export complete chats into your notes.
- Prompts and responses should be numbered.
- Change the overall behavior of the LLM with prompts: E. g. “No Yes-man mode”, “advocatus diaboli mode”.

Details can be found in the Github repository.

Trust

Why trust is so important to humans

- Trust simplifies things. And simplifying things has always been a major driving force for humans. So, humans want to trust
- Humans are evolutionarily tuned to detect and predict other humans, and to cooperate with them.
- LLMs pass the Turing-test and show behavior that resembles kindness, empathy or patience
- When we see this our brain defaults to "There must be a human behind this".
- And we are tempted to trust, but this produces an unwarranted sense of safety.

LLMs have no intentions

- LLMs have no consciousness, no intentions and no inner emotional life.
- What appears as kindness, empathy, or patience is the result of statistical pattern generation. These behaviors are simulations, not manifestations of genuine affects.
- LLMs do not want to help, nor can they decide to deceive. Outputs are just statistically plausible continuations of text.
- An interaction with an LLM is not a social encounter.

Tool-based trust vs social trust

- The appropriate category of trust is not social trust but tool-based trust.
- Some tools in scientific and engineering practice are "battle tested" and that's why we trust them.
- Examples include: the Unix stream editor `sed`, Linux and LAPACK.

Software Development

Making expensive things cheap

Some things may require a lot of effort:

- Analyzing error messages
- Looking for answers in bulky manuals
- Analyzing GitHub repositories

LLMs are surprisingly helpful and significantly lower the effort needed to tackle these tasks.

Prototyping: Turning ideas into computational plans

- Ask for a minimal working example for a concrete research task.
- Get an instant draft in Python/Julia/C++/Fortran/...
- You still verify, but the barrier to prototyping collapses.

This is about leverage. It is like having a bright grad student working for you. You certainly have to check his (or her) work, but he is enthusiastic, works 24/7 and never complains.

Case study: Iterative development with an LLM

A more dangerous way of LLM assisted software development:

- Build a Go tool for encrypting a directory tree (example *without* security measures).
- Add features via LLM prompts.
- Iterate by copy-pasting compiler errors back to the LLM.
- Result: a working executable after a few days of tinkering.

Warning: Don't do this to create production code!

Details can be found in the GitHub repository.

What this teaches (and why it is dangerous)

Lessons learned:

1. Feasible even without prior language knowledge.
2. Having no guardrails is scary (Again: Don't do this for real!).
3. Strong temptation to stop checking changes over time.

Agents and Automation

What we have now:

- A mixture of genuine AI assistance and manual steps
- A clumsy workflow
- A strong incentive to further automate the process

And here is why this is so scary:

- Agent-like tools can execute code, issue system commands and fix errors on their own. Human intervention is no longer necessary.
- Risks explode. Negative consequences range from subtle logical errors to serious system-level problems.
- This is so new that we don't even know what we don't know about the risks.

Safeguards become essential, not optional. Some of them are:

- Sandboxed execution, limited permissions.
- Clear audit trails and logging of sessions.
- Mandatory human checkpoints.

Pragmatic workflow

Despite these risks, the idea is extremely appealing. Here is a pragmatic workflow that emerged over the last year or so:

- General-purpose LLMs like ChatGPT are used to design the computational plan,
- Agentic tools such as Codex CLI handle the concrete tasks of writing and debugging code.
- Before making changes to the code the current state of the files in a directory is saved using a version control program such as Git or Fossil.
- Establish tests and make sure that all tests are being passed before making new changes.

Literature Search

The real bottleneck: search, not access

Search for literature and search in literature are common tasks:

- Access to papers is often solvable (subscriptions, preprints, OA).
- Cognitive resources are limited.
- Use specialized LLMs that are able to search huge curated collections of literature with minimal effort.
- Search with the help of an LLM in a collection of documents.
- Even the best LLMs have limitations and we have to understand what they are.

Question: What is the path signature?

Answer:



The **path signature** is a mathematical concept used to describe a path through a collection of iterated integrals. It provides a universal and characteristic feature map for sequential data, capturing the essential information of the path in a graded sequence of statistics [Chevyrev, 2016](#).

Path signatures are particularly useful in rough path theory, where they help describe the interactions between highly oscillatory and non-linear systems [Lyons, 2014](#). They transform a path into a group-like element in the tensor algebra, summarizing the path's information in a way that is useful for various applications, including machine learning and differential equations [Lyons, 2011](#).

Source: www.scienceos.ai

Chevyrev, 2016 ×



Source



Network



Citations

A Primer on the Signature Method in Machine Learning

2016, I. Chevyrev, A. Kormilitzin

REVIEW

ArXiv



Semantic Scholar



PDF

TLDR

An introduction to the signature method is provided, focusing on its basic theoretical properties and recent numerical applications, and current progress in applications of signatures to machine learning problems is reviewed.

ABSTRACT

In these notes, we wish to provide an introduction to the signature method, focusing on



184



20



Add to library

A Primer on the Signature Method in Machine Learning

Ilya Chevyrev^a and Andrey Kormilitzin^{a,b}

^aMathematical Institute, University of Oxford, Andrew Wiles Building, Woodstock Road, Oxford, OX2 6CG, UK

^bOxford-Man Institute, University of Oxford, Eagle House, Walton Well Road, Oxford, OX2 6ED, UK

E-mail: ilya.chevyrev@maths.ox.ac.uk,

andrey.kormilitzin@maths.ox.ac.uk

ABSTRACT: In these notes, we wish to provide an introduction to the signature method, focusing on its basic theoretical properties and recent numerical applications.

The notes are split into two parts. The first part focuses on the definition and fundamental properties of the signature of a path, or the *path signature*. We have aimed for a minimalistic approach, assuming only familiarity with classical real analysis and integration theory, and supplementing theory with straightforward examples. We have chosen to focus in detail on the principle properties of the signature which we believe are fundamental to understanding its role in applications. We also present an informal discussion on some of its deeper properties and briefly mention the role of the signature in rough paths theory, which we hope could serve as a light introduction to rough paths for the interested reader.

- Upload one or more PDFs and form a private collection of related documents (currently up to 4000 documents).
- Ask questions about the content of a single document or of the documents in a collection.
- Helpful feature: answers are linked to exact locations in source PDFs.

Example: A long review paper

Question: Which of the applications uses cross validation?

Answer:

Cross-validation has been used in the following applications:

1. **Diagnosing Alzheimer's Disease:** 10-fold cross-validation was used for training to select input features via Lasso regularization for binary logistic regression tasks Lyons, p59 .
2. **Early Sepsis Detection:** Stratified five-fold cross-validation was employed for hyperparameter optimization Lyons, p61 .

The PartialSOFA and SOFA-Deterioration variables are included since deterioration of the SOFA score is a requirement of the Sepsis-3 definition. The size of the look-back windows chosen for the Counter and Max/Min variables are treated as hyperparameters and optimised during training.

Once the data has been augmented to include these hand-crafted features, the signature transform is applied to the resulting time series. A sliding window approach is used; signature features are computed for each time point over a window of pre-determined look-back size. The input paths were augmented to include a time dimension, and the cumulative sum followed by the lead-lag transformation were applied as further augmentations prior to truncated signatures being computed. Both the truncation level and the look-back window length are treated as hyperparameters to be optimised during training.

The challenge data was prelabelled with the value 1 at any location of sepsis occurrence or predefined window around sepsis onset and zero otherwise. The method proposed in [HKLMNS19] creates an alternative labelling that accounts for information about the utility score to enable the classifier to place greater importance on points that lead to a larger score if predicted correctly. If $U_y(x, t)$ denotes the utility score of predicting y for patient x at time t , then the *modified utility score* (MUS) is defined as $U_M(x, t) := U_1(x, t) - U_0(x, t)$. It is against this labelling that the regressor is trained.

Stratified five-fold cross-validation, with a uniform distribution of time points and sepsis labels in each fold, is used for hyperparameter optimisation. Precise details may be found in [HKLMNS20]. The final values of the parameters can be found in Table 2 in [HKLMNS19]. They may be summarised as follows. The Counter variables are computed over a look-back window of size 8, whilst the Max/Min variables are computed over a look-back window of size 6. The streams given by PartialSOFA, MAP and BUN/CR are augmented with a time dimension and a lead-lag transformation, before signatures truncated to depth 3 were computed using a look-back window of size 7. For the remaining non-stationary streams, the cumulative sum augmentation is applied followed by the lead-lag transformation, before signatures truncated to depth 3 are computed using a look-back window of size 7.

Creating summaries is easy — remembering is hard

- LLM summaries are easy to generate and store.
- Storage in a note system does not imply long-term learning.
- Cognitive psychology: encoding, storage, retrieval.

A practical workflow to mitigate the problem (a little bit...)

- Store the LLM summary in your notes.
- Add your own 3-sentence meta-summary:
 - What is this about?
 - What is the core idea?
 - Why is it important / what surprised me?
- Add a concept map as an additional retrieval route.

The Expertise Paradox

- Beginners benefit: orientation, terminology, scaffolding.
- Experts suffer: oversimplifications, missing assumptions, edge cases.
- “Almost correct” outputs can be the most dangerous.

RAG helps — but does not provide real expertise

- Resource Augmented Generation (RAG) reduces hallucinations and improves traceability.
- But models tend to summarize rather than critically evaluate sources.
- Uncertainty and methodological weaknesses are often omitted or glossed over unless prompted.

Discussion

Cognitive Offloading: What Changed

- Predating LLMs science has always offloaded cognition to tools: calculators, numerical methods, symbolic systems.
- Outputs of pre-LLM tools are constrained by explicit algorithms and models. Outputs of LLMs are plausible outputs.
- The novelty is not complexity, but exploitation of our tendency to trust.

Consequences

- LLMs generate the most probable next statement, not the correct one.
- Fluency and coherence mask the absence of semantic grounding.
- Errors look often plausible, local, and are therefore hard to detect.

Implications for Scientific Use

- Risk depends on error severity and detectability.
- Verification costs must be weighed against cognitive savings.
- LLM outputs must be treated as hypotheses, not results.

*An LLM can produce results
without knowing why they are right
and without noticing when they are wrong.*

*If this were acceptable in physics,
derivations would be optional.*

Questions?