Machine learning algorithms for the conservative-to-primitive conversion in relativistic hydrodynamics

Thibeau Wouters

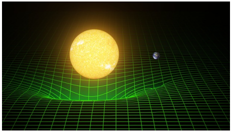
June 30, 2023

Table of Contents

Table of Contents

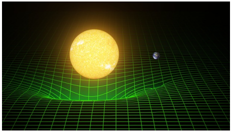
Gravitational waves

- General relativity is the most accurate theory of gravity to date.
- “Space-time tells matter how to move, matter tells space-time how to curve.”

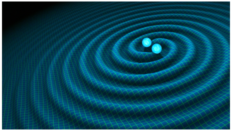


Gravitational waves

- General relativity is the most accurate theory of gravity to date.
- "Space-time tells matter how to move, matter tells space-time how to curve."

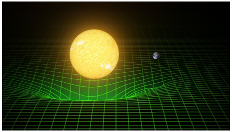


- In 2015, we observed the first gravitational waves (GW) from a binary black hole system.
- Yesterday, discovered stochastic background.

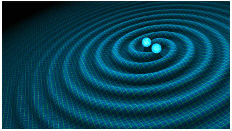


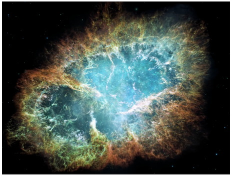
Gravitational waves

- General relativity is the most accurate theory of gravity to date.
- “Space-time tells matter how to move, matter tells space-time how to curve.”



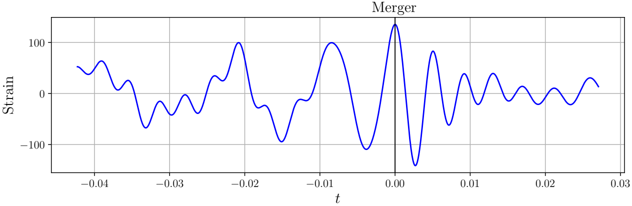
- In 2015, we observed the first gravitational waves (GW) from a binary black hole system.
- Yesterday, discovered stochastic background.
- Future? Insights into nuclear physics from neutron stars and supernovae!





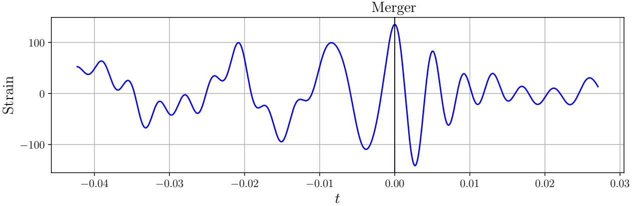
How do we detect GWs?

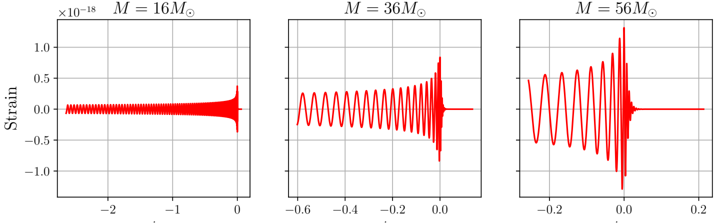
Gravitational wave signals are detected using **template matching**. **Numerical relativity simulations** provide large template banks for matching.



How do we detect GWs?

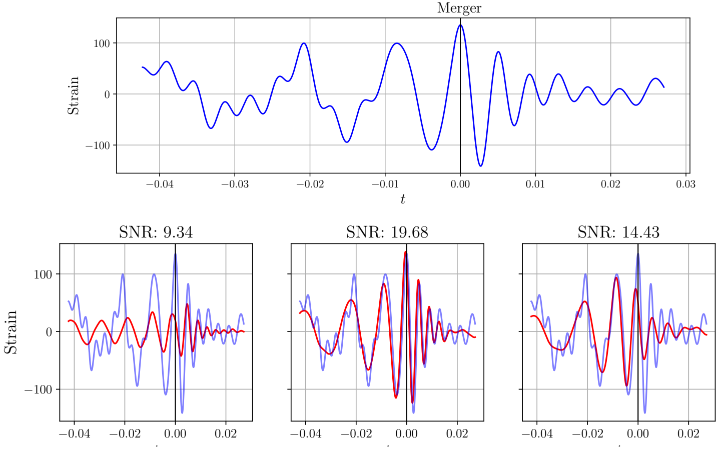
Gravitational wave signals are detected using **template matching**. **Numerical relativity simulations** provide large template banks for matching.





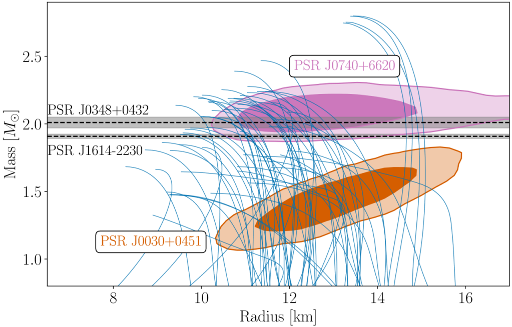
How do we detect GWs?

Gravitational wave signals are detected using **template matching**. **Numerical relativity simulations** provide large template banks for matching.



Every blue curve:

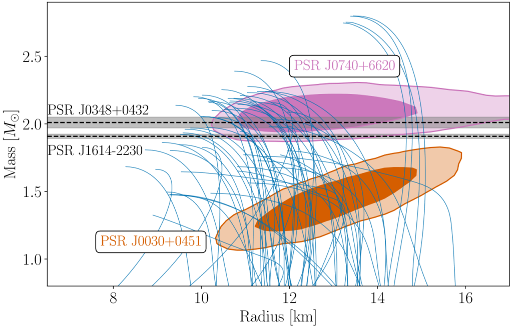
- is a proposed nuclear physics theory,
- influences the GW of a supernova/neutron star,
- has to be simulated!



Supernovae simulations

Every blue curve:

- is a proposed nuclear physics theory,
- influences the GW of a supernova/neutron star,
- has to be simulated!



Goal of the thesis

Leverage machine learning to speed up simulations.

The C2P bottleneck

Simulations need to keep track of two sets of variables:

- conserved variables $\mathcal{C}$: fluid dynamics; numerically evolved
- primitive variables $\mathcal{P}$: GW, source fluxes; *not* evolved, computed from $\mathcal{C}$ variables.

The C2P bottleneck

Simulations need to keep track of two sets of variables:

- conserved variables $\mathcal{C}$: fluid dynamics; numerically evolved
- primitive variables $\mathcal{P}$: GW, source fluxes; *not* evolved, computed from $\mathcal{C}$ variables.

Going from $\mathcal{C}$ to $\mathcal{P}$ (C2P) is a major **bottleneck** [1, 2]:

- No closed form $\rightarrow$ root-finding techniques
- 10^9 calls per ms
- $\sim 40\%$ of total simulation cost
- Load 300MB of external data

The C2P bottleneck

Simulations need to keep track of two sets of variables:

- conserved variables $\mathcal{C}$: fluid dynamics; numerically evolved
- primitive variables $\mathcal{P}$: GW, source fluxes; *not* evolved, computed from $\mathcal{C}$ variables.

Going from $\mathcal{C}$ to $\mathcal{P}$ (C2P) is a major **bottleneck** [1, 2]:

- No closed form $\rightarrow$ root-finding techniques
- 10^9 calls per ms
- $\sim 40\%$ of total simulation cost
- Load 300MB of external data

Goal of the thesis (specified)

Optimize the C2P conversion with machine learning.

Table of Contents

Evaluation criteria & methods

Wishlist for numerical methods for simulations [2]:

1. Speed: ideally, reduce cost of methods
2. Accuracy: predictions have to be accurate
3. Robustness: make sure simulations converge to true solution

Evaluation criteria & methods

Wishlist for numerical methods for simulations [2]:

1. Speed: ideally, reduce cost of methods
2. Accuracy: predictions have to be accurate
3. Robustness: make sure simulations converge to true solution

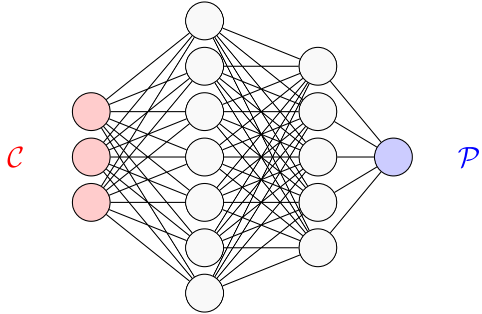
- Simulations use Fortran
- Use neural networks for flexibility

```fortran
subroutine nn_compute(x, y, neuralnet)
  implicit none

  double precision, intent(in)      :: x(INPUT_SIZE)
  double precision, intent(out)     :: y
  type(neural_network), intent(in)  :: neuralnet

  double precision :: xx(HIDDEN_SIZE_1)
  double precision :: yy(HIDDEN_SIZE_1)
  double precision :: xxx(HIDDEN_SIZE_2)
  double precision :: yyy(HIDDEN_SIZE_2)
  double precision :: y(OUTPUT_SIZE)

  xx    = matmul(neuralnet%weight0, x)   + neuralnet%bias0(:,1)
  call sigmoid(xx, yy)
  xxx   = matmul(neuralnet%weight2, yy)  + neuralnet%bias2(:,1)
  call sigmoid(xxx, yyy)
  y_vec = matmul(neuralnet%weight4, yyy) + neuralnet%bias4(:,1)
  y     = y_vec(1)
end subroutine nn_compute
```

Table of Contents

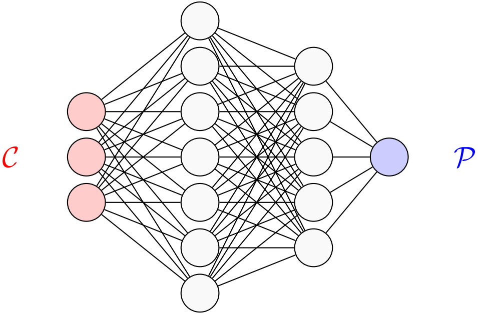
Neural network for C2P

1st (naïve) idea: Approximate $f : \mathcal{C} \to \mathcal{P}$ with a neural network.



Neural network for C2P

1st (naïve) idea: Approximate $f : \mathcal{C} \to \mathcal{P}$ with a neural network.



- Data generated with the *analytic* $f^{-1} : \mathcal{P} \to \mathcal{C}$
- MLP with 504, 127 hidden neurons; sigmoid activation functions
- Trained with Adam & adaptable learning rate

Results of naïve approach

1. **Speed?**

Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

2. **Accuracy?**

Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

2. ~~Accuracy?~~
   Squared difference: $\sim 10^{-3}$, vs. $\sim 10^{-8}$ for existing methods

Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

2. ~~Accuracy?~~
   Squared difference: $\sim 10^{-3}$, vs. $\sim 10^{-8}$ for existing methods

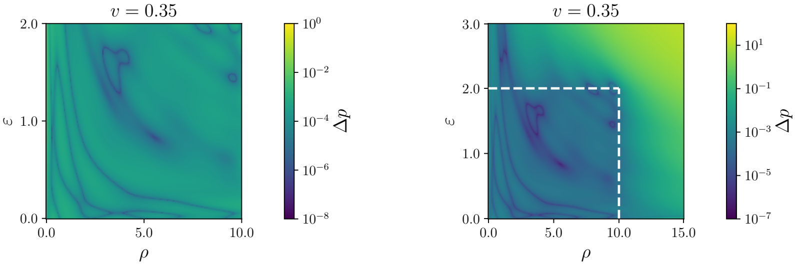
3. **Robustness?**

Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

2. ~~Accuracy?~~
   Squared difference: $\sim 10^{-3}$, vs. $\sim 10^{-8}$ for existing methods

3. ~~Robustness?~~
   Not guaranteed by MLP (*e.g.*, performance outside training domain).



Results of naïve approach

1. ~~Speed?~~
   $\sim 5\times$ slower than existing methods

2. ~~Accuracy?~~
   Squared difference: $\sim 10^{-3}$, vs. $\sim 10^{-8}$ for existing methods

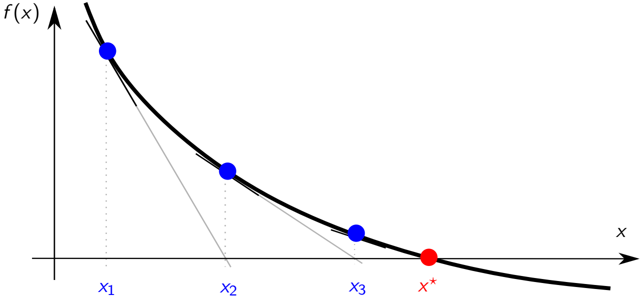
3. ~~Robustness?~~
   Not guaranteed by MLP (*e.g.*, performance outside training domain).

- Overall, worse performance!
- Robustness most crucial: difficult for ML models, built into existing methods
- *Can we speed up the existing methods?*

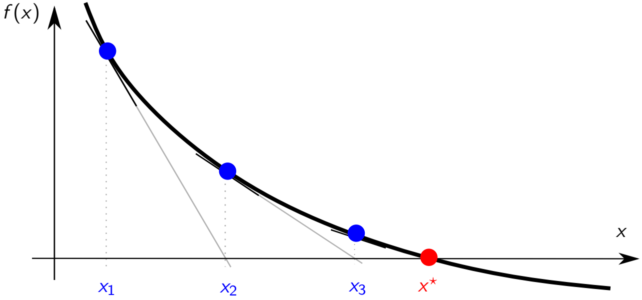
Table of Contents

Existing methods: root-finding algorithms

Current methods use root-finding algorithms: find root $x^\star$ of master function f by iteratively improving estimates x_i (*e.g.*, Newton-Raphson).



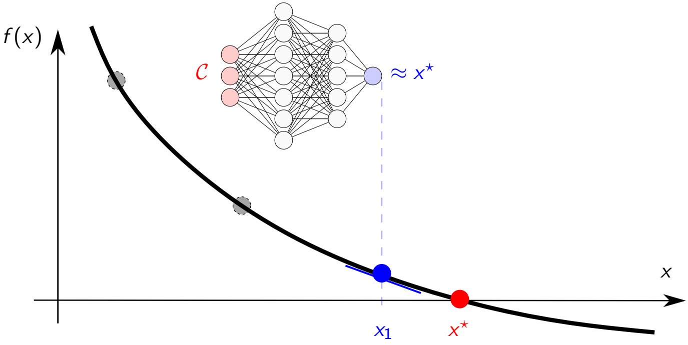
Current methods use root-finding algorithms: find root $x^\star$ of master function f by iteratively improving estimates x_i (*e.g.*, Newton-Raphson).



- **Slow**: evaluating $f(x)$ is costly.
- **Accurate**: accuracy tolerance as stopping criterion
- **Robust**: well-designed master function (Kastaun *et al.* [3])

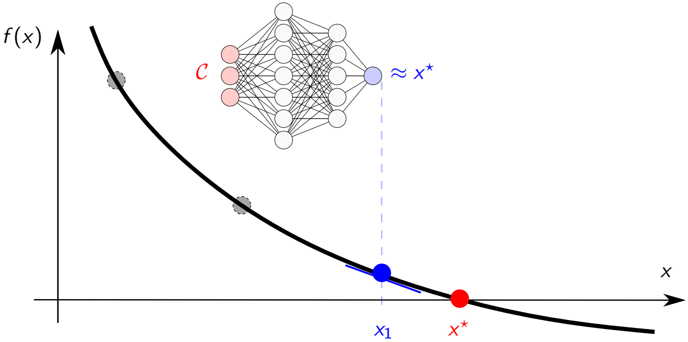
Hybrid approach

2nd idea: Neural network gives an initial guess, to be refined with root-finding algorithm.



Hybrid approach

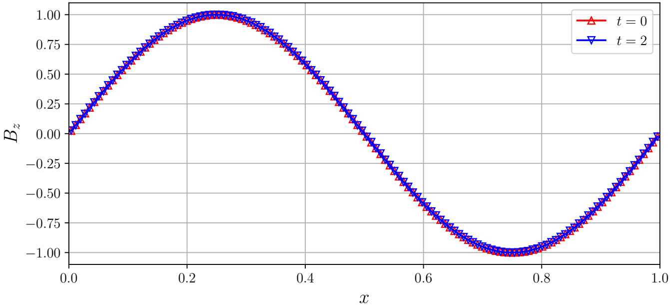
2nd idea: Neural network gives an initial guess, to be refined with root-finding algorithm.



- **Faster?** Amount of iterations, size & accuracy network,...
- **Accurate**: as accurate as existing method
- **Robust**: as robust as existing method

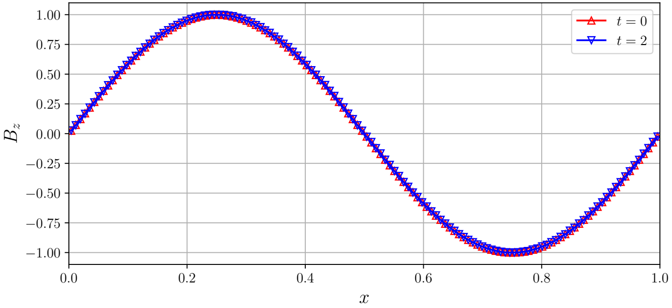
Hybrid approach: proof of concept

Test case: magnetic field B_z of Alfvén wave:



Hybrid approach: proof of concept

Test case: magnetic field B_z of Alfvén wave:



- MLP with 2 hidden layers, each 20 hidden neurons
- Sigmoid or ReLU activation functions
- Training data: sampled directly from the simulation

Hybrid approach: proof of concept

Faster! Compare time-to-completion (TTC):

- Standard: (23.48 ± 0.54) seconds
- Hybrid, ReLU activation function: (18.84 ± 0.19) seconds
- Speed-up of $\sim 25\%$!

Hybrid approach: proof of concept

Faster! Compare time-to-completion (TTC):

- Standard: (23.48 ± 0.54) seconds
- Hybrid, ReLU activation function: (18.84 ± 0.19) seconds
- Speed-up of $\sim 25\%$!

What about larger networks? (w: influences accuracy)

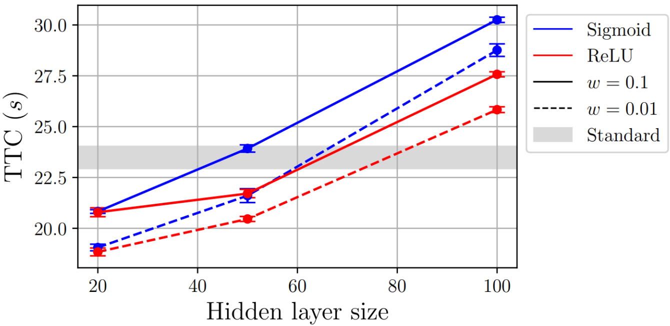


Table of Contents

Future work

- Extend to realistic simulations with nuclear physics

Future work

- Extend to realistic simulations with nuclear physics
- How to switch between nuclear physics theory? Recall the plethora of *proposed theories*: 1 curve = 1 dataset

Future work

- Extend to realistic simulations with nuclear physics
- How to switch between nuclear physics theory? Recall the plethora of *proposed theories*: 1 curve = 1 dataset
- Enable online training of neural networks

Conclusion

- Gravitational wave astrophysics needs templates obtained with accurate simulations. The C2P is a major bottleneck to be tackled
- Existing methods using root-finding algorithms are guaranteed to be accurate and robust
- Machine learning models, such as neural networks, are not guaranteed to be robust, which is a major drawback for simulations
- Hybrid approaches, combining machine learning with existing root-finding methods, can potentially speed up simulations

References

[1] Tobias Dieselhorst et al. “Machine learning for conservative-to-primitive in relativistic hydrodynamics”. In: *Symmetry* 13.11 (2021), p. 2157.

[2] Daniel M Siegel et al. “Recovery schemes for primitive variables in general-relativistic magnetohydrodynamics”. In: *The Astrophysical Journal* 859.1 (2018), p. 71.

[3] Wolfgang Kastaun, Jay Vijay Kalinani, and Riccardo Ciolfi. “Robust recovery of primitive variables in relativistic ideal magnetohydrodynamics”. In: *Physical Review D* 103.2 (2021), p. 023018.