Learning Kernels for Real-Time Complex Langevin

Enno Carstensen

6. May 2025

FWF Austrian Science Fund

UNI GRAZ

1 Real time evolution in QFT

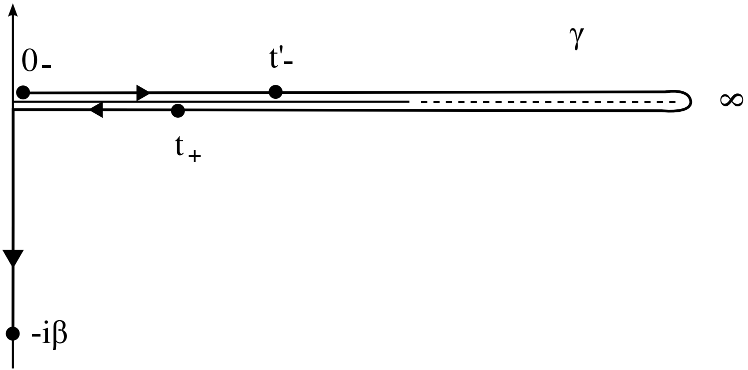
- A simple test case

- Machine learning kernels

What is real time evolution useful for?

- Calculating time-separated correlators
- Useful for both equilibrium and non-equilibrium systems:
  - Phase transitions
  - Baryogenesis
  - Gravitaional wave production
  - Heavy-Ion collisons

The Schwinger-Keldysh formalism in equilibrium



Approaches to evaluating the contour

- Perturbative expansion
- Naive lattice approaches lead to strong sign problem
- Mitigations: combine Lattice with:
  - Classical-Statistical
  - Schwinger-Dyson
  - Contour deformation
  - **Complex Langevin**

The complex Langevin equation

- Stochastic differential equation
  - Action S
  - Degrees of freedom ϕ
  - Gaussian noise η with $\mathrm{Var}(\eta) = 2$
  - Langevin time τ

$$\frac{\partial \phi}{\partial \tau} = -\frac{\delta S}{\delta \phi} + \eta$$

$$\phi_{\tau+\epsilon} = \phi_\tau - \epsilon \frac{\delta S}{\delta \phi}\bigg|_\tau + \sqrt{\epsilon}\eta$$

- Real time evolution in QFT

2 A simple test case

- Machine learning kernels

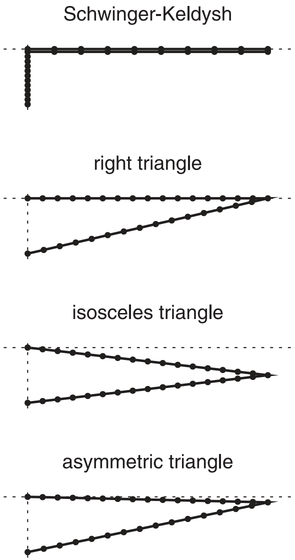
0+1-dimensional scalar field theory

$$\mathcal{L} = \left(\partial_t \phi(t)\right)^2 + m^2 \phi^2(t) + \frac{\lambda}{4!}\phi^4(t)$$

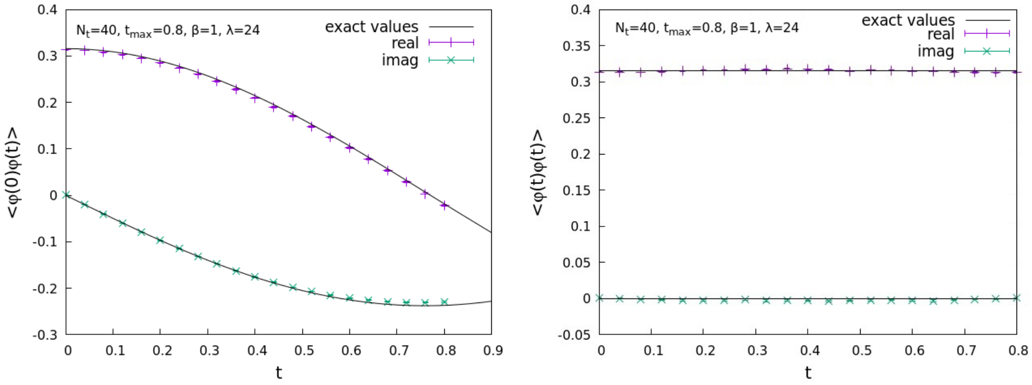
- Conceptionally and computationally simple
- Also known as: anharmonic oscillator
- Analytically solvable by diagonalizing the Hamiltonian
- Observables:
    - Unequal-time correlator: oscillations
    - Equal-time correlator: constant
- 2 parameters: mass $m = 1$ and coupling $\lambda = 24 \rightarrow$ strongly coupled

Lattice setup

- asymmetric triangle contour
- skew $= 0.1\%$
- $N_t = 40$
- adaptive step size ϵ
- maximum $\epsilon = 10^{-5}$

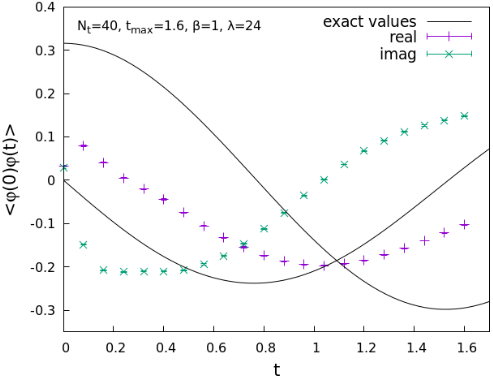


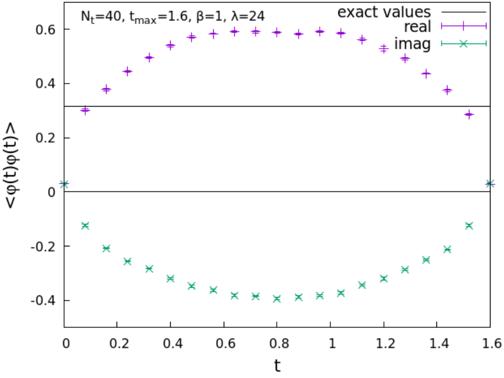
The anharmonic oscillator in action



Lampl and Sexty, 2023, arXiv:2309.06103

Wrong convergence for larger t_{max}





Lampl and Sexty, 2023, arXiv:2309.06103

Real time evolution in QFT

A simple test case

3 Machine learning kernels

What is a kernel?

- Any arbitrary holomorphic function $K(\phi, \tau, ...)$ with $K = H^T H$
- Does not change result (in real Langevin)
- Can change convergence behavior
- Optimal kernel can fix wrong convergence (in complex case)
- Constant kernel $\implies \frac{\delta K}{\delta \Phi} = 0$

$$\frac{\partial \Phi}{\partial \tau} = -K \frac{\delta S}{\delta \Phi} + \sqrt{K}\eta + \frac{\delta K}{\delta \Phi}$$

$$\Delta \phi_i = -\epsilon (H^T)_i^j H_j^k \frac{\partial S}{\partial \phi^k} + \sqrt{\epsilon} H_i^j \eta_j$$

Learning optimized kernels with gradient descent

- Introduced in Alvestad et al., 2022, arXiv:2211.15625
- Optimize kernel according to a loss function L

$$\Delta K_{ij} = -r \cdot \frac{\partial L(\Phi, K)}{\partial K_{ij}}$$

- Descend kernel along loss function gradient with rate r
- Use trained kernel to solve system

Which loss function to use?

- Ideally some function that quantifies "wrongness of results"
- Incorporate as much prior information as possible (2211.15625)
- Or: Use unitarity norm $U(\Phi)$ as cheap proxy (2310.08053, 2309.06103)
- Evaluated on next-step field Φ'

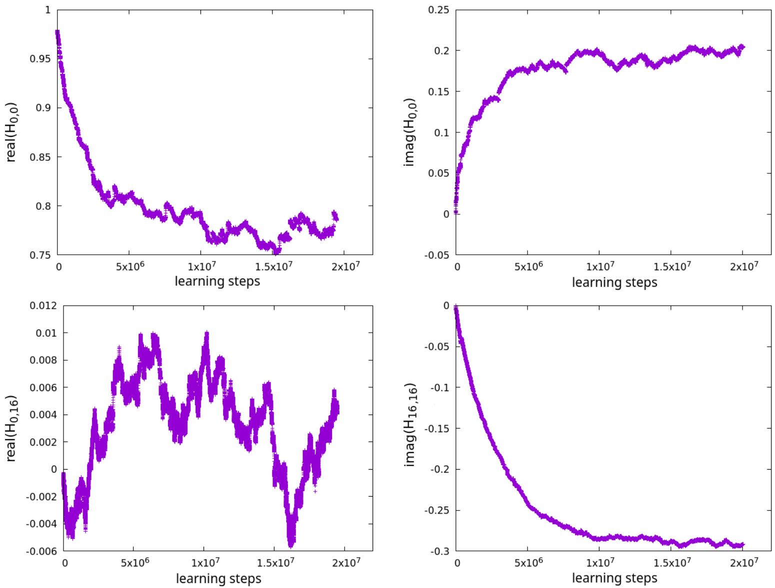
$$U(\Phi) = \sum_i^N \mathfrak{Im}(\phi_i)^2$$

$$\begin{aligned} L(\Phi, K) &= U(\Phi'(\Phi, K)) \\ &= U(\Phi + \Delta\Phi(\Phi, K)) \end{aligned}$$

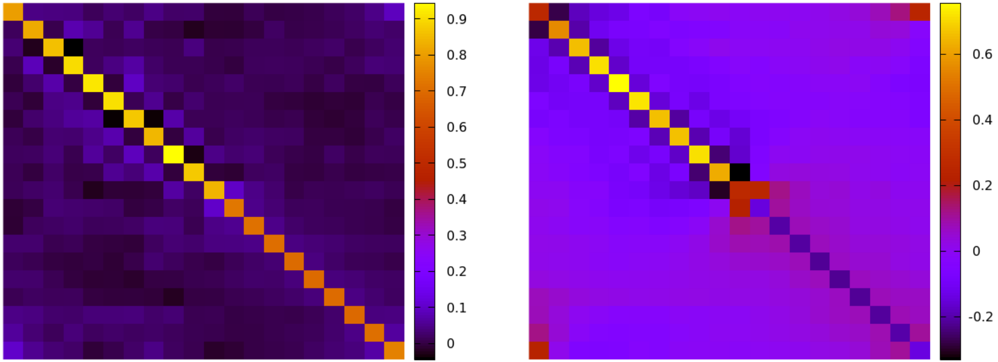
How to train your kernel

- Kernel starts out at identity
- One training step consists of:
  - Themalization for $10 - 100$ Langevin time
  - Averaging the loss function gradient for $0.1 - 1$ Langevin time
  - Applying the gradient to the kernel with learning rate $r = 10^{-4} - 10^{-3}$
  - Rescale kernel such that sum of squares is fixed
- Repeat for $10^6 - 10^7$ times
- Reset Φ every $\sim 10^5$ training steps to combat runaway feedback effects
- Only $N_t = 20$ since gradient descent is very expensive

Kernel training in action

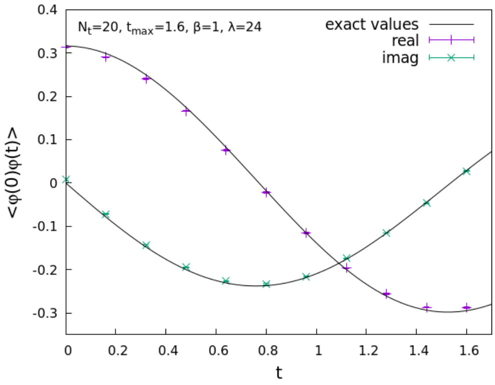


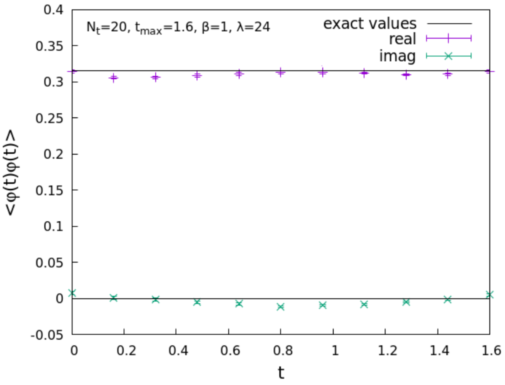
What does the kernel learn?



Lampl and Sexty, 2023, arXiv:2309.06103

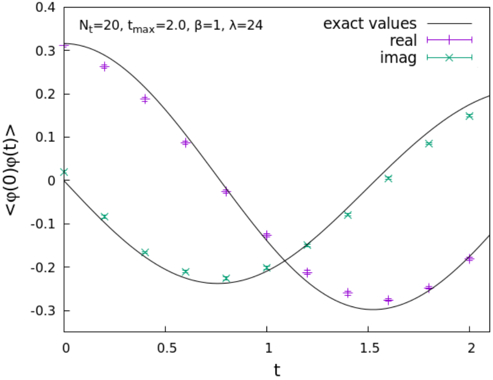
Kernel improves real time extent considerably

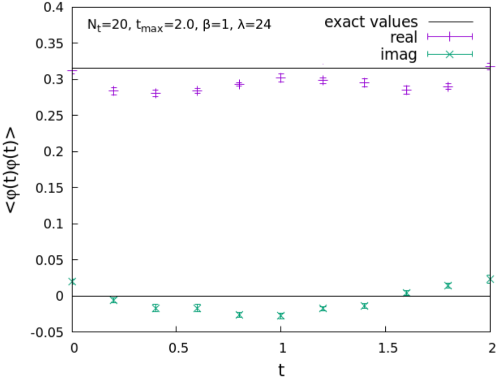




Lampl and Sexty, 2023, arXiv:2309.06103

Limit of constant kernel is $t \approx 2\beta$





Lampl and Sexty, 2023, arXiv:2309.06103

Next step: Linear kernel

- Field dependent kernel with N^3 parameters:

$$H_i^j(\Phi) = a_i^{jk}\phi_k + b_i^j$$

- More parameters and field dependence should lead to better results (?)
- Overfitting with simple loss function (?)
- Easily expandable to polynomial kernels

Neural net: When polynomial kernels are not enough

- Neural network can approximate any function (Universal approximation theorem)
- Holomorphicity and universal approximation at the same time are difficult (if not impossible) Voigtlaender, 2012.03351
- Other considerations:
  - N inputs (Φ), N^2 outputs (K)
  - Different possible topologies
  - Different activation functions (can lead to non-universality)

$$N : \mathbb{C}^n \to \mathbb{C}^{n \times n} : \Phi \mapsto K = N(\Phi)$$

Summary

- Kernels can improve real time extent
- Constant kernels hit barrier at $t_{\text{max}} \approx 2\beta$
- Linear field dependend kernels should reach higher times
- Using neural networks is possible, but brings problems