Parallel-in-time methods with ML

Viktor Csomor
s1984842@ed.ac.uk

THE UNIVERSITY *of* EDINBURGH

February 17, 2020

Supervisors: Rupert Nash, Anna Roubíčková

epcc

Overview

Motivation

Differential equations

- numerous scientific and engineering problems are described by time-dependent differential equations
- ODEs and PDEs
- many different numerical methods exist for solving them [1]

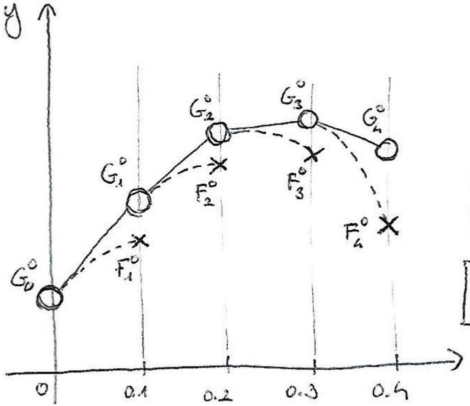
Parallelism

- traditionally, parallelised across the system or across space [2]
- good weak scaling, limited strong scaling
- but for some problems only wall time matters (e.g. stock trading)
- need to exploit parallelism better to reduce wall clock time

Parallel-in-time methods

- solution parallelised across the time steps
- more CPU time, less wall clock time
- parareal algorithm [3] for solving IVPs
- relies on two operators, coarse G and fine F
- for every iteration, G executed serially, F in parallel
- G must be fast but preferably also accurate enough

Parareal algorithm



$$y'(t) = f(y(t), t)$$
$$y(t_0) = y_0$$
$$t_0 \leq t \leq T$$

$$G_n^k = G(y_n^k, t_n, t_{n+1})$$
$$F_n^k = F(y_n^k, t_n, t_{n+1})$$

$$\boxed{y_{n+1}^k = G_n^k + \underbrace{F_n^{k-1} - G_n^{k-1}}_{\text{correction term}}}$$

Hypotheses

Hypotheses

1. an ML (regression) model can be trained and used as G to solve differential equations in a parareal framework
2. for some problems, the ML model based G operator is better than traditional coarse operators in terms of performance and accuracy
3. for some problems, the ML accelerated parareal framework achieves shorter wall time than a conventional parareal algorithm or a state-of-the-art traditional solver using the same number of CPU cores
   a. training time ignored
   b. training time taken into account

Related work

Related work

- neural network for solving time-dependent differential equations [4]
- neural network to approximate the gradients of high-dimensional PDEs [5]
- physics informed neural network (PINN) for solving PDEs [6] [7] [8]
- parareal solver using a PINN as F [9]

Challenges

Challenges

- numerical analysis is a complex field
- fast moving area of research
- many potential avenues (PDE solving methods, model training, tech stack)
- aiming high

Plan

Implementation

- develop a simple parareal framework
  - already implemented first version in C with POSIX threads
- construct an ML model (PINN vs standard models)
- train the model using targets provided either by G or F
- automatise the model training as part of the framework
- use the trained model for inference as G
- extend the framework to discretise PDEs or handle PDEs directly through the model

Problem iterations

- ODE (Lotka-Volterra)
- simple PDE (2D diffusion)
- high-dimensional PDE (Black-Scholes)

Performance

- compare the ML version of G to traditional ODE solvers of similar accuracy
- compare the performance of the parareal framework to traditional ODE and PDE solvers and analyse the impact of the model training on performance
- try different ML models (linear regression, ANN, RNN, etc.)
- take advantage of Cirrus' new GPUs

Tech stack

- C/C++ faster than Python but most scientific Python libraries are just wrappers on top of C bindings
- Python better suited for ML and numerical solver libraries are easier to use
- mpi4py [10] for distributed computing
- scikit-learn [11] and keras [12] for ML
- FEniCS [13] (finite element) and FiPy [14] (finite volume) to compare performance against

References I

[1] Endre Süli and David F Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003.

[2] C. W. Gear. Parallel methods for ordinary differential equations. *CALCOLO*, 25(1):1–20, Mar 1988.

[3] J. Lions. Résolution d'EDP par un schéma en temps ≪pararéel ≫A "parareal" in time discretization of PDE's. *Academie des Sciences Paris Comptes Rendus Serie Sciences Mathematiques*, 332(7):661–668, Apr 2001.

[4] Francesco Regazzoni, Luca Dede, and Alfio Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 07 2019.

References II

[5] Jiequn Han, Arnulf Jentzen, and Weinan E. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[6] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

[7] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, Dec 2018.

[8] Lu Lu, Xuhui Meng, Zhiping Mao, and George E. Karniadakis. Deepxde: A deep learning library for solving differential equations, 2019.

References III

[9] Xuhui Meng, Zhen Li, Dongkun Zhang, and George Em Karniadakis. Ppinn: Parareal physics-informed neural network for time-dependent pdes, 2019.

[10] Lisandro Dalcín, Rodrigo Paz, and Mario Storti. Mpi for python. *Journal of Parallel and Distributed Computing*, 65(9):1108 – 1115, 2005.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[12] François Chollet et al. Keras. `https://keras.io`, 2015.

References IV

[13] Martin S. Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The fenics project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[14] Jonathan E. Guyer, Daniel Wheeler, and James A. Warren. FiPy: Partial differential equations with Python. *Computing in Science & Engineering*, 11(3):6–15, 2009.