FEniCS Adjoint Documentation

Release 2019.1.2

Sebastian Mitusch

May 20, 2022

CONTENTS

The dolfin-adjoint project automatically **derives the discrete adjoint and tangent linear models** from a forward model written in the Python interface to FEniCS and Firedrake.

These adjoint and tangent linear models are key ingredients in many important algorithms, such as data assimilation, optimal control, sensitivity analysis, design optimisation, and error estimation. Such models have made an enormous impact in fields such as meteorology and oceanography, but their use in other scientific fields has been hampered by the great practical difficulty of their derivation and implementation. In his recent book, Naumann (2011) states that

> [T]he automatic generation of optimal (in terms of robustness and efficiency) adjoint versions of large-scale simulation code is **one of the great open challenges in the field of High-Performance Scientific Computing**.

The dolfin-adjoint project aims to solve this problem for the case where the model is implemented in the Python interface to FEniCS/Firedrake.

CHAPTER

ONE

NEWS

27.05.2019: dolfin-adjoint 2019.1 released (compatible with FEniCS 2019.1) ChangeLog.rst.

20.03.2019: Firedrake docker images with dolfin-adjoint preinstalled are now available

18.02.2019: New example demonstrating the shape-optimisation capabilities of dolfin-adjoint now available

05.10.2018: dolfin-adjoint 2018.1 released (compatible with FEniCS 2018.1) ChangeLog.rst.

08.01.2018: dolfin-adjoint/pyadjoint 2017.2 released (compatible with FEniCS 2017.2) (this website) dolfin-adjoint/libadjoint 2017.2 released (compatible with FEniCS 2017.2) (documentation)

dolfin-adjoint-difference

31.05.2017: dolfin-adjoint 2017.1 released (compatible with FEniCS 2017.1) **`ChangeLog`_**.

21.02.2017: dolfin-adjoint 2016.2 is now available as Conda package.

20.12.2016: dolfin-adjoint 2016.2 released (compatible with FEniCS 2016.2) **`Changelog`_**.

08.9.2016: We now offer Docker images to simplify the installation procedure.

30.8.2016: dolfin-adjoint 2016.1 released (compatible with FEniCS 2016.1) **`Changelog`_**.

20.8.2015: dolfin-adjoint 1.6 released (compatible with FEniCS 1.6) **`Changelog`_**.

11.6.2015: P. E. Farrell, S. W. Funke, D. A. Ham and M. E. Rognes were awarded the 2015 Wilkinson prize for numerical software for dolfin-adjoint.

CHAPTER

TWO

FEATURES

dolfin-adjoint has the following features:

- Works for both steady and **time-dependent problems** and for both linear and **nonlinear problems**.
- Using it is very **easy**: given a differentiable forward model, employing dolfin-adjoint involves changing on the order of ten lines of code.
- The adjoint and tangent linear models exhibit **optimal theoretical efficiency**. If every forward variable is stored, the adjoint takes 0.2-1.0x the runtime of the forward model, depending on the precise details of the structure of the forward problem.
- If the forward model runs in parallel, the adjoint and tangent linear models also **run in parallel** with no modification.
- If instructed, the adjoint model can automatically employ **optimal checkpointing** schemes to mitigate storage requirements for long nonlinear runs.
- **Rigorous verification** routines are provided, so that users can easily verify for themselves the correctness of the derived models.
- Solves **optimisation problems** constrained by partial differential equations by interfacing to powerful optimisation algorithms

For more details, see the features page.

CHAPTER

THREE

LIMITATIONS

To do all this, dolfin-adjoint requires some cooperation from the model developer:

- Works only with the **Python** interface of FEniCS and Firedrake.
- For the adjoint to be consistent, the discretisation must be differentiable.
- All changes to object values (matrices, vectors, functions) must happen through the FEniCS/Firedrake interface, though custom operations can be recorded manually.

CHAPTER

FOUR

HOW IT WORKS

The traditional approach to deriving adjoint and tangent linear models is called algorithmic differentiation (also called automatic differentiation). The fundamental idea of algorithmic differentiation is to *treat the model as a sequence of elementary instructions*. An elementary instruction is a simple operation such as addition, multiplication, or exponentiation. Each one of these operations is differentiated individually, and the derivative of the whole model is then composed with the chain rule.

The dolfin-adjoint project is instead based on a very different approach. The model is considered as *a graph of high-level operations*. This abstraction is similar to the fundamental abstraction of algorithmic differentiation, but operates at a much higher level of abstraction. This idea is implemented in a software library, pyadjoint. When this new idea is combined with the high-level abstraction of the FEniCS system, many of the difficult problems associated with algorithmic differentiation dissolve.

For more technical details on pyadjoint and dolfin-adjoint, see the papers.

CHAPTER

FIVE

CONTRIBUTORS

The dolfin-adjoint project is developed and maintained by the following authors:

- Sebastian Mitusch (Simula Research Laboratory)
- Jørgen S. Dokken (Simula Research Laboratory)
- Patrick E. Farrell (Mathematical Institute, University of Oxford)
- Simon W. Funke (Simula Research Laboratory)
- David A. Ham (Department of Mathematics and Department of Computing, Imperial College London)
- Marie E. Rognes (Simula Research Laboratory)
- James R. Maddison (School of Mathematics, University of Edinburgh)

CHAPTER

SIX

LICENSE

Like the core FEniCS components, The dolfin-adjoint software is freely available under the GNU LGPL, version 3.