Algorithmic differentiation

January 2, 2019

1 Introduction

The goal is to reverse engineering a solution field to find the parameter, in our case the diffusion coefficient (ν), used in the original solution using Algorithmic Differentiation (AD).

2 Prerequisite

Install “Fenics_adjoint” using the procedure described here:

http://www.dolfin-adjoint.org/en/latest/download/

Also check out the tutorial on the same website:

http://www.dolfin-adjoint.org/en/latest/documentation/tutorial.html#dolfin-adjoint-tutorial

3 Steps

1. Generate a reference solution using an arbitrary diffusion coefficient (save the value somewhere).
2. Save the result in an hdf5 file (for example: "temp.h5").

```
hdf = HDF5File(mesh.mpi_comm(), "temp.h5", "w")
hdf.write(u, "temp")
del hdf
```

3. Start a new python script.
4. Load the fenics_adjoint python module.

```
from fenics_adjoint import *
```

5. Load in the reference temperature previously generated.

```
hdf = HDF5File(mesh.mpi_comm(), "temp.h5", "r")
attr = hdf.attributes("temp")
dataset = "temp/vector_0"
attr = hdf.attributes(dataset)
hdf.read(temp_ref, dataset)
```

6. Check that the loaded temperature is correct with a terminal printing or a VTK file.
7. Define an initial diffusion coefficient and step size (α).
8. Start AD loop.
    (a) Solve the temperature field for the current diffusion coefficient

(b) The function we are trying to minimizing is:

$$J(u) = \int_{\Omega} \langle u_{ref}\left(T\right) - u\left(T\right), u_{ref}\left(T\right) - u\left(T\right)\rangle \, \mathrm{d}\Omega$$

Define this function using a build-in function.

(c) Compute the derivative of Jacobian ($\frac{\mathrm{d}J}{\mathrm{d}\nu}$) using a build in function.

(d) Update the diffusion coefficient:

$$\nu = \nu - \alpha \frac{\mathrm{d}J}{\mathrm{d}\nu}$$

9. Print the latest diffusion coefficient. Does it match the original one?

10. Et voilà!

4 Test cases

Apply these steps on two test cases:

1. A simple coarse rectangular test case to debug the python code (case1).

2. A more demanding test case like a 2d piston (case4).

For each test case:

- Confirm that the resulting diffusion coefficient reverse engineered corresponds to the original one.

- Comment on the effect of increasing and decreasing the step size α.

- Provide a convergence plot for each step size tested. If possible, provide them on one graph for comparison purpose.

- Does the mesh size influence the convergence rate?

- Does parallelization works in AD?

5 Bonus question

Does the solution converge if slight noise is introduced to the reference solution? If yes, how much?