Computational Differentiation (WS1617)

J. Hüser, M.Sc., und Prof. Dr. U. Naumann

LuFG Informatik 12: Software and Tools for Computational Engineering
RWTH Aachen

Tutorial Sheet 2 (26.10.16)

(SAC and lDAG, Tangent Mode AD, Gradient Descent)

1 SAC and lDAG

Consider again Nesterov's Chebyshev-Rosenbrock function $f : \mathbb{R}^n \to \mathbb{R}$ with

$$f(x) = \frac{1}{4}(x_0 - 1)^2 + \sum_{i=0}^{n-2}(x_{i+1} - 2x_i^2 + 1)^2 \ .$$

Task: For $n = 2$ write down a single assignment code (SAC) and the corresponding linearized directed acyclic graph (lDAG) like described in the lecture slides (slide 90 and the following). Also write down a way of computing the elements of the gradient $\nabla f(x)$ by following the chain rule on the lDAG.

2 Tangent Mode AD

During the exercise session we saw the following example of how to use dco/c++ for first order scalar tangent mode algorithmic differentiation (AD):

```
#include <iostream>
#include <cmath>
using namespace std;

#include "dco.hpp"
using namespace dco;

template <typename T>
T f(T *x) {
        return (sin(x[0]) * x[1] * x[1] + 5 * x[0]);
}

int main() {
        gt1s<double>::type x[2], y;
        x[0] = 2.0;
        x[1] = 2.0;
        derivative(x[0]) = 1.0;
        derivative(x[1]) = 0.0;
        y = f(x);
        cout << y << endl;
        cout << derivative(y) << endl;

        return 0;
}
```

For the function $f(x) = \sin(x_0)x_1^2 + 5x_0$ the above example prints $\texttt{y} = f(2,2)$ and

$$\texttt{derivative(y)} = \nabla f(2,2)\begin{pmatrix}1\\0\end{pmatrix} = \frac{\partial f}{\partial x_0}(2,2)\,.$$

The following demonstrates how to compile the above code under my Linux system:

- I downloaded dco/c++ and the licence file as described on the previous exercise sheet. My dco/c++ installation is in `/home/jonathan/Downloads/dco_cpp_v3.1.4_trial_lin64_gcc` and my licence file is `/home/jonathan/Downloads/keys.txt`. The paths will probably be different on your computer so you need to change them accordingly.
- Activate the licence key file by typing

  ```
  export NAG_KUSARI_FILE=/home/jonathan/Downloads/keys.txt
  ```

  into you shell. Note that the export will only be valid for the shell session in which you enter the command. You need to do all the compilation and execution involving dco/c++ within that same shell session.
- Compile the example program in `example.cpp` with

  ```
  g++ -I/home/jonathan/Downloads/dco_cpp_v3.1.4_trial_lin64_gcc/include -c
  example.cpp -o example.o
  ```
- Link the static library part of dco/c++ (the order of object file and libary matters)

  ```
  g++ -I/home/jonathan/Downloads/dco_cpp_v3.1.4_trial_lin64_gcc/include example.o
  /home/jonathan/Downloads/dco_cpp_v3.1.4_trial_lin64_gcc/lib/libdcoc.a -o
  example
  ```
- Finally you can run the program with

  ```
  ./example
  ```

Task: Based on your implementation of Nesterov's Chebyshev-Rosenbrock from the previous exercise sheet write a function that computes the complete gradient using dco/c++ tangent mode AD. The tangent mode AD gradient function should have the same interface as the finite difference gradient function from the previous exercise sheet.

3 Gradient Descent

Download the `heat_equation.zip` archive from the `261016` folder in the L2P. Go over the finite difference gradient descent implementation (in `fd_grad.cpp`) for the heat equation example developed during the last tutorial session. Also go over the tangent mode AD gradient implementation (in `gt1s.cpp`) developed during the tutorial session. You should be able to get the example to run with dco/c++ with the above instructions and replacing the file name of the `.cpp` file.
Task: Merge the two implementations into a gradient descent that uses tangent mode AD instead of finite differences.