The Manual of ReverseAD

Mu Wang

September 19, 2016

Abstract

(TODO)

Contents

Preliminaries of Automatic Differentiation

(TODO)

Chapter 1

Basic Usage of ReverseAD

1.1 Download And Installation

1.2 The One Minute Example

```
#include <memory>
#include <iostream>

#include "reversead/reversead.hpp"

using ReverseAD::adouble;
using ReverseAD::TrivialTrace;
using ReverseAD::BaseReverseHessian;
using ReverseAD::DerivativeTensor;
using ReverseAD::trace_on;
using ReverseAD::trace_off;

template <typename T>
T foo(T x1, T x2) {
  return pow(x1+1, 2) + pow(x1*x1 - x2, 2);
}

int main() {
  adouble x1, x2;
  adouble y;
```

```
21   double vy;
22   trace_on<double>(); // begin tracing
23   x1 <<= 2.0; // independent variable #0
24   x2 <<= 3.0; // independent variable #1
25   y = foo<adouble>(x1, x2); // function evaluation
26   y >>= vy; // dependent variable
27   std::shared_ptr<TrivialTrace<double>> trace =
28       trace_off<double>(); // end tracing
29   std::cout << "y = " << vy << std::endl;
30
31   std::unique_ptr<BaseReverseHessian<double>> hessian(
32       new BaseReverseHessian<double>(trace));
33   std::shared_ptr<DerivativeTensor<size_t, double>> tensor = hessian
      ->compute(2,1);
34
35   // retrieve results
36   size_t size;
37   size_t** tind;
38   double* values;
39   // adjoints : dep[0].order[1]
40   tensor->get_internal_coordinate_list(0, 1, &size, &tind, &values);
41   std::cout << "size of adjoints = " << size << std::endl;
42   for (size_t i = 0; i < size; i++) {
43     std::cout << "A["<< tind[i][0] << "] = " << values[i] << std::
      endl;
44   }
45   // hessian : dep[0].order[2]
46   tensor->get_internal_coordinate_list(0, 2, &size, &tind, &values);
47   std::cout << "size of hessian = " << size << std::endl;
48   for (size_t i = 0; i < size; i++) {
49     std::cout << "H["<< tind[i][0] << ", " << tind[i][1]
50               << "] = " << values[i] << std::endl;
51   }
52 }
```

Listing 1.1 One Minute Example

1.3 Additional Options and Examples

1.3.1 Disk Tape

1.3.2 Forward-over-Reverse Mode

Chapter 2

Internal Implementation

2.1 Function Trace

```
private:
 std::shared_ptr<VirtualTape<opbyte>> op_tape;
 std::shared_ptr<VirtualTape<locint>> loc_tape;
 std::shared_ptr<VirtualTape<Base>> val_tape;
 std::shared_ptr<VirtualTape<Base>> param_tape;
 std::shared_ptr<VirtualTape<double>> coval_tape;
```

2.2 Active Type

2.3 Indexing Scheme

2.4 Derivative Computation Classes

2.4.1 Adjoints, Hessian and Third Order

2.4.2 Generic Reverse Mode

2.4.3 Generating Flat Code

2.5 Forward-over-Reverse Mode

2.6 Time-Step Functions