Google Summer of Code 2021

Utilize second order derivatives from Clad in ROOT

Student: Baidyanath Kundu
Mentors: Vassil Vassilev, Ioana Ifrim

Overview

What is Clad?
Clad is Clang plugin for Automatic Differentiation.

What is TFormula and ROOT?
ROOT is a data processing framework created by CERN.

TFormula is a ROOT class which bridges compiled and interpreted code.

Aim:
The project aimed to add second order derivative support in TFormula using `clad::hessian`.

Outcomes

Clad

Array differentiation in forward mode

```
#include <iostream>
#include <clad/Differentiator/Differentiator.h>

double f(double *x, double n) {
  double sq_sum = 0;
  for (int i = 0; i < n; i++)
    sq_sum += x[i] * x[i];
  return sq_sum;
}

int main() {
  auto f_dx = clad::differentiate(f, "x[1]");
  f_dx.dump();

  double x[3] = {1, 2, 3};

  double f_dx0 = f_dx.execute(x, 3);
  printf("dx0 = %g\n", f_dx0);
}
```

```
The code is: double f_darg0_1(double *x, double n) {
    double _d_n = 0;
    double _d_sq_sum = 0;
    double sq_sum = 0;
    {
        int _d_i = 0;
        for (int i = 0; i < n; i++) {
            _d_sq_sum += (i == 1) * x[i] + x[i] * (i == 1);
            sq_sum += x[i] * x[i];
        }
    }
    return _d_sq_sum;
}

dx0 = 4
```

Clad

Array differentiation in reverse mode

```cpp
#include <iostream>
#include <clad/Differentiator/Differentiator.h>

double f(double *x, double n) {
  double sq_sum = 0;
  for (int i = 0; i < n; i++)
    sq_sum += x[i] * x[i];
  return sq_sum;
}

int main() {
  auto f_dx = clad::gradient(f);
  f_dx.dump();

  double x[3] = {1, 2, 3};

  double dx[3] = {0};
  clad::array_ref<double> dx_ref(dx, 3);
  double dn = 0;

  f_dx.execute(x, 3, dx_ref, &dn);
  printf("dx = {%g, %g, %g}\ndn = %g\n", dx[0], dx[1], dx[2], dn);
}
```

```cpp
The code is: void f_grad(double *x, double n, clad::array_ref<double> _d_x, clad::array_ref<double> _d_n) {
    double _d_sq_sum = 0;
    unsigned long _t0;
    int _d_i = 0;
    clad::tape<double> _t1 = {};
    clad::tape<int> _t2 = {};
    clad::tape<double> _t3 = {};
    clad::tape<int> _t4 = {};
    double sq_sum = 0;
    _t0 = 0;
    for (int i = 0; i < n; i++) {
        _t0++;
        sq_sum += clad::push(_t3, x[clad::push(_t2, i)]) * clad::push(_t1, x[clad::push(_t4, i)]);
    }
    double f_return = sq_sum;
    goto _label0;
  _label0:
    _d_sq_sum += 1;
    for (; _t0; _t0--) {
        double _r_d0 = _d_sq_sum;
        _d_sq_sum += _r_d0;
        double _r0 = _r_d0 * clad::pop(_t1);
        _d_x[clad::pop(_t2)] += _r0;
        double _r1 = clad::pop(_t3) * _r_d0;
        _d_x[clad::pop(_t4)] += _r1;
        _d_sq_sum -= _r_d0;
    }
}

dx = {2, 4, 6}
dn = 0
```

Clad

Array differentiation in hessian mode

```
#include <iostream>
#include <clad/Differentiator/Differentiator.h>

double h(double *x) {
  return x[0] * x[1] * x[2];
}

int main() {
  auto f_hess = clad::hessian(h, "x[0:2]");
  f_hess.dump();

  double x[3] = {1, 2, 3};

  double hess_mat[9] = {0};
  clad::array_ref<double> hess_mat_ref(hess_mat, 9);
  double dn = 0;

  f_hess.execute(x, hess_mat_ref);
  printf("hessian_matrix = \n{%g, %g, %g},\n{%g, %g, %g},\n{%g, %g, %g}\n",
                 hess_mat[0], hess_mat[1], hess_mat[2],
                 hess_mat[3], hess_mat[4], hess_mat[5],
                 hess_mat[6], hess_mat[7], hess_mat[8]);
}
```

```
The code is: void h_hessian(double *x, clad::array_ref<double> hessianMatrix) {
    h_darg0_0_grad(x, hessianMatrix.slice(0UL, 3UL));
    h_darg0_1_grad(x, hessianMatrix.slice(3UL, 3UL));
    h_darg0_2_grad(x, hessianMatrix.slice(6UL, 3UL));
}

hessian_matrix =
{0, 3, 2},
{3, 0, 1},
{2, 1, 0}
```

ROOT

Hessian mode in ROOT

```
❯ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.25/01                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Aug 24 2021, 15:45:34                 |
  | From heads/master@v6-25-01-1836-g5d536b29d7                      |
  | With c++ (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0                    |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0] TFormula f("f", "x*sin([0]) + y*cos([1])");
root [1] double x[] = {3, 4};
root [2] double p[] = {1.57079633, 0};
root [3] f.SetParameters(p);
root [4] TFormula::CladStorage hess(4);
root [5] f.HessianPar(x, hess);
root [6] printf("{%g, %g},\n{%g, %g}\n", hess[0], hess[1], hess[2], hess[3]);
{-3, 0},
{0, -4}
```

Thankyou