Differentiating GATE/Geant4 with Derivgrind

Max Aehle

Scientific Computing Group,
University of Kaiserslautern-Landau

& SIVERT Research Training Group

Workshop on Differentiable and Probabilistic Programming for Fundamental Physics

MIAPbP Munich, Germany
June 16th, 2023

Algorithmic Differentiation

- Given a computer-implemented function, AD provides computer-implemented derivative functions, e. g. for gradient-based optimization.
- Based on the idea to insert AD logic for all real-arithmetic calculations performed by the primal program.
- Forward-mode AD propagates dot values alongside the primal program execution.
- Reverse-mode AD back-propagates adjoint values in a reverse pass, after a forward-pass recording.

Ways to Add AD Logic

Classification of AD tools according to mechanism used for instrumentation:

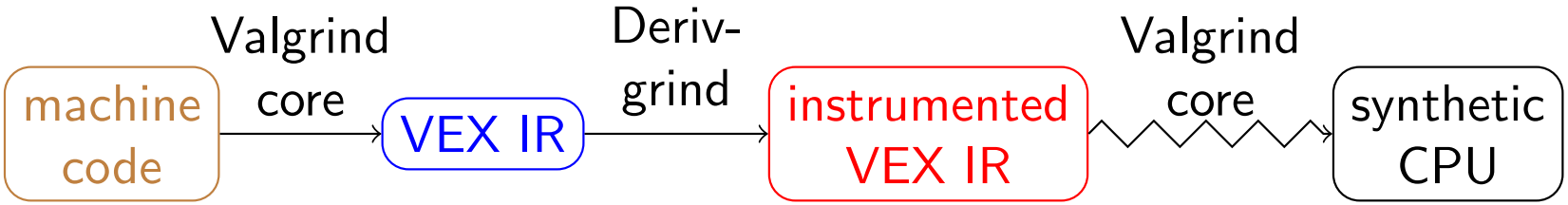
- Source Transformation: TAPENADE, ...
- Operator Overloading: ADOL-C, autograd, CoDiPack, ...
- Compiler-Based: CLAD, Enzyme, ...
- Machine Code-Based: Derivgrind

Machine code-based AD reduces interaction with the primal program's source code to the bare minimum, which is desirable if

- the program is written in a (combination of) programming languages for which there is no suitable source code-based AD tool, or
- libraries used by the program are only available in compiled form, or
- to reduce the effort of integrating an AD tool into a primal program, e. g. for exploratory studies.

Valgrind and VEX IR

We implemented Derivgrind as a "tool" in the Valgrind instrumentation framework for building dynamic analysis tools.



E. g., the x86 instruction `mulsd %xmm0, %xmm1` to multiply scalar double-precision value in register `%xmm0` to `%xmm1` could be translated as

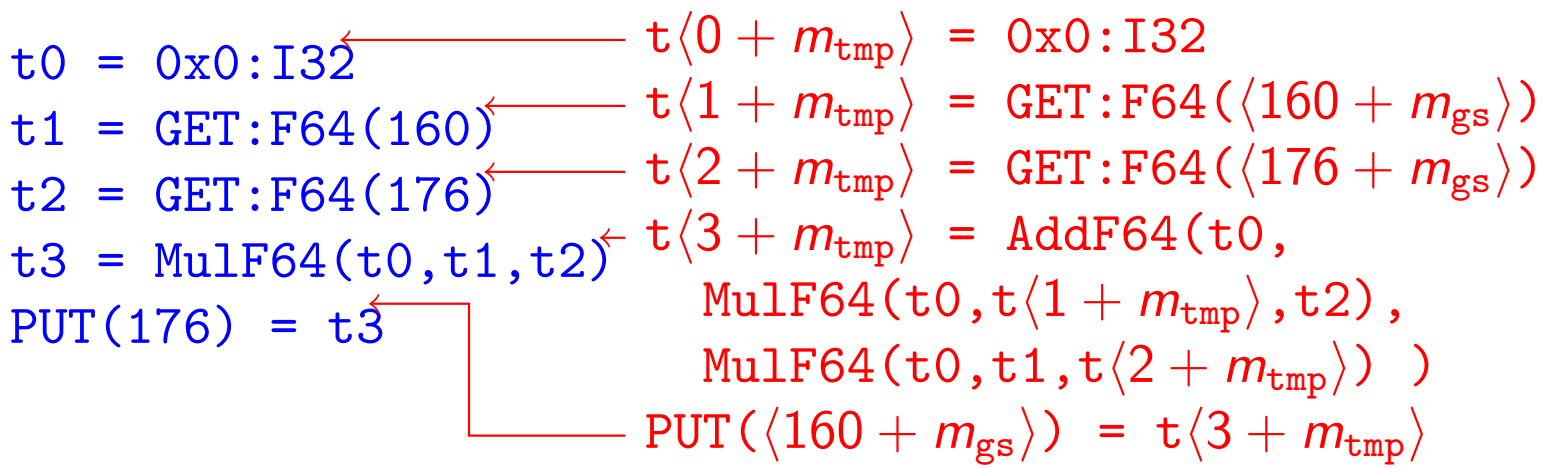
```
t0 = 0x0:I32
t1 = GET:F64(160)
t2 = GET:F64(176)
t3 = MulF64(t0,t1,t2)
PUT(176) = t3
```

Forward Mode: Propagation of Dot Values

For every number a, keep track of $\dot{a} = \frac{\partial a}{\partial x}$ for single AD input x.

- Store dot value of temporary $\mathtt{t}\langle i \rangle$ in shadow temporary $\mathtt{t}\langle i + m_{\mathtt{tmp}} \rangle$.
- Store dot value of register with byte offset j in shadow register at byte offset $j + m_{\mathtt{gs}}$.
- Store dot value of memory using a shadow memory tool.
- The dot value of a constant is zero.
- Use differentiation rules like

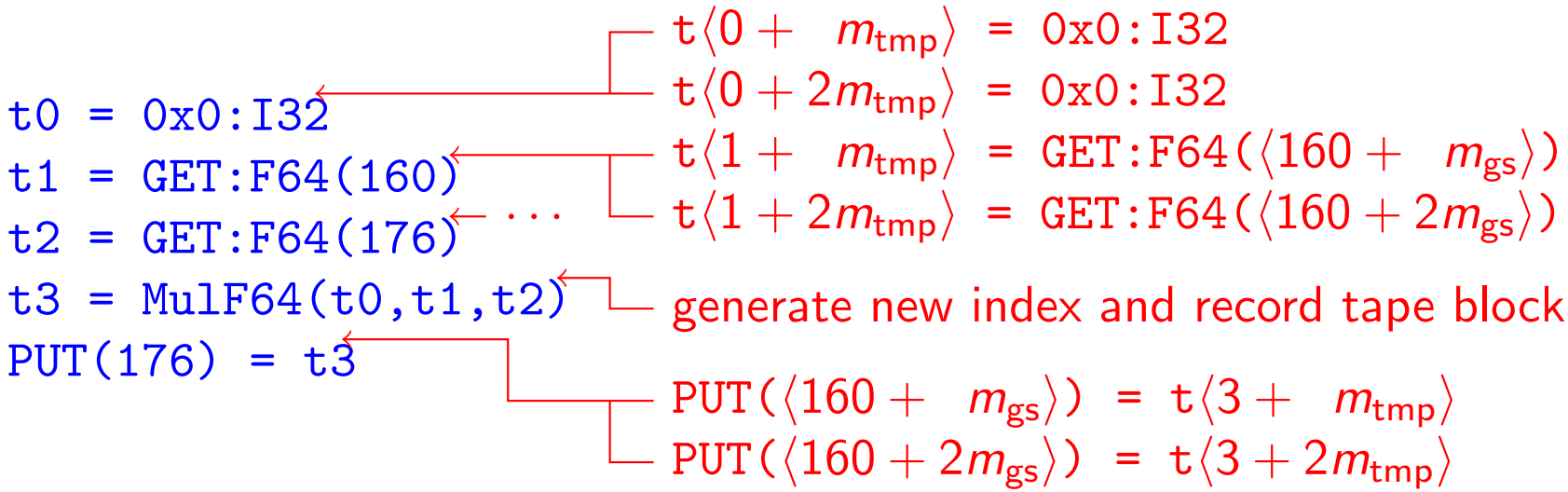
$$a_3 = a_1 \cdot a_2 \rightsquigarrow \dot{a}_3 = \dot{a}_1 \cdot a_2 + a_1 \cdot \dot{a}_2.$$



Reverse Mode: Tape Recording

For every number *a*, identify it by an *index* and record a *tape*

- Store indices in two layers of shadow memory.
- Passive variables have the default index 0.
- Linear index management and Jacobian tape.



Bit-Tricks

Supported "bit-tricks":

- Compute the negative value of a floating-point number by flipping the sign bit with a bitwise logical "xor".
- ...

Unsupported "bit-tricks":

- Double a number by incrementing the exponent bits.
- ... ...

⇝ Assumed not to be present in the primal program. (OK in libm.)

⇝ (Would also be problematic for source-code-based AD tools, to a lesser extent.)

Specifying Input and Output Variables

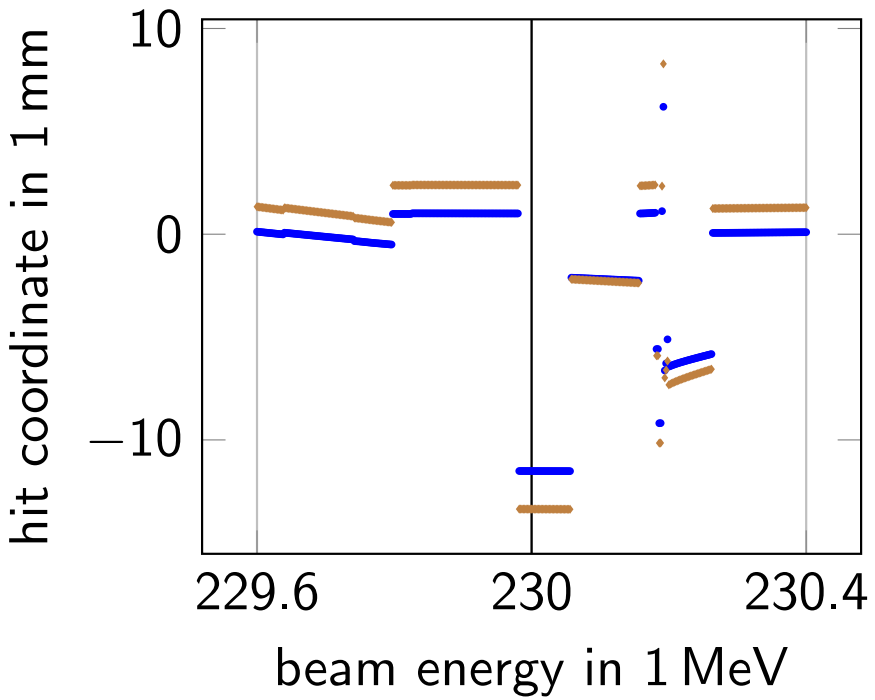
In order to identify input and output variables in the compiled program, limited access to the primal program's source code is necessary.

Client requests interface: Call specific C macros/functions from the primal code. Requires write access and recompilation of specific source files.

We will now consider the following example:
Derivgrind's forward mode is applied to GATE/Geant4.

Application to GATE/Geant4

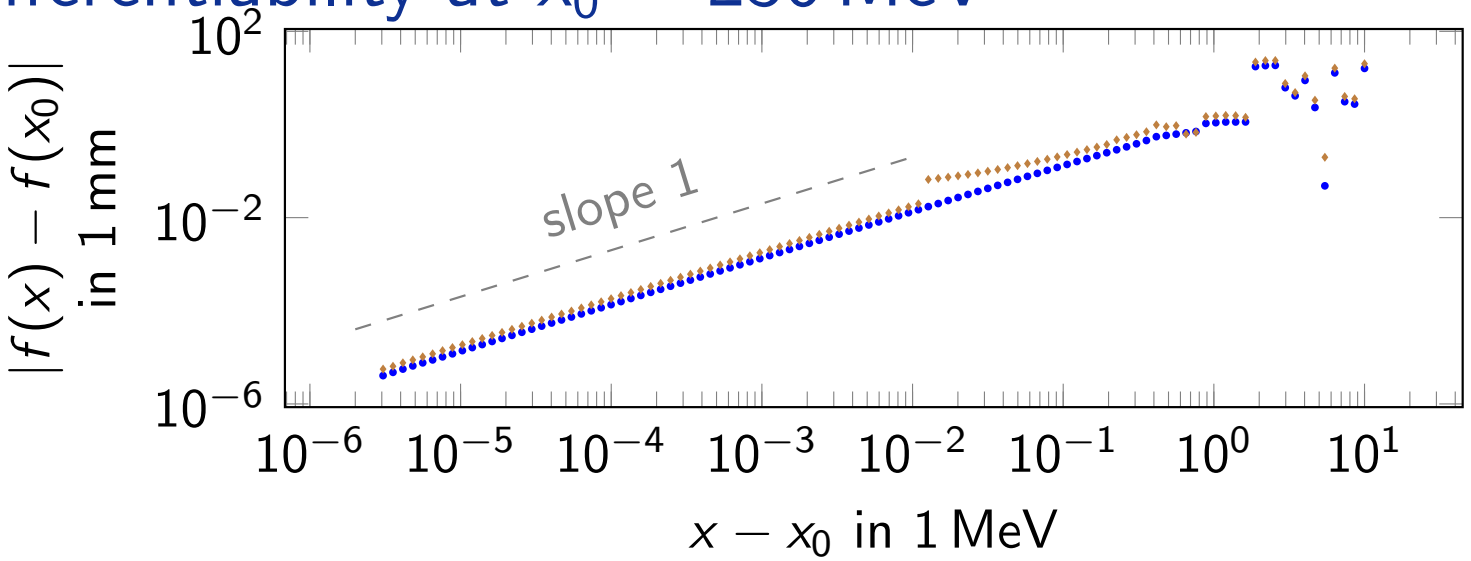
- Monte Carlo simulation for particle physics, $>$ 1 million lines of C++ code.
- In our setup, a single energetic proton passes through a human head and a digital tracking calorimeter. Our input is the beam energy, and our output is a hit coordinate in the first two tracking layers.
- The dependency is piecewise differentiable in an interval around 230 MeV.



- First tracking layer.
- Second tracking layer.

Adapted from Aehle, Alme et al., Derivatives in Proton CT.

Differentiability at $x_0 = 230\,\text{MeV}$



$\rightsquigarrow$ The hit coordinate $f(x)$ is differentiable in the beam energy x at x_0.

Derivatives in $\frac{\text{mm}}{\text{MeV}}$	• first layer	♦ second layer
Difference quotient	-0.082016	-0.130653

Integrating Derivgrind into Geant4 with Client Requests

```
+ #include "/somepath/include/valgrind/derivgrind.h"
```

"Seeding" dot values of the input variable (beam energy):

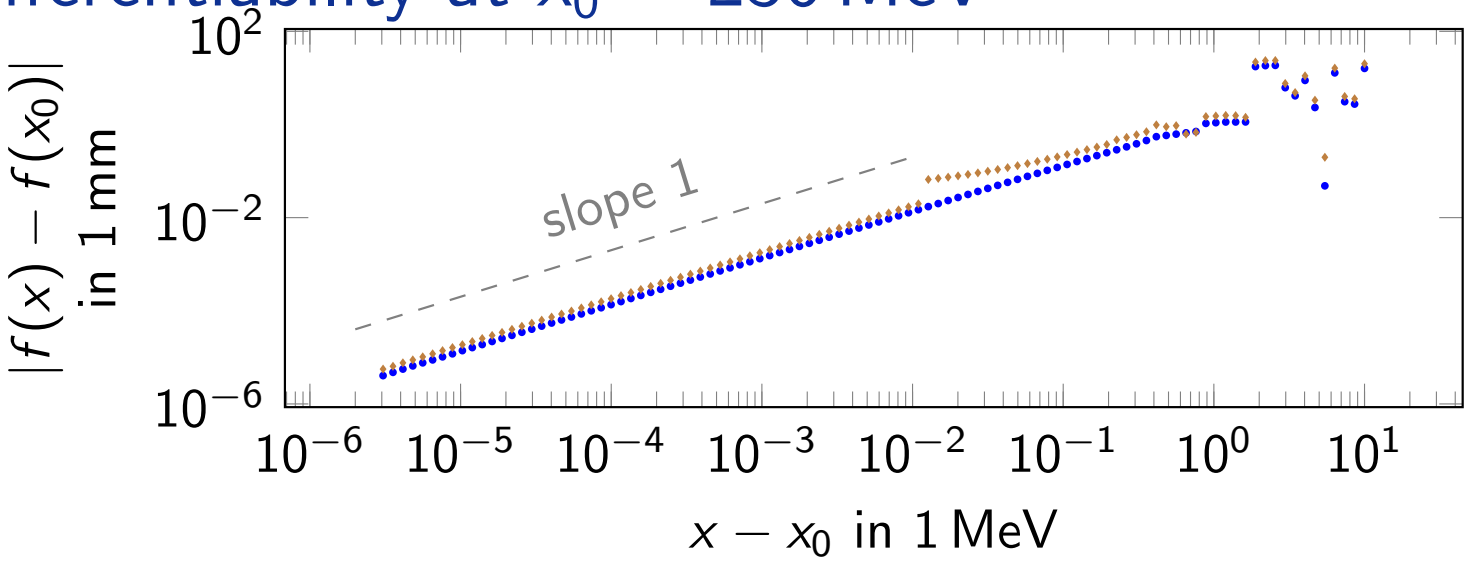
```
    if (command == pIonCmd) {
      pSourcePencilBeam->SetIonParameter(newValue);
    }
    if (command == pEnergyCmd) {
      double energy = pEnergyCmd->GetNewDoubleValue(newValue);
+     double one = 1.0;
+     DG_SET_DOTVALUE(&energy,&one,sizeof(double));
      pSourcePencilBeam->SetEnergy(energy);
    }
```

Obtaining dot values of output variables (hit position):

```
    if (m_rootHitFlag) m_treeHit->Fill();
+   float pos = *(float*)(m_treeHit->GetBranch("posX")->GetAddress());
+   float pos_d;
+   DG_GET_DOTVALUE(&pos,&pos_d,sizeof(float));
+   std::cout << "pos_d=" << pos_d << "\n";
```

Running GATE under Derivgrind: `valgrind --tool=derivgrind Gate ⟨args⟩`

Differentiability at $x_0 = 230\,\text{MeV}$



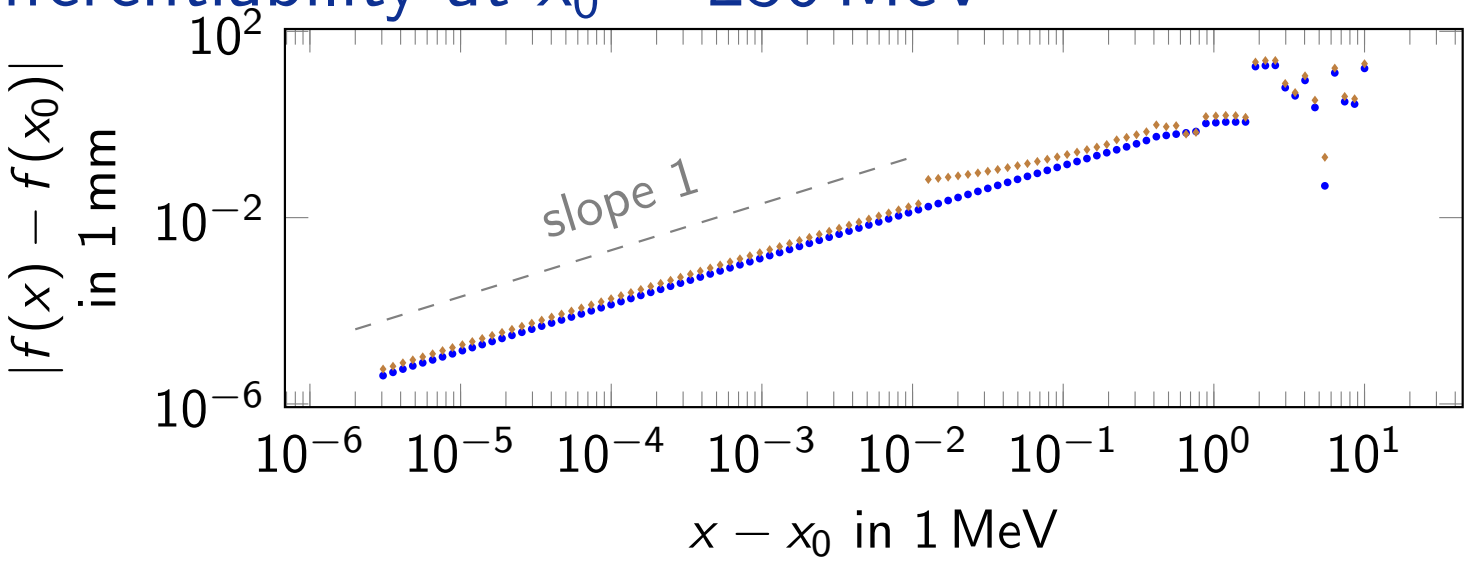
$\rightsquigarrow$ The hit coordinate $f(x)$ is differentiable in the beam energy x at x_0.

Derivatives in $\frac{\text{mm}}{\text{MeV}}$	• first layer	♦ second layer
Difference quotient	−0.082016	−0.130653
Derivgrind	−0.068512	−0.113841

Bit-Tricks in Geant4

- Geant4 defines and uses a function `G4Log` adapted from the VDT math library.
- `G4Log` first performs a "range reduction", scaling its argument by power of 2 to map it into $[0.5, 1)$.
- This is done by setting the exponent bits to `0b01111111110` via bitwise operations.
- Derivgrind's normal dot value propagation does not recognize this as an arithmetic operation.
- Fix: Edit Geant4 source code, replacing body of `G4Log` by call to `log`.

Differentiability at $x_0 = 230\,\text{MeV}$



$\rightsquigarrow$ The hit coordinate $f(x)$ is differentiable in the beam energy x at x_0.

Derivatives in $\frac{\text{mm}}{\text{MeV}}$	• first layer	♦ second layer
Difference quotient	−0.082016	−0.130653
Derivgrind	−0.068512	−0.113841
Derivgrind, `G4Log` replaced	−0.081339	−0.130524

$\rightsquigarrow$ After the fix, Derivgrind's derivative agrees well with difference quotients.

Summary

- Derivgrind implements forward-mode AD and operator-overloading-style reverse-mode AD for compiled programs.
- Very limited access to primal source code needed, only in order to identify input/output variables (and for debugging in case of failure).
- In the case of unsupported bit-tricks, Derivgrind might miss or misunderstand real-arithmetic dependencies.
- Successfully tested on "big" software projects Geant4, CPython and LibreOffice Calc, with bit-trick issues observed in corner cases or easily fixable.

Next Step

Debugging tools, applications, ...

Contact Information

Max Aehle, max.aehle@scicomp.uni-kl.de,

and Johannes Blühdorn, Max Sagebaum, Nicolas R. Gauger

Chair for Scientific Computing
University of Kaiserslautern-Landau (RPTU)

Code: https://github.com/SciCompKL/derivgrind

Project E-Mail Address: derivgrind@projects.rptu.de

Video (7 min) about Derivgrind+LibreOffice Calc:
https://t1p.de/tt4ne