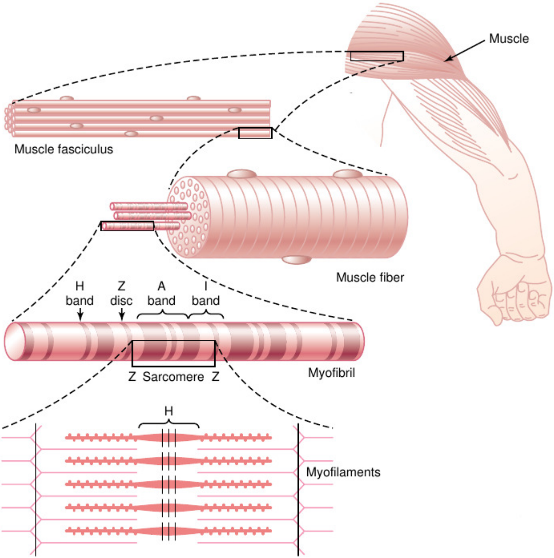
Not to code

Lars Lubkoll

May 19, 2015

Muscle tissue



A model for muscle tissue

$$\begin{aligned} W(F) = c\left[\exp\left(b(\bar{\iota}_1 - 3)\right) - 1\right] \\ + d\left(\exp\left(a\left[\bar{\iota}_6 - 1\right]^2\right) - 1\right) \\ + \frac{e}{50}\left(j^5 - j^{-5} - 2\right), \end{aligned}$$

with

$$\begin{aligned} \bar{\iota}_1 &= \mathrm{tr}(F^T F)\,\det(F^T F)^{-1/3}, \\ \bar{\iota}_6 &= \mathrm{tr}(F^T F M^2)\,\det(F^T F)^{-1/3}, \\ j &= \det(F). \end{aligned}$$

Automatic differentation

Provide suitably prepared function, compute derivative.

Automatic differentation

Provide function + derivatives

Advantages:

- Easy to use (no code preparation, “arbitrary input”).
- Efficient reuse of intermediate results (caching vs. lazy evaluation).
- Simple extension to an arbitrary number of variables.

Function generation

Idea: Implement (up to third derivative + operator overloads)

- chain rule $[f\,(g(x))]' = f'(g(x))g'(x)$

  ```
  template <class F, class G> class Chain;
  ```

- sum rule $[f(x) + g(x)]' = f'(x) + g'(x)$ and

  ```
  template <class F, class G> class Sum;
  ```

- product rule $[f(x)g(x)]') = f'(x)g(x) + f(x)g'(x)$

  ```
  template <class F, class G> class Product;
  ```

Implement $f(x) = \sqrt{x^3} + \sin(\sqrt{x}) = (h \circ g)(x)$, with

$$h(x) = x^3 + \sin(x) \quad \text{and} \quad g(x) = \sqrt{x}$$

Implement $f(x) = \sqrt{x^3} + \sin(\sqrt{x}) = (h \circ g)(x)$,
with

$$h(x) = x^3 + \sin(x) \quad \text{and} \quad g(x) = \sqrt{x}$$

```
auto generateFunction()
{
  using namespace RFFGen::CMath;

  // Chain< Sum< Pow<3> , Sin > , Sqrt >
  auto f = ( Pow<3>() + Sin() ) << Sqrt();

  return RFFGen::Finalize<decltype(f),true>(f);
}
```

Usage:

```
auto f = generateFunction();

// update function argument
f.update(1.);

// access value and derivatives
auto value = f(); // or f.d0();
auto firstDerivative  = f.d1();
auto secondDerivative = f.d2();
auto thirdDerivative  = f.d3();
```

A model for muscle tissue

$$W(F) = c\left[\exp\left(b(\bar{\iota}_1 - 3)\right) - 1\right] + d\left(\exp\left(a\left[\bar{\iota}_6 - 1\right]^2\right) - 1\right)$$

```
template <class Matrix>
auto generateFunction(const Matrix& F,
    const Matrix& M) {
  using RFFGen::CMath::Exp;
  using namespace RFFGen::LinearAlgebra;

  auto i1 = ShiftedFirstModified...<Matrix>();
  auto i6 = ShiftedThirdModified...<Matrix>(F,M);

  auto f0 = c*( ( Exp() << ( b*i1 ) ) - 1 );
  auto f1 = d*( ( Exp() << ( a*( i6^2 ) ) ) - 1 );

  ...
```

A model for muscle tissue

$$W(F) = c\left[\exp\left(b(\bar{\iota}_1 - 3)\right) - 1\right] + d\left(\exp\left(a\left[\bar{\iota}_6 - 1\right]^2\right) - 1\right)$$

```
template <class Matrix>
auto generateFunction(const Matrix& F,
    const Matrix& M) {

  ...

  auto f = ( f0 + f1 )
    << LeftCauchyGreenStrainTensor<Matrix>(F);
  return RFFGen::Finalize<decltype(f)>(f);
}
```

A model for muscle tissue

Usage:

```
// given matrices F,M,dF0,dF1,dF2
auto f = generateFunction(F,M);

// update function argument
f.update(F);

// access value and derivatives
auto value = f();   // or f.d0();
auto firstDerivative    = f.d1(dF0);
auto secondDerivative   = f.d2(dF0,dF1);
auto thirdDerivative    = f.d3(dF0,dF1,dF2);
```

More variables?

More variables?

```
template <class Arg>
struct Identity : Base {
  ...
  const Arg& d0() const noexcept
  { return x; }

  const Arg& d1(const Arg& dx) const noexcept
  { return dx; }

private:
  Arg x;
};
```

Identity $f : x \mapsto x$ with directional derivative $f'(x)\delta x = \delta x$.

Variable with id

```
template <class Arg, int id>
struct Variable : Base {
  ...
  const Arg& d0() const noexcept
  { return x; }

  const Arg& d1(const Arg& dx) const noexcept
  { return dx; }

private:
  Arg x;
};
```

Variable with id

```
template <class Arg, int id>
struct Variable : Base {
  ...
  const Arg& d0() const noexcept
  { return x; }

  template <int id1,
            std::enable_if_t< id == id1 > >
  const Arg& d1(const Arg& dx) const noexcept
  { return dx; }

private:
  Arg x;
};
```

An example with two variables

$$f(x,F) = \sqrt{x}\mathrm{tr}(F) = \sqrt{x}\,(F_{00} + F_{11} + F_{22})$$

```
template <class Mat>
auto generateFunction() {
  using namespace RFFGen::CMath;
  using namespace RFFGen::LinearAlgebra;

  auto x = RFFGen::variable<0>(1.);
  auto F = RFFGen::Variable<Mat,1>();

  auto sqrt_x = Sqrt() << x;
  auto f = ( Trace<Mat>() << F ) * sqrt_x;
  return RFFGen::Finalize<decltype(f)>(f);
}
```

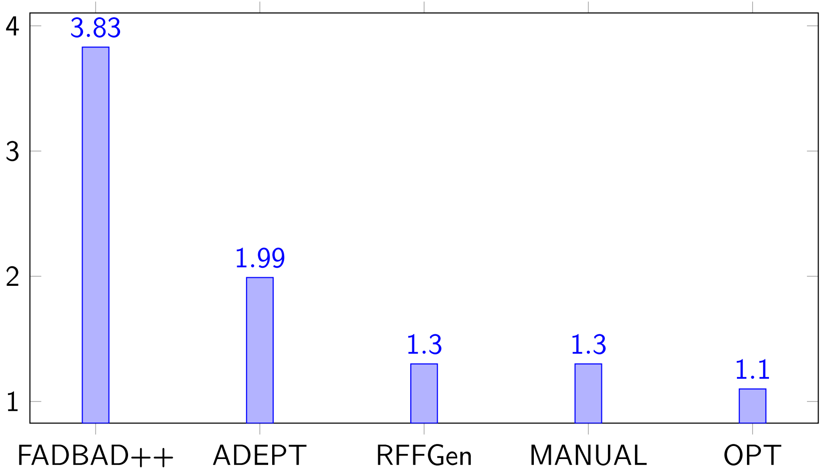
An example with two variables

Usage:

```
// given matrices F, dF and scalar x
auto f = generateFunction<Matrix>();
// update function arguments
f.template update<0>(x);
f.template update<1>(F);
// access value and derivatives
auto value       = f(); // or f.d0();
auto df_dx       = f.template d1<0>(1);
auto df_dF       = f.template d1<1>(dF);
auto ddf_dFdx    = f.template d2<1,0>(dF,1);
auto dddf_dxdxdF = f.template d3<0,0,1>(1,1,dF);
```

$$f(x) = x\left(e^{\sqrt{x}} + 1\right) + \sin\left(e^{\sqrt{x}} + 1\right)$$

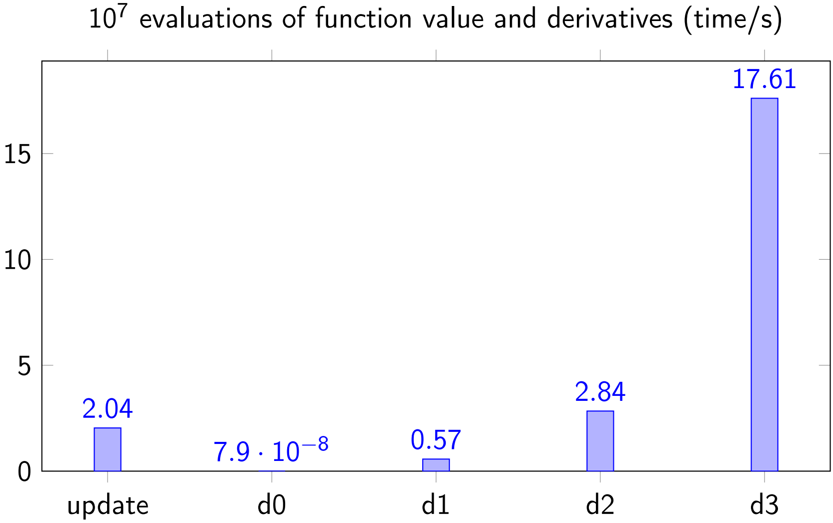
10^7 evaluations of function value and derivative (time/s)



Optimization strategies

- Simplicity
- Elimination of compile-time zeroes
- Caching
- Compiler parameters:
  - max-inline-insns-auto=5000
  - early-inlining-insns=5000
  - inline-unit-growth=100

$$W(F) = c\left[\exp\left(b(\bar{\iota}_1 - 3)\right) - 1\right] + d\left(\exp\left(a\left[\bar{\iota}_6 - 1\right]^2\right) - 1\right)$$



Interested?
→ github.com/lubkoll/RFFGen

Contact: lars.lubkoll@posteo.de