Differentiable DSP in Faust

Thomas Rushton

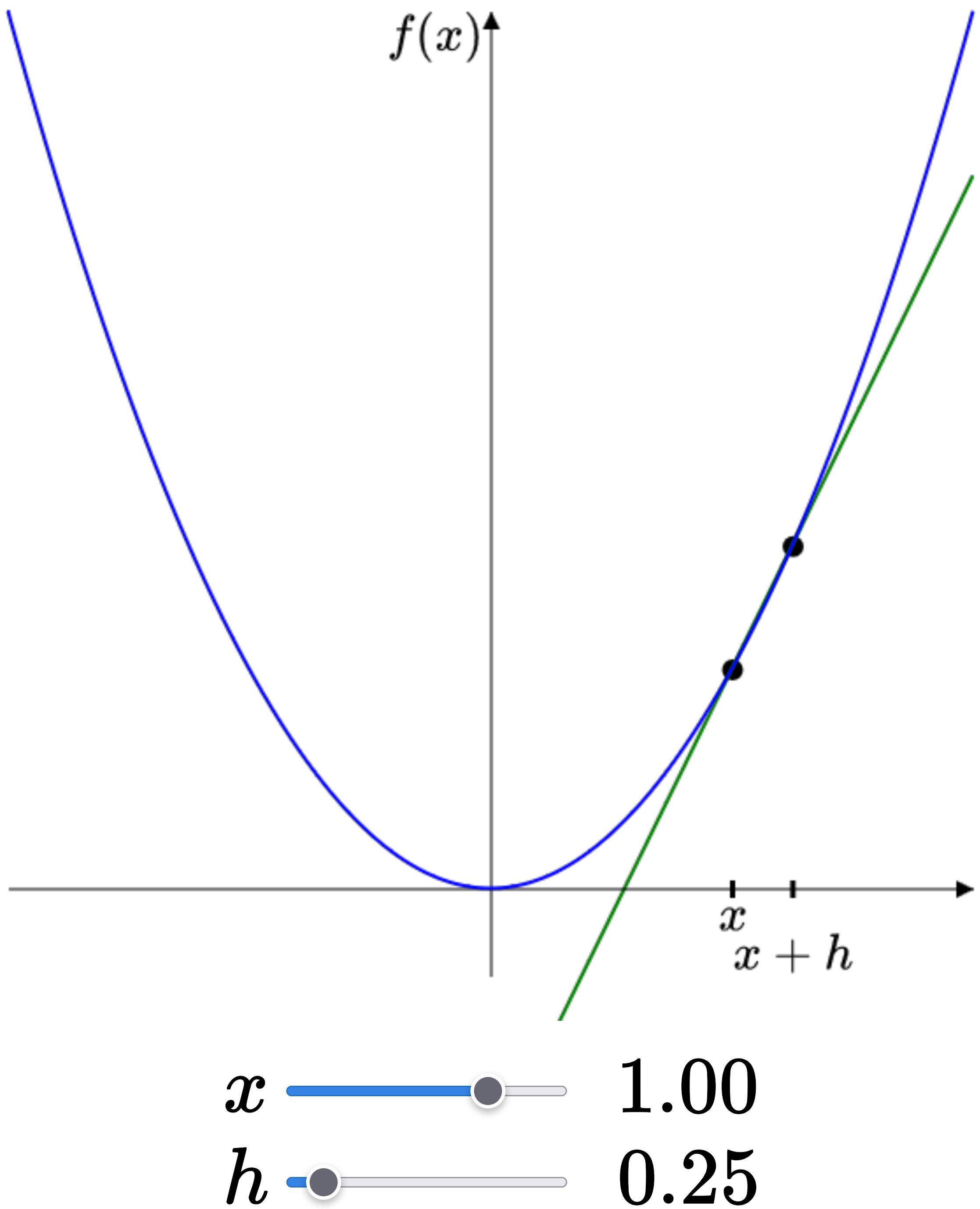
Inria, Emeraude Team — INSA Lyon

Agenda

- Differentiation
- Dual Number arithmetic
- Automatic Differentiation
- Differentiable Programming in Faust
- Gradient-based Parameter Optimisation

Differentiation

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



$$f(x) = x^2$$

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

$$\approx \frac{1.56 - 1.00}{0.25} = 2.25$$

$$\approx 2.25x$$

Infinitessimal Jest

$$\text{Let } h = \varepsilon\dot{x}$$

$\dot{x}$ is Newton's notation for the derivative.

ε is a *nilpotent symbol*: $\varepsilon \neq 0\,, \quad \varepsilon^2 = 0$

$$f'(x) = \frac{f(x + \varepsilon\dot{x}) - f(x)}{\varepsilon\dot{x}}$$

$$f(x + \varepsilon\dot{x}) = f(x) + \varepsilon\dot{x}f'(x)$$

Baydin, Pearlmutter, Radul, and Siskind. 2018. 'Automatic Differentiation in Machine Learning: A Survey'. Journal of Machine Learning Research 18.

Rall. 1986. 'The Arithmetic of Differentiation'. Mathematics Magazine 59 (5): 275–82.

$$\text{Let } f(x) = x$$

$$f(x) + \varepsilon\dot{x}f'(x) = x + \varepsilon\dot{x}$$

This is a DUAL NUMBER.

The first component is the PRIMAL expression; the second is the TANGENT.

Possible interpretation — truncated Taylor series expansion at $\varepsilon\dot{x} = 0$:

$$f(x + \varepsilon\dot{x}) = x + \varepsilon\dot{x}f'(x) + \cancel{(\varepsilon\dot{x})^2\frac{f''(x)}{2}} + \cancel{(\varepsilon\dot{x})^3\frac{f'''(x)}{6}} + \cdots$$

Dual Number Arithmetic

The basic rules of differentiation arithmetic arise from application of dual numbers.

$$(u + \varepsilon\dot{u}) + (v + \varepsilon\dot{v}) = u + v + \varepsilon(\dot{u} + \dot{v})$$

$$(u + \varepsilon\dot{u}) - (v + \varepsilon\dot{v}) = u - v + \varepsilon(\dot{u} - \dot{v})$$

$$(u + \varepsilon\dot{u}) \times (v + \varepsilon\dot{v}) = uv + \varepsilon(u\dot{v} + v\dot{u})$$

$$\begin{aligned}(u + \varepsilon\dot{u}) \div (v + \varepsilon\dot{v}) &= (u + \varepsilon\dot{u})\frac{1}{(v + \varepsilon\dot{v})} \\ &= (u + \varepsilon\dot{u})\left(\frac{1}{v} - \varepsilon\frac{\dot{v}}{v^2}\right) \\ &= \frac{u}{v} + \varepsilon\left(\frac{v\dot{u} - u\dot{v}}{v^2}\right)\end{aligned}$$

The truncated Taylor series gives us some other useful results.

$$\sin(u + \varepsilon\dot{u}) = \sin(u) + \varepsilon\dot{u}\cos(u)$$

$$\ln(u + \varepsilon\dot{u}) = \ln(u) + \varepsilon\dot{u}\frac{1}{u}$$

$$e^{(u+\varepsilon\dot{u})} = e^{u} + \varepsilon\dot{u}e^{u}$$

$$(u + \varepsilon\dot{u})^{(v+\varepsilon\dot{v})} = u^{v} + \varepsilon u^{v-1}(\dot{v}u\ln(u) + \dot{u}v)$$

Multiple applications of $f(x + \varepsilon \dot{x})$ give us the chain rule.

$$
\begin{aligned}
f(g(x + \varepsilon \dot{x})) &= f(g(x) + \varepsilon \dot{x} g'(x)) \\
&= f(w + \varepsilon \dot{w}) \ , \quad w = g(x) \ , \quad \dot{w} = \dot{x} g'(x) \\
&= f(w) + \varepsilon \dot{w} f'(w) \\
&= {\color{blue} f(g(x))} + {\color{green} \varepsilon \dot{x} g'(x) f'(g(x))}
\end{aligned}
$$

E.g. let $g(x) = \sin(x)$ and $f(v) = v^2$:

$$
\begin{aligned}
\sin^2(x + \varepsilon \dot{x}) &= \sin^2(x) + \varepsilon \dot{x} \cos(x) 2v \\
&= {\color{blue} \sin^2(x)} + {\color{green} \varepsilon 2 \sin(x) \cos(x)}
\end{aligned}
$$

A Change of Notation

$$(u + \varepsilon\dot{u}) \to \langle u, u' \rangle$$

$$UV = \langle u, u' \rangle \langle v, v' \rangle = \langle uv, uv' + vu' \rangle$$

Possible implementation:

```
class Dual {
  //...
  float primal, tangent;
  //....
  Dual operator*(Dual& d) {
    Dual result;
    result.primal = this->primal * d.primal;
    result.tangent = this->primal * d.tangent + d.primal * this->tangent;
    return result;
  }
}
```

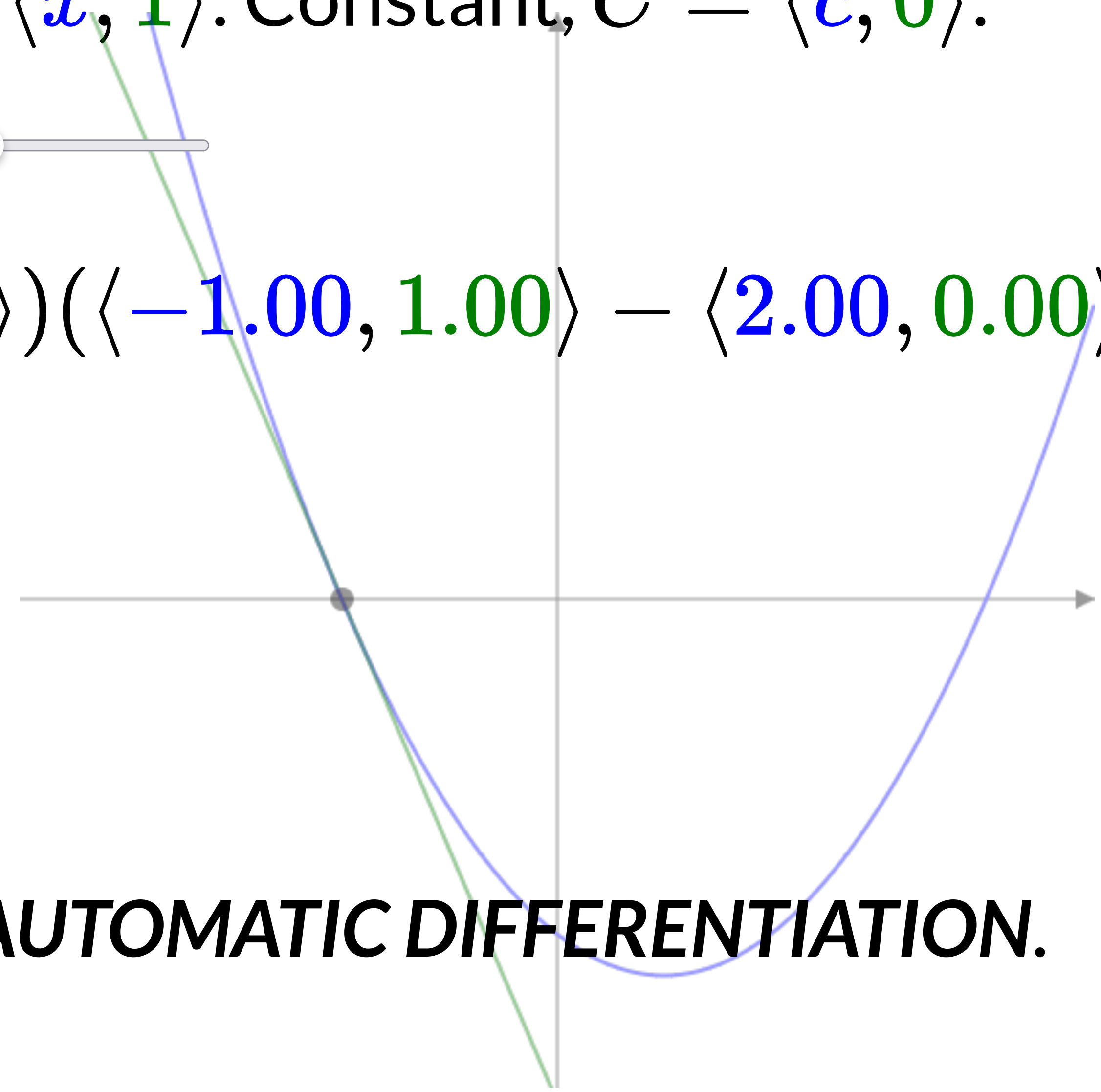
Yu and Blair. 2013. 'DNAD, a Simple Tool for Automatic Differentiation of Fortran Codes Using Dual Numbers'. Computer Physics Communications 184 (5): 1446–52.

A Numerical Example

Compute the primal and tangent of $(x+1)(x-2)$.

Independent variable, $X = \langle {\color{blue}x}, {\color{green}1} \rangle$. Constant, $C = \langle {\color{blue}c}, {\color{green}0} \rangle$.

x



$$
\begin{aligned}
&(\langle {\color{blue}-1.00}, {\color{green}1.00} \rangle + \langle {\color{blue}1.00}, {\color{green}0.00} \rangle)(\langle {\color{blue}-1.00}, {\color{green}1.00} \rangle - \langle {\color{blue}2.00}, {\color{green}0.00} \rangle) \\
=&\langle {\color{blue}0.00}, {\color{green}1.00} \rangle \langle {\color{blue}-3.00}, {\color{green}1.00} \rangle \\
=&\langle {\color{blue}0.00}, {\color{green}-3.00} \rangle
\end{aligned}
$$

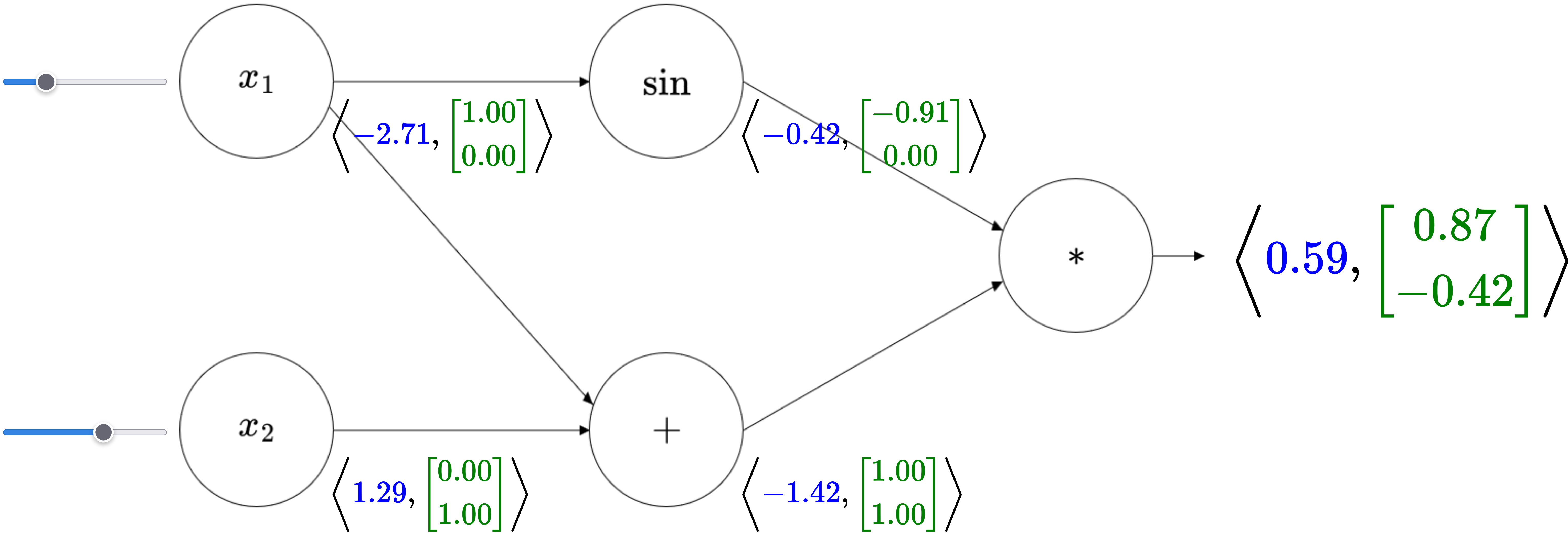
What we've achieved here is ***AUTOMATIC DIFFERENTIATION***.

Multivariate Dual Numbers

$$\langle u, u' \rangle \to \langle u, \nabla u \rangle$$

$$\nabla u = \begin{bmatrix} \frac{\partial u}{\partial x_1} \\ \vdots \\ \frac{\partial u}{\partial x_N} \end{bmatrix}$$

$$f(x_1, x_2) = \sin(x_1)(x_1 + x_2)$$



Why Would We Want To Do Any Of This?

- *End-to-end differentiability* is fundamental to **gradient-based** optimisation methods.
- Gradient descent is fundamental to contemporary approaches to machine learning.
- *Automatic Differentiation* and *Differentiable Programming* ensure end-to-end differentiability.

What Does This Have To Do With Faust?

- Many of Faust's primitive operators are trivially differentiable.
- Dual-number automatic differentiation can be implemented in Faust.
- Pattern matching can be used to *override* Faust's primitives with differentiable implementations.

Gradient-based Parameter Optimisation

Take ground truth circuit producing target output signal y, and optimisable circuit producing estimate output $\hat{y}$.

Compare y and $\hat{y}$ via a loss function.

$$\mathcal{L}(y, \hat{y})[n] = ||\hat{y}[n] - y[n]||$$

Partial derivatives of the loss function are the **gradients** in the direction that minimises its value.

Backpropagating the gradients, we can update the parameters of the optimisable circuit.

Examples

Provisos, Caveats

- Automatic Differentiation has a reverse mode too — and it may be more efficient.
- Some of Faust's primitives don't have well-defined derivatives.
- The derivative of a variable delay defies a closed-form solution.
- Pattern matching has its limitations.
- Loss taken in time-domain; only works for deterministic input, not perceptually-informed.
- Optimisation ≠ generalisation.

Thank you

https://github.com/hatchjaw/faust-ddsp