LIEDRIVERS — A Toolbox for the Efficient Computation of Lie Derivatives Based on the Object-Oriented Algorithmic Differentiation Package ADOL-C

Klaus Röbenack

Institute of Control Theory
Faculty of Electrical and Computer Engineering
Technische Universität Dresden
Germany

EOOLT'2011
4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools

Structure of the Talk

$\mathbb{V}$ ... n-dimensional vector space consisting of vectors (column vectors)
$\mathbb{V}^*$ ... dual space of $\mathbb{V}$ consisting of covectors (row vectors)

A multi-linear map

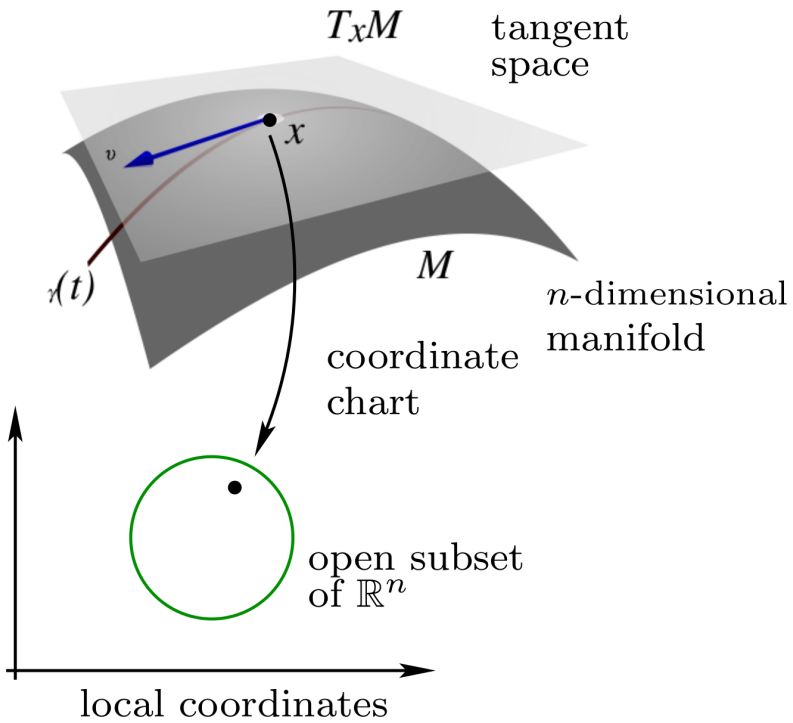
$$(\mathbb{V}^*)^p \times \mathbb{V}^q \longrightarrow \mathbb{R}$$

is called p-covariant q-contravariant tensor, or, (p, q)-tensor.
The set $\mathbb{V}_q^p$ of (p, q)-tensors on $\mathbb{V}$ is a vector space.

Isomorphisms:

$$\begin{array}{lcll} \mathbb{V}_0^0 & \cong & \mathbb{R} & \text{(scalars)} \\ \mathbb{V}_0^1 & \cong & \mathbb{V} & \text{(vectors)} \\ \mathbb{V}_1^0 & \cong & \mathbb{V}^* & \text{(covectors)} \\ \mathbb{V}_1^1 & \cong & \mathbb{V} \otimes \mathbb{V}^* & \text{(matrices)} \end{array}$$

$\mathcal{M}$ ... n-dimensional smooth manifold
$T_{\mathbf{x}}\mathcal{M}$ ... tangent space at $\mathbf{x} \in \mathcal{M}$
$T^*_{\mathbf{x}}\mathcal{M}$ ... cotangent space at $\mathbf{x} \in \mathcal{M}$



$\mathcal{M}$	...	n-dimensional smooth manifold
$T_{\mathbf{x}}\mathcal{M}$	...	tangent space at $\mathbf{x} \in \mathcal{M}$
$T^*_{\mathbf{x}}\mathcal{M}$	...	cotangent space at $\mathbf{x} \in \mathcal{M}$

A (p, q)-tensor field $\mathbf{S}$ is a map

$$\text{(point)} \qquad \mathcal{M} \ni \mathbf{x} \mapsto \mathbf{S}(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}^p_q \qquad (p, q)\text{-tensor}$$

Special cases:

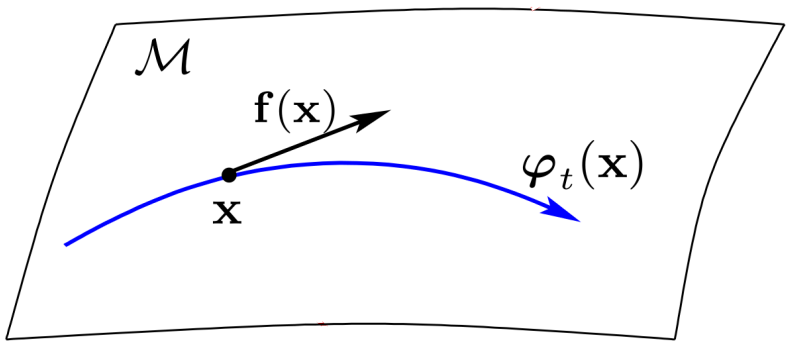
$\mathbf{S}(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}^0_0 \cong \mathbb{R}$	...	$\mathbf{S}$ is a scalar field
$\mathbf{S}(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}^1_0 \cong T_{\mathbf{x}}\mathcal{M}$	...	$\mathbf{S}$ is a vector field
$\mathbf{S}(\mathbf{x}) \in T_{\mathbf{x}}\mathcal{M}^0_1 \cong T^*_{\mathbf{x}}\mathcal{M}$	...	$\mathbf{S}$ is a covector field

The flow φ_t of a vector field $\mathbf{f}$ is the general solution of the ODE

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$$

The Lie derivative of the scalar field $h : \mathcal{M} \to \mathbb{R}$ along $\mathbf{f}$ is given by

$$L_{\mathbf{f}} h(\mathbf{x}) := \frac{\mathrm{d}}{\mathrm{d}t} h(\varphi_t(\mathbf{x}))\Big|_{t=0} = h'(\mathbf{x}) \cdot \dot{\varphi}_t(\mathbf{x})|_{t=0} = h'(\mathbf{x})\,\mathbf{f}(\mathbf{x})$$

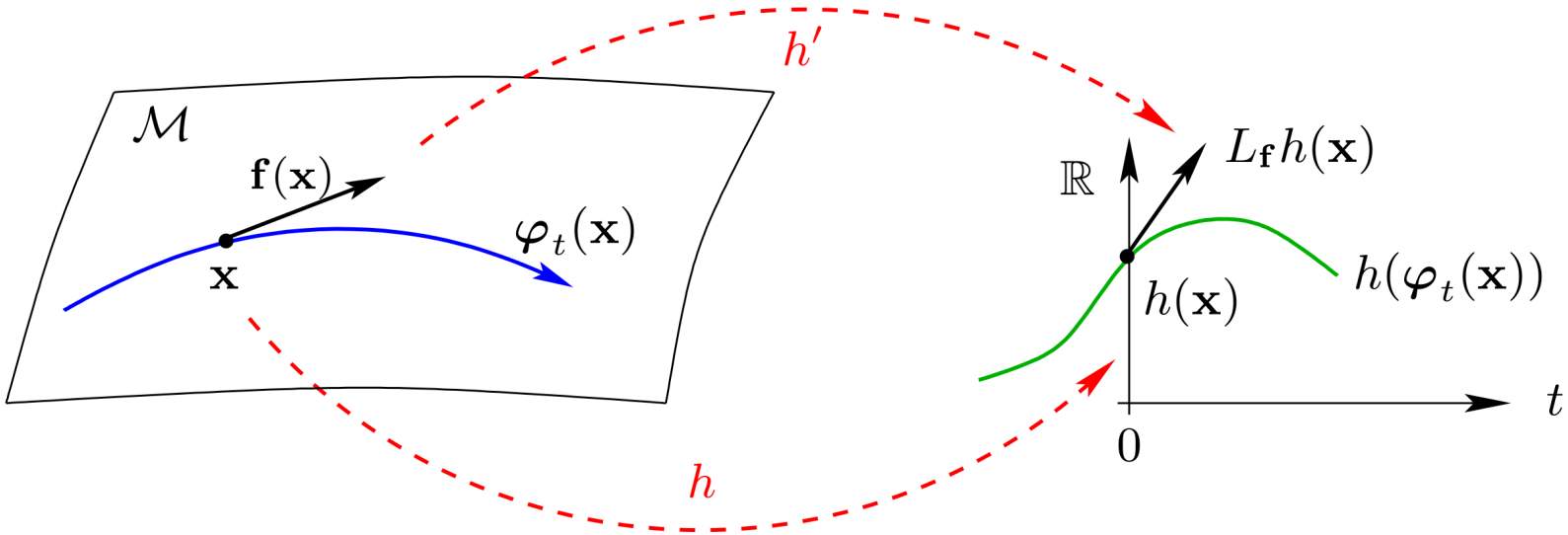


The flow φ_t of a vector field $\mathbf{f}$ is the general solution of the ODE

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t))$$

The Lie derivative of the scalar field $h : \mathcal{M} \to \mathbb{R}$ along $\mathbf{f}$ is given by

$$L_{\mathbf{f}} h(\mathbf{x}) := \frac{\mathrm{d}}{\mathrm{d}t} h(\varphi_t(\mathbf{x}))\Big|_{t=0} = h'(\mathbf{x}) \cdot \dot{\varphi}_t(\mathbf{x})|_{t=0} = h'(\mathbf{x})\,\mathbf{f}(\mathbf{x})$$



Automonous system with vector field $\mathbf{f}$ and scalar field h

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) \\ y(t) &= h(\mathbf{x}(t)) \end{aligned}$$

Lie derivative of the scalar field h along $\mathbf{f}$

$$L_{\mathbf{f}} h(\mathbf{x}) := \dot{y}(0) = \left. \frac{\mathrm{d}}{\mathrm{d}t} h(\mathbf{x}(t)) \right|_{t=0} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})$$

Higher order Lie derivatives

$$L_{\mathbf{f}}^2 h(\mathbf{x}) := \ddot{y}(0) = \frac{\partial L_{\mathbf{f}} h(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})$$

Lie series (Taylor series, coeffients: Lie derivatives)

$$y(t) = h(\varphi_t(\mathbf{x}_0)) = \sum_{k=0}^{\infty} L_{\mathbf{f}}^k h(\mathbf{x}_0) \frac{t^k}{k!}$$

Automonous system with vector field $\mathbf{f}$ and scalar field h

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) \\ y(t) &= h(\mathbf{x}(t)) \end{aligned}$$

Lie derivative of the scalar field h along $\mathbf{f}$

$$L_{\mathbf{f}} h(\mathbf{x}) := \dot{y}(0) = \frac{\mathrm{d}}{\mathrm{d}t} h(\mathbf{x}(t)) \Big|_{t=0} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})$$

Higher order Lie derivatives

$$L_{\mathbf{f}}^2 h(\mathbf{x}) := \ddot{y}(0) = \frac{\partial L_{\mathbf{f}} h(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})$$

Lie series (Taylor series, coeffients: Lie derivatives)

$$y(t) = h(\varphi_t(\mathbf{x}_0)) = \sum_{k=0}^{\infty} L_{\mathbf{f}}^k h(\mathbf{x}_0) \frac{t^k}{k!}$$

Automonous system with vector field $\mathbf{f}$ and scalar field h

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) \\ y(t) &= h(\mathbf{x}(t)) \end{aligned}$$

Lie derivative of the scalar field h along $\mathbf{f}$

$$L_{\mathbf{f}}h(\mathbf{x}) := \dot{y}(0) = \left.\frac{\mathrm{d}}{\mathrm{d}t}h(\mathbf{x}(t))\right|_{t=0} = \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}}\,\mathbf{f}(\mathbf{x})$$

Higher order Lie derivatives

$$L_{\mathbf{f}}^2h(\mathbf{x}) := \ddot{y}(0) = \frac{\partial L_{\mathbf{f}}h(\mathbf{x})}{\partial \mathbf{x}}\,\mathbf{f}(\mathbf{x})$$

Lie series (Taylor series, coeffients: Lie derivatives)

$$y(t) = h(\boldsymbol{\varphi}_t(\mathbf{x}_0)) = \sum_{k=0}^{\infty} L_{\mathbf{f}}^k h(\mathbf{x}_0)\frac{t^k}{k!}$$

Different Lie derivatives along a vector field $\mathbf{f} : \mathcal{M} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$:

- Lie derivative of a scalar field $h : \mathcal{M} \rightarrow \mathbb{R}$

$$L_{\mathbf{f}} h(\mathbf{x}) = h'(\mathbf{x}) \mathbf{f}(\mathbf{x})$$

- Lie derivative of a vector field $\mathbf{g} : \mathcal{M} \rightarrow \mathbb{R}^n$ (Lie bracket):

$$L_{\mathbf{f}} \mathbf{g}(\mathbf{x}) = ad_{\mathbf{f}} \mathbf{g}(\mathbf{x}) = [\mathbf{f}, \mathbf{g}](\mathbf{x}) = \mathbf{g}'(\mathbf{x}) \mathbf{f}(\mathbf{x}) - \mathbf{f}'(\mathbf{x}) \mathbf{g}(\mathbf{x})$$

- Lie derivative of a covector field $\boldsymbol{\omega} : \mathcal{M} \rightarrow (\mathbb{R}^n)^*$:

$$L_{\mathbf{f}} \boldsymbol{\omega}(\mathbf{x}) = \mathbf{f}^T(\mathbf{x}) \left(\frac{\partial \boldsymbol{\omega}^T}{\partial \mathbf{x}} \right)^T + \boldsymbol{\omega}(\mathbf{x}) \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}$$

Symbolic differentiation

- Efficient for lower order derivatives
- Exponential expression growth higher order derivatives
- Time-consuming computations

Numeric differentiation

- Finite differences such as

$$\frac{F(x+h)-F(x)}{h} \quad \text{or} \quad \frac{F(x+h)-F(x-h)}{2h}$$

  may result in large errors (cancellation, trunction)
- *Not applicable for higher order derivatives!*

Automatic/Algorithmic differentiation

- Systematic application of elementary differentiation rules with chain rule
- Intermediate values are floating point numbers
- No truncation errors (exact w.r.t. floating point numbers)

Symbolic differentiation

- Efficient for lower order derivatives
- Exponential expression growth higher order derivatives
- Time-consuming computations

Numeric differentiation

- Finite differences such as

$$\frac{F(x+h)-F(x)}{h} \quad \text{or} \quad \frac{F(x+h)-F(x-h)}{2h}$$

  may result in large errors (cancellation, trunction)
- *Not applicable for higher order derivatives!*

Automatic/Algorithmic differentiation

- Systematic application of elementary differentiation rules with chain rule
- Intermediate values are floating point numbers
- No truncation errors (exact w.r.t. floating point numbers)

Symbolic differentiation

- Efficient for lower order derivatives
- Exponential expression growth higher order derivatives
- Time-consuming computations

Numeric differentiation

- Finite differences such as

$$\frac{F(x+h)-F(x)}{h} \quad \text{or} \quad \frac{F(x+h)-F(x-h)}{2h}$$

  may result in large errors (cancellation, trunction)
- *Not applicable for higher order derivatives!*

Automatic/Algorithmic differentiation

- Systematic application of elementary differentiation rules with chain rule
- Intermediate values are floating point numbers
- No truncation errors (exact w.r.t. floating point numbers)

Example: $z = F(x, y) = [\sin(xy) + x] \cdot [\sin(xy) - y]$

F			
x			3.0
y			4.0
v_1	:=	$x \cdot y$	12.0
v_2	:=	$\sin(v_1)$	-0.5366
v_3	:=	$v_2 + x$	2.4634
v_4	:=	$v_2 - y$	-4.5366
v_5	:=	$v_3 \cdot v_4$	-11.1755
z	:=	v_5	-11.1755

Example: $z = F(x, y) = [\sin(xy) + x] \cdot [\sin(xy) - y]$

F				$\partial F/\partial x$
x		3.0	$\dot{x}$	1.0
y		4.0	$\dot{y}$	0.0
v_1	$:= x \cdot y$	12.0	$\dot{v}_1 := \dot{x}y + \dot{y}x$	4.0
v_2	$:= \sin(v_1)$	-0.5366	$\dot{v}_2 := \dot{v}_1 \cos(v_1)$	3.3754
v_3	$:= v_2 + x$	2.4634	$\dot{v}_3 := \dot{v}_2 + \dot{x}$	4.3754
v_4	$:= v_2 - y$	-4.5366	$\dot{v}_4 := \dot{v}_2 - \dot{y}$	3.3754
v_5	$:= v_3 \cdot v_4$	-11.1755	$\dot{v}_5 := \dot{v}_3 v_4 + \dot{v}_4 v_3$	-11.5345
z	$:= v_5$	-11.1755	$\dot{z} := \dot{v}_5$	-11.5345

Example: $z = F(x, y) = [\sin(xy) + x] \cdot [\sin(xy) - y]$

F				$\partial F/\partial x$	$\partial F/\partial y$
x		3.0	$\dot{x}$	1.0	0.0
y		4.0	$\dot{y}$	0.0	1.0
v_1	$:= x \cdot y$	12.0	$\dot{v}_1 := \dot{x}y + \dot{y}x$	4.0	3.0
v_2	$:= \sin(v_1)$	-0.5366	$\dot{v}_2 := \dot{v}_1 \cos(v_1)$	3.3754	2.5316
v_3	$:= v_2 + x$	2.4634	$\dot{v}_3 := \dot{v}_2 + \dot{x}$	4.3754	2.5316
v_4	$:= v_2 - y$	-4.5366	$\dot{v}_4 := \dot{v}_2 - \dot{y}$	3.3754	1.5316
v_5	$:= v_3 \cdot v_4$	-11.1755	$\dot{v}_5 := \dot{v}_3 v_4 + \dot{v}_4 v_3$	-11.5345	-7.7119
z	$:= v_5$	-11.1755	$\dot{z} := \dot{v}_5$	-11.5345	-7.7119

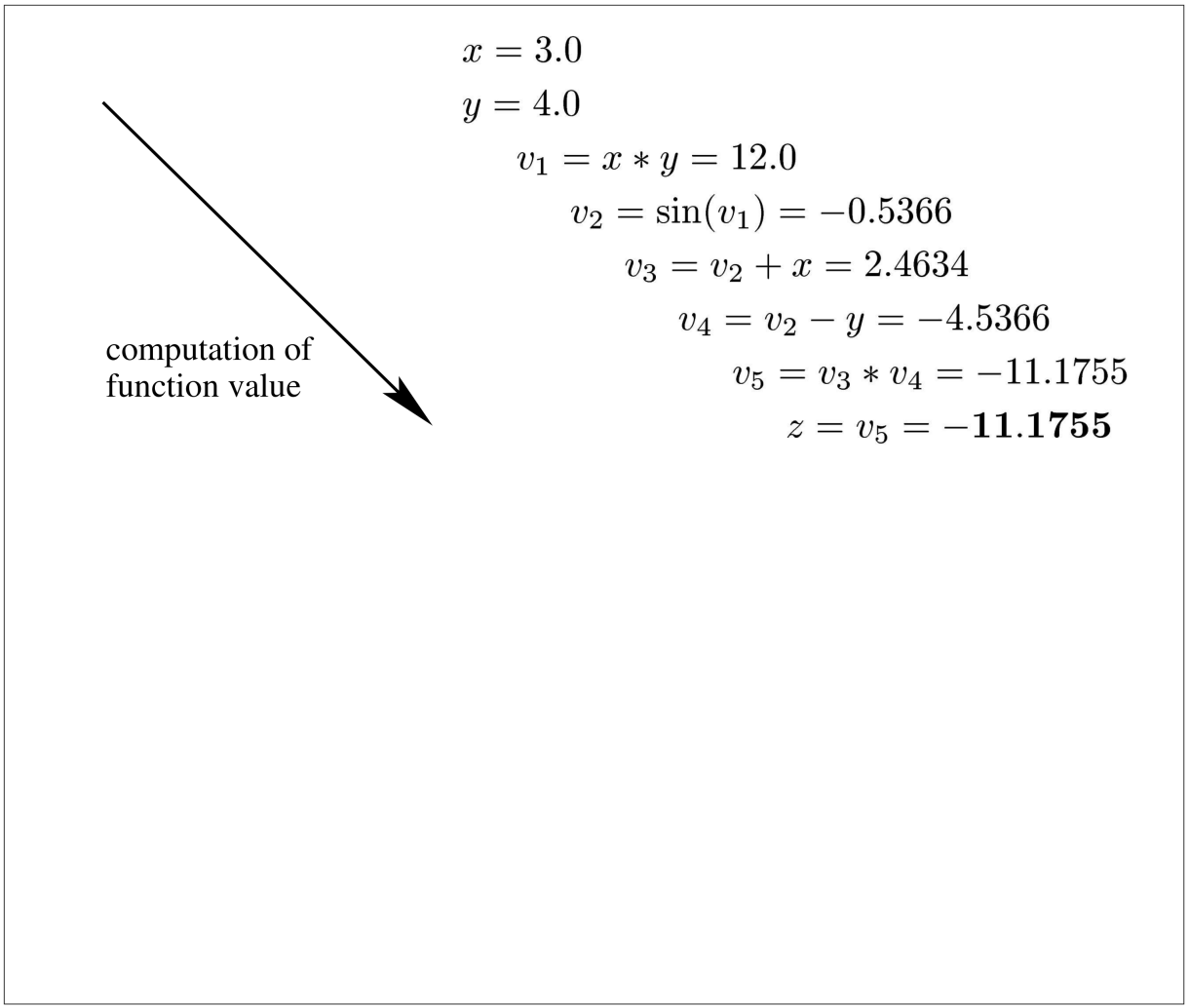
Replace floating point type `double` by a new class, e.g. `ddouble`:

```
class ddouble
{
public:
  double val; // function value
  double der; // derivative value
};
```

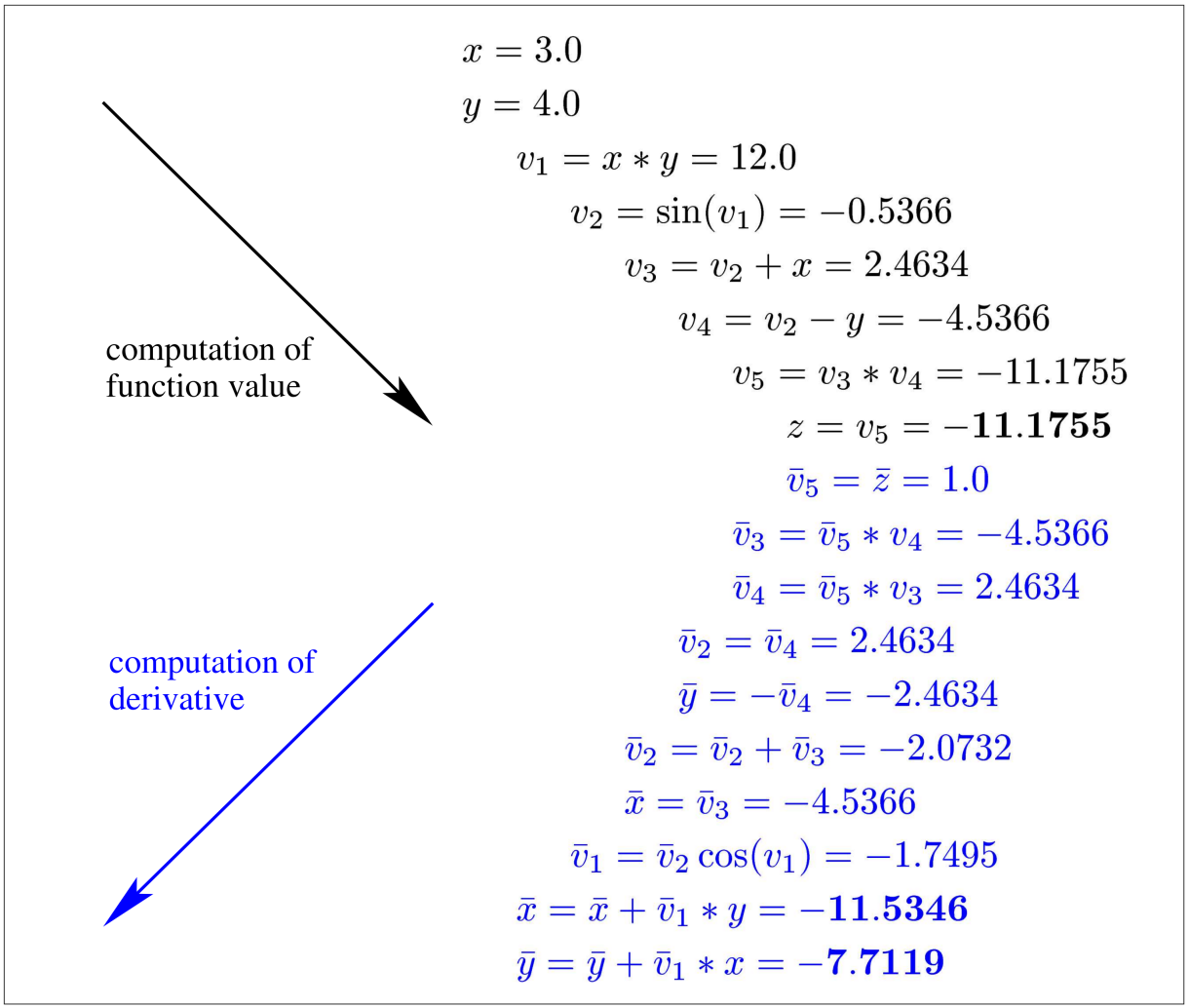
Overload all operations for additional derivative calculation:

```
ddouble operator * (ddouble x, ddouble y)
{
ddouble z;
z.val = x.val*y.val;
z.der = x.val*y.der+y.val*x.der;
}
```

Example: $z = F(x, y) = [\sin(xy) + x] \cdot [\sin(xy) - y]$



Example: $z = F(x, y) = [\sin(xy) + x] \cdot [\sin(xy) - y]$



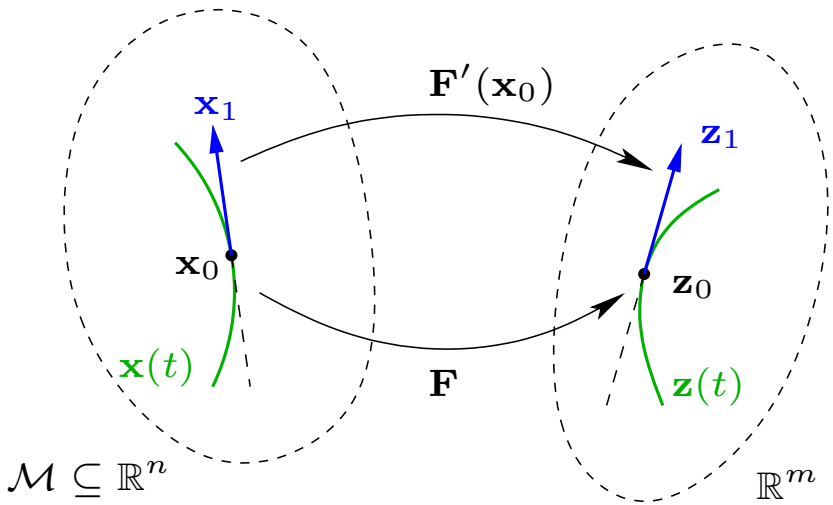
$$x = 3.0$$
$$y = 4.0$$
$$v_1 = x * y = 12.0$$
$$v_2 = \sin(v_1) = -0.5366$$
$$v_3 = v_2 + x = 2.4634$$
$$v_4 = v_2 - y = -4.5366$$
$$v_5 = v_3 * v_4 = -11.1755$$
$$z = v_5 = \mathbf{-11.1755}$$
$$\bar{v}_5 = \bar{z} = 1.0$$
$$\bar{v}_3 = \bar{v}_5 * v_4 = -4.5366$$
$$\bar{v}_4 = \bar{v}_5 * v_3 = 2.4634$$
$$\bar{v}_2 = \bar{v}_4 = 2.4634$$
$$\bar{y} = -\bar{v}_4 = -2.4634$$
$$\bar{v}_2 = \bar{v}_2 + \bar{v}_3 = -2.0732$$
$$\bar{x} = \bar{v}_3 = -4.5366$$
$$\bar{v}_1 = \bar{v}_2 \cos(v_1) = -1.7495$$
$$\bar{x} = \bar{x} + \bar{v}_1 * y = \mathbf{-11.5346}$$
$$\bar{y} = \bar{y} + \bar{v}_1 * x = \mathbf{-7.7119}$$

Smooth map $\mathbf{F} : \mathcal{M} \subseteq \mathbb{R}^n \to \mathbb{R}^m$ and truncated Taylor series

$$\begin{aligned}
\mathbf{x}(t) &= \mathbf{x}_0 + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \cdots + \mathbf{x}_d t^d + \mathcal{O}(t^{d+1}), \quad \mathbf{x}_k = \tfrac{1}{k!}\mathbf{x}^{(k)}(0) \\
\mathbf{z}(t) = \mathbf{F}(\mathbf{x}(t)) &= \mathbf{z}_0 + \mathbf{z}_1 t + \mathbf{z}_2 t^2 + \cdots + \mathbf{z}_d t^d + \mathcal{O}(t^{d+1}), \quad \mathbf{z}_k = \tfrac{1}{k!}\mathbf{z}^{(k)}(0)
\end{aligned}$$

with coefficient vectors $\mathbf{x}_0, \ldots, \mathbf{x}_d \in \mathbb{R}^n$ and $\mathbf{z}_0, \ldots, \mathbf{z}_d \in \mathbb{R}^m$ with

$$\begin{aligned}
\mathbf{z}_0 &= \mathbf{F}(\mathbf{x}_0) \\
\mathbf{z}_1 &= \mathbf{F}'(\mathbf{x}_0)\,\mathbf{x}_1 \\
\mathbf{z}_2 &= \mathbf{F}'(\mathbf{x}_0)\,\mathbf{x}_2 + \tfrac{1}{2}\,\mathbf{F}''(\mathbf{x}_0)\,\mathbf{x}_1\,\mathbf{x}_1 \\
\mathbf{z}_3 &= \mathbf{F}'(\mathbf{x}_0)\,\mathbf{x}_3 + \mathbf{F}''(\mathbf{x}_0)\,\mathbf{x}_1\,\mathbf{x}_2 \\
&\quad + \tfrac{1}{6}\,\mathbf{F}'''(\mathbf{x}_0)\,\mathbf{x}_1\,\mathbf{x}_1\,\mathbf{x}_1 \\
&\vdots \\
\mathbf{z}_k &\equiv \mathbf{z}_k\left(\mathbf{x}_0 \ldots, \mathbf{x}_k\right)
\end{aligned}$$



Storage: Replace double by a new class tdouble

```
#include <vector>
class tdouble:public std::vector<double>;
```

Calculation: Overloading elementary operations

$$z = x \pm y \quad \Rightarrow \quad z_k = x_k \pm y_k$$

$$z = x \cdot y \quad \Rightarrow \quad z_k = \sum_{i=0}^{k} x_i \cdot y_{k-i}$$

$$z = x/y \quad \Rightarrow \quad z_k = \frac{1}{y_0}\left(x_k - \sum_{i=1}^{k} y_i x_{k-i}\right) \qquad (k \geq 1)$$

$$z = e^x \quad \Rightarrow \quad z_k = \frac{1}{k}\sum_{i=0}^{k-1}(k-i) z_i x_{k-i} \qquad (k \geq 1)$$

$$z = \ln(x) \quad \Rightarrow \quad z_k = \frac{1}{x_0}\left(x_k - \frac{1}{k}\sum_{i=1}^{k-1} i z_i x_{k-i}\right) \qquad (k \geq 1)$$

Tools: ADOL-C (Germany); TADIFF, FADBAD++ (Denmark)

Storage: Replace double by a new class tdouble

```
#include <vector>
class tdouble:public std::vector<double>;
```

Calculation: Overloading elementary operations

$$z = x \pm y \quad \Rightarrow \quad z_k = x_k \pm y_k$$

$$z = x \cdot y \quad \Rightarrow \quad z_k = \sum_{i=0}^{k} x_i \cdot y_{k-i}$$

$$z = x/y \quad \Rightarrow \quad z_k = \frac{1}{y_0}\left(x_k - \sum_{i=1}^{k} y_i x_{k-i}\right) \qquad (k \geq 1)$$

$$z = e^x \quad \Rightarrow \quad z_k = \frac{1}{k}\sum_{i=0}^{k-1}(k-i) z_i x_{k-i} \qquad (k \geq 1)$$

$$z = \ln(x) \quad \Rightarrow \quad z_k = \frac{1}{x_0}\left(x_k - \frac{1}{k}\sum_{i=1}^{k-1} i z_i x_{k-i}\right) \qquad (k \geq 1)$$

Tools: ADOL-C (Germany); TADIFF, FADBAD++ (Denmark)

Storage: Replace double by a new class tdouble

```
#include <vector>
class tdouble:public std::vector<double>;
```

Calculation: Overloading elementary operations

$$z = x \pm y \quad \Rightarrow \quad z_k = x_k \pm y_k$$

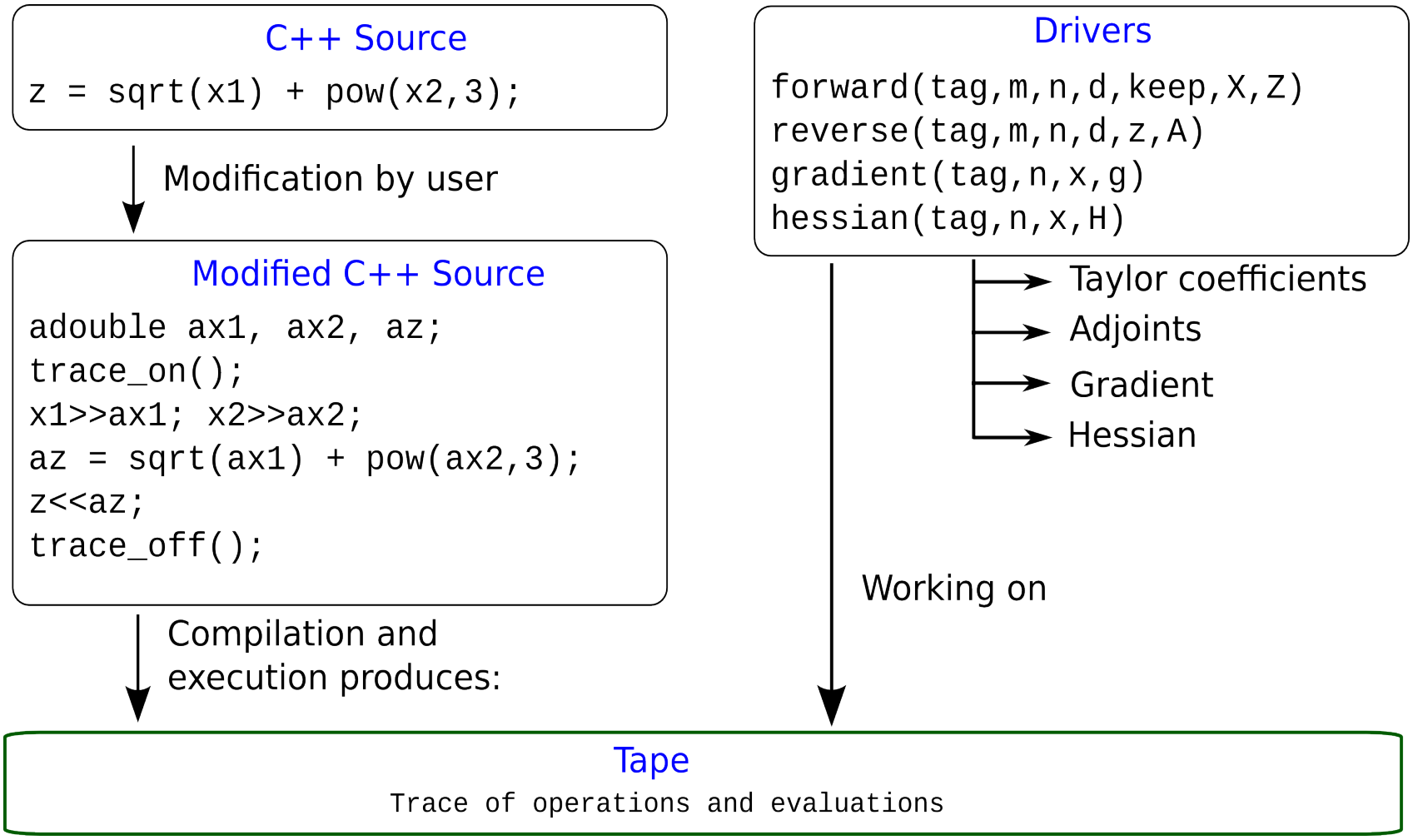
$$z = x \cdot y \quad \Rightarrow \quad z_k = \sum_{i=0}^{k} x_i \cdot y_{k-i}$$

$$z = x/y \quad \Rightarrow \quad z_k = \frac{1}{y_0}\left(x_k - \sum_{i=1}^{k} y_i x_{k-i}\right) \qquad (k \geq 1)$$

$$z = e^x \quad \Rightarrow \quad z_k = \frac{1}{k}\sum_{i=0}^{k-1}(k-i) z_i x_{k-i} \qquad (k \geq 1)$$

$$z = \ln(x) \quad \Rightarrow \quad z_k = \frac{1}{x_0}\left(x_k - \frac{1}{k}\sum_{i=1}^{k-1} i z_i x_{k-i}\right) \qquad (k \geq 1)$$

Tools: ADOL-C (Germany); TADIFF, FADBAD++ (Denmark)


C++ Source
z = sqrt(x1) + pow(x2,3);
Modification by user
Modified C++ Source
adouble ax1, ax2, az;
trace_on();
x1>>ax1; x2>>ax2;
az = sqrt(ax1) + pow(ax2,3);
z<<az;
trace_off();
Compilation and
execution produces:
Drivers
forward(tag,m,n,d,keep,X,Z)
reverse(tag,m,n,d,z,A)
gradient(tag,n,x,g)
hessian(tag,n,x,H)
Taylor coefficients
Adjoints
Gradient
Hessian
Working on
Tape
Trace of operations and evaluations

Initial value problem

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{M} \subseteq \mathbb{R}^n$$

with vector field $\mathbf{f} : \mathcal{M} \to \mathbb{R}^n$. Series expansions:

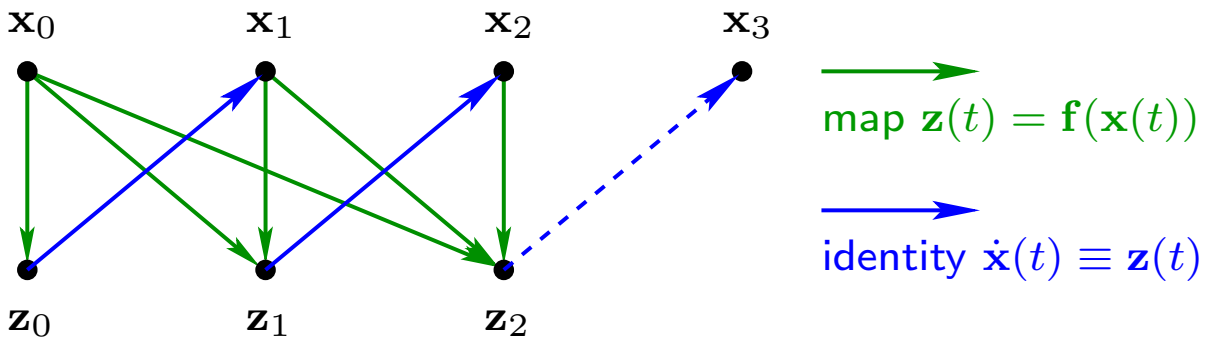
$$\begin{aligned} \text{flow:} \quad \mathbf{x}(t) &= \varphi_t(\mathbf{x}_0) &&= \mathbf{x}_0 + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \cdots + \mathbf{x}_d t^d + \mathcal{O}(t^{d+1}) \\ \text{map:} \quad \mathbf{z}(t) &= \mathbf{f}(\mathbf{x}(t)) &&= \mathbf{z}_0 + \mathbf{z}_1 t + \mathbf{z}_2 t^2 + \cdots + \mathbf{z}_d t^d + \mathcal{O}(t^{d+1}) \end{aligned}$$

Since

$$\dot{\mathbf{x}} \equiv \mathbf{z}$$

we have

$$\mathbf{x}_{k+1} = \frac{1}{k+1}\mathbf{z}_k$$



Consider the autonomous system

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)), \quad y(t) = h(\mathbf{x}(t)), \quad \mathbf{x}(0) = \mathbf{x}_0 \in \mathcal{M} \subseteq \mathbb{R}^n$$

with $\mathbf{f} : \mathcal{M} \to \mathbb{R}^n$ and $h : \mathcal{M} \to \mathbb{R}$.
The Taylor coeffients $\mathbf{x}_1, \ldots, \mathbf{x}_d \in \mathbb{R}^n$ and $y_0, \ldots, y_d \in \mathbb{R}$ can be computed with the forward mode of automatic differentiation:

$$\begin{aligned} \text{flow:} \quad & \mathbf{x}(t) = \varphi_t(\mathbf{x}_0) && = \mathbf{x}_0 + \mathbf{x}_1 t + \mathbf{x}_2 t^2 + \cdots + \mathbf{x}_d t^d + \mathcal{O}(t^{d+1}) \\ \text{output:} \quad & y(t) = h(\varphi_t(\mathbf{x}_0)) && = y_0 + y_1 t + y_2 t^2 + \cdots + y_d t^d + \mathcal{O}(t^{d+1}) \end{aligned}$$

Iterated Lie derivatives of the scalar field h along the vector field $\mathbf{f}$:

$$y(t) = h(\varphi_t(\mathbf{x}_0)) = \sum_{k=0}^{\infty} L_{\mathbf{f}}^k h(\mathbf{x}_0) \frac{t^k}{k!} \quad \Longrightarrow \quad L_{\mathbf{f}}^k h(\mathbf{x}_0) = k!\, y_k$$

C interface of LIEDRIVERS library:

```
int Lie_scalarc(Tape_F,Tape_H,n,x0,d,res)
short Tape_F;       // tape tag of vector field f
short Tape_H;       // tape tag of scalar field h
short n;            // dimension n
double x0[n];       // vector x_0
short d;            // highest degree d
double res[d+1];    // Lie derivatives
```

Also supported:

- Simulaneous computation for multiple scalar fields
- Gradients of Lie derivatives of scalar fields
- Lie derivatives of vector fields and covector fields
- C++ interface

C interface of LIEDRIVERS library:

```
int Lie_scalarc(Tape_F,Tape_H,n,x0,d,res)
short Tape_F;      // tape tag of vector field f
short Tape_H;      // tape tag of scalar field h
short n;           // dimension n
double x0[n];      // vector x_0
short d;           // highest degree d
double res[d+1];   // Lie derivatives
```

Also supported:

- Simulaneous computation for multiple scalar fields
- Gradients of Lie derivatives of scalar fields
- Lie derivatives of vector fields and covector fields
- C++ interface

We consider computational problems occurring in controller and observer design for nonlinear control systems

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{f}(\mathbf{x}(t)) + \mathbf{g}(\mathbf{x}(t))\, u \\ y(t) &= h(\mathbf{x}(t)) \end{aligned}$$

with

state	$\mathbf{x}(t) \in \mathcal{M} \subseteq \mathbb{R}^n$	(open subset)
input	$u(t) \in \mathbb{R}$	
output	$y(t) \in \mathbb{R}$	

and

$$\begin{aligned} \mathbf{f}, \mathbf{g} : \mathcal{M} &\to \mathbb{R}^n && \text{(vector fields)} \\ h : \mathcal{M} &\to \mathbb{R} && \text{(scalar field)} \end{aligned}$$

- Exact input-output linearization (Byrnes; Isidori):

  $h(\mathbf{x}),\ L_{\mathbf{f}}h(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{r}h(\mathbf{x}),\ L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x}),\quad r$ ... relative degree

- Exact input-state linearization (Byrnes; Isidori):

  $$\mathbf{g}(\mathbf{x}),\ ad_{\mathbf{f}}\mathbf{g}(\mathbf{x}),\ \ldots,\ ad_{\mathbf{f}}^{n-1}\mathbf{g}(\mathbf{x})$$

- High-gain observer (Gauthier, Hammouri, Othman):

  $$\boldsymbol{\omega}(\mathbf{x}) := h'(\mathbf{x}),\ L_{\mathbf{f}}\boldsymbol{\omega}(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{n-1}\boldsymbol{\omega}(\mathbf{x})$$

- Extended Luenberger observer (Bestle, Zeitz):

  $\mathbf{v}(\mathbf{x}),\ ad_{-\mathbf{f}}\mathbf{v}(\mathbf{x}),\ \ldots,\ ad_{-\mathbf{f}}^{n}\mathbf{v}(\mathbf{x})$ with $\langle L_{\mathbf{f}}^{k}\boldsymbol{\omega}(\mathbf{x}),\mathbf{v}(\mathbf{x})\rangle = \delta_{k,n-1}$

- Exact input-output linearization (Byrnes; Isidori):

$$h(\mathbf{x}),\ L_{\mathbf{f}}h(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{r}h(\mathbf{x}),\ L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x}), \quad r \ \ldots \text{ relative degree}$$

- Exact input-state linearization (Byrnes; Isidori):

$$\mathbf{g}(\mathbf{x}),\ ad_{\mathbf{f}}\mathbf{g}(\mathbf{x}),\ \ldots,\ ad_{\mathbf{f}}^{n-1}\mathbf{g}(\mathbf{x})$$

- High-gain observer (Gauthier, Hammouri, Othman):

$$\boldsymbol{\omega}(\mathbf{x}) := h'(\mathbf{x}),\ L_{\mathbf{f}}\boldsymbol{\omega}(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{n-1}\boldsymbol{\omega}(\mathbf{x})$$

- Extended Luenberger observer (Bestle, Zeitz):

$$\mathbf{v}(\mathbf{x}),\ ad_{-\mathbf{f}}\mathbf{v}(\mathbf{x}),\ \ldots,\ ad_{-\mathbf{f}}^{n}\mathbf{v}(\mathbf{x}) \quad \text{with} \quad \langle L_{\mathbf{f}}^{k}\boldsymbol{\omega}(\mathbf{x}), \mathbf{v}(\mathbf{x})\rangle = \delta_{k,n-1}$$

- Exact input-output linearization (Byrnes; Isidori):

$$h(\mathbf{x}),\ L_{\mathbf{f}}h(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{r}h(\mathbf{x}),\ L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x}), \quad r \ \ldots \text{ relative degree}$$

- Exact input-state linearization (Byrnes; Isidori):

$$\mathbf{g}(\mathbf{x}),\ ad_{\mathbf{f}}\mathbf{g}(\mathbf{x}),\ \ldots,\ ad_{\mathbf{f}}^{n-1}\mathbf{g}(\mathbf{x})$$

- High-gain observer (Gauthier, Hammouri, Othman):

$$\boldsymbol{\omega}(\mathbf{x}) := h'(\mathbf{x}),\ L_{\mathbf{f}}\boldsymbol{\omega}(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{n-1}\boldsymbol{\omega}(\mathbf{x})$$

- Extended Luenberger observer (Bestle, Zeitz):

$$\mathbf{v}(\mathbf{x}),\ ad_{-\mathbf{f}}\mathbf{v}(\mathbf{x}),\ \ldots,\ ad_{-\mathbf{f}}^{n}\mathbf{v}(\mathbf{x}) \quad \text{with} \quad \langle L_{\mathbf{f}}^{k}\boldsymbol{\omega}(\mathbf{x}), \mathbf{v}(\mathbf{x})\rangle = \delta_{k,n-1}$$

- Exact input-output linearization (Byrnes; Isidori):

$$h(\mathbf{x}),\ L_{\mathbf{f}}h(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{r}h(\mathbf{x}),\ L_{\mathbf{g}}L_{\mathbf{f}}^{r-1}h(\mathbf{x}),\quad r\ \ldots \text{ relative degree}$$

- Exact input-state linearization (Byrnes; Isidori):

$$\mathbf{g}(\mathbf{x}),\ ad_{\mathbf{f}}\mathbf{g}(\mathbf{x}),\ \ldots,\ ad_{\mathbf{f}}^{n-1}\mathbf{g}(\mathbf{x})$$

- High-gain observer (Gauthier, Hammouri, Othman):

$$\boldsymbol{\omega}(\mathbf{x}) := h'(\mathbf{x}),\ L_{\mathbf{f}}\boldsymbol{\omega}(\mathbf{x}),\ \ldots,\ L_{\mathbf{f}}^{n-1}\boldsymbol{\omega}(\mathbf{x})$$

- Extended Luenberger observer (Bestle, Zeitz):

$$\mathbf{v}(\mathbf{x}),\ ad_{-\mathbf{f}}\mathbf{v}(\mathbf{x}),\ \ldots,\ ad_{-\mathbf{f}}^{n}\mathbf{v}(\mathbf{x})\quad \text{with}\quad \langle L_{\mathbf{f}}^{k}\boldsymbol{\omega}(\mathbf{x}),\mathbf{v}(\mathbf{x})\rangle = \delta_{k,n-1}$$

Summary and Outlook

Summary

- Computation of typical types of Lie derivatives based on algorithmic differentiation
- Suitable for complex equation based models (branches, loops, encapsulation, ...)
- C interface for micro controller implementations

Outlook

- Native integration in ADOL-C
- Specialized functions for nonlinear control

Summary and Outlook

Summary

- Computation of typical types of Lie derivatives based on algorithmic differentiation
- Suitable for complex equation based models (branches, loops, encapsulation, ...)
- C interface for micro controller implementations

Outlook

- Native integration in ADOL-C
- Specialized functions for nonlinear control