arXiv:2601.10828v1 [math.NA] 15 Jan 2026

ADTAYL 软件包中由泰勒级数计算高阶李导数

Nedialko S. Nedialkov[1] 和 John D. Pryce[2]

[1] 加拿大安大略省汉密尔顿麦克马斯特大学
nedialk@mcmaster.ca[2] 英国卡迪夫卡迪夫大学 PryceJD1@cardiff.ac.uk

摘要。 高阶李导数在非线性系统分析中至关重要。若采用符号方法计算，随着阶数增加，求值的成本会不断上升。本文介绍一种紧凑而高效的数值方法，利用 MATLAB Adtayl 软件包计算标量场、向量场和余向量场的李导数。该方法利用了 R¨obenack 指出的一项事实：这些导数与围绕轨迹点进行泰勒展开所得表达式的泰勒系数相同（相差一个阶乘比例因子）；必要时，该表达式还包括相应的变分矩阵。龙门起重机模型的计算结果表明，与使用 MATLAB 符号数学工具箱进行符号求值相比，该方法的速度提升了多个数量级。

关键词： 李导数、泰勒级数、自动微分、非线性控制系统

1 引言

我们开发了 Adtayl [9], 一个用于泰勒级数运算的 MATLAB 数值工具包。本文介绍了该工具包在非线性动力系统中高效求值高阶李导数方面的应用；当导数阶数较高时，符号方法往往难以胜任这项任务。

此类李导数在非线性系统理论中发挥着重要作用，应用于输出微分、可观性与可控性分析、反馈线性化以及正规形构造 [4,14]。计算机代数系统 Maple [6],支持李导数的符号计算，也可以借助 MATLAB 符号数学工具箱和 SymPy [7]进行计算。然而，符号方法通常会导致表达式膨胀，随着导数阶数增加，高阶求值的计算成本也会不断上升；例如参见 [12,13]。

基于自动微分的数值方法提供了一种有效的替代方案。R¨obenack 及其合作者此前指出，利用这样一个事实可以高效计算高阶李导数：经过阶乘缩放的李导数与适当构造的表达式的泰勒系数一致，这些表达式涉及系统流，并在需要时涉及相应的变分矩阵 [10,11,12]。他们的 ADOL-C 实现相比符号方法显著提升了速度 [13]。

利用 R¨obenack [10,12], 引入的数学恒等式，我们使用泰勒级数算术编写了实现。对于标量场、向量场和余向量场的李导数，其核心计算分别只需两行 MATLAB 代码。同样的代码无需修改即可应用于这类场的函数族。这种简洁性得益于 ADTAYL 的特性和现有机制。

在龙门起重机模型上进行的数值实验展示了该方法的效率，并将其与使用 MATLAB 符号数学工具箱进行求值的效率进行了比较。

本文其余部分安排如下。第2节定义李导数并陈述计算问题。第3 节介绍 ADTAYL 中与我们实现相关的功能。第4 节介绍标量、向量和余向量场的高阶李导数计算，并将我们的方法与 LIEDRIVERS [13] 实现进行比较。第5节给出性能结果，第6节总结全文。

2 李导数：定义与问题陈述

定义 1（李导数）。
设$\mathcal{M} \subseteq \mathbb{R}^n$ 为一个开集，并设 $f\colon \mathcal{M} \to \mathbb{R}^n$ 为定义自治常微分方程的光滑向量场

$$\dot{x} = f(x). \tag{1}$$

（***a***）标量场的李导数。设 $h : \mathcal{M} \to \mathbb{R}$ 是一个光滑标量场。h 沿 f 的李导数是方向导数 $L_f h : \mathcal{M} \to \mathbb{R}$，定义为

$$L_f\, h(x) = h'(x)\, f(x), \tag{2}$$

其中 $h'(x) \in \mathbb{R}^{1\times n}$ 是 h 的行梯度。

（***b***）向量场的李导数。设 $g : \mathcal{M} \to \mathbb{R}^n$ 是一个光滑向量场。g 沿 f 的李导数是李括号

$$L_f\, g(x) = [f, g](x) = g'(x)\, f(x) - f'(x)\, g(x), \tag{3}$$

其中 $f'(x)$、$g'(x) \in \mathbb{R}^{n\times n}$ 是雅可比矩阵。

（***c***）余向量场的李导数。设 $\omega : \mathcal{M} \to (\mathbb{R}^n)^*$ 是一个光滑余向量场。ω 沿 f 的李导数定义为

$$L_f\, \omega(x) = \left[\omega'(x) f(x)\right]^\top + \omega(x) f'(x), \quad \textit{where} \quad \omega'(x) = \frac{\partial}{\partial x}\, \omega(x)^\top \in \mathbb{R}^{n\times n}. \tag{4}$$

对于给定的 M 和其上的向量场 f，映射 $X \mapsto L_f X$ 是给定类型的光滑场空间（标量 h、向量 g 或余向量 ω）到其自身的一个算子。它是线性的：对于所有这样的 X，Y 以及标量 α，β，

$$L_f\, (\alpha X + \beta Y) = \alpha\, L_f\, X + \beta\, L_f\, Y.$$

迭代李导数 L_f^k 是该算子的 k 次幂：其中 X 表示 h、g 或 ω,

$$L_f^0 X = X, \qquad L_f^k X = L_f L_f^{k-1} X, \quad \text{for } k \geq 1. \tag{5}$$

通常用 L_f^k 表示向量场的伴随算子 ad_f^k。

问题陈述。给定一个光滑向量场 f：$\mathrm{M} \to \mathbb{R}^n$、M 上的光滑场 X（标量、向量或余向量）、初始状态 $x_0 \in \mathrm{M}$ 以及整数 $p \geq 1$，计算迭代李导数

$$L_f^k X(x_0), \quad k = 0, 1, 2, \ldots, p. \tag{6}$$

我们称

$$\frac{1}{k!} L_f^k X(x_0), \qquad k \geq 0,$$

为 X 在 x_0 处沿 f 的李系数。由于这些系数在我们的理论中自然产生，我们计算李系数；实际导数 $L_f^k X(x_0)$ 随后通过乘以 $k!$ 获得。

3 Adtayl 概述

Adtayl（基于泰勒级数的自动微分）是一个纯数值 MATLAB 软件包，包含两个主要组件：

– adtayl，用于单变量泰勒级数运算的类，以及– odets，一个用于基于泰勒级数的积分、求解常微分方程（ODE）初值问题（IVP）的求解器。

adtayl 和 odets 共同提供了用于计算李系数的工具。

小节 3.1 介绍了 adtayl 中与第4节实现相关的特性。小节 3.2 介绍了 odets的相应特性。小节 3.3总结了两个组件中用于计算泰勒系数的方法。该软件包的完整说明请参见 Adtayl用户指南 [9]。

3.1 adtayl类

adtayl 类实现了 MATLAB 风格的数组，其元素是关于单个自变量的截断泰勒级数。具体而言，adtayl 数组中的每个条目都存储了标量函数（此处记为 $u(t)$）关于某个$t = t_0$ 、截至用户指定阶数 $p \geq 0$ 的截断泰勒展开,

$$u(t_0 + s) = u_0 + u_1 s + \cdots + u_p s^p + O\left(s^{p+1}\right), \qquad u_k = \frac{u^{(k)}(t_0)}{k!}.$$

也就是说，对应于 $u(t)$ 的条目存储系数 u_0，u_1，……，u_p。adtayl 数组中的所有条目都存储相同阶数的级数。

当部分或全部操作数为 adtayl 对象时，此类允许在不修改用户代码的情况下计算 MATLAB 表达式。表达式可以自由混合使用 adtayl

和大小兼容的数值数组，前提是所有 adtayl 操作数具有相同的阶数，或者其中一个操作数的阶数为零；数值数组被视为零阶 Taylor 级数。涉及 adtayl 对象的二元运算遵循 MATLAB 对大小兼容数组进行逐元素运算的标准规则。

此功能通过重载逐元素算术运算符（加、减、乘等）以及一整套初等函数（例如 exp、log、sin、asin 等）实现。这些运算符和函数均作用于截断泰勒级数，并舍弃 $> p$阶的项。此外，adtayl 还提供访问和数组操作（重塑、拼接、索引、切片）功能，用于操作存储的系数。

以下两个矩阵级运算将在第4节中使用。

设 A 和 B 为 adtayl 对象，分别表示矩阵值函数 $A(t)$ 和$B(t)$。

– 矩阵乘积 A∗B 计算乘积$A(t)B(t)$的截断泰勒展开。– 反斜杠运算符 A\B 计算 $X(t)$的截断泰勒展开，使得 $A(t)\ X(t)\ =\ B(t)$，前提是 $A(t)$的常数项为非奇异矩阵。

在这两种情况下，A 和 B 的数组维度和阶数都必须兼容。

默认情况下，adtayl 使用双精度运算。它也支持 MATLAB 的可变精度算术（vpa）。在双精度和 vpa 模式下，adtayl 均支持对所有标准函数进行实数和复数求值。

3.2 odets 求解器

odets 求解器旨在求解以下形式的初值问题

$$\dot{x} = F(t,x), \quad x(t_0) = x_0, \tag{7}$$

其中，F 由四种基本算术运算和适当的标准函数构成。此处不包括 abs、min 和 max 等不可微函数，也不包括“if”语句等条件结构。

给定一个用于计算常微分方程右端项 F 的 MATLAB 函数，odets 会使用运算符重载对该函数进行预处理，在内存中构造一个代码列表对象，用以编码 F [8] 的求值。此预处理步骤在概念上类似于 ADOL-C [3] 中的记录机制。

在每个积分步中，odets 会解读此代码列表，以计算当前时刻解的泰勒系数。随后，它通过计算所得的截断泰勒级数，并采用合适的步长来推进解。

odets 求解器也接受自治形式的初值问题常微分方程

$$\dot{x} = f(x), \qquad x(t_0) = x_0, \tag{8}$$

这是此处关注的情形。除了计算解的泰勒系数 $x(t)$（对应8），odets 还可以计算 $J(t)\ = \partial x(t)/\partial x_0$的泰勒系数，该函数满足变分方程

$$\dot{J}(t) = f'\big(x(t)\big)\, J(t), \qquad J(t_0) = I, \tag{9}$$

其中 I 是单位矩阵。

该taylcoeffs 函数。对于上述计算，我们提供了函数

```
[x,J] = taylcoeffs(f,x0,p)
```

参数 f 可以是用于计算 f 的函数句柄，也可以是根据 f 预先构建的代码列表；参数 x0 保存初始状态向量 x_0；p 为所需的展开阶数 p。

在内部，`taylcoeffs` 使用 odets 的代码列表基础架构，计算解 $x(t)$ 到 8) 的泰勒系数，直到阶数 p，并将其存储在adtayl对象 `x`中。当请求输出 `J` 时，它会同时计算 $J(t)$的泰勒系数，并将其返回到adtayl对象 `J`中。

3.3 计算泰勒系数

Adtayl 中泰勒系数的计算基于五个递推关系：四个用于基本算术运算（BAOs），+、−、×、/，以及一个用于下文所述的子常微分方程运算 ⊙。我们在此对其进行概述。有关子常微分方程方法的详细信息，请参见 [8]。

基本算术运算的递推关系。已知 $u(t)$和 $v(t)$的泰勒系数，且已知至 k阶。对于基本算术运算，第 k阶泰勒系数为

$$(u \pm v)_k = u_k \pm v_k, \tag{10}$$

$$(uv)_k = \sum_{i=0}^{k} u_i \, v_{k-i}, \tag{11}$$

$$(u/v)_k = \frac{1}{v_0}\Big(u_k - \sum_{i=0}^{k-1} v_{k-i}\,(u/v)_i\Big), \qquad v_0 \neq 0. \tag{12}$$

在（12）中，系数 $(u/v)_k$ 取决于 u_k 以及严格低阶的系数。

假设（8）的解在 $t = t_0$ 处可展开为泰勒级数，则 $(k+1)$阶泰勒系数 $x(t)$ 满足

$$x_{k+1} = \frac{1}{k+1} f_k, \tag{13}$$

其中 $x_{k+1} = \frac{x^{(k+1)}(t_0)}{(k+1)!}$，且 $f_k = \frac{1}{k!}\frac{\mathrm{d}^k}{\mathrm{d}t^k} f\big(x(t)\big)\Big|_{t=t_0}$。

如果 f仅由基本算术运算构成，那么（13）结合（10）、（11）和（12）即可构成计算系数 x_1、x_2、..... 的完整步骤。

通过子常微分方程处理标准函数。我们先给出一个例子，然后介绍一般方案。

例 *1.* 假设已知 $u(t)$ 的泰勒系数，最高阶数为 k，我们希望计算 $v(t) = \exp\big(u(t)\big)$ 至相同阶数的系数。该指数函数满足

$$\frac{\mathrm{d}v}{\mathrm{d}u} = v,$$

因此满足常微分方程

$$\dot{v} = v\dot{u}. \tag{14}$$

将（13）和（11）应用于（14）后， k对于 $k \geq 1$ 的阶系数为

$$v_k = \frac{1}{k}(v\dot{u})_{k-1} = \frac{1}{k}\sum_{i=1}^{k} i\, u_i\, v_{k-i}.$$

该递推关系以 $v_0 = \exp（u_0）$ 开始，这可以视为 $t = 0$ 处（14）的初始条件。

一般而言，设 $\psi(u)$ 是标量自变量的 m-向量函数，并令 $v = \psi(u)$。如果导数 $\psi'(u) \in \mathbb{R}^m$ 关于标量 u 可以写成如下形式

$$\psi'(u) = \varphi\big(u, \psi(u)\big), \tag{15}$$

则 $v(t)$ 满足子常微分方程

$$\dot{v} = \varphi(u, v)\,\dot{u}. \tag{16}$$

例*2*。 标准函数的子常微分方程示例：

(i) $v = \exp(u)$ has $\dot{v} = v\dot{u}$, $\varphi(u, v) = v$

(ii) $v = u^c$, (c constant) has $\dot{v} = (cv/u)\dot{u}$, $\varphi(u, v) = cv/u$

并定义 $\mathrm{cs}(u) = \begin{bmatrix} \cos(u) \\ u \end{bmatrix} \sin()$,

(iii) $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \mathrm{cs}(u)$ has $\dot{v} = \begin{bmatrix} -v_2 \\ v_1 \end{bmatrix}\dot{u}$, $\varphi(u, v) = \begin{bmatrix} -v_2 \\ v_1 \end{bmatrix}$

实际应用中， $\varphi(u,\ v)$ 仅包含基本算术运算。对于具有子常微分方程（$\psi(u)$ 的标准函数16），系数通过 $\odot$ 运算计算

$$v_k = (\varphi \odot_\psi u)_k = \begin{cases} \psi(u_0), & k = 0, \\ \dfrac{1}{k}\displaystyle\sum_{i=1}^{k} i\, u_i\, \varphi_{k-i}, & k > 0. \end{cases} \tag{17}$$

这里 φ_j 表示 $\varphi\big(u(t)\, v(t)\big)$ 的第 j 个泰勒系数，。

例*3*。 对于前一个例子中的情况 (ii)，使用（17）可计算得到系数

$$v_k = \begin{cases} {u_0}^c, & k = 0, \\ \frac{1}{k}\sum_{i=1}^{k} i\, u_i\, (cv/u)_{k-i}, & k > 0, \end{cases}$$

其中 $(cv/u)_{k-i}$ 由（12）求得。

注 *1.* 有了子常微分方程机制，实现标准函数就变得格外简洁。在 odets 中，我们用约 130 行 MATLAB 代码（在 Adtayl 中约 170 行）实现了指数函数、对数函数和幂函数，所有直接和反三角函数及双曲函数；平方根、n 次根等根函数，以及 expm1、log1p 和 atan2 等函数。

备注 *2*。我们的方法不同于 TIDES [1],ATOMFT [2], ADOL-C [3], TAYLOR [5], 和 FADBAD++ [15]等成熟系统。在这些软件包中，每种标准函数都实现了单独的系数递推关系。

泰勒系数的雅可比矩阵。写作

$$x(t) = \sum_{k=0}^{\infty} x_k \, t^k.$$

每个 x_k 都是初始 x_0 的函数。对其求导可得

$$J(t) = \frac{\partial x(t)}{\partial x_0} = \sum_{k=0}^{\infty} \frac{\partial x_k}{\partial x_0} \, t^k. \tag{18}$$

为计算（18）中的系数，我们扩展递推关系（10）、（11）、（12）和（17），以传播关于 x_0的梯度。例如，（11）可扩展为

$$\nabla(uv)_k = \sum_{i=0}^{k} \Big(\nabla u_i \, v_{k-i} + u_i \, \nabla v_{k-i} \Big).$$

输入 i 阶梯度初始化为第 i 个单位向量。

4 计算李系数

以下三个小节展示了我们如何计算标量场、向量场和余向量场在 (6) 中的李系数。第 4.4 小节说明，在每种情况下，Adtayl的实现都可以直接扩展到此类场的族，而无需修改。第 4.5小节强调了与 [13] 中方法的差异。

在下面的代码片段中，计算得到的李系数以 adtayl 对象 L 的形式返回。其数值可通过 L.gettc(k) 提取，其中 k = $k = 0,1,\ldots,p$。

4.1 标量场

这里，h 是一个标量场。设 $y(t) = h\big(x(t)\big)$，其中 $x(t)$ 满足 (8)。那么

$$y^{(k)}(t) = L_f^k \, h\big(x(t)\big), \tag{19}$$

因此， $y(t)$ 在 $t = 0$ 处的第 k 个泰勒系数为

$$y_k = \frac{y^{(k)}(0)}{k!} = \frac{1}{k!} \frac{d^k}{dt^k} h\big(x(t)\big)\Big|_{t=0} = \frac{1}{k!} L_f^k \, h(x_0). \tag{20}$$

因此， k在 x_0 处的李系数，恰好是 k的泰勒系数，其中 $h\big(x(t)\big)$ 在$t = 0$处；参见 [11]。以下 Adtayl 代码计算式 y_k （ 20 ）：

```
x  =taylcoeffs(f,x0,p); L  =
```

代码列表 1.1：标量场的李系数。

– 此处，f 是用于计算向量场 f 的函数句柄或代码列表对象。– h 是用于计算 h 的函数句柄。– adtayl 对象 x 包含 $x_0, x_1, \ldots, x_p$。– 因此，h(x) 会在 adtayl 算术中计算 h(x)，并将 x 传播到定义 h 的表达式中。由此得到的 adtayl 对象 L 包含 $y(t) = \mathrm{h}\big(x(t)\big)$ 的系数 $y_0, \ldots, y_p$，根据 (20)，这些正是所需的李系数。

4.2 向量场

此处 g 是一个向量场。令 $Z(t)$ 定义为

$$\dot{Z}(t) = -Z(t)\, f'\big(x(t)\big), \qquad Z(0) = I. \tag{21}$$

容易看出 $Z(t) = J^{-1}(t)$作为形式泰勒级数（因为 $J(0) = I$），其中 $J(t)$满足变分方程（9）。利用 [12, 式（25）], ，可得

$$\begin{aligned}\sum_{k=0}^{\infty} \frac{1}{k!}\, \mathrm{ad}_f^k\, g(x_0)\, t^k &= Z(t)\, g\big(x(t)\big) \\ &= J^{-1}(t)\, g\big(x(t)\big).\end{aligned} \tag{22}$$

因此，

Lie coefficient $\dfrac{1}{k!}\, \mathrm{ad}_f^k\, g(x_0)$ is the kth Taylor coefficient of $J^{-1}(t)\, g\big(x(t)\big)$ at $t = 0$.

在 Adtayl 中，计算这些量的方法是

```
[x,J] =taylcoeffs(f,x0,p);
```

列表 1.2：向量场的李系数。

– adtayl 对象 x 和 J 分别包含 $x(t)$ 和 $J(t)$ 的泰勒系数，阶数为p。– g(x) 是一个函数句柄，返回一个 adtayl 对象，其中包含 $g\big(x(t)\big)$ 的泰勒系数，阶数为 p。– 由于 $J(0) = I$ 是非奇异的，幂级数 $J(t)$ 可逆，而 L=J\g(x) 计算 $J^{-1}(t)\ g\big(x(t)\big)$ 的泰勒系数，阶数为 p，并将结果返回到 L 中。

4.3 余向量场

这里 $\omega(x)$ 是一个余向量场。利用 [10,第 5 节2], 中的结果可得

$$\sum_{k=0}^{\infty} \frac{1}{k!} L_f^k \, \omega(x_0) \, t^k = \omega\big(x(t)\big) \, J(t), \tag{23}$$

其中 $J(t)$ 满足（9）。因此,

$$\text{Lie coefficient } \frac{1}{k!} L_f^k \, \omega(x_0) \text{ is the } k\text{th Taylor coefficient of } \omega\big(x(t)\big) \, J(t).$$

在 Adtayl 中，计算这些量的方法是

```
[x,J] =taylcoeffs(f,x0,p); L =
```

列表 1.3：余向量场的李系数

这里，w(x) 返回一个行向量（余向量），因此 L 表示 $\omega\big(x(t)\big) \, J(t)$。

4.4 标量场、向量场和余向量场族。

无需修改，列表 1.1、1.2、1.3 即可处理场族。由于向量场以列向量的形式实现，adtayl 支持将它们并排堆叠成矩阵的一维数组。同理，由于余向量场是行向量，adtayl支持将它们垂直堆叠成矩阵的一维数组。标量场可以存放在形状任意的数组中。

标量场。一个adtayl数组可以有任意多个维度。一般来说，h(x)在列表 1.1 中可以返回一个大小为 $[s_1, s_2, 、 \ldots 、 \ s_l]$的数组，表示对由 $h(x)$组成的 $s_1 \times s_2 \times \cdots \times s_l$个独立标量场数组求值的结果。列表 1.1仍然有效，并可同时计算所有这些标量场的李系数，并且

$$\frac{1}{k!} L_f^k \, h(x_0) \quad \text{is the } s_1 \times s_2 \times \cdots \times s_l \text{ array L.gettc(k).}$$

在控制理论中，我们关注这种 h 是否为输出映射，通常将其视为列向量:

定义 2（输出映射）。设$\mathcal{M}$ 是表示动力系统状态空间的光滑流形。输出映射是一个光滑函数

$$h : \mathcal{M} \to \mathbb{R}^m$$

它将每个状态 $x \in \mathcal{M}$ 映射到一个由可观测量组成的向量 $y = h(x) \in \mathbb{R}^m$ 。向量 y称为系统输出；其分量 $h_1, 、 \ldots 、 \ h_m$ 是$\mathcal{M}$上的标量场。
因此，在这种情况下，我们按分量计算

$$L_f^k \, h(x) = \big[\, L_f^k \, h_1(x), \; L_f^k \, h_2(x), \; \ldots, \; L_f^k \, h_m(x) \, \big]^\top \in \mathbb{R}^m.$$

向量场。 假设我们有 m 光滑向量场 $g_1(x)$, $g_2(x)$, ... $g_m(x)$。设

$$g(x) = \left[\, g_1(x),\ g_2(x),\ \ldots,\ g_m(x) \,\right] \in \mathbb{R}^{n\times m}.$$

我们按列定义该矩阵上的 ad_f^k：

$$\mathrm{ad}_f^k\, g(x) = \left[\mathrm{ad}_f^k\, g_1(x),\ \mathrm{ad}_f^k\, g_2(x),\ \ldots,\ \mathrm{ad}_f^k\, g_m(x)\right], \qquad k \ge 0.$$

方程（22）对这样的 g 仍然成立。清单 1.2 中的计算依然有效，并且

$$\frac{1}{k!}\ \mathrm{ad}_f^k\, g(x_0) \quad \text{is the } n\times m \text{ matrix } \mathrm{L.gettc(k)}.$$

余向量场。假设我们有 m 光滑余向量场 $\omega_i, i = 1,$ ....., m。令

$$\omega(x) = \begin{bmatrix} \omega_1(x) \\ \omega_2(x) \\ \vdots \\ \omega_m(x) \end{bmatrix} \in \mathbb{R}^{m\times n},$$

其中每个 $\omega_i(x)$ 都是一个行向量。我们按行定义该矩阵上的 L_f^k:

$$L_f^k\, \omega(x) = \begin{bmatrix} L_f^k\, \omega_1(x) \\ L_f^k\, \omega_2(x) \\ \vdots \\ L_f^k\, \omega_m(x) \end{bmatrix}, \qquad k \ge 0.$$

方程（23）对这样的 ω 仍然成立。清单 1.3 中的计算依然有效，并且

$$\frac{1}{k!} L_f^k\, \omega(x_0) \quad \text{is the } m\times n \text{ matrix } \mathrm{L.gettc(k)}.$$

上述简洁非凡的 Adtayl 计算总结如下

场类型	Size	表达式
标量	any	h(x)
向量	$n\times m$	J\g(x)
余向量	$m\times n$	w(x)*J

4.5 与 LIEDRIVERS 的比较

我们总结了 LIEDRIVERS [13] 的 ADOL-C 实现，并指出它与 ADTAYL的两个关键区别；另请参阅 [10,11,12]。

LIEDRIVERS 通过对 f 进行前向遍历，从 ADOL-C 磁带获取 $x(t)$的系数。字段 X 也会被记录到磁带上，并通过对该磁带进行前向遍历，计算 $X(x(t))$的系数；所有运算均针对 $x(t)$的截断泰勒级数进行。

(1) 我们方法的一个关键区别在于，我们不为 X构造代码列表对象；也就是说，X “不会被记录到磁带上”。相反， $X(x(t))$ 直接通过 adtayl 算术求值。(2) 第二个区别在于雅可比矩阵系数的计算方式。在向量和余向量的情况下，LIEDRIVERS 会随后对 f进行反向遍历，以计算雅可比矩阵 $f'(x(t))$的系数 A_i，

$$f'(x(t)) = A_0 + A_1 t + \cdots + A_k t^k + \cdots,$$

通过反向模式自动微分。这些系数随后用于递推关系（24）和（25）。相比之下，我们只使用前向模式自动微分。

向量场。 LIEDRIVERS 计算 $Z(t)$ $g(x(t))$在（22）中的系数。 $Z(t)$ 的系数使用以下方法构成：

$$Z_{k+1} = -\frac{1}{k+1}\sum_{i=0}^{k} Z_i A_{k-i}, \qquad Z_0 = I. \tag{24}$$

该递推关系直接由（13）和（11）应用于（21）得到。记作

$$g(x(t)) = g_0 + g_1 t + \cdots + g_k t^k + \cdots,$$

$Z(t)$ $g(x(t)$ 的第 k 个泰勒系数可写为 $\sum_{i=0}^{k} Z_i\, g_{k-i}$，从而得到$\frac{1}{k}$ ！ $\mathrm{ad}_f^k\, g(x_0)$。

—

余向量场。 LIEDRIVERS 使用以下方法计算 $J(t)$ 的系数：

$$J_{k+1} = \frac{1}{k+1}\sum_{i=0}^{k} A_{k-i}\, J_i, \qquad J_0 = I. \tag{25}$$

这同样可由将（13）和（11）应用于（9）直接得到。同样地，记作

$$\omega(x(t)) = \omega_0 + \omega_1 t + \cdots + \omega_k t^k + \cdots,$$

可得 $\omega(x(t))\, J(t)$ 的第 k 个系数为 $\sum_{i=0}^{k} \omega_i J_{k-i}$。

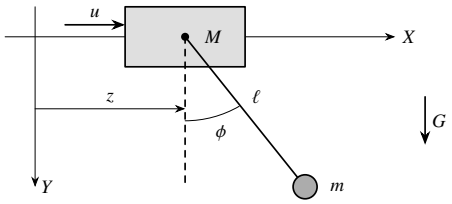


图 1：龙门起重机模型示意图。

5 性能

我们报告了该方法在以下龙门起重机模型上的性能： [13],图 1。该系统由一辆质量为 M、沿水平方向运动并承载质量为 m的负载的小车组成，负载由一根长度为 、与垂直方向夹角为 ϕ的无质量缆绳悬挂。输入 $u(t)$是施加于小车的水平力，重力 G 垂直向下作用。

运动方程（其中 z 表示水平位置）为

$$\begin{pmatrix} m+M & m\ell\cos\phi \\ m\ell\cos\phi & m\ell^2 \end{pmatrix}\begin{pmatrix} \ddot{z} \\ \ddot{\phi} \end{pmatrix} + \begin{pmatrix} -m\ell\dot{\phi}^2\sin\phi \\ m\ell G\sin\phi \end{pmatrix} = \begin{pmatrix} u \\ 0 \end{pmatrix}. \tag{26}$$

以状态向量 $x = (z, \phi, \dot{z}, \dot{\phi})$, (26) 表示的一阶系统形式如下

$$\dot{x} = f(x) + g(x)u. \tag{27}$$

向量场 f和 g以及负载位置的输出映射 $y = h(x)$ 的表达式见附录 A。从控制理论的角度来看，这具有常见的 控制仿射 形式，这意味着输入 u 仅以线性形式出现。

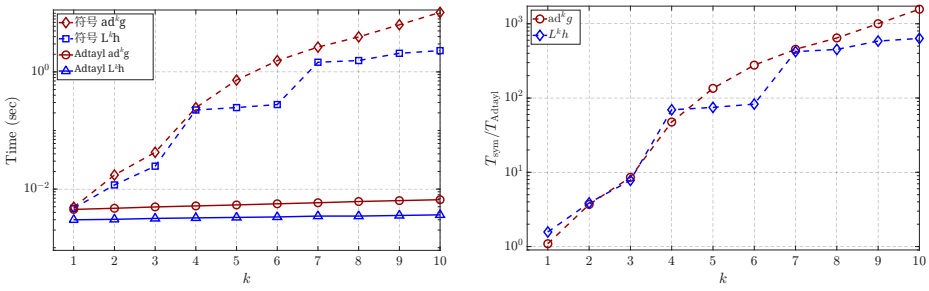
测试内容包括计算迭代李括号 $\mathrm{ad}_f^k\, g(x_0)$ 和李导数 $L_f^k h(x_0)$，阶数为 $k = 1, \ldots,$ 10。我们使用的参数值为 $M = 1, m = 1, l = 1$和 $G = 9.81$，初始状态为 $x_0 = (1,$ $0.2, -0.5, -0.4)^\top$，这些值取自 ADOL-C 发行版中的 GantryCrane.cpp 文件。

我们通过以下三种方式计算导数：(a) 使用 Adtayl 软件包；(b) 使用 MATLAB 符号数学工具箱，并对所得符号表达式求值；(c) 使用 ADOL-C 重新运行 GantryCrane.cpp。三组数值非常接近，这让我们对 Adtayl 的算法充满信心。

测试在配备 Apple M4 Max 和 64GB RAM 的 Mac Studio 上使用 MATLAB R2025b 完成。下文报告的时间为实际运行时间。对于符号方法，这些时间包括构造符号表达式及随后对其进行数值求值所需的时间。对于 Adtayl，这些时间包括为向量场 f生成代码列表，以及计算相应导数所需的时间。

图 2a 展示了计算 $\mathrm{ad}_f^k\, g(x_0)$ 和 $L_f^k\, \mathrm{h}(x_0)$ 所需时间随阶数 k 的变化。图 2b 展示了在阶数 k 时，符号计算时间与 Adtayl 计算时间之比 $T_{\mathrm{sym}}(k)/T_{\mathrm{Adtayl}}(k)$，分别对应 $\mathrm{ad}_f^k\, g$ (x_0) 和 $L_f^k \mathrm{h}(x_0)$。图 3 更清晰地展示了 Adtayl 的计时结果。

Adtayl 的耗时随 k 缓慢增长（在此范围内近似呈线性增长），而符号方法则随着 k 增大而大致呈指数增长，这是由于



(a) 时间与阶数。 (b) 符号计算时间与 Adtayl 时间之比。

图 2：使用 MATLAB 符号数学工具箱和 Adtayl 求值 $\mathrm{ad}_f^k g(x_0)$ 和 $L_f^k h(x_0)$ 的性能比较。时间以对数刻度显示。

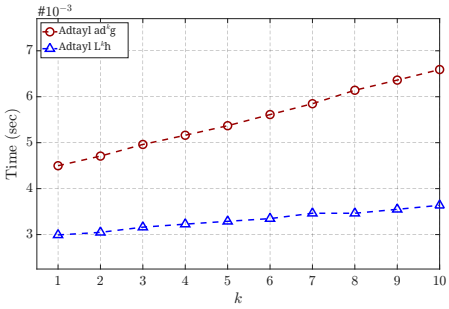


图 3：Adtayl 计算 $\mathrm{ad}_f^k g(x_0)$ 和 $L_f^k h(x_0)$ 所需的时间。

中间符号表达式的规模和复杂度不断增加所致。对于 $k = 10$，符号计算成本约为 Adtayl 方法的 1000 倍。

图4展示了用于符号表达式构造和代码列表生成的时间占总运行时间的百分比。对于符号计算，表达式构造所需的时间随着导数阶数 k 的增加而逐渐占据主导地位；对于 $L^k h$，其占比趋近于 100%；对于 $\mathrm{ad}_f^k g$，其占比在 70% 到 90% 之间波动 $k \geq 4$。相比之下，对于 Adtayl，构建代码列表所需的时间是固定的。因此，随着 k 的增加，其相对于总运行时间的占比呈线性下降，参见图3。

6 结论

6.1 软件结构

本文主要探讨如何计算各种李导数，同时也揭示了数值软件包设计方面的一些见解。在 Adtayl 基本完成后，有人建议我们将计算李导数作为一项应用，而第 4 节展示的代码精简程度令我们颇感意外。

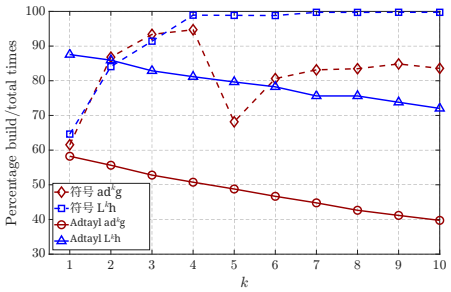


图 4：符号工具箱中用于符号表达式构造的运行时间占比，以及 Adtayl 中用于代码列表生成的运行时间占比。

这得益于我们尽可能让 odets 接受与 MATLAB 求解器语法相同的常微分方程函数 f。此外，我们还努力让 adtayl 对象在索引、拼接、切片和基本矩阵代数方面尽可能表现得像 MATLAB 原生数组。因此，f 或其代码列表既可以对数值数组求值，也可以对 adtayl 数组求值；若对后者求值，则既可以通过纯代数运算一次完成，也可以使用求解微分方程所需的多遍递推关系。

通过这种方式，Adtayl 力求迈向可微编程范式。粗略地说，这一范式旨在将导数和微分——无论是普通微分还是偏微分，无论是相对于变量、参数还是初始条件——变成计算平台的一等对象和运算符。

Adtayl之所以未能完全做到这一点，是因为各条代码列表指令都是标量。即使是简单的常微分方程 $\dot{x} = 2x$，其中 n为向量 x——即 n 相互独立的方程 $\dot{x}_i = 2x_i$——也需要将 $2x$ 转化为代码中针对 f （8）的循环。将代码列表“向量化”能让Adtayl进一步迈向可微编程，但这需要付出巨大努力。

6.2 性能与数值计算

Adtayl 中阶数的运行时间扩展Adtayl。如果计算最高至k 阶的导数所需成本主要来自算术运算（包括数组索引），则总运行时间预计会随 k呈二次增长。然而，对于此处考察的阶数范围（$k \le 10$），观测到的运行时间随 k近似线性增长。

代码列表每个阶数执行一次。对于此处考虑的 k 范围，性能分析表明，开销主要来自与阶数无关的部分，包括用于解码每个代码列表条目的数组访问和标量赋值（例如，加载操作码和操作数索引）、用于分派操作码执行的分支语句，以及特殊情况检查。尽管算法还包含卷积型运算（11）、（12）和（17），其开销与 k成正比，但当 k 较小时，这些运算的循环仍然很短，对总耗时的影响也很小。当 k 不超过 10 时，这些卷积的开销呈线性增长；只有在阶数高得多时，二次增长趋势才会显现（但仍不明显）——在双精度下，$k \gtrsim 50$。

数值误差。对于数值误差，我们采用 MATLAB vpa 以 40 位精度计算出的解作为参考解。在双精度下，我们发现 Adtayl 的（范数相对）误差在 $L_f^k h$ 从 1 增至 10 的过程中仅略有增长——从接近舍入误差增至其四倍。$\mathrm{Ad}_f^k g$ 误差增长得更多，约为原来的 200 倍。

对于 LIEDRIVERS，$L_f^k h$ 在此 k 范围内的误差同样出色。然而，我们观察到，其针对 $\mathrm{ad}_f^k g$ 的误差迅速增大了几个数量级。由于两者的区别在于后者使用了雅可比矩阵，而前者没有，因此我们想知道 ADOL-C 中用于计算该雅可比矩阵的方法是否存在问题。

6.3 其他工作

根据 [10], 中所述的理论，我们可以使用 *Adtayl* 简洁地计算可观测性矩阵，代码与上述计算李导数的代码类似。我们计划在今后的工作中报告这一内容。

致谢。 第一作者感谢加拿大自然科学与工程研究委员会（NSERC）项目 RGPIN-2019-07054 的资助。

门式起重机： f、 g和h 表达式

With

$$x = \left(x_1, x_2, x_3, x_4\right)^\top = \left(z, \phi, \dot{z}, \dot{\phi}\right)^\top,$$

其中， z 表示小车质心的位置， ϕ 表示缆绳与垂直方向的夹角，而$(\dot{z},\ \dot{\phi})$ 为对应的速度，式（ f 和 g （27）中的

$$f(x) = \begin{pmatrix} x_3 \\ x_4 \\ \dfrac{m\ell x_4^2 \sin x_2 + mG \sin x_2 \cos x_2}{m \sin^2 x_2 + M} \\ -\dfrac{m\ell x_4^2 \sin x_2 \cos x_2 + (m+M) G \sin x_2}{\ell\left(m \sin^2 x_2 + M\right)} \end{pmatrix} \quad \text{and} \quad g(x) = \begin{pmatrix} 0 \\ 0 \\ \dfrac{1}{m \sin^2 x_2 + M} \\ \dfrac{-\cos x_2}{\ell\left(m \sin^2 x_2 + M\right)} \end{pmatrix}$$

输出映射$h(x) = \left(l \sin x_2 + x_1,\ l \cos x_2\right)^\top$给出负载的笛卡尔坐标。

参考文献

1. Abad, A., Barrio, R., Blesa, F., Rodr´ıguez, M.: 算法924：TIDES，一种用于微分方程的泰勒级数积分器。ACM数学软件汇刊 **39**(1), 1–28 (2012). doi：10.1145/2382585.2382590

2. Chang, Y., Corliss, G.F.：ATOMFT：使用泰勒级数求解常微分方程和微分代数方程。计算数学及其应用 28, 209 6 3 233 1994 10 1016 0898 1221 94 00193 755 – ( ). doi: . / - ( ) - . Griewank, A., Juedes, J., Utke, J.：算法：ADOL-C：用于对以C/C++编写的算法进行自动微分的软件包。ACM数学软件汇刊 22(2), 131–167 (1996). doi:10.1145/229473.229474 4. Isidori, A.：非线性控制系统。施普林格 (1995) 5. Jorba, A., Zou, M.：一种利用高阶泰勒方法对常微分方程进行数值积分的软件包。Experimental Mathematics 14(1), 99–117 (2005). doi:10.1080/10586458.2005.10128904 6. Maplesoft, 滑铁卢, 安大略省, 加拿大：微分几何软件包 (2024)。URL https://www.maplesoft.com/support/help/Maple/view.aspx?path= DifferentialGeometry。用于符号微分几何的Maple文档，包括李导数的计算 7. Meurer, A., Smith, C.P., Paprocki, M., ˇCert´ık, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., 等：SymPy：Python符号计算。PeerJ Computer Science 3, 103 8 103 2017 10 7717 e ( ). doi: . /peerj-cs. . Nedialkov, N.S., Pryce, J.D.：子常微分方程可简化常微分方程的泰勒级数算法。SIAM科学计算杂志 47(5), A2746–A2773 (2025). doi:10.1137/24M1716161 9. Pryce, J.D., Nedialkov, N.S.：ADTAYL用户指南 (2025)。编写中。将在arxiv.org上发布 10. R¨obenack, K.：利用自动微分计算非线性控制器与观测器设计所需的张量场李导数。载于：PAMM：应用数学与力学论文集，卷

5, 181 184 2005 10 1002 200510069

11 ¨

pp. – ( )。doi: . /pamm. . Robenack, K., Gerbet, D.：借助自动微分和高增益微分器实现状态估计。优化方法与软件 pp. 1–16 (2024). doi:10.1080/10556788.2024.2320737 12. R¨obenack, K., Reinschke, K.J.：基于自动微分计算李导数与李括号。应用数学与力学杂志 84(2), 114–123 (2004). doi:10.1002/zamm.200320077 13. R¨obenack, K., Winkler, J., Wang, S.：LIEDRIVERS——一种基于面向对象的算法微分软件包ADOL-C高效计算李导数的工具箱。载于：基于方程的面向对象建模语言与工具国际研讨会 (2011)。URL https://api.semanticscholar.org/CorpusID:14490038 14. Sontag, E.D.：数学控制理论：确定性有限维系统。施普林格 (1998) 15. Stauning, O., Bendtsen, C.：FADBAD网页 (2003)。URL http://uning.dk/ fadbad.html。访问日期：2024-12-01