*Acta Physica Sinica*

# 自动微分以及它在物理模拟中的应用*

刘金国 [1)]　许开来 [2)]

1) (哈佛大学物理系, 坎布里奇　02138)

2) (斯坦佛大学, 斯坦佛　94305)

自动微分是利用计算机自动化求导的技术，最近几十年因为被用于机器学习研究而被很多人了解。如今越来越多的物理学者意识到高效，高效的，自动化的求导可以对很多物理问题的求解提供新的思路。其中自动微分在物理模拟问题中的应用尤为重要且具有挑战性。本文介绍如何将自动微分技术运用到物理模拟的求导中，介绍了共轭态法，前向自动微分，后向自动微分以及可逆计算自动微分的基本概念，并横向对比了它们在物理模拟中的优势和劣势。

关键词: 自动微分，科学计算，可逆计算，treeverse，物理模拟

**PACS:** 02.60.Pn, 02.30.Jr, 91.30.−f

## 1 引　言

自动微分是指自动获取一个目标计算机程序导数的技术。很多人了解它是因为它在机器学习中的成功应用，人们可以用它优化带有千亿参数的神经元网络 [Rosset(2019)]。与很多人印象不同的是，自动微分其实是个很古老的技术。Nolan 曾在他 1953 年的博士论文中就提出过 [Nolan(1953)] 自动化求导计算机程序的构想，而最近十几年，自动微分在科学计算中的应用越来越广泛并拓展了人们可解决问题的范畴。一个典型的例子是变分蒙特卡洛算法。过去，人们会将变分基态假设为 Gutzwiller 波函数 [Gutzwiller(1963)] 这样具有良好解析性质的函数。直到 2017 年，Carleo 等人 [Carleo and Troyer(2017), Deng *et al.*(2017)Deng, Li, and Das Sa

---

*

† 通信作者. E-mail: jinguoliu@g.harvard.edu

把机器学习中的一些变分模型限制波尔兹曼机（RBM）带入到了大家的视野。由于 RBM 的解析性质很好，计算所需的求导过程可以通过解析推导，所以当时人们并没有强烈的利用计算机辅助求导的动机。但受此启发，大家意识到不光是 RBM 这样解析性质好的函数，任何机器学习中的神经元网络也可以被用做波函数的猜测形式 [Cai and Liu(2018)]。而且如果可以用流行的机器学习框架，比如脸书公司开发的 PyTorch 和谷歌公司开发的 TesnorFlow 来编写生成波函数的代码，人们可以避免手动推导导数的麻烦，从而可以把波函数的表达变得更加自由。而现在，随着人们对微分的认知更加深刻，波函数不光可以表达为以张量运算为主体的神经元网络，还可以是几乎任意的计算机程序。除了这个例子，科学家们利用方便的，自动化的计算机辅助求导还解决了很多包括量子模拟 [Luo *et al.*(2019)Luo, Liu, Zhang, and Wang]，张量网络 [Liao *et al.*(2019)Liao, Liu, Wang, and Xiang]，组合优化 [Liu *et al.*(2020)Liu, Wang, and Zhang] 等领域中的问题。甚至是一些非解析的蒙特卡洛抽样过程，人们也设计出了一些办法对其自动化的求导 [Zhang *et al.*(2019)Z

虽然有这么多成功的例子，但当人们把自动微分技术应用到物理模拟过程，比如电路仿真，海洋学 [Heimbach *et al.*(2
和地震学 [Symes(2007), Zhu *et al.*(2020)Zhu, Xu, Darve, and Beroza] 等，常见的机器学习库经常无法直接胜任。一方面现有的机器学习框架中的自动微分需要存储程序每一步计算的部分状态以便在后向传播过程中取出用于计算局域梯度。而另一方面，物理模拟过程经常包含物理量对时间的积分，积分步长很小而步骤数很多，导致记录中间状态对空间的需求巨大。为了解决自动微分对空间的需求，人们可以假设在较短时间内可逆来回溯状态，也被称为共轭态法 [Plessix(2006), Chen *et al.*(2018)Chen, Rubanova, Bettencourt, and Duvenaud]。事实上，除了 Leap frog 积分器在时间反演不变的哈密顿量问题中可以做到严格时间反演对称，大多数问题中的积分器并不能保证可逆性，所以共轭态法往往存在一定的由积分器带来的系统性误差。后来，有人把机器学习中的 treeverse 算法带入到了物理模拟的状态回溯中，使得自动微分不再有积分器不可逆带来的系统误差。同时，treeverse 算法可以在对数时间和空间下做到状态的严格回溯。本文提出了第三种节省空间的方案，那就是利用可逆编程的方案利用 Bennett 算法来进行状态回溯，并横向对比了共轭态 (Adjoint-State) 方法，前向自动微分以及基于 treeverse 算法和可逆计算的后向自动微分在处理物理模拟问题中的优劣。

章节 2 介绍了共轭态法和自动微分的基本原理。章节 3 介绍了基于检查点和可逆编程的两种后向自动微分的基础理论，尤其两者如何权衡程序的运行时间和空间。章节 4 介绍了不同自动微分技术在地震波模拟过程中的应用。

## 2 对物理模拟的自动微分方法

考虑一个 ODE 方程

$$\frac{ds}{dt} = f(s, t, \theta)$$

其中 $s$ 为状态，$t$ 为时间，$\theta$ 为控制参数。以及一个 ODE 方程求解器

$$s_n = \text{ODESolve}(s_0, f, t_0, t_n, \theta)$$

其中 $t_0$ 和 $t_n$ 分别为初始和末了状态，$s_0$ 为初始状态。这个 ODE 求解器在求解过程中会把时间离散化，作 $n$ 步叠代过程。最后我们还可以定义一个损失函数 $\mathcal{L} = \text{loss}(s_n)$。自动微分的目标则是求解损失量对参数的导数 $\frac{\partial \mathcal{L}}{\partial s_0}$, $\frac{\partial \mathcal{L}}{\partial \theta}$, $\frac{\partial \mathcal{L}}{\partial t_0}$ 和 $\frac{\partial \mathcal{L}}{\partial t_n}$。

### 2.1 共轭态方法

共轭态方法 [Plessix(2006), Chen *et al.*(2018)Chen, Rubanova, Bettencourt, and Duvenaud] 是专门针对积分过程反向传播的传统方法。在研究中，人们发现积分过程的导数的反向传播同样是一个积分过程，只不过方向相反。于是人们通过构造一个可以同时更新原函数和导数的拓展函数，以对拓展函数的逆向积分的形式完成导数的计算，如算法 1所示。

**算法 1:** 共轭态法

**输入:** 动力学参数 $\theta$，开始时间 $t_0$，结束时间 $t_n$，末态 $s_n$，以及需要回传的导数 $\partial\mathcal{L}/\partial s_n$

**输出:** $\frac{\partial \mathcal{L}}{\partial s_0}$, $\frac{\partial \mathcal{L}}{\partial \theta}$, $\frac{\partial \mathcal{L}}{\partial t_0}$, $\frac{\partial \mathcal{L}}{\partial t_n}$

$\frac{\partial \mathcal{L}}{\partial t_n} = \frac{\partial \mathcal{L}}{\partial s_n}^T f(s_n, t_n, \theta)$ # 计算损失函数对终了时间的导数

**function** aug_dynamics([$s$, $a$, -, -], $t$, $\theta$) # 定义拓展动力学函数

  $s' = f(s, t, \theta)$

  **return** [$s'$, $-a^T \frac{\partial s'}{\partial s}$, $-a^T \frac{\partial s'}{\partial \theta}$, $-a^T \frac{\partial s'}{\partial t}$]

**end**

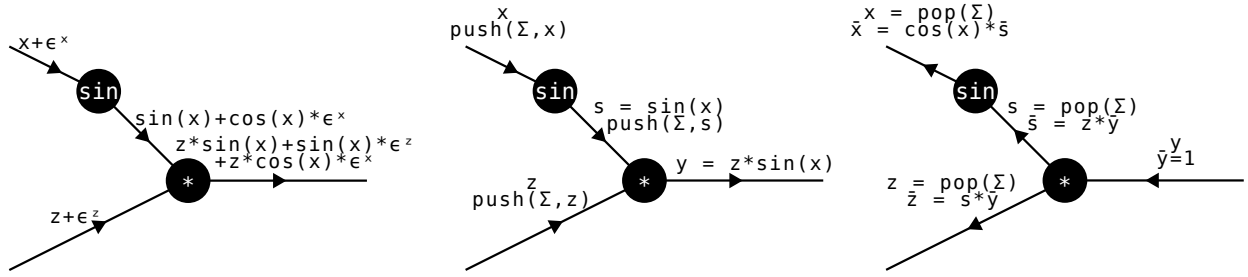
$S_0 = [s_n, \frac{\partial \mathcal{L}}{\partial s_n}, 0, -\frac{\partial \mathcal{L}}{\partial t_n}]$ # 计算拓展动力学函数的初始状态

$[s_0, \frac{\partial \mathcal{L}}{\partial s_0}, \frac{\partial \mathcal{L}}{\partial \theta}, \frac{\partial \mathcal{L}}{\partial t_0}] = \text{ODESolve}(S_0, \text{aug\_dynamics}, t_n, t_0, \theta)$ # 对拓展动力学反向积分

该算法的描述来自文献 [Chen *et al.*(2018)Chen, Rubanova, Bettencourt, and Duvenaud]，其中可以找到详细的推导过程，这里对原算法中的符号做了替换以方便读者理解。

### 2.2 前向自动微分

自动微分可以分为两大类，分别是前向传播（Forward propagation）[Wengert(1964)] 和后向传播（Backward propagation） [Boltyanski *et al.*(1960)Boltyanski, Gamkrelidze, Mishchenko, and Pontryagin]。




(a) 前向自动微分　　(b) 后向自动微分中的前向计算　　(c) 后向自动微分中的梯度反向传播

图 1: 利用自动微分计算对于计算过程 $y = z * \sin(x)$ 的导数 $\overline{x} \equiv \frac{\partial y}{\partial x}$ 和 $\overline{z} \equiv \frac{\partial y}{\partial z}$。其中圆圈代表函数，线代表变量，箭头代表运算方向，$\Sigma$ 是一个用于缓存中间计算结果的全局堆栈而 `push` 和 `pop` 分别代表了对该堆栈的入栈和出栈操作。

前向自动微分和数学中的链式法则推导导数的方法极为相似。它通修改程序运算中的变量，使之携带若干无穷小量 $\epsilon^i$，并通过这个无穷小量的运算规则完成对程序的求导。当函数 $f$ 作用于一个标量，它的运算规则变为

$$f(x + \sum_{i=1}^{k} y_i \epsilon^i) = f(x) + \frac{\partial f(x)}{\partial x} \sum_{i=1}^{k} y_i \epsilon^i \tag{1}$$

这里的上标 $i$ 代表它对应第 $i$ 个变量。容易验证这个无穷小量的运算规则与数学分析中的无穷小量并无差别。程序中仅记录一阶小量的系数，因此无穷小量满足运算规则 $\epsilon^i \epsilon^j = 0$。值得注意的是，前向自动微分的计算时间随着需求导的变量的数目线性增长。因为随着变量数目的增加，$\epsilon^i$ 的数目线性增加，函数的操作数也随之线性增加。图 1 (a) 所示的是对函数 $y = z * \sin(x)$ 的前向自动微分，这个函数仅仅包含两步运算 sin 与 $*$。程序中每个变量都记录了三个域，分别是（变量本身，$\epsilon^x$ 的系数，$\epsilon^z$）的系数，每当经过一个函数便更新数值和两个无穷小量的系数，因此整个函数的运行空间和一次求导的变量的个数也成正比。但在实践中，由于内存大小有限且需要求导变量较多的情况下，一般不会一次性的对所有变量都求导。而是重复运行多次计算过程，一次只处理若干个变量的导数这样分批次求导。例子中这样简单的表达式对于人类来说处理起来也毫无难度，但真实的程序可能会包含数以亿计的这样的基础操作，虽然结果依然是解析的，但是人们很难通过人力得到解析的导数表达式。然而计算机恰恰很擅长这样繁琐但是规则简单的任务。

## 2.3 后向自动微分

图 (b) 和 (c) 展示了后向自动微分的前向过程和后向过程，前向过程中函数除了运算本身，还会将部分中间结果放入一个全局堆栈 $\Sigma$ 中，而后向过程将这些缓存的结果出栈并利用如下递推公式计算导数

$$\overline{x} = \frac{\partial f(x)}{\partial x} \overline{y} \tag{2}$$

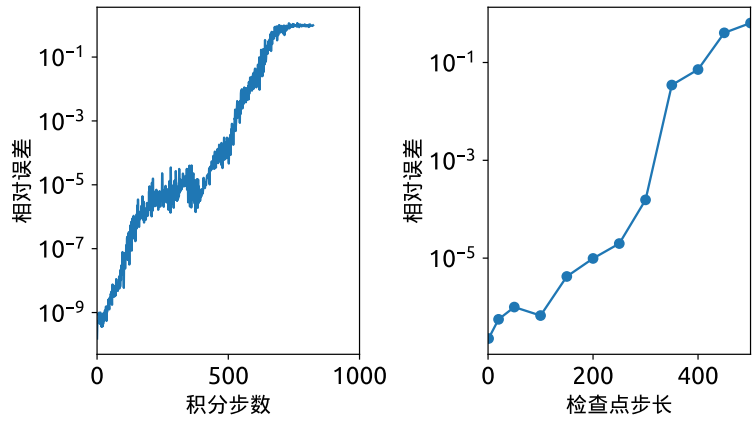



图 2: 利用 Neural-ODE 求导时，$l^2$ 误差与积分步长的关系。其中一个点代表了在该步长下，对 100 个初始点求导得到的 $l^2$ 误差中位数。缺失的数据代表该处出现数值溢出的情况。

其中，$\cdot$ 代表了相应变量的梯度 $\frac{\partial \mathcal{L}}{\partial \cdot}$，这个定义同时隐含了程序最终会输出一个标量 $\mathcal{L}$ 作为损失函数。这个算法的复杂度对于需要求导的变量数目不敏感，相反因为这里假设了输出变量只有一个，该算法的实际计算量会随着输出变量的数目而线性增加。但考虑到实际应用中，输出往往只有一个，也就是损失函数，因此它在求导 $\sim 10^2$ 以上的输入变量的优化问题中对比前向自动微分拥有优势。值得注意的是，对堆栈的操作也带来了正比于计算步骤数 $(\mathcal{O}(T))$ 的空间开销以及频繁访问内存带来的性能下降的问题，这是后向自动微分框架设计以及使用的复杂性的来源。

## 3 后向自动微分的时间与空间的权衡

在处理很多 ODE 的时候，由于计算步骤数很多，在使用后向自动微分时缓存所有的中间变量会使得计算空间开销巨大。如何在一定的运行时间限制下，尽可能少的空间开销回溯中间状态成了其中的核心问题。图 1 (b), (c) 演示的是如何利用堆栈来缓存中间状态并回溯。但其实还有一种更加简单的做法是，那就是仅记录程序的初始状态，当需要获取第 $k$ 个计算状态 $s_k$，程序从初始状态 $s_0$ 开始进行 $k$ 步运算。这个算法虽然没有额外的空间开销，但是需要 $O(T^2)$ 的运行时间。但很多实践中，不管是 $O(T^2)$ 的时间开销或者是 $O(T)$ 的空间开销都不实际，人们更需要的是一个均衡的计算时间和计算空间的方案。人们设计出了检查点方案，它的核心思路是在确定在程序运行的何时，何处进行状态拷贝。最优的关于时间和空间交换的检查点方案在 1992 年被 Griewank [Griewank(1992)] 提出，它是很多广义自动微分框架的基础。

回溯中间状态还有另一种做法,那就是可逆计算。可逆计算的代码不需要借助全局堆栈也可以被回溯。但它并非免费的午餐，因为它也需要额外的空间保证不丢弃信息以保证可逆性。它并不像传统代码一样可以随意的擦除或释放内存，可逆计算“释放”内存的方式是通过反计算把变量内容恢复到 0 状态，也就是

| 方法 | 时间 | 空间 |
|---|---|---|
| 共轭态法 | $\mathcal{O}(T)$ | $\mathcal{O}(TS)$ |
| 前向自动微分 | $\mathcal{O}(NT)$ | $\mathcal{O}(S)$ |
| 基于检查点的后向自动微分 | $\mathcal{O}(T\log T)$ | $\mathcal{O}(S\log T)$ |
| 基于可逆计算的后向自动微分 | $\mathcal{O}(T^{1+\epsilon})$ | $\mathcal{O}(S\log T)$ |

表 1: 不同方案的时间与空间复杂度。其中共轭态法考虑了缓存部分中间结果以保证反向积分的正确性，因此空间会有与时间线性增长。前向自动微分中的 $N$ 代表了需要求导的参数个数。可逆计算中的时间复杂度为多项式，且 $\epsilon > 0$。

用反计算的时间交换空间。回溯计算状态问题有着简单的数学模型，它可以被抽象的概括为如下的鹅卵石游戏。

### 3.1 鹅卵石游戏

鹅卵石游戏是在一维格子上的单人游戏，游戏开始，玩家拥有一堆鹅卵石以及一个一维排布的 $G$ 个格子，标记为 $0, 1, 2 \dots G$，并且在 0 号格子上有一个预先布置的鹅卵石。游戏目标是从自己的堆中取尽可能最少的鹅卵石，或是使用尽可能少的步骤数。

它最初被提出描述可逆计算中的时间与空间的权衡，其规则为

**鹅卵石游戏-可逆计算版本**

**放置规则**：如果第 $i$ 个格子上有鹅卵石，则可以从自己堆中取一个鹅卵石放置于第 $i+1$ 个格子中，

**回收规则**：仅当第 $i$ 个格子上有鹅卵石，才可以把第 $i+1$ 个格子上的鹅卵石取下放入自己的堆中，

**结束条件**：第 $G$ 个格子上有鹅卵石。

我们稍微修改可以得到检查点版本的规则，它为用户增加了一支画笔用于涂鸦格点，其规则描述为

**鹅卵石游戏-检查点版本**

**放置规则**：如果第 $i$ 个格子上有鹅卵石，则可以从自己堆中取一个鹅卵石放置于第 $i+1$ 个格子中，

**回收规则**：可以随意把格子上的鹅卵石取下放入自己的堆中，收回鹅卵石不计步骤数，

**涂鸦规则**：当第 $i$ 个格子有鹅卵石，且第 $i+1$ 个格子被涂鸦或 $i = G$，可以涂鸦第 $i$ 个格子，涂鸦不记入步骤数，

**结束条件**：涂鸦完所有的格点。

鹅卵石游戏是对计算过程的时间和空间交换的一种理想化的描述，它假设了每个计算步骤都需要固定的内存开销以及时间开销，并用一个鹅卵石代表了一个单位的内存，而放置和取回鹅卵石的步骤数代表了一个单位的运算时间。在可逆计算中，内存的释放必须通过反计算来实现，因此需要消耗步骤数且要求其前置状态存在以保证反计算的可行。检查点方案中，涂鸦过程代表了梯度反向传播的过程，它要求按照以与程序正常运行方向相反的顺序访问计算状态。

有了鹅卵石游戏的图像，我们看下检查点方案的算法的梗概如图 5 (a) 所示，算法把计算过程分割成 $d$

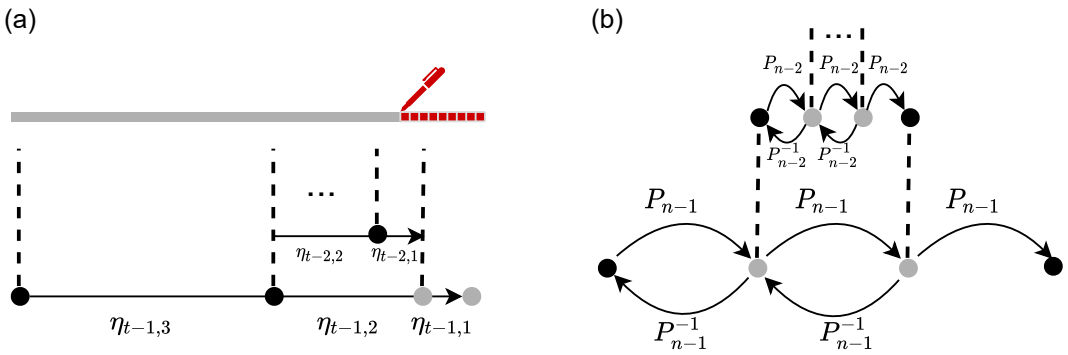



图 3: (a) 广义自动微分中常见的 Treeverse 算法 [Griewank(1992)]，其中 $\eta(\tau,\delta) \equiv \binom{\tau+\delta}{\delta} = \frac{(\tau+\delta)!}{\tau!\delta!}$。(b) 可逆计算时空交换的 Bennett 算法。 [Bennett(1973), Levine and Sherman(1990)] 其中，$P$ 和 $Q$ 分别代表了计算和反计算。

个区块，并记录每个区块的开始状态，每个区块的大小由函数 $\eta(\tau,\delta)$ 决定，其中 $\delta = 1 \dots d$ 为从末尾开始数的区块的指标，$\tau = 1 \dots t$ 是一个超参数并被初始化为 $t$。最后的区块最小，因此可以用上述 $O(T^2)$ 的算法来获取中间状态，回溯完最后一个区块后，程序释放最后一个区块开始处被缓存的状态。于是在接下来计算倒数第二个区块的过程中，我们可以在这个计算过程中多缓存一个状态而不增加最大内存开销。类似的，倒数第 $\delta$ 个分块可以使用的额外状态缓存数为 $\delta - 1$。对每个分开递归的调用这个算法 $t$ 次，每次将 $\tau \leftarrow \tau - 1$，最后得到的分块的大小为 1。整个算法的时间和空间开销的关系是

$$T_c = tT, S_c = dS, \tag{3}$$

其中，$T = \eta(t,d)$ 是初始计算时间。选择适当的 $t$ 或者 $d$，在时间和空间维度上的额外复杂度可以都是 $\log(T)$。

如图 5 (b) 所示的是可逆计算框架下时间和空间最优交换方案，也被称为 Bennett 算法。它将程序均匀的分割为 $k$ 等份，先是像前执行 $k$ 步得到结果，然后从最后第 $k-1$ 步运算开始执行反计算以清除中间计算结果释放内存。在每个区块中，递归的分割为 $k$ 等份做同样的计算-拷贝-反计算，直到程序无法再分割。

$$T_r = (2k-1)^n, S_r = n(k-1). \tag{4}$$

其中，$k$ 与 $n$ 满足 $T = k^n$。可以看出可逆计算的时间复杂度和原时间为多项式关系。

检查点和可逆编程的都可以做到和原程序一样的时间复杂度，但是此时都有 $O(TS)$ 的空间复杂度。只有检查点方案才能做到没有额外的空间开销，但是需要有 $\mathcal{O}(T^2)$，而可逆计算至少为 $\mathcal{O}(\log(\frac{T}{S}))$ 的空间复

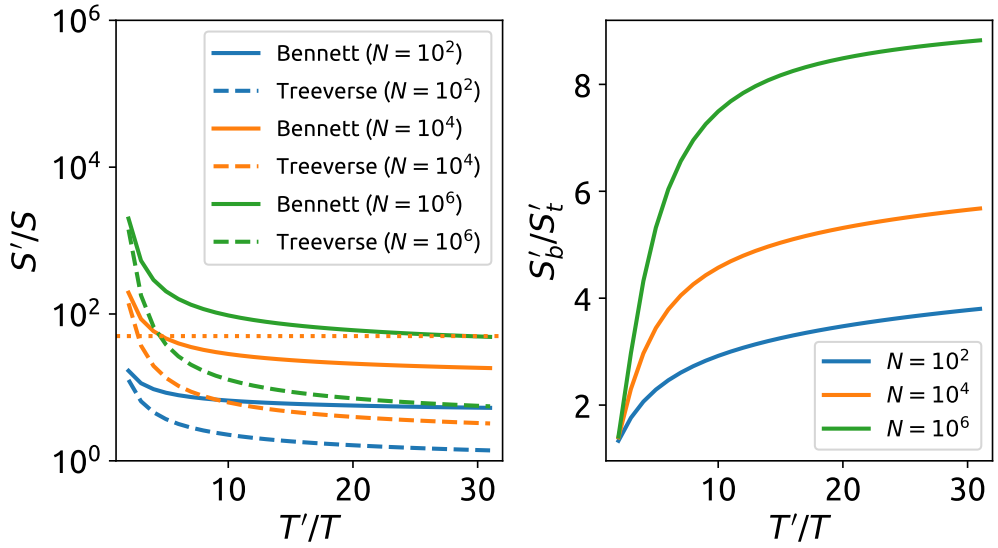



图 4: 为了回溯中间状态, 时间和空间在两种最优时间-空间交换策略下的关系。(a) 固定横轴为状态回溯的计算时间与原函数计算时间的比值，对比再允许固定时间开销下，内存的额外开销。其中 Bennett 算法代表了可逆计算下的最优策略，而 Treeverse 则是传统计算允许的最优策略，黄色点状横线对应 $S'/S = 50$。(b) 对比 Bennett 算法与 Treeverse 算法空间开销的比值。

杂度，对应时间复杂度 $\mathcal{O}(T(\frac{T}{S})^{0.585})$。可逆计算的优点是可以很方便的利用可逆性。当我们固定时间，两种方案的空间消耗关系如图 4(a) 和 (b) 所示。当时间的额外开销较小，两者的差异并不明显，反之则可以有接近一个数量级的差别。

后来在地震学的模拟中，它被用来微分地震波传播的过程 [Symes(2007)]。自动微分也在其中发挥着重要的作用 [Zhu *et al.*(2020)Zhu, Xu, Darve, and Beroza]。

## 4 自动微分在物理中的应用

### 4.1 地震波的模拟

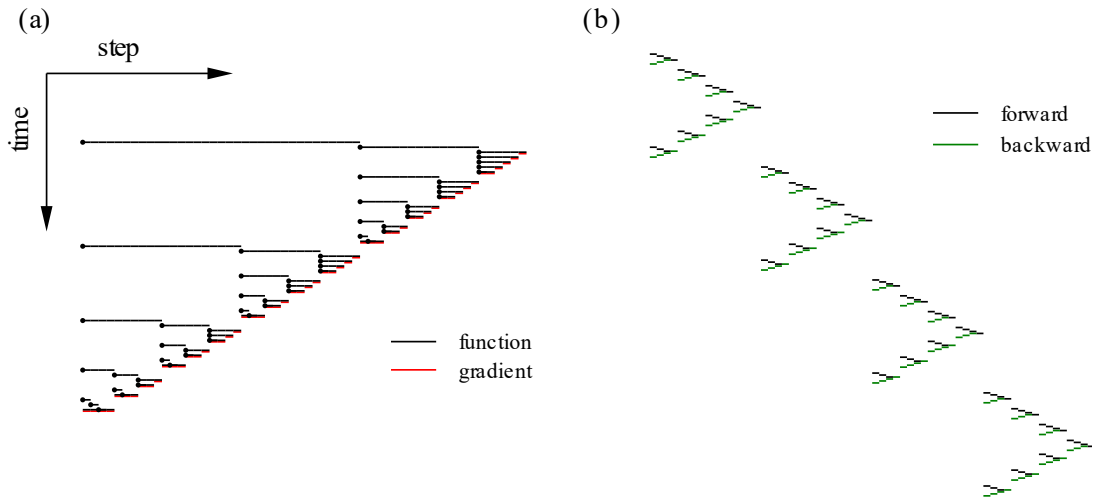



图 5: (a) treeverse 算法 ($t = 5$, $d = 3$) 和 (b) Bennett 算法 ($k = 4$, $n = 3$) 中，函数的执行过程，其中横向是鹅卵石游戏的格子，纵向随着进入 treeverse 函数或者 Bennett 函数的次数向下延展。

## 5 结　论


## 致谢

感谢王磊老师的讨论。


## 附录 A1

标题排列和编号方式为 A1, A2, A3.

## 附录 A2


[Rosset(2019)] C. Rosset, Microsoft Blog (2019).

[Nolan(1953)] J. F. Nolan, *Analytical differentiation on a digital computer*, Ph.D. thesis, Massachusetts Institute of Technology (1953).

[Gutzwiller(1963)] M. C. Gutzwiller, \bibfield journal \bibinfo journal Phys. Rev. Lett.\ \textbf \bibinfo

volume 10,\ \bibinfo pages 159 (\bibinfo year 1963).

[Carleo and Troyer(2017)] G. Carleo and M. Troyer, \bibfield journal \bibinfo journal Science\ \textbf \bibinfo volume 355,\ \bibinfo pages 602—606 (\bibinfo year 2017).

[Deng *et al.*(2017)Deng, Li, and Das Sarma] D.-L. Deng, X. Li, and S. Das Sarma, \bibfield journal \bibinfo journal Physical Review X\ \textbf \bibinfo volume 7 (\bibinfo year 2017),\ 10.1103/physrevx.7.021021.

[Cai and Liu(2018)] Z. Cai and J. Liu, \bibfield journal \bibinfo journal Phys. Rev. B\ \textbf \bibinfo volume 97,\ \bibinfo pages 035116 (\bibinfo year 2018).

[Luo *et al.*(2019)Luo, Liu, Zhang, and Wang] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, "Yao.jl: Extensible, efficient framework for quantum algorithm design," (2019), arXiv:1912.10877 [quant-ph] .

[Liao *et al.*(2019)Liao, Liu, Wang, and Xiang] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, \bibfield journal \bibinfo journal Physical Review X\ \textbf \bibinfo volume 9 (\bibinfo year 2019),\ 10.1103/physrevx.9.031041.

[Liu *et al.*(2020)Liu, Wang, and Zhang] J.-G. Liu, L. Wang, and P. Zhang, "Tropical tensor network for ground states of spin glasses," (2020), arXiv:2008.06888 [cond-mat.stat-mech] .

[Zhang *et al.*(2019)Zhang, Wan, and Yao] S.-X. Zhang, Z.-Q. Wan, and H. Yao, "Automatic differentiable monte carlo: Theory and application," (2019), arXiv:1911.09117 [physics.comp-ph] .

[Heimbach *et al.*(2005)Heimbach, Hill, and Giering] P. Heimbach, C. Hill, and R. Giering, Future Generation Computer Systems **21**, 1356 (2005).

[Symes(2007)] W. W. Symes, \bibfield journal \bibinfo journal Geophysics\ \textbf \bibinfo volume 72,\ \bibinfo pages SM213 (\bibinfo year 2007).

[Zhu *et al.*(2020)Zhu, Xu, Darve, and Beroza] W. Zhu, K. Xu, E. Darve, and G. C. Beroza, "A general approach to seismic inversion with automatic differentiation," (2020), arXiv:2003.06027 [physics.comp-ph] .

[Plessix(2006)] R.-E. Plessix, \bibfield journal \bibinfo journal Geophysical Journal International\ \textbf \bibinfo volume 167,\ \bibinfo pages 495 (\bibinfo year 2006), https://academic.oup.com/gji/article-pdf/167/2/495/1492368/167-2-495.pdf .

[Chen *et al.*(2018)Chen, Rubanova, Bettencourt, and Duvenaud] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, in \emph \bibinfo booktitle Advances in Neural Information Processing Systems, Vol. 31, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018).

[Wengert(1964)] R. E. Wengert, Communications of the ACM **7**, 463 (1964).

[Boltyanski *et al.*(1960)Boltyanski, Gamkrelidze, Mishchenko, and Pontryagin] V. Boltyanski, R. Gamkrelidze, E. Mishchenko, and L. Pontryagin, IFAC Proceedings Volumes **1**, 464 (1960).

[Griewank(1992)] A. Griewank, \bibfield journal \bibinfo journal Optimization Methods and software\ \textbf \bibinfo volume 1,\ \bibinfo pages 35 (\bibinfo year 1992).

[Bennett(1973)] C. H. Bennett (1973).

[Levine and Sherman(1990)] R. Y. Levine and A. T. Sherman, \bibfield journal \bibinfo journal SIAM Journal on Computing\ \textbf \bibinfo volume 19,\ \bibinfo pages 673 (\bibinfo year 1990).

# Automatic differention in physics simulation *

Jin-Guo Liu[1)] Kai-Lai Xu[2)]

1) (Harvard University, Cambridge 02138)

2) (Stanford University, Stanford 94305)

1) ( *Massachusetts Hall, Cambridge, MA 02138* )

2) ( *450 Serra Mall, Stanford, CA 94305* )

## Abstract

To determine the probe made of amino acids arranged in a linear chain and joined together by peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. The sequence of amino acids in a protein is defined by a gene and encoded in the genetic code. This can happen either before the protein is used in the cell, or as part of control mechanisms.

**Keywords:** automatic differentiation, scientific computing, reversible programming, treeverse, physics simulation

**PACS:** 02.60.Pn, 02.30.Jr, 91.30.−f

* Project supported by the State Key Development Program for Basic Research of China (Grant No. 2011CB00000), the National Natural Science Foundation of China (Grant Nos. 123456, 567890), and the National High Technology Research and Development Program of China (Grant No. 2011AA06Z000).

† Corresponding author. E-mail: jinguoliu@g.harvard.edu