# 微分万物:深度学习的启示*

王 磊[1,2,†] 刘金国[3]
(1 中国科学院物理研究所 北京 100190)
(2 松山湖材料实验室 东莞 523808)
(3 哈佛大学物理系 剑桥 02138)

2020－12－09收到
† email:wanglei@iphy.ac.cn
DOI:10.7693/wl20210201

# Differentiate everything: a lesson from deep learning

WANG Lei[1,2,†] LIU Jin-Guo[3]
(1 Institute of Physics, Chinese Academy of Sciences, Beijing 100190, China)
(2 Songshan Lake Materials Laboratory, Dongguan 523808, China)
(3 Department of Physics, Harvard University, Cambridge 02138, USA)

**摘 要** 深度学习教会了人们一种新的和计算机打交道的方式:将一些可微分的计算单元组合形成一段程序,再通过梯度优化的方法调整程序参数,使其达成期望的目标。这就是微分编程的思想。深度学习技术的快速发展为微分编程提供了趁手的工具,也为计算物理开辟了一番新天地。文章介绍微分编程的基本概念,并举例说明它在建模、优化、控制、反向设计等物理问题中的应用。

**关键词** 微分编程,自动微分,计算物理

**Abstract** Deep learning taught us a new way to play with computers: compose differentiable components into a computer program, then tune its parameters via gradient optimization until it achieves what we want. This is the key idea of differentiable programming. The rapid development of deep learning technology offers convenient tools for differentiable programming, and also opens a new frontier for computational physics. This article introduces the basic notion of differentiable programming and its physics applications including modeling, optimization, control, and inverse design.

**Keywords** differentiable programming, automatic differentiation, computational physics

## 1 引言

深度学习在做什么?对于这个问题,人们的第一反应往往是“训练人工神经网络”。这个解释让深度学习听起来既神奇又神秘,却没有正面回答问题。让我们再追问一句:训练神经网络又是在做什么?纯粹从编程的角度看,深度学习将一些可微分的计算单元搭建成为一段程序,再使用梯度优化的方法调节程序参数,使其达成所需的功能。其实,人工神经网络是一类特别简单的计算机程序。大多数神经网络中主要涉及的是张量运算,例如张量收缩和元素非线性变换等。而所谓“深层”的神经网络,只不过意味着这类张量计算程序比较长而已。在这个意义下,人工神经网络只是一小类可以被训练的程序。推而广之,我们可以对一般的计算机程序求导,利用梯度信息调节程序的行为,这就是微分编程。

以Yann LeCun(Fackbook的首席AI科学家)和

* 国家自然科学基金(批准号:11774398)资助项目

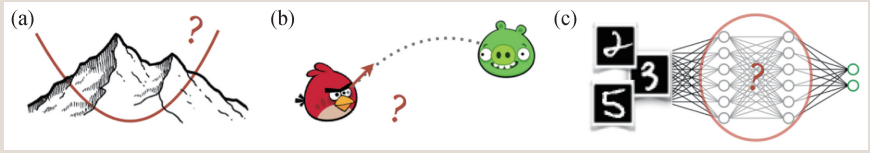


图1　三个微分编程实例　(a)逆薛定谔方程；(b)愤怒的小鸟；(c)图像识别

表1　三个微分编程实例的基本特征

| | 逆薛定谔方程 | 愤怒的小鸟 | 图像识别 |
|---|---|---|---|
| 问题性质 | 反向设计问题 | 控制问题 | 建模问题 |
| 学习对象 | 外势场 | 初始速度 | 神经网络参数 |
| 计算过程 | 本征方程求解 | 运动方程积分 | 张量运算 |
| 目标函数 | 密度差 | 距离差 | 识别正确率 |
| 优化方式 | 微分编程 | | |

Andrej Karpathy(Tesla的AI部门主管)为代表的一些机器学习专家认为微分编程是深度学习的未来[1]。深度学习中所发明的各式各样的神经网络结构，例如卷积、循环、注意力机制等，都是针对特定任务的可微分模块。设计这类模块的指导思想是尽可能地利用关于问题本身的先验知识，例如对称性、局域性、层级结构等等。微分编程通过学习一般的计算机程序，可以更加充分地利用问题相关的特定领域的先验知识，同时还能保证模型的可解释性。另一方面，越来越多其他领域的科学家们也认识到微分编程的价值，认为它是联系深度学习与科学计算的一条纽带。

## 2　微分编程：从薛定谔方程到愤怒的小鸟

让我们通过几个例子来直观地认识微分编程。图1(a)定义了这样一个问题：如何设计合适的外势场，使得其中的自由电子的基态密度分布达到期望的形式。这个问题的正向求解是相对容易的，只需要根据外势场构造哈密顿量，再求解相应的本征值问题就能得到基态的电子密度。按照微分编程的思路解决反向问题，我们首先定义正向计算的密度与目标密度的差别作为目标函数，再穿过正向计算过程求导得到目标函数对于外势场的梯度。之后，我们可以利用梯度信息优化外势场，使其中的电子密度分布逐渐逼近目标分布。这类反向求解自由电子势函数的问题是密度泛函理论研究中的基本问题[4]，也是一个有代表性的哈密顿量反向设计问题。微分编程是解决这类问题简单而通用的方法。读者可以参考笔者编写的演示程序，使用深度学习框架PyTorch解决一维薛定谔方程的逆问题[5]。

图1(b)示意了“愤怒的小鸟”游戏的基本设置。我们希望通过调整小红飞出的初始速度，使得它能够命中目标。与反复的试错相比，一种更加高效的调整方式需要用到和目标的距离对初始速度的导数信息。这要求能够穿过整个飞行轨迹求导。微分编程可以帮助我们做到这一点。更妙的是，为了对运动方程的积分计算求导，我们只需要沿着时间反向，对另一套运动方程积分[6]。

图1(c)是一个典型的深度学习任务：图像识别。我们希望调节神经网络的参数，使得网络输出正确的图片标签。基于网络参数梯度的优化方案是目前最成功的神经网络训练方法。关于这点有一个简单的理解：多元函数的函数值只是一个标量的信息，而梯度是高维的矢量。从信息的角度，越是高维的函数，梯度优化相对于非梯度优化方法的优势就越明显。现代的深度学习应用甚至可以训练含有一千多亿个参数的神经网络。如果不使用梯度的信息，实在难以想象如何处理这么高维的优化问题。

表1总结了以上三个例子的基本特征。可以看出，无论是从问题性质、学习对象、计算过程，还是目标函数的角度，这三个问题看起来都毫无关联。然而，它们其实都是微分编程的特例：我们穿过整个计算过程求导，得到目标函数相对于学习对象的梯度，从而解决问题。

和通常的深度学习问题不同，图1(a，b)这两个例子中都包含针对特定物理领域的计算过程(求解薛定谔方程和牛顿方程)。另外，读者也许注意到了，在这两个例子中甚至都没有谈到训练数据。事实上，有不少科学问题都是如此。我们并

不能够、也不需要事先准备大量的训练数据。对于科学问题，“一个好的模型胜过一千个数据”[7]。而微分编程将模型表达成计算机程序，再利用梯度信息学习其中的未知成分。这背后的关键既不是大数据，也不是神经网络，而是自动微分。

## 3 什么是自动微分？

顾名思义，自动微分就是自动化地计算程序的输出相对于输入的导数。这项技术的本质是机械地对计算过程使用链式法则，从而得到数值精确的导数信息。我们可以用计算图来形象化地表达程序运行和自动微分的过程。计算图是一个有向无环图(directed acyclic graph)，其中的节点和箭头代表程序运行时变量之间的依赖关系。图2展示了一类典型的计算图。其中变量 $x_1$ 代表程序输入，$x_2$，$x_3$ 是计算中间结果，$\theta_1$，$\theta_2$ 是程序的参数。最后，计算图末端的节点 $\mathscr{L}$ 代表程序输出的标量损失函数。图1中的三个例子所对应的计算图都具有这样的结构[8]。

为了考虑自动微分，我们定义计算图中某一变量 $x$ 的梯度为其伴随变量(adjoint variable)，记为 $\bar{x}=\partial\mathscr{L}/\partial x$。自动微分的目的是把计算图最末端 $\bar{\mathscr{L}}=1$ 的信息反向传递回所有需要计算梯度的节点。这可以使用反向传播算法实现：中间节点收集下游节点的伴随变量信息，根据链式法则得到自己的伴随变量，再往它的上游节点传递这个信息。在反向传播的每一步，我们只需要计算下游传来的伴随变量与当前计算步骤雅可比矩阵的乘积。遵循这种信息传递的方法，计算机可自动地对任意复杂的计算过程微分。理论保证了使用这个办法求导与程序前向运行的计算复杂度是一样的。此外，因为梯度的计算过程也可以表达为计算图，我们甚至可以迭代地使用自动微分方法计算程序的高阶导数。

自动微分相对于使用数值差分的方法计算梯度有不少优点。首先，它可以保证计算的梯度是数值精确的，而不依赖于额外的差分参数。其次，图2所展示的自动微分方式，仅通过向前和向后遍历计算图两遍，就能高效地计算出所有变量的梯度。这远比数值差分高效：因为数值差分需要对参数的每一个分量分别做微扰，再反复地执行前向计算来得到参数在各个分量上的梯度。由于图2所示的计算方法有一个从后向前回溯计算图的过程，这被称为反向模式(reverse mode)的自动微分。实现反向模式的自动微分往往需要付出额外的存储代价，以记录前向计算时的中间变量信息。还有一类前向模式(forward mode)的自动微分：通过计算雅可比矩阵和变量梯度的乘积，向前传递对于程序输入或参数的扰动。前向模式求导的方向与程序的运行方向相同，因此不需要付出额外的存储代价。但是前向模式不如反向模式高效：它的计算复杂度和数值差分是一样的，因此通常不被用于含有大量参数的优化问题。

值得强调的是，自动微分的对象不见得非要是数学函数。微分编程完全可以应用于包含循环和控制流等指令的一般程序。在程序的运行过程中，取决于不同的输入数据与分支条件，动态生成的计算图可以具有完全不同的拓扑结构与深度。一旦确定了计算图的结构，在自动微分计算的每一步，仅需要关心计算单元局部的信息传递。这里的计算单元可以是基本的四则运算，也可以是一系列基本运算的复合，例如本征方程、微分方程、自洽方程求解等等。只要我们有办法指定计算单元的伴随变量回传规则，就可不必关心其内部计算的细节，也不需要求导进入计算单元内部。利用这样的方式控制自动微分的颗粒度，可以保证数值稳定性、提升运行效率、降低反向模式计算的内存消耗。如何针对具体问题设计合

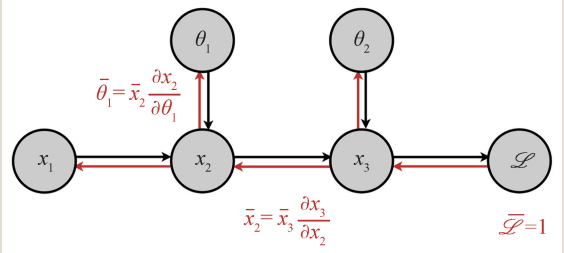


**图2** 计算图表达计算过程中变量之间的依赖关系。自动微分方法回溯计算图，得到程序输出相对于所有变量的梯度

适的计算单元，正是微分编程的艺术。模块化和复合性是深度学习的名片，也是微分编程的关键。

## 4 微分编程的科学应用

微分编程并不是一个从天而降的崭新方法。科学家早在20世纪50、60年代就开始探索对于计算机程序的自动微分[2, 3]。在控制和反向设计等领域，也早已存在与微分编程密切相关的伴随变量方法。而物理学家熟知的微扰论和线性响应理论，可以说就是穿过物理过程的微分计算。如今，在基于张量运算的自动微分应用——深度学习——成为一门显学的时代，微分编程获得了长足的进步和更普遍的关注。它也给物理学中的建模、优化、控制和反向设计问题提供了一个统一而灵活的研究方法。下面我们分别举例说明。

### 4.1 建模问题

建模是指根据观测数据推断实际物理过程，从而预测新的情形。建模的目标是能够举一反三，亦即从少量的观测数据出发做出正确的预测。用机器学习的术语来说，就是训练出具有泛化能力的模型。图1(c)所示的图像识别任务就是一个建模问题。然而，不同于常见的图像识别任务，我们通常难以为待解决的物理问题准备大量准确的训练数据。仅仅依赖于少量数据建模，容易发生过拟合现象。这就需要精心地设计模型，最好让它本身就契合待研究的物理过程。微分编程是实现这个目标的一种方式。

蛋白质折叠是一个公认的难题。从氨基酸序列出发预测蛋白质的结构，中间涉及复杂的物理过程。完全基于第一性原理的计算恐怕是极其困难的。因此，人们寄希望于从已知的序列——蛋白质结构数据中寻找规律，直接预测蛋白质结构。DeepMind的AlphaFold凭着精妙的模型设计与充沛的算力，已在这方面取得了了不起的成果[11]。但这并没有妨碍人们另辟蹊径，探索不一样的技术路线。文献[9]将氨基酸序列到蛋白质结构的映射建模成一段分子动力学模拟，并从现有的序列——结构数据中学习分子模拟所需的能量函数。利用微分编程技术，可以穿过如图3(a)所示的整个模拟过程学习模型参数，以预测未知的蛋白质结构。

对于量子效应更加显著的化学分子，预测其结构需要快速准确地求解量子多体问题的基态。Google Research的李力等[12]根据密度泛函理论建立模型，将Kohn—Sham密度泛函计算当作可微分的程序，穿过整个自洽计算过程来学习交换关联势。他们仅仅需要非常少的分子基态数据就可以成功地训练模型，预测一维氢分子随着原子间距的整条势能曲线。

以上两个工作的共同特点是模型中包含了特定的物理计算(分子动力学模拟、Kohn—Sham方程)，因此不再需要从零开始学习关于物理问题的基本规律，而只需填补成熟的理论框架中空缺

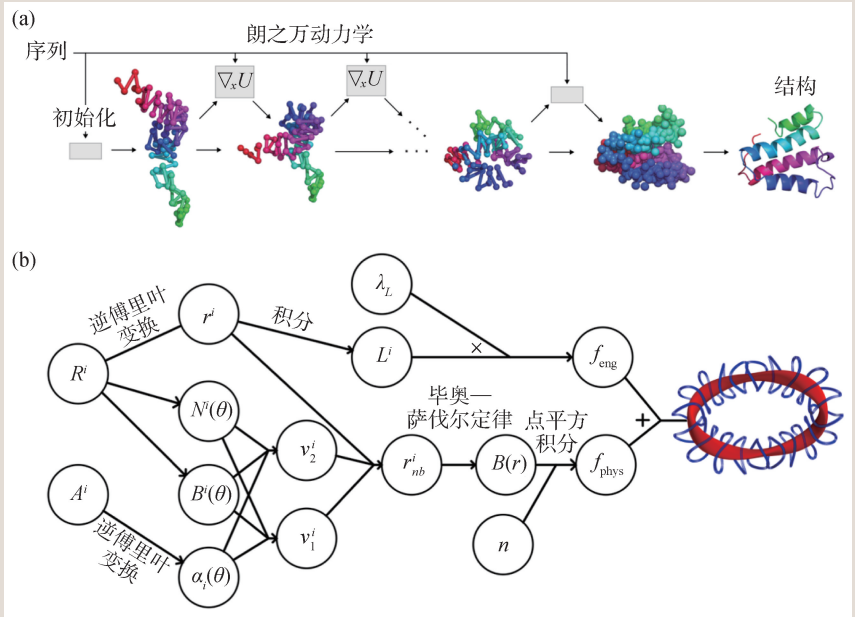


**图3** 微分编程应用于蛋白质折叠和仿星器设计 (a)从氨基酸序列到蛋白质结构的计算图；(b)从仿星器的线圈参数到目标函数的计算图(摘自文献[9]和[10])

的部分(分子模拟的能量函数、密度泛函理论的交换关联势)。这样的建模方式有利于从小样本中学习到具有足够泛化能力的模型。但是由于涉及到比神经网络复杂得多的计算，这些工作目前还只停留在小规模概念验证的阶段。时间将会告诉我们，基于物理机制的微分编程是否还能带我们走得更远。

### 4.2 优化问题

优化问题在物理学中无处不在。自然界偏爱低能的状态。更一般地看，甚至可以说最小作用量原理支配了整个物理学。在具体的物理学研究中，可以利用自动微分计算梯度帮助求解优化问题。这是微分编程非常自然的一类应用。

一个有代表性的例子是全变分的Hartree—Fock方法[13]。在传统的Hartree—Fork计算中，人们通常经验性地选取一些基函数展开分子的哈密顿量，再通过自洽计算优化这些基函数上的组合系数来近似体系的基态。而在全变分的计算中，可以利用自动微分穿过自洽循环，计算Hartree—Fock能量相对于基函数中经验参数的导数，从而彻底优化基函数的位置和形状。因为优化空间变大了，全变分的Hartree—Fock方法可以得到比传统方法更低的变分能量。

另一个例子是我们与合作者们发展的张量网络微分编程[14]。张量网络方法经过多年的发展已经成为经典统计、量子多体物理，乃至机器学习等领域重要的理论和计算工具。然而，针对张量网络的优化仍是一个长期存在的难点。传统的优化方法要么是基于一些近似的方案，要么依赖于繁琐的人工解析推导梯度。这限制了张量网络方法在复杂问题中的广泛应用。采用微分编程的方法穿过整个张量网络重正化计算进行优化，不仅大大降低了程序实现的门槛，还更加高效和可靠。例如，作者对于二维无穷大方格子上的反铁磁海森伯模型得到了最低的变分能量。对于更有挑战的阻挫磁性问题[15](如笼目格(Kagome)格子上的反铁磁海森伯模型等)张量网络微分编程也都给出比之前更低的变分能量。特别的，对于六角格子上的Kitaev模型，通过对张量网络进行梯度优化可以直接得到物理上定性正确的非磁性基态。这些结果给了我们更强的信心：假以时日，对张量网络的微分编程可以带给我们新的物理！

上面举的两个例子涉及到对于Hartree—Fock自洽方程或张量网络重正化这类迭代计算的求导。可以简单直接地将迭代过程展开，然后利用现成的自动微分工具穿过整个计算过程求导。除此以外，还可以把达到迭代收敛条件视作基本的计算单元，利用隐函数定理得出伴随变量的回传规则[16]。后面这种做法更加契合微分编程的思想，可以让程序更加高效和省内存。

### 4.3 控制问题

控制问题是指通过调整体系的初始条件和外场等，以实现对于含时物理过程的调控。图1(b)所示的愤怒的小鸟游戏就是一个控制问题。通过对计算机模拟的经典物理过程求导，微分编程已经在计算机图形学、机器人控制、分子模拟等领域彰显威力。例如，文献[17]利用可微分的分子模拟手段调整胶体颗粒的相互作用形式，以控制其自组装过程中的动力学行为。

量子计算中也有类似的控制问题。例如，如何调控微波或光脉冲以实现特定的量子门？从微分编程的视角审视现有的量子控制方法，会发现它们有的使用了不够高效的前向微分模式，有的依赖于不够灵活的离散控制方案。利用图1(b)中我们见过的穿过微分方程求导的方法[6]，可以类似地控制含时薛定谔方程的演化[18]。这类微分编程方法不仅高效，还适用于任意的控制手段、限制条件和目标函数，是研究量子最优控制问题的一般方法。另外，在实际的控制问题中，计算机模拟未必能够完全精确地描述物理过程。因此还需要将基于微分方程的最优控制和上文提到的建模方法相结合[7]。一方面根据实际观测的结果对物理过程建立准确的模型，另一方面利用学习到的模型做出正确的控制预测。和彻底黑箱化的控制手段(例如无模型的强化学习方案)相比，微分编程可以充分利用关于控制过程的物理认识，因此优化效率要高很多。

### 4.4 反向设计问题

反向设计是指通过设计合适的物理结构以实现特定的力场、光场、电磁场等。图1(a)所示的逆薛定谔问题就是一个反向设计问题。一个类似的例子是光子晶体的反向设计。光子晶体通过人为加工的微结构来调制光的传播。因为计算机模拟的光学现象可以几乎完美地在样品中实现，这是一个特别适合于反向设计的领域。斯坦福大学的Momchil Minkov等人[19]通过对麦克斯韦方程组求解过程求导，以优化光子晶体的色散关系和品质因子。他们体会到，自动微分方法将光子晶体领域里常用的伴随变量方法拓展到了任意的计算图，可以辅助设计更加灵活多样的光学器件。

另一个值得一提的应用是使用自动微分设计仿星器[10]。仿星器是可控核聚变的途径之一。如图3(b)所示，为了实现它，需要设计合适的线圈以产生磁场来约束几亿度的等离子体。这样的极端条件对于设计提出了极高的要求。微分编程的方法可以统一地考虑产生磁约束的物理过程以及各种现实的工程限制。深度学习专家David Duvenaud读到这篇文章之后，兴奋地在Twitter上发文说："这增强了我对人类文明的信念。当我听说仿星器时，我就想'希望他用上了自动微分，但我担心他们其实还没有。我希望有人告诉他们怎么做。'"。诚然，已经充分享受微分编程成果和乐

#### 微分编程工具

本文中列举的应用涉及到对多种复杂运算的求导，例如非线性方程求解、快速傅里叶变换、矩阵分解等等。因为这些应用的推动，人们开发了越来越多的微分编程基本单元和工具。这些进展进一步丰富了微分编程的生态。下方表格介绍了一些微分编程工具。此外，www.autodiff.org网站还列举了更多的微分编程工具。

基于这些编程工具，科学家们打造了针对具体领域的微分编程软件。这包括张量网络[20]、分子模拟[21]、量子线路[22—24]、系外行星[25]等。此外还有面向可微分的科学计算工具：ξ-torch[26]和ADCME[27]等。SciML[28]和dolfin-adjoint[29]对基于微分方程的可微分科学计算有丰富的支持。一般来说，如果程序中主要包含的是线性代数与张量计算，可以直接使用现有的深度学习框架，因为这些框架的本质是支持硬件加速的可微分张量计算库。而如果程序中主要包含的是微分方程求解、自洽迭代等计算，则需要根据具体情况选用专门的工具以充分发挥微分编程的性能。

| 编程语言 | 软件名 | 开发组织 | 简介 |
|---|---|---|---|
| Python等 | TensorFlow | Google | 张量运算性能出色。对工业级别应用支持完善。 |
| Python等 | PyTorch | Facebook | 方便高效，在研究领域流行程度逐渐超越了Tensorflow。 |
| Python | Autograd | Dougal Maclaurin等 | 通过实现约两百个可微分的基本单元，几乎支持对于整个Numpy库的自动微分。 |
| Python | Jax | James Bradbury等 | Autograd的后代。包括自动矢量化以及对于专用加速硬件的支持等。 |
| Julia | Zygote | FluxML | Julia语言中最流行的后向模式微分库之一。 |
| Julia | ForwardDiff/ ReverseDiff | JuliaDiff | ForwardDiff是性能出色的前向自动微分库。ReverseDiff是后向模式自动微分库，性能比Zygote更好。 |
| Julia | NiLang | Jin-Guo Liu, Taine Zhao | 在可逆计算编程语言中实现的源代码自动微分，能够更灵活地平衡内存消耗与计算时间。 |
| C/Fortran | Tapenade | Laurent Hascoët, Valérie Pascual | 较为古老的微分编程工具之一。它通过源代码转换实现了对于一般程序的前向和后向自动微分。性能出色，并且在洋流和大气模拟等科学计算中有一些应用。 |
| C++ | Adept | Robin Hogan | C++语言中性能最好的通用自动微分库之一。包括前向和后向自动微分。 |

趣的深度学习专家，自然希望它能够被更广泛地应用于各个领域，特别是那些与人类命运休戚相关的问题上。

## 5 结语

科幻作家Arthur Clarke有一句名言：“任何充分先进的技术看起来都无异于魔法”。技术进步可以解放人们的思想，去探索更有创意的工作。微分编程的技术与应用都还在快速的发展中。例如，微分编程与可逆计算相结合，可以缓解反向模式自动微分的内存瓶颈[30]。此外，微分编程的应用范围甚至还逐渐扩展到对于离散参数或是结构信息等“不可微分”对象的学习上。因此，本文所介绍的例子还仅仅是微分编程牛刀小试，希望可以让更多的读者了解和喜欢上微分编程，让它在现实问题中大显身手。

**致 谢** 感谢与谢浩、张潘、李伟、谢志远、廖海军和向涛的合作与讨论。

## 参考文献

[1] 其实，也可以说微分编程是深度学习的过去，见文献2，3
[2] Nolan J F. Analytical differentiation on a digital computer. Massachusetts Institute of Technology，1953
[3] Wengert R E. Communications of the ACM，1964，7：463
[4] Jensen D S，Wasserman A. International Journal of Quantum Chemistry，2018，118(1)：e25425
[5] https://github.com/QuantumBFS/SSSS/blob/master/1_deep_learning/schrodinger.py
[6] Chen R T Q，Rubanova Y，Bettencourt J *et al*. 2018，arXiv:1806.07366
[7] Rackauckas C *et al*. 2020，arXiv:2001.04385
[8] 其实逆薛定谔方程和愤怒的小鸟例子中的计算图更简单：它们没有参数节点，是纯链状结构的。
[9] Ingraham J *et al*. Learning Protein Structure with a Differentiable Simulator. International Conference on Learning Representations，2019
[10] McGreivy N，Hudson S R，Zhu C X. 2020，arXiv:2009.00196
[11] AlphaFold: a solution to a 50-year-old grand challenge in biology. https://deepmind.com/blog/article/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology
[12] Li L *et al*. Phys. Rev. Lett.，2021，126：036401
[13] Tamayo-Mendoza T *et al*. ACS Cent. Sci.，2018，4(5)：559
[14] Liao H J，Liu J G，Wang L *et al*. Phys. Rev. X，2019，9：031041
[15] Liao H J. Talk at Computational Approaches to Quantum Many-body Problems. ISSP，2019. https://www.issp.u- tokyo.ac.jp/public/caqmp2019/slides/726S_Liao.pdf
[16] Kolter Z，Duvenaud D，Johnson M. NeurIPS 2020 tutorial，Deep Implicit Layers- Neural ODEs，Deep Equilibirum Models，and Beyond. https://implicit-layers-tutorial.org/
[17] Goodrich C P *et al*. 2020，arXiv: 2010.15175
[18] Güngördü U，Kestner J P. 2020，arXiv:2011.02512
[19] Minkov M *et al*. ACS Photonics，2020，7( 7)：1729
[20] https://github.com/jurajHasik/peps-torch
[21] https://github.com/google/jax-md
[22] Bergholm V *et al*. 2018，arXiv:1811.04968
[23] Luo X Z，Liu J G，Zhang P *et al*. Quantum，2020，4：341
[24] Broughton M *et al*. 2020，arXiv:2003.02989
[25] Morvan M *et al*. 2020，arXiv:2011.02030
[26] Kasim M F，Vinko S M. 2020，arXiv:2010.01921
[27] Xu K L. Automatic Differentiation Library for Computational and Mathematical Engineering. https://github.com/kailaix/ADCME.jl
[28] https://sciml.ai/
[29] http://www.dolfin-adjoint.org/en/latest/
[30] Liu J G，Zhao T. 2020，arXiv:2003.04617

**读者和编者**

## 《物理》有奖征集封面素材

为充分体现物理科学的独特之美，本刊编辑部欢迎广大读者和作者踊跃投寄与物理学相关的封面素材。要求图片清晰，色泽饱满，富有较强的视觉冲击力和很好的物理科学内涵。

一经选用，均有稿酬并赠阅该年度《物理》杂志。

请将封面素材以附件形式发至：physics@iphy.ac.cn；联系电话：010-82649029。

《物理》编辑部