基于编译器库工具实现神经网络的切线线性与伴随模式

Sa Xiao, [a, b] Hao Jing* [a,b] Honglu Sun [a,b] Haoyu Li, [a, b]

[a] 强天气气象科学与技术国家重点实验室（LaSW），中国北京[b] 中国气象局地球系统数值预报中心（CEMC），中国北京

摘要

本文介绍了 TorchNWP，这是一种编译库工具，可用于高效耦合人工智能组件与传统数值模式。该工具旨在解决以 Fortran 开发的业务数值模式与基于 Python 的深度学习框架之间跨语言兼容性差、耦合灵活性不足和数据传输效率低等问题。该工具由独立、可插拔且统一配置的编译和运行脚本组成。它基于 LibTorch，对统一的应用层调用接口进行优化设计，将 PyTorch 框架下的深度学习模型转换为静态二进制格式，并提供 C/C++接口。随后，通过 Fortran/C/C++混合编程，实现深度学习模型在数值模式中的部署。将 TorchNWP 集成到数值模式中，只需将其编译为可调用的链接库，并在编译和链接阶段将其链接，即可生成可执行文件。在此基础上，在 C/C++ 层实现了基于神经网络的切线线性模式和伴随模型，既可屏蔽神经网络模型的内部结构，又能简化四维变分资料同化系统的构建过程。同时，该工具支持在异构平台上部署，兼容主流神经网络模型，并可实现不同并行粒度的映射和高效并行执行。使用该工具只需对原数值模式进行少量代码修改，从而降低耦合成本。它可高效集成到 CMA-GFS 和 MCV 等数值天气预报模式中，并已应用于基于深度学习的物理参数化方案（如辐射、非地形重力波拖曳）的耦合，以及相应切线线性模式和伴随模型的开发，显著提升了数值天气预报的准确性和效率。

1. 引言

人工智能赋能天气预报有两条技术路线。一条以数值天气预报（NWP）为核心，利用人工智能技术提升

整个工作流程的准确性和计算效率，包括观测处理、数据同化、模式预报和产品解读。另一条以人工智能为核心，建立融合物理机制的数据驱动型气象预报模型。对于第一条路线，传统数值天气预报模式主要使用 Fortran 这一标准编程语言开发。然而，人工智能模型的核心组件大多使用基于动态类型语言 Python 的机器学习框架进行训练或部署。这意味着，以静态类型语言 Fortran 编写的数值天气预报模式无法直接将人工智能模型作为函数调用。

传统数值模式与机器学习模型是相互独立的模块，各自拥有内部数据结构、数据分布和并行分解。理想情况下，应在不引入额外依赖、也不过度修改原本完整的程序代码的前提下，实现二者之间灵活的数据传输，同时支持完整的变量场数据重构和细粒度数据交换，从而解决数据传输灵活性问题。要将深度学习模型与传统数值模式耦合，一种方法是从训练好的深度学习模型中导出参数（如权重和偏置），再以硬编码的形式将其嵌入 Fortran 代码中。然而，这种方法开发效率较低：每当深度学习模型的参数发生变化，都需要相应修改相关代码。同时，Fortran 生态系统中缺乏成熟且易用的机器学习框架和工具，导致移植深度学习模型的成本非常高。

一种方法是采用类似 Fortran-Keras 深度学习桥接工具（Ott 等人，2020）的人工智能模型接口。该接口可以从 Fortran 程序外部导入经过训练的深度学习参数，将其存储在 TXT 文件中，并在深度学习模型推理和预测时调用这些参数。基于 Fortran 的实现可确保其在数值模式中具有相对灵活的部署方式，只需修改 TXT 文件中的参数即可替换模型，而无需更改深度学习模型的网络结构；然而，该接口主要支持基于全连接层的深度学习模型，无法支持 Stable Diffusion 或 Transformer 模型等许多最先进的网络架构。

另一种常用的耦合方法是将基于 Fortran 的主机模式与基于 Python 的深度学习组件分为两组进程，并在耦合运行期间通过进程间通信（IPC）配合同步控制来实现数据传输（Wang 等人，2022）。在耦合执行期间，IPC 数据接口从主机模式接收输入变量，对原始数据执行汇集操作以提高数据传输效率，随后进行浮点数端序转换，并通过 IPC 机制将数据发送至人工智能模型。IPC 数据接口支持多种通信方式，包括 MPI、FIFO 管道和高速数据缓冲区。人工智能推理引擎完成计算后，首先将数据写回 IPC 数据接口，然后通过数据汇集和浮点数端序转换将结果返回给主机模式。在整个数据

耦合过程中，同步控制器负责管理主机模式与人工智能推理引擎之间的运行同步。

Fortran Torch 适配器 (FTA) 是一种基于 LibTorch 的通用工具，通过 FTA 库为用户提供直接使用 Fortran 操作神经网络模型的接口（Mu 等，2023）。该工具支持将 PyTorch 框架下的深度学习模型转换为静态二进制格式，并提供 C/C++ 接口；随后，利用 Fortran 与 C/C++ 混合编程技术，将深度学习模型部署到数值模型中。与其他耦合方法相比，FTA 更加灵活，但混合编程过程中仍存在一些代码兼容性问题。同时，由于 LibTorch 遵循 C/C++ 编程规范，耦合执行期间的调试和稳定性测试在灵活性与效率方面仍显不足，而且 FTA 功能有限，无法满足人工智能与数值天气预报融合的全部需求。

人工智能模型与数值模型的耦合通常需要频繁调用同一网络进行快速推理，并进行密集通信，因此适合使用基于 LibTorch 的编译库工具进行实时链接，以确保混合模式高效运行。因此，开发一种全面的耦合中间件，使其支持异构平台部署、兼容主流神经网络模型，并能够映射不同的并行粒度以及实现高效并行执行，是实现人工智能与数值天气预报深度融合的核心要求。

2. TorchNWP 引言

为解决人工智能推理模块与数值模型耦合过程中面临的技术挑战，包括跨语言接口设计不一致、运行模式不兼容以及数据传输方式不统一等问题，我们开发了 TorchNWP——一款功能完备、高效稳定且适应性强的编译库工具，可有效实现人工智能组件与传统数值模型的耦合。
TorchNWP 基于标准化的接口设计、跨语言绑定、高效的模型加载与推理、全面的性能优化以及严格的测试与验证，使深度学习模型能够以低成本、高效率的方式集成到传统数值天气预报模式中。同时，通过标准化构建切线线性与伴随模式，它简化了四维变分资料同化系统的开发，为人工智能与数值天气预报技术的深度融合提供核心支持，并有助于提高数值天气预报的准确性和计算效率。该编译库工具的结构流程图如图1所示。具体说明如下：

（1）接口设计。接口设计基于 C++-TorchScript 开发，核心目标是在 Fortran数值模型与 PyTorch深度学习模型之间构建简洁、统一且高效的交互桥梁，屏蔽底层技术复杂性，降低用户调用难度。具体而言，通过在 C++ 层显式定义三个核心接口函数，实现 Fortran 与 PyTorch 模型的全流程交互。各函数的功能与作用如下：

The model_new 函数：主要负责模型初始化，包括分配模型运行所需的内存资源、加载模型配置参数以及初始化 LibTorch运行时环境，为后续的模型加载与推理奠定基础。该函数支持接收用户配置的参数，例如模型存储路径和运行时设备（CPU/GPU），灵活性良好，可适应不同场景下的模型初始化需求。

The model_forward 函数：作为模型推理的核心接口，负责接收从 Fortran 数值模型传入的输入数据，调用模型完成推理计算，并将推理结果返回给 Fortran 数值模型。该函数集成了数据格式验证和维度转换等内置辅助功能，以确保输入数据符合模型推理要求，同时保证推理结果的准确性和完整性。

The model_delete 函数：主要负责模型运行后的资源释放，包括释放模型占用的内存资源、销毁 LibTorch运行时环境，以及清除输入和输出数据缓存。这可防止内存泄漏问题，确保工具长期稳定运行，并满足数值模式长期持续集成的运行要求。

上述三个核心接口函数协同工作，构成了 Fortran数值模型调用 PyTorch 模型的完整链路。用户无需关注底层 C++ 和 LibTorch 的实现细节，只需调用相应的接口函数，即可完成深度学习模型的初始化、推理和资源释放，从而显著降低跨语言耦合的难度。

（2）Fortran 与 C++ 的绑定。由于 Fortran 开发的传统数值模型与封装在 C++ 中的深度学习模型接口之间存在固有的语言壁垒，包括数据类型、内存布局和函数调用规范方面的差异，本发明采用 Fortran/C/C++ 混合编程技术，并借助 ISO_C_Binding 模块，实现 Fortran 与 C++ 的深度绑定。这彻底解决了跨语言兼容性问题，并实现了两种语言之间的高效数据传输和函数调用。作为 Fortran 语言提供的标准化接口，ISO_C_Binding 模块定义了 Fortran 与 C++ 中各种数据类型之间明确的对应关系。它可以将 Fortran 中的基本数据类型（例如整数、浮点数、逻辑值）、数组类型和指针类型准确转换为其

在 C++ 中对应的类型，从而消除因数据类型不兼容而导致的调用失败、数据损坏等问题。例如，它将 Fortran 中的 REAL(8) 类型转换为 C++ 中的 double 类型，并将 Fortran 中的二维数组转换为 C++ 中的指针数组，确保两种语言之间的数据传输正确无误。

(3) 模型加载与推理。在模型加载方面，该工具利用 LibTorch 提供的 torch::jit::load 函数，加载基于 PyTorch 框架开发并导出的预训练深度学习模型（通常以 .pt 文件格式存储）。.pt 文件完整保留了模型的所有结构信息（包括网络层数、每层的神经元数量、卷积核参数、激活函数类型等）和参数信息（包括权重、偏置、归一化参数等）。torch::jit::load 函数能够准确读取文件中的信息，重建模型结构、加载模型参数，并完成模型的初始加载。该加载方法支持各种主流 PyTorch 模型，包括卷积神经网络（CNN）、Transformer、扩散模型、残差神经网络（ResNet）等，适用于数值天气预报领域的各种应用场景。此外，该方法还具有加载速度快、稳定性高的特点，能够满足数值模式的实时运行要求。

在模型推理方面，模型加载后，会调用 module.forward 函数对输入数据执行推理计算。作为 LibTorch 框架提供的标准化推理接口，module.forward 函数可以接收经过格式转换的输入数据，按照模型的网络结构依次执行各层的计算操作，并最终输出推理结果。该函数支持批量推理，能够高效处理数值模式传入的大规模输入数据，从而提高推理效率。

需要注意的是，由于 Fortran 和 C++ 在数组内存布局方面存在根本差异（Fortran 使用列优先存储，而 C++ 使用行优先存储），直接将 Fortran 中的输入数据传递给模型进行推理会导致数据损坏，并影响推理结果的准确性。因此，在推理过程中，该工具会调整输入数据的维度，以适配 Fortran 的内存布局，确保模型能够正确识别和处理输入数据。具体而言，通过调整数据维度的顺序，将以 Fortran 列优先格式存储的数组转换为模型可识别的行优先格式。这既能避免由内存布局差异导致的推理错误，也能尽可能减少数据调整过程中的性能损失。

模型加载与推理的整个过程都封装在 C++ 接口函数中。Fortran 用户无需关注底层实现细节，只需通过接口函数传入输入数据并获取推理结果，即可完成深度学习模型的调用。这大幅降低了耦合难度，并提升了耦合效率。

（4）性能优化。针对深度学习模型与数值模型耦合过程中数据传输效率低、内存使用量过大、运行速度慢等问题，为确保该工具能够满足数值模型长期持续集成和大规模数据处理的需求，本工具采取了多种有针对性的性能优化措施，全面提升运行效率和稳定性。具体优化策略如下：

首先，为了减少 Fortran 与 C++ 之间因内存布局差异造成的性能开销，使用 PyTorch 中的 permute函数调整输入和输出数据的维度，从而实现数据布局的自适应转换，并避免不必要的数据复制。permute函数可以快速重新排列维度顺序，将 Fortran 以列优先方式存储的数据转换为 C++ 的行优先格式，或进行反向转换。整个过程只修改索引映射，不会产生额外的数据复制，从而显著降低数据传输过程中的性能开销，并提高数据传输效率。

其次，该工具支持多设备执行。用户可以根据实际计算资源，灵活选择在 CPU 或 GPU 上运行深度学习模型，实现计算资源的合理分配并加速模型推理。当 GPU 资源可用时，该工具会自动启用 GPU加速推理，利用 GPU 的并行计算能力，大幅提升大规模数据的推理效率。当没有可用的 GPU 时，该工具会自动切换至 CPU 模式，以确保正常运行。

此外，该工具采用即时内存释放机制，在推理完成后立即释放输入/输出数据和中间计算结果所占用的内存，从而防止内存泄漏和过度的内存消耗。为处理数值天气预报中常见的大数据量，工具采用分块数据处理技术，将大规模数据拆分为小块，并以批次加载和推理，以避免单个批次数据过大导致的运行时卡顿和内存溢出。同时，利用高效的压缩算法优化模型参数的存储方案，在不影响推理精度的情况下，降低存储和内存开销。这些策略共同提升了该工具的整体性能。

（5）切线线性与伴随模式的标准化构建。基于神经网络的切线线性与伴随技术在 C++ 层实现，以屏蔽神经网络模型内部结构的影响，并提供标准化的切线线性与伴随接口。具体实现过程如下：

正向计算：数值模型输入的物理量（包括数值天气预报过程中使用的各种物理参数，如温度、湿度、风速等）经应用接口层转换后，传递至封装在 C/C++ 层的深度学习模型，完成正向推理并获得模式输出结果。

切线性计算：利用 LibTorch 的自动微分工具，跟踪正向计算过程中的所有可微操作（如卷积、矩阵乘法、激活函数等），并计算深度学习模型输出相对于输入的雅可比矩阵，从而生成切线性模式。切线性模式满足y′ ，其中 为输入扰动，y 为输出扰动， 为雅可比矩阵，可以准确表征输入扰动对输出结果的影响。

伴随计算：构造一个标量损失函数 y ∗（其中 y 是模型的正向输出结果，∗ 是伴随变量）。利用 LibTorch 的自动微分工具，计算标量损失函数相对于输入物理量的梯度，从而得到伴随模型的结果 ∗ T ∗（其中 T 是雅可比矩阵的转置）。

接口封装：将上述切线性计算过程和伴随计算过程封装为标准化的 C/C++ 接口，并通过 ISO_C_Binding 模块与 Fortran 语言绑定。这样，Fortran数值模型便可直接调用这些接口，从而屏蔽神经网络内部结构的差异。

此外，还需要验证通过自动微分算法得到的神经网络参数导数与正向模型梯度的一致性，以及通过自动微分构建的机器学习伴随模型与基于伴随模式的传统梯度方法的一致性。通过严格的验证流程，确保切线线性与伴随模式的正确性和可靠性，为构建四维变分资料同化系统提供有力支持。

（6）测试与验证。在该工具的开发过程中，我们进行了严格的测试与验证，涵盖功能测试、性能测试、兼容性测试等方面。我们针对多种 Fortran 代码和 PyTorch 模型开展了测试，以确保该工具在不同场景下的稳定性和效率。

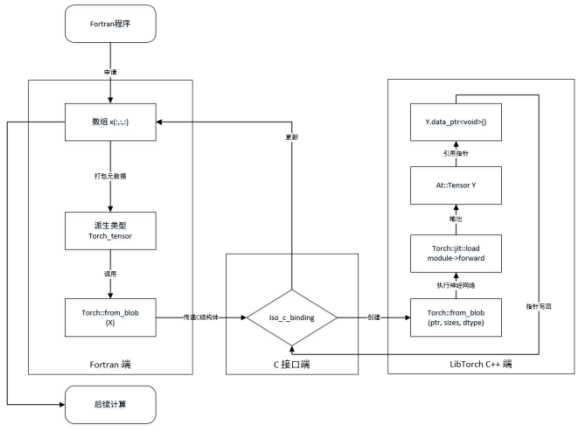

图1. TorchNWP编译器库工具的结构流程图。

3. 使用 TorchNWP 构建基于神经网络的切线线性与伴随模式

本节通过一个具体示例，说明 TorchNWP 的实现方式和运行流程。我们以中国气象局开发的中国气象局全球预报系统（CMA-GFS）作为应用平台。作为国家级核心业务数值天气预报模式，CMA-GFS 每天运行四次业务预报，并提供高精度的全球 10 天天气预报。作为其核心物理过程之一，非地形重力波拖曳模块在物理过程模拟中占用了大量计算时间。我们使用 TorchNWP 库，将经过充分训练的高精度非地形重力波拖曳神经网络参数化方案耦合到 CMA-GFS 模式中。同时，我们基于 TorchNWP 构建了与该神经网络对应的切线线性与伴随模式，以支持四维变分资料同化系统。具体实现如下：

3.1 非地形重力波拖曳高精度神经网络模型的训练与耦合实现

首先，根据非地形重力波拖曳的物理机制，选取深度学习模型的输入和输出物理量，并构建覆盖不同季节和天气状况的高质量训练数据集。此举旨在使深度学习模型能够准确识别和学习非地形重力波拖曳过程中的非线性特征、时空演变规律及复杂影响因素，为后续模型训练和数值模型耦合奠定数据基础。具体而言，输入物理量包括温度、湿度、风速、气压、位势高度和纬度等 11 个核心气象参数；输出物理量包括与非地形重力波拖曳相关的 5 个关键物理量。数据集总时长为一年。采用冷启动方法进行模式积分，每个案例积分 10 天，并按小时输出数据，最终数据总量为 41 TB。所有数据均经过严格的质量控制、异常值剔除、缺失值填补和标准化预处理，以确保数据的准确性和完整性。

考虑到数值模型的业务特性，基于深度学习的非地形重力波拖曳参数化方案采用了针对单个垂直网格柱进行独立处理的设计理念。也就是说，该模型将 CMA-GFS 模型中的单个垂直网格柱作为基本处理单元，输入和输出数据均为由多物理变量组成的一维向量。这种设计能够与原数值模型的网格结构和计算逻辑完美匹配，显著降低神经网络模型与数值模型之间的耦合难度，并提高耦合后的业务稳定性。因此，深度学习模型的输入和输出数据分别为大小为 (11, 89) 和 (5, 89) 的二维数组，其中第一维对应输入/输出物理变量的数量，第二维对应

垂直模型层的数量。特别需要指出的是，纬度对非地形重力波拖曳的模拟精度有显著影响。因此，将纬度作为核心输入变量纳入模型，以确保模型能够适应不同纬度区域非地形重力波拖曳的特征。

在数据预处理阶段，归一化方法的选择会直接影响模型训练性能和推理精度。经过多轮对比实验，最终确定了最优归一化策略，具体如下：对于输入变量，采用逐变量逐层归一化方法，即分别针对每个输入物理变量和每个垂直模式层进行归一化处理。这样可以消除不同变量和垂直层之间的量级差异，避免模型训练过程中某些变量或层级占据主导地位。对于输出变量，采用逐变量全局归一化方法，即对每个输出物理变量进行全局统一归一化，以确保输出结果的一致性和合理性。该归一化策略的优势在于，能够准确适应非地形重力波拖曳过程中不同高度层物理特性的显著差异，有效提升模型捕捉微弱信号的能力，并减少系统偏差。

深度学习模型采用优化后的残差神经网络（ResNet）架构，在保证推理精度的同时，尽可能减少模型参数量和计算开销，从而满足数值模式长期持续集成的需求。通过使用一维卷积神经网络，模型能够高效提取物理量在垂直方向上的时空特征，并捕捉非地形重力波拖曳过程的垂直演变；网络的最后几层为全连接层，用于深入挖掘不同物理变量之间的非线性相关性，从而提升模型的拟合能力和泛化能力。为进一步提高耦合效率，输入和输出数据的归一化与反归一化操作被集成到深度学习模型内部，无需在数值模式中额外编写代码，简化了耦合流程。

实验结果表明，增加深度学习模型的参数数量对提高预测精度有显著的积极作用；然而，参数数量增加也会导致推理速度变慢。综合考虑预测精度和推理速度，并经过多轮网络结构优化后，目前的最优网络由两个包含 16 个输出通道的残差模块组成，每个模块包含两个一维卷积层；在第二个残差模块之后，再接一个额外的一维卷积层，将通道数从 16 降至 5。由于模型输出由三个向量和两个标量组成，因此在卷积层之后连接两个并行的全连接层，分别对向量和标量进行预测。

非地形重力波拖曳神经网络模型经过离线训练和优化。在测试集上，神经网络预测的非地形重力波拖曳相关物理量与模式输出的参考值之间的偏差极小，表明

训练得到的代理模型能够准确拟合非地形重力波拖曳过程。基于 TorchNWP 编译库工具，将上述训练好的深度学习模型耦合到原始数值天气预报模式 CMA-GFS 中。首先，将训练好的 PyTorch 模型保存为 .pt 文件，完整保留深度学习模型的所有结构信息、参数信息和归一化配置，以确保模型转换和调用过程中的无损精度。其次，使用 TorchNWP 库提供的编译脚本对 .pt 文件进行编译和转换，生成可供 Fortran 模型调用的 C++ 链接库；在编译过程中，可以灵活配置编译器类型、PyTorch 版本以及运行设备（CPU/GPU）等参数，以适应 CMA-GFS 模式的运行环境。第三，在编译并链接 CMA-GFS 模型以生成可执行程序时，将上述生成的 C++ 链接库与原始模式进行链接；同时，在原始模式中需要启用深度学习模型的位置编写少量接口调用代码，以调用 TorchNWP 提供的三个核心接口（model_new、model_forward、model_delete），完成模型初始化、推理和资源释放。

对耦合混合模式进行了长期持续集成测试和精度验证。结果表明，积分6小时后，与原数值模型相比，混合模式的10米风速平均绝对误差（MAE）为0.055 m/s，500百帕风速为0.027 m/s，500百帕温度为0.013 K。预报性能的详细评估见图2。混合模式能够稳定完成10天连续积分，且不会出现内存泄漏、运行时卡顿或解发散，因此满足数值天气预报的业务要求。在计算速度方面，原模型和深度学习模型目前均在CPU上运行；当并行规模低于8000核时，深度学习模型的推理速度优于原有的非地形重力波拖曳物理方案，计算效率提高一倍。然而，在更大的并行规模下，深度学习版本会出现反向加速异常，这一问题可通过将深度学习组件卸载至GPU运行来避免。在整个耦合过程中，数值模型与人工智能推理模块在同一进程内计算，无需额外的进程间通信，从而彻底消除了进程间通信造成的开销，进一步提高了耦合效率和运行稳定性。

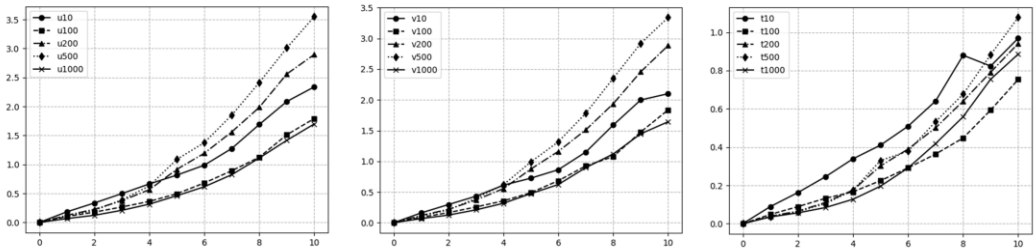

图2. 耦合非地形重力波拖曳高精度神经网络模型后的预报性能评估。

3.2 神经网络切线线性与伴随模式的实现

神经网络模型与数值模型耦合后，为适应四维变分资料同化系统，需要构建与神经网络模型相对应的切线线性与伴随模式。采用传统方法构建切线线性与伴随模式时，需要深入理解神经网络的内部结构，并手动推导切线线性关系和伴随方程。这种方法不仅工作量巨大、流程繁琐，而且极易出现推导错误。同时，它无法适用于结构不同的神经网络模型，通用性较差。本发明通过在C/C++ 层实现基于神经网络的切线线性与伴随技术，并借助 LibTorch 的自动微分工具屏蔽神经网络模型内部结构的影响，从而能够快速、准确地构建标准化的切线线性与伴随模式。

设 表示输入，y 表示输出， 表示神经网络，则切线性模式为 ，其中 是输入扰动， 是输出扰动。

$$J=\begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \cdots & \cdots & \cdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

表示神经网络关于输入的雅可比矩阵。对于伴随模型，点 x 处的伴随模型可定义为 $\left(\partial M(x)\ \partial x\ \ x=\right.$ $)_T$，并且以下关系也成立：

$$\frac{\partial k}{\partial y}=z$$

$$\frac{\partial k}{\partial x_0}=\left(\frac{\partial y}{\partial x_0}\right)^T\frac{\partial k}{\partial y}=\left(\frac{\partial M(x)}{\partial x}|_{x=x_0}\right)^T z\ ,$$

结合图 3 中的计算图，已知点 y 处的伴随为 ，将$_0$输入神经网络模型 ，即可得到推理状态 y。然后通过 y 与 的点积构造一个标量。该标量关于 的梯度给出点$_0$处的伴随结果。

基于上述推导，在 Python 中，雅可比矩阵的计算以及相对于$_0$的梯度计算，都可以借助 PyTorch 的自动微分工具实现。其根本原因在于，神经网络是由一系列可微操作（例如卷积、激活函数和矩阵乘法）构成的复合函数，而自动微分能够准确追踪这些操作基于链式法则的导数计算过程。它既能避免数值微分误差和符号微分中的冗余计算，又能高效输出雅可比矩阵（多输入多输出情况下的完整偏导数）和梯度（单个输出相对于多个输入的偏导数），满足神经网络的导数计算需求。

这也表明，通过自动微分计算梯度的机制，本质上与传统切线线性与伴随模式的计算方案完全相同。

考虑到需要与基于 Fortran 的四维变分资料同化（4D-Var）系统耦合，我们采用编译器库工具。通过使用 Torch 的 C++ 版本，我们借助 C++ 在耦合中间层实现了自动微分功能，从而构建了基于神经网络的切线线性与伴随模式。由于模型信息完整保存在 .pt 文件中，这种切线线性与伴随模式的实现方案能够屏蔽内部模型结构的影响，并简化四维变分资料同化系统的构建过程。

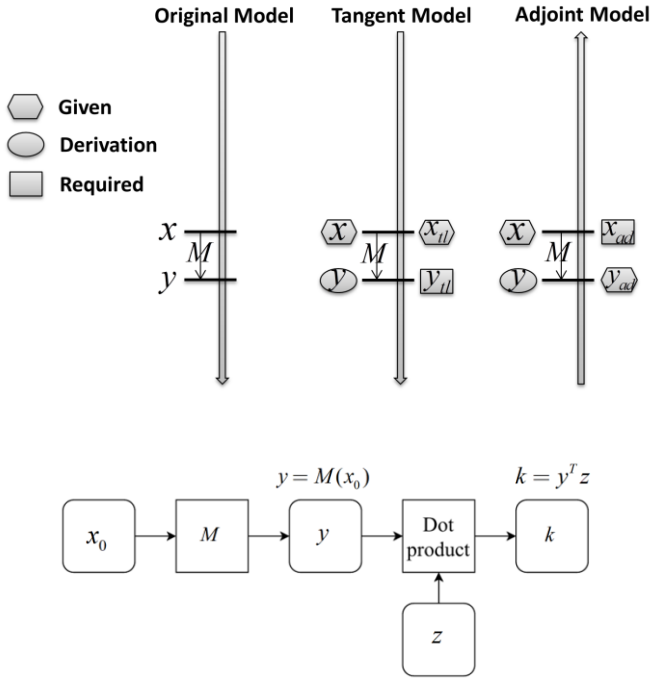

图 3. 切线线性与伴随模式的计算图。

4. 结论

TorchNWP 已成功集成到 CMA-GFS 等国家级数值天气预报（NWP）模式中，支持辐射和非地形重力波拖曳等多种物理过程的深度学习参数化方案耦合。其核心优势总结如下：

(1) 兼容性：解决了基于 Fortran 的数值模式与基于 Python 的深度学习框架之间的跨语言兼容性问题，支持残差网络（ResNet）和 Transformer 等主流深度学习模型，并兼容 CPU/GPU 异构平台；

(2) 高效性：模型调用在同一进程内完成，消除了通信开销，实现了高数据传输效率和快速模型推理；(3) 低侵入性：仅需编译、链接并编写少量接口调用代码，无需对原有数值模式进行大幅修改，从而降低耦合成本；(4) 可扩展性：支持灵活配置编译器、模型参数和并行粒度，能够适应不同数值模式的定制化需求；(5) 同化适应性：统一切线性模型/伴随模型的接口，屏蔽神经网络的黑盒特性，简化四维变分资料同化（四维变分）系统的构建过程，并降低同化适配成本。

致谢

本研究获得国家自然科学基金委员会（项目编号：U2342220）资助。我们还感谢张林博士和刘永柱博士参与讨论。

参考资料

[1] Ott J、Pritchard M、Best N 等。一种用于科学计算的 Fortran-Keras 深度学习桥接工具。Scientific Programming，2020，2020：1-13。[2] Wang X、Han Y、Xue W 等。使用包含大气湿物理和辐射进程神经网络参数化的真实大气环流模式进行稳定气候模拟。Geoscientific Model Development，2022，15(9)：3923-3940。[3] Mu B、Chen L、Yuan S 等。一种耦合至 WRF 的辐射传输深度学习模型及通用 Fortran Torch 适配器。Frontiers in Earth Science，2023，11：1149566。