arXiv:2601.04826v1 [cs.CE] 8 Jan 2026

Zoomy：用于自由表面流的灵活建模与仿真软件

Ingo Steldermann[1*]和 Julia Kowalski[1*]

[1]亚琛工业大学计算工程中基于模型的开发方法讲席，机械工程学院，艾尔弗斯霍恩施泰因街 18 号，亚琛，邮编 52062，德国。

*通讯作者。电子邮箱：steldermann@mbd.rwth-aachen.de；kowalski@mbd.rwth-aachen.de；

摘要

自由表面流与许多水资源工程、地质灾害评估以及海岸与河流工程领域的研究者密切相关。人们提出了多种自由表面模型，其建模复杂度涵盖从静水Saint-Venant方程到雷诺平均纳维–斯托克斯方程的不同层次。通过深度平均可以推导出特别高效的方法，从而得到降维模型。通常，这会产生一系列模型——其系统结构随流动变量的多项式展开而变化的模型——这些模型需要进行分析和数值求解。对这类模型进行描述、分析和仿真颇具挑战性，而现有软件解决方案仅涵盖这些模型层级所生成模型的特定子集。我们提出了一种新的软件框架来解决这一问题。*Zoomy* 支持对自由表面流模型的深度平均模型层级进行高效描述、符号分析和数值求解。*Zoomy* 可处理非结构网格上一个和两个空间维度的数值离散。

借助这一框架，可以对深度平均自由表面流的层级进行系统评估。此外，我们的开源框架让有意采用高效自由表面流方法的应用工程师更容易使用这些深度平均系统。

关键词： 浅层流，浅水，RSE，矩方法，非静水自由表面流，模型层次结构

1 引言

计算性能谱的一端可由一维静水Saint-Venant方程 [1] 来定义。它也称为一维及更高维度的浅水方程（SWE）。雷诺平均纳维-斯托克斯方程（RANS） [2, 3] 则代表另一端。与典型的RANS模拟 [4],相比，SWE的计算效率更高。这是因为沿流动的垂向范围进行深度平均，并假设静水压力分布，从而降低了模型的维度。

实际上，在许多应用领域中，SWE 过于受限，尤其是在关注速度分布时。低复杂度的 SWE 与复杂度更高的 RANS 方程之间存在各种替代模型，其中一些列于表1中。

表1：浅层流建模概览

Model	维度	分层	Code 可用性	参考文献
浅水方程（SWE）	一维,二维	✗	✓	[1]
浅水方程（SWE）	3D	✗	✗	[5]
浅层动量方程（SME）	一维、二维	✓	✗	[6–8]
非流体静力浅水矩（N-SME）	1D	✓	✗	[9]
Serre-Green-Naghdi（SGE）	一维、二维	✗	✓	[10, 11]
垂向平均动量（VAM）	一维、二维	✓	✗	[12–16]
剪切浅流（SSF）	一维、二维	✗	✗	[17, 18]
双层非水静力（L2NH）	1D	✗	✗	[19]
多层非水静力（LDNH）	1D	✓	✗	[20]

表 1 所示的众多已提出模型中，商业软件和开源代码的可用性有限。其次，本文尤其关注的是这些模型中许多模型所具有的层次结构特性。

这里的“层次结构”指方程组规模可变。这是因为表1中的深度平均模型通过依赖于垂直方向的基函数 $\phi_i(z)$，将速度（u, v, w）或压力 p 等流动变量表示为多项式展开。例如$u(t, x, y, z) = \sum_{i=0}^{N} \alpha_i(t, x, y)\phi_i(z)$，其系数为 $\alpha_i(t, x, y)$，基函数为$\phi_i(z)$。

目前，还没有统一框架能够高效地分析并数值求解此类分层模型。为了说明一些软件设计原则，我们首先考虑以下一维 SME系统：

$$
\begin{aligned}
&\partial_t(h) + \partial_x\left(h\alpha_0\right) = 0 \\
&\partial_t\left(h\alpha_k\right)\langle\phi_k,\phi_k\rangle + \partial_x\left(\sum_{i,j=0}^{N} h\alpha_i\alpha_j\langle\phi_i\phi_j,\phi_k\rangle + \frac{g}{2}e_z h^2\langle 1,\phi_k\rangle\right) \\
&\quad - u_m\partial_x\left(h\alpha_k\right)\langle\phi_k,\phi_k\rangle + \sum_{i,j=1}^{N}\alpha_i\partial_x\left(h\alpha_j\right)\langle\phi_i\Phi_j,\phi_k'\rangle \\
&\quad = -\left(\frac{\tilde{\sigma}_{xz}}{\rho}\phi_k\right)\Big|_{\zeta=0}^{1} + \left\langle\frac{\tilde{\sigma}_{xz}}{\rho},\phi_k'\right\rangle - \langle 1,\phi_k\rangle g\, h\partial_x b,
\end{aligned}
\tag{1}
$$

其中 $k \in \{0,...,N\}$， h 为水深， b 为底部地形，$\tilde{\sigma}_{xz}$ 为剪切应力， $\alpha_0 = \bar{u}$ 为平均垂向速度， g 为重力加速度。此外，令 $\langle\cdot,\ \cdot\rangle$ 表示沿垂直方向的深度积分，而 $\Phi_j(\zeta) = \int\phi_j(\zeta)\,d\zeta$。

有关技术细节的深入说明，例如 σ坐标变换（从 $z \in [b,s]$ 到 $\zeta \in [0,1]$ 的映射，其中 s 为自由表面）、一般推导过程，以及特定符号的更多细节，可参见 [6]。

方程（1）突显了我们对分层自由表面流模型提出的大部分要求：

- 分层特性。系统（1）由$(N+1)$ 个方程组成。
- 灵活的基函数定义。所得系统取决于基函数的选择 $\{\phi_i\}$，且应当可以互换。

- 基函数符号积分。对基函数乘积进行深度积分，例如 $\langle\phi_i\phi_j,\ \phi_k\rangle$，需要进行符号积分。这对于线性稳定性分析或计算系统特征结构等重要建模任务是必需的。

- 材料闭合。不同的材料闭合会为所得偏微分方程组产生不同的源项。我们希望能够灵活地指定不同材料。
- 非守恒系统。对于包含 $N > 0$ 的任何系统，该系统都包含非守恒项。这要求使用合适的数值方法进行离散化。

其他分层模型（例如 VAM 方程）还需要处理非静水压力分布。

仅凭快速原型开发框架（例如 FenicX [21], Firedrake [22], 或 DUNE- FEM [23], ）即可灵活定义模型，并且非常适合在我们所关注的领域中实现和比较深度平均模型。然而，在模型开发过程中，代数运算、特征结构的解析计算或线性稳定性分析都是关键的建模任务，仅凭这些数值框架无法完成。

下面我们将介绍*Zoomy*，这是一款可高效描述、符号分析和数值求解分层自由表面模型的软件。

这段代码旨在弥合有限的一维示例中孤立的模型开发与应用工程师需要在真实世界测试案例中测试和比较这些可能颇具吸引力的模型之间的差距。尤其是，它支持自动探索由模型层级生成的不同模型。

我们的框架涵盖一般的降维自由表面流应用，其方程形式如下

$$\begin{cases} \partial_t \mathbf{Q} + \nabla \cdot \underline{\underline{F}}(\mathbf{Q}) + \underline{\underline{\underline{N}}}(\mathbf{Q}) : \nabla \mathbf{Q} = \mathbf{S}(\mathbf{Q}, \nabla \mathbf{Q}, \nabla^2 \mathbf{Q}, ..., \mathbf{P}, \nabla \mathbf{P}, \nabla^2 \mathbf{P}, ..) \\ \mathbf{R}(\mathbf{Q}, \nabla \mathbf{Q} \nabla^2 \mathbf{Q}, ..., \mathbf{P}, \nabla \mathbf{P}, \nabla^2 \mathbf{P}, ...) = \mathbf{0} \quad . \end{cases} \tag{2}$$

其中 $\mathbf{Q} \in \mathbb{R}^N$ 是由 N 个未知（输运）变量组成的向量， $\nabla \cdot \mathbf{F}(\mathbf{Q})$ 是守恒通量， $N(\mathbf{Q}) : \nabla \mathbf{Q}$ 是一个非守恒乘积，而 $\mathbf{S}(\mathbf{Q}\ \nabla \mathbf{Q} \nabla^2 \mathbf{Q} ...)$ 是一个源项。 $\mathbf{P} \in \mathbb{R}^M$ 是由 M 个未知量组成的向量，通常与泊松型方程相关；这类方程由非静水压力约束产生，并由残差向量函数 $\mathbf{R}$ 定义。请注意，表1 中所列模型的所有偏微分方程均遵循偏微分方程结构（2）。

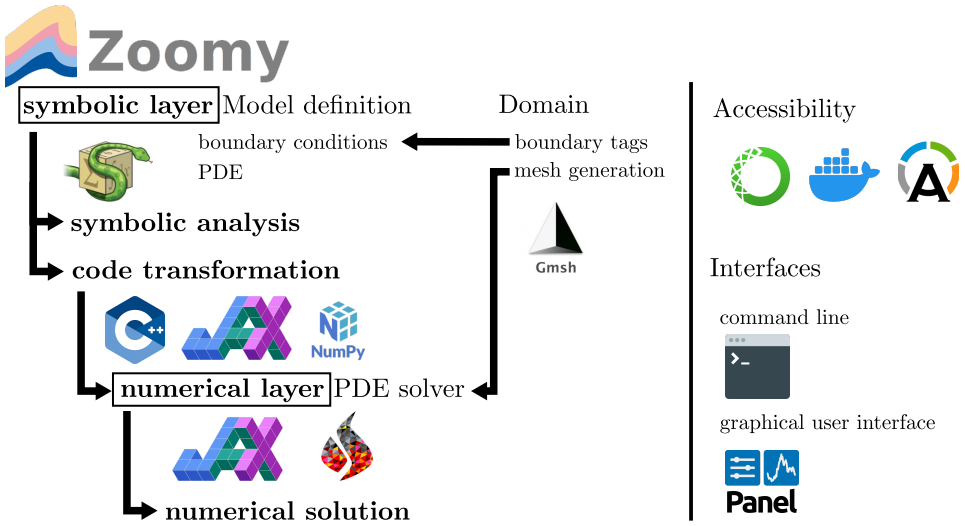

图 1：Zoomy 软件概览。该软件由两层组成：符号层和数值层。前者可对偏微分方程进行符号数学运算，后者则以数值方式求解偏微分方程。图标表示本项目中所用现有软件包的依赖关系及使用方式。图标说明：符号层：SymPy；代码转换：C++、JAX、NumPy；数值层：JAX、FenicsX；定义域：GMSH；可访问性：Conda、Docker、Apptainer；接口：命令行、Panel

图 1展示了通用软件架构的概览：

•*Zoomy* 由符号和数值两层组成。在符号层中，使用 Python 包SymPy 创建偏微分方程组的符号表示（2）以及边界条件。

• 符号层与数值层通过 SymPy 中提供的代码打印器连接。该打印器可将符号表达式转换为适用于不同后端的形式，包括 Python 包 NumPy 和 JAX，以及标准 C。

• 数值层由网格和求解器组成。为支持非结构化网格，我们使用开源工具 GMSH [24]创建的网格。我们提供了一些组件，可用于构建定制的有限体积法（FVM）求解器。我们还提供了一些通用实现，例如用于非保守系统的非结构网格双曲型有限体积法求解器，以及用于非静水压问题的校正预测格式。

代码打印器便于从符号层过渡到定制数值求解器或现有离散化框架。具体而言，我们提供了一个面向基于统一编程语言（UFL）的框架 FenicsX 的接口。深度平均自由表面流模型的推导既繁琐又难以理解，因为封闭的偏微分方程组包含大量方程，而且它们与底层纳维-斯托克斯系统之间的联系并不直观。因此，我们认为，让*Zoomy* 软件易于使用，并使结果具有可比性，是至关重要的。我们通过将*Zoomy* 实现为开源软件来确保其易用性，并以 conda环境、Docker容器和 Apptainer镜像的形式提供可直接使用的安装包，适用于高性能计算环境。此外，我们还提供一个简单的图形用户界面，可部署为云服务，以便用户无需安装即可使用。通过后处理步骤可以实现模型间的可比性：将系统的内部变量($\mathbf{Q}$，$\mathbf{P}$) 插值到非约化空间(t，x，y，z)中，从而得到输出变量 h，($\mathbf{u}$ p)。这一相对简单的补充对于模型的可解释性、可比性以及为用户提供友好的入门途径都至关重要。

本文的结构安排如下：第2节讨论软件的需求、架构和功能。第3节列举一系列示例，展示本软件在开发深度平均自由表面流求解器方面的适用性。第4 节讨论局限性、尚待解答的问题以及未来发展方向。

2 软件设计

Zoomy 源于这样一种需求：需要一个能够处理由浅层动量方程 [6]生成的模型层级的偏微分方程求解器。否则，要系统比较不同阶数的这类偏微分方程组，就必须手动实现每个模型，而对于高阶系统，这很快便会变得不可行。同样，高阶方程的符号分析也会变得愈发困难。围绕这一核心应用，还开发了图形用户界面（GUI）、用于自动执行后处理任务的脚本，以及其他偏微分方程求解器后端等工具，以便为自由表面流模型研究者提供更便利的软件。

2.1 核心

Zoomy 的核心 由一个 符号 层和一个 数值 层组成。其关键设计理念是，为式（2）中的偏微分方程寻找一种合适的符号表示，使其适用于典型的解析建模任务，随后可将其传递至数值 层，并转换为供数值求解器使用的数值代码。

2.1.1 符号层

该符号层需要

•从建模层级生成偏微分方程模型，需要对基函数乘积进行解析积分，并使系统（2）的规模具有灵活性，

• 对雅可比矩阵进行解析计算 $\frac{\partial F}{\partial}$ Q 以符号方式计算系统（2）的特征结构，并允许将（2）转换为完全拟线性形式，以供某些数值求解器使用，

• 对雅可比矩阵进行解析计算 $^{\partial}$S $_{\partial \mathbf{Q}}$ 以分析源项的刚性，或通过算子分裂为源项构造隐式求解器，

• 计算非静水压形式的线性稳定性分析，以及
•将符号表示转换为可用于计算数值求解的格式。

SymPy [25] 是一个用于符号数学的 Python 软件包，并提供代码打印器，以便将公式提供给数值层。偏微分方程模型的蓝图定义在（虚拟）类 Model 中。类规范的部分内容如下：

```
类 Model：""" 通用（虚拟）模型实现。 # 用户输入
名称：字符串 维度：整数 边界_条件：边界条件 初始条件：初始条件_ 场数：整数_ 变量：可迭代命名空间 辅助_变量：可迭代命名空间 参数：可迭代命名空间 定义 通量（自我）： 返回 [零矩阵（自我.n_场，1） 对于 d 在范围内 （自我.维度）] 定义 非保守_矩阵（自我）
```

返回 [零矩阵（自我.n_场，自我.n_场）**对于** d **在范围内**（自我.维度）]
定义 源项（自我）：**返回** 零矩阵（自我.n_场，1）**定义** 残差（自我）：
返回 零矩阵（自我.n_场，1）

上述代码片段凸显了该模型与偏微分方程密切相关，因为其定义要求对变量进行符号表示（$\mathbf{Q}$）、辅助变量和参数。引入辅助变量是为了表示常量标量参数场，以及注册运行时相关场，例如梯度，如 $\partial_x \mathbf{Q_i}$ 或状态方程。辅助变量的更新属于后文所述的数值层。类 边界条件可对后文讨论的边界条件进行符号表示。函数 通量、非保守_矩阵和源项的定义与公式（2）中的函数密切对应 F，N 和$\mathbf{S}$。

请注意，在实际应用中，数值实现公式（2）有多种方式。对于静水系统，压力相关变量 $\mathbf{P}$ 消失，所得系统通常是纯双曲型的。对于非静水系统，可以使用预测-校正法迭代求解 $\mathbf{Q}$ 和 $\mathbf{P}$。在我们的框架中，两者都可以通过两个模型来表示：第一个模型描述基础双曲系统，第二个模型则使用残差函数来表示压力相关约束。

通过继承基类 Model，可以构建浅水方程 (SWE)、浅层动量方程 (SME)、垂向平均动量方程 (VAM) 或 Serre-Green-Nagdhi 方程等偏微分方程组。

得益于符号表示，可以借助 SymPy 计算拟线性矩阵 $A \cdot \mathbf{n} = \left(\frac{\partial F}{\partial \mathbf{Q}}\right) + N) \cdot \mathbf{n}$ 的解析特征结构，其中 ~~$\mathbf{n}$~~是符号方向向量。

定义良态瞬态偏微分方程需要初始条件和边界条件。这两类函数都可以提供给模型。单个 BoundaryCondition 使用 SymPy 的符号表示，为定义域中的一个物理部分编码一个函数。由于边界条件通常需要针对特定偏微分方程进行定制，我们只提供了几种通用边界条件，包括 周期性边界条件、外推和 $Lambda$ 边界条件。后者提供了一个简单的接口来定义入流或流出边界条件，例如，它可以用自定义函数覆盖特定变量的外推条件。以下代码片段展示了如何为浅水方程 提供亚临界入流边界、流出边界处的外推边界条件，以及定义域顶部和底部的周期边界条件。方程的变量为 $\mathbf{Q} = (h, hu, hv)^T$，其中 physical_tag 对应于在 $GMSH$ [24]输入网格中创建的物理名称：

```
1 import library.model.boundary_conditions as BC 2 boundary_conditions = BC.边界条件(3 [4 # 入流流量为 0.1 m^3/s 5 BC.Lambda(physical_tag='inflow', prescribe_fields={6 1:lambda t,x,dx,q,qaux,参数,normal:0.1,7}),8 BC.外推(physical_tag='outflow'),9 BC.周期性边界条件(physical_tag='top',10 periodic_to_physical_tag='bottom'),11 BC.周期性边界条件(physical_tag='bottom',12 periodic_to_physical_tag='top'),13 ]14)
```

请注意，创建初始模型时无需定义网格。数值层会验证网格与模型的物理_标签是否一致。

最后一个符号层组成部分是模型层次结构的表示方式，正如引言中所述。对于式（1）所示的 SME 特例，我们添加了两个新的成员变量：level 和 basisfunctions。前者是一个整数，用于定义水平速度场的多项式阶数 u。后者是一个专为 SME 模型设计的类，用于计算模型公式中所需的基函数乘积的解析积分 $\phi_i(\zeta)$。通常使用勒让德多项式作为基函数。随后即可基于这类包含基函数乘积积分的张量，构建任意多项式阶数的 SME。

例如，SME 的非守恒_矩阵 可以写成系统（1）的形式

```
1 2 类 B a s i s 矩阵: 3 def __i n i t __( s e l f , basis =Legendre_shifted ( )
, 4 use_cache=True , cache_path=" . cache " ) : 5 s e l f . b a s i s f u n c t i o n
s =basis 6 7 def _B ( s e l f , k , i , j ) : " " " 8 9 Compute <( p h i ’ ) _k , p h i _
j , i n t ( p h i ) _j >" " " 10 11 return i n t e g r a t e ( 12 d i f f ( s e l f . b a s i s f
u n c t i o n s . eval ( k , z ) , z ) 13 *i n t e g r a t e ( s e l f . b a
s i s f u n c t i o n s . eval ( j , z ) , z ) 14 *s e l f . b a s i s f u n c
t i o n s . eval ( i , z ) , 15 ( z , 0 , 1 ) , 16 ) 17 18 19 20 class 浅层矩 ( Model ) :
21 def n o n c o n s e r v a t i v e _ m a t r i x ( s e l f ) : 22 nc_x =矩阵 ( [ [ 0 f
o r i in range ( s e l f . n _ f i e l d s ) ] 23 f o r j in range ( s e l f . n _ f i e l d s ) ] )
24 h =s e l f . v a r i a b l e s [ 0 ] 25 ha =s e l f . v a r i a b l e s [ 1 : 1 + s e l
f . l e v e l s + 1 ] 26 p =s e l f . 参数 27 um =ha [ 0 ] / h 28 f o r k in range (
s e l f . l e v e l s + 1 ) : 29 nc_x [ k + 1 , k + 1 ] —= um 30 f o r i in
range ( 1 , s e l f . l e v e l s + 1 ) : 31 f o r j in range ( 1 , s e l f . l e v e l s + 1 )
: 32 nc_x [ k + 1 , i + 1 ] += ( ha [ j ] / h
33 *s e l f . b a s i s m a t r i c e s . B [ k , i , j ]
34 / s e l f . b a s i s m a t r i c e s . M [ k , k ] 35 ) 36 return [ nc_x ]
```

代码片段展示了实现与（1）中的数学公式遵循得有多么一致。

符号公式可以转换为数值层的多种格式。对于我们的应用而言，转换为 JAX、NumPy 和 C 很有意义。转换要求符号层与数值层之间的接口固定。这意味着 Model 类中的所有偏微分方程相关函数都具有明确定义的输入和输出结构。

例如，我们规定通量或源项函数依赖于三个输入（Q、Qaux、参数），其中每个输入的类型均为 jax.numpy.ndarray、

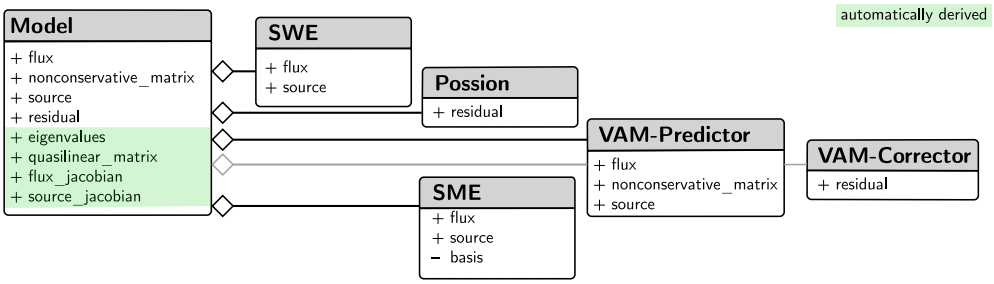

图 2：从 Model 基类派生的类。

numpy.ndarray 或双精度浮点数*，具体取决于代码转换。其他函数依赖于更多输入；例如，eigenvalues(Q, Qaux, parameters, normals) 还需要一个表示法向方向的额外输入。

当需要对 NumPy 或 JAX 中的向量化表达式进行代码转换时，会出现一个微妙却关键的问题。出于性能考虑，通常建议使用向量化表达式，尤其是在使用 NumPy 时。然而，对常数进行代码转换（常数常见于边界条件中，例如固定的狄利克雷值，或源项中，例如常数源项）会产生非向量化代码，因为常数并非与 Q 长度相同的数组。为解决此问题，我们采用一个简单的修复方法：为每个仅包含常数的表达式添加一个非常小的数（$10^{-20}\mathbf{Q}_0$），从而解决这一转换问题。

要在 Python 中创建运行时模型，可以使用例程 _create_runtime_model；而要将其转换为 C 库，则可以调用例程 create_runtime_model_C_library。

本文讨论的各种模型的继承关系和类组合如图 2 所示。基类`Model` 的继承机制允许创建新的符号模型，而固定接口则确保数值例程保持兼容。

2.1.2 数值层

数值求解器通常针对当前特定的偏微分方程模型进行定制。我们提供构建模块，以便构造新的数值格式。从较高层次来看，存在一个通用基类 Solver，其接口为

```
类 求解器：定义 _初始化（自身，Model，网格，设置）：""
"初始化 Q、Qaux，并收集基于网格的边界条件列表 """。。。
返回Q、Qaux定义 _加载_运行时_Model（自身，Model）：。。。
```

11 **返回** 运行时_偏微分方程、运行时_边界条件12 **定义** _获取_边界_算子（自身，网格，运行时_边界条件）：13 。。。14 @jax.jit15 **定义** 边界_算子（时间，Q，Qaux，参数）：16 。。。17 **返回**Q18 **返回**边界_算子19 **定义** solve（自身，网格，Model，设置）：20 。。。21 **返回**Q、Qaux

函数 `_initialize` 通过为两个数组分配内存，在符号模型与网格之间建立关联，`Q` 和 `Qaux`，并创建必要的映射，以确定两个向量中的哪些自由度属于哪个边界。函数 `_load_runtime_model`会触发从符号模型到数值模型的转换，并生成与 JAX兼容的函数。对于由（2）描述的任何系统，边界条件都是必需的，因此已纳入基础求解器。许多函数，例如 `get_边界_算子` 函数，都遵循以下模式

```
    定义 get_func(self, *静态_参数):...
    @jax.jit定义 func(*动态_参数):...返回
    return func
```

外层函数 get_func 接收在运行时视为静态的输入参数。内层函数则使用 (jax.jit) 构建并编译，且仅依赖动态参数，例如 (Q, Qaux, 参数)。这种模式便于追踪代码中哪些变量会参与自动微分。对于所有需要编译的耗时构件，也会重复采用相同的模式。

函数 `solve` 是 求解器 类的公共接口，其中包含数值格式的实现。我们已经提供了几种 求解器 类的专门实现，例如瞬态双曲型偏微分方程求解器（`HyperbolicSolver`）或稳态求解器（`SteadyResidualSolver`）。图 3总结了构建模块及其在一些求解器中的用法：

- 基于 [26–28] ，实现了适用于一维、二维和三维非结构网格上非守恒系统的有限体积格式，以求解双曲型_步骤。
- 将双曲型_步骤与一个边界_算子和时间循环相结合，实质上构成了一个瞬态双曲型求解器（HyperbolicSolver）。
- 同样，我们基于 JAX 中提供的 GMRES线性求解器实现了牛顿求解器（NewtonSolver）。该构建模块可用于创建一个稳态残差求解器 （SteadyResidualSolver）。

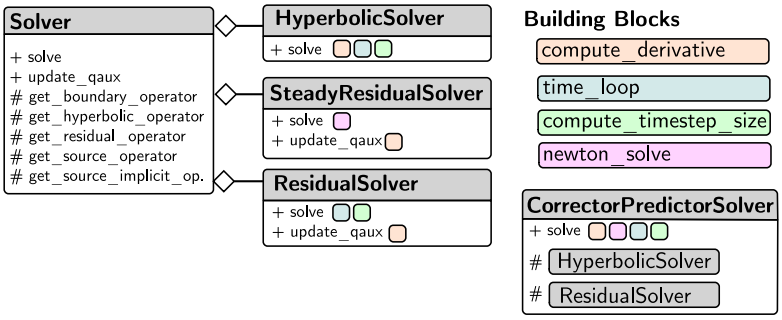

图 3：构建模块与可用的求解器类

由于我们的代码支持通过 JAX 进行自动微分，因此可以对残差函数求导，以构造牛顿求解器所需的解析雅可比矩阵，从而无需用户提供额外输入。

另一个构建模块是基于最小二乘重构的梯度重构，支持任意多项式阶数（compute_导数）。该实现支持一维、二维和三维非结构网格。为避免在边界处复杂地管理幽灵单元，我们将实现限制为单层幽灵单元，并通过收集网格内部区域中足够大的单元邻域，在边界处重构梯度。

非静水深度平均自由表面模型需要构建更先进的数值求解器。按照算子分裂格式 [16], 我们将以下半隐式格式确定为值得实现的候选方案:

explicit hydrostatic predictor (hyperbolic PDE)

$$\frac{(\mathbf{Q}^* - \mathbf{Q}^n)}{dt} + \nabla \cdot \underline{\underline{F}}(\mathbf{Q}^n) + \underline{\underline{\underline{N}}}(\mathbf{Q}^n) : \nabla \mathbf{Q}^n = \mathbf{0} \tag{3a}$$

semi-implicit non-hydrostatic corrector (Possion type PDE)

$$\mathbf{R}(\mathbf{Q}^{**}, \nabla \mathbf{Q}^{**}, \nabla^2 \mathbf{Q}^{**}, ..., \mathbf{P}, \nabla \mathbf{P}, \nabla^2 \mathbf{P}, ...) = \mathbf{0} \tag{3b}$$

$$\frac{(\mathbf{Q}^{**} - \mathbf{Q}^*)}{dt} = \mathbf{S}^P \left(\mathbf{Q}^*, \nabla \mathbf{Q}^*, \nabla^2 \mathbf{Q}^*, ..., \mathbf{P}, \nabla \mathbf{P}, \nabla^2 \mathbf{P}, ...\right) \tag{3c}$$

显式（或隐式）摩擦项（常微分方程）

$$\frac{\left(\mathbf{Q}^{n+1} - \mathbf{Q}^{**}\right)}{dt} = \mathbf{S}^F(\mathbf{Q}^{**}, \nabla \mathbf{Q}^{**}, \nabla^2 \mathbf{Q}^{**}, ...) \tag{3d}$$

在上述分步方案中，$\mathbf{Q}^n$,$\mathbf{Q}^*$,$\mathbf{Q}^{**}$ 和 $\mathbf{Q}^{n+1}$ 分别表示每个时间步 n的旧解、两个中间解和新解。源项分为压力部分和摩擦部分 $\mathbf{S}^P$ 和 $\mathbf{S}^F$，其中所有涉及压力 $\mathbf{P}$ 的项都并入$\mathbf{S}^P$。

注意，压力 $\mathbf{P}$ (3b) 的求法是，先将 $\mathbf{Q}^{**}$ 代入式(3c)，再代入式(3b)，得到一个仅与已知值 $\mathbf{Q}^{*}$ 和未知量 $\mathbf{P}$有关的泊松型方程。求出 $\mathbf{P}$后，$\mathbf{Q}^{**}$ 通过式(3c)中的更新步骤求得。

根据（3a）至（3d），我们确定了所需的代码构建模块

- 步进_双曲用于（3a）的,
- 牛顿_求解器用于（3b）的,
- 步进_源项用于（3c）和（3d）的,
- 步进_源项_隐式用于（3d）的隐式模式下,
- 边界_算子,
- 并计算_导数，以构造 $\nabla^{k}\mathbf{Q}$和 $\nabla^{l}\mathbf{P}$,

以实现预测-校正格式。

目前的实现提供了适用于同位网格数值格式的构件。尽管网格类可以访问节点和单元面，但仍需在后续工作中添加便捷函数，例如单元、节点和面之间的投影格式，以便在交错网格上轻松构建方法。

上述构件可用于构建许多不同的求解器，而不仅限于（3a)-(3d)中描述的瞬态半隐式格式。由于牛顿求解器的残差定义具有灵活性，全隐式格式同样可行。在当前版本中，我们提供了图3所示的求解器。

2.2 易于比较、获取和使用

高级深度平均自由表面模型的数值模型通常难以推导、修改和解读。我们认为，主要难点在于五个主要流动变量—— h，$(u,\ v,\ w)$ 以及表示水深、速度和压力的 p ——采用了不同的变量集进行描述，例如，h，$(u0,\ u1,\ v0,\ v1,\ w0,\ w1,\ w2)$，$(p0,\ p1,\ p2)$ 用于 VAM 模型 [16]。如果读者缺乏相关专业知识，也未仔细审视其推导过程，就很容易忽略变量之间的联系及其背后的物理直觉。

为了推广深度平均自由表面流模型的应用，我们软件的一项核心设计原则是确保不同模型之间具有可比性，让任何用户都能轻松使用该软件，并简化对新模型的初次体验，使用户无需具备偏微分方程方面的先验知识。

2.2.1 后处理

在工程应用中进行模型选择，或对新开发的模型进行基准测试等场景，都需要借助现有的参考实现，并要求这些实现具有可比性。为此，专门用于自由表面流模型的建模软件很有用。

因此，每个模型都包含一个可实现的 interpolate_3d 函数，用于恢复三维不可压缩纳维-斯托克斯或雷诺平均纳维-斯托克斯设置中的流动变量 h，$(u,\ v,\ w)$ 以及 p

该函数还可以扩展，以包含感兴趣的辅助变量。

这一简便的后处理功能有助于将使用三维流动求解器的广大科学家群体与深度平均自由表面模型联系起来。

我们会在求解器运行期间，将解存储为自定义的 $hdf5$ 格式，以便重新启动仿真；原则上，这也使多个数值后端与 $Python$ 和 $C++$ 等编程语言之间能够兼容。对于后续后处理，目前我们支持将数据转换为 VTK 格式，例如可在广受欢迎的开源科学可视化软件 $ParaView$ 中打开 [29]。

2.2.2 跨平台开源软件

Zoomy 作为开源软件发布，可通过以下网址获取：https://github.com/mbd-rwth/Zoomy。我们支持通过 $conda$ 包管理器 [30] 或从源代码手动安装，使项目更易于使用。此外，我们还支持适用于 Apptainer [31]和 Docker [32] 的预构建容器环境。

2.2.3 跨平台图形用户界面

提供以*Jupyter Notebook*形式编写的教程，例如托管在 Binder[33] 或 Google Collab [34] 等云服务上，可以让用户只需投入少量时间，就能轻松使用新软件。基于这一理念，我们旨在创建一个图形用户界面（GUI），作为用户探索我们的软件以及深度平均自由表面模型的便捷入口。与*Jupyter Notebook*不同，我们的目标是让用户与软件的首次交互尽可能简单，因此需要提供无需数学知识、无需编写代码的体验。另一方面，图形用户界面不应只是演示工具，还应允许用户探索和修改现有模型。

因此，我们使用 Python 软件包 Panel 开发了一个简单的图形用户界面。Panel 是一个用于开发仪表板和其他图形用户界面应用程序的工具。与其他解决方案相比，我们选择 Panel 作为图形用户界面平台，因为它具备以下特点：

- 可跨平台并支持云服务，因为 Panel 可以采用客户端-服务器架构运行
- 可原生支持 Matplotlib 等常见绘图库
- 可集成 ParaView Lite 等更复杂的可视化软件 [29]
- 提供开箱即用的代码编辑器界面

请注意，从我们的角度来看，最后一点至关重要，主要是因为它免去了我们构建和维护大量输入字段（如滑块或文本字段）的需要。相反，它让我们能够向用户提供两种不同的视图：精简简单的卡片视图 和功能齐全的代码视图。

图 *4*展示了卡片视图的示例。卡片视图显示多个卡片，用户可从中选择不同的现有网格。配置并运行仿真的工作流分为多个选项卡：

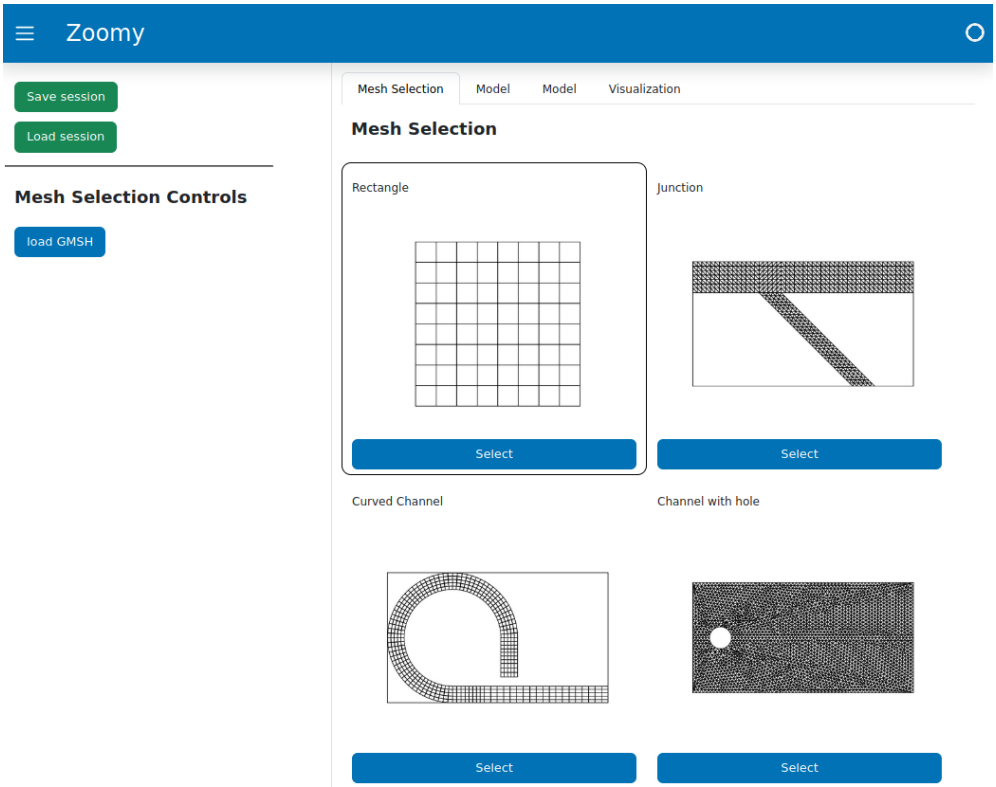

图 4：图形用户界面（GUI）。展示了用于网格选择的卡片系统。同一套卡片系统也用于选择模型、数值求解器和可视化工具。可以通过高级设置自定义所选模型，也可以直接在图形用户界面的代码编辑器中编辑各个类的代码。

网格页面、模型页面、求解器页面和可视化页面。每个页面都有一组卡片供用户选择。卡片视图左侧的侧边栏可用于放置经典图形用户界面组件，例如按钮、滑块或文本字段。在此图形用户界面的初始版本中，我们仅实现了一组最基本的附加元素，因为仿真仍可在代码视图中完全配置。

点击卡片上的相应按钮即可访问代码视图。这会打开一个新的页面，其中的代码编辑器会显示底层 Python 文件中的相应代码。

利用 Panel 提供的组件，我们开发了 页面和卡片视图，特别是引入了以下类：卡片、卡片管理器、页面和页面管理器。这些类之间的交互关系在我们的图形用户界面设计中进行了总结，如图所示。5。

目前我们支持三种不同的可视化后端：*ParaView Lite* 和*PyVista* [35] 用于二维和三维数据，Matplotlib [36]用于一维数据。

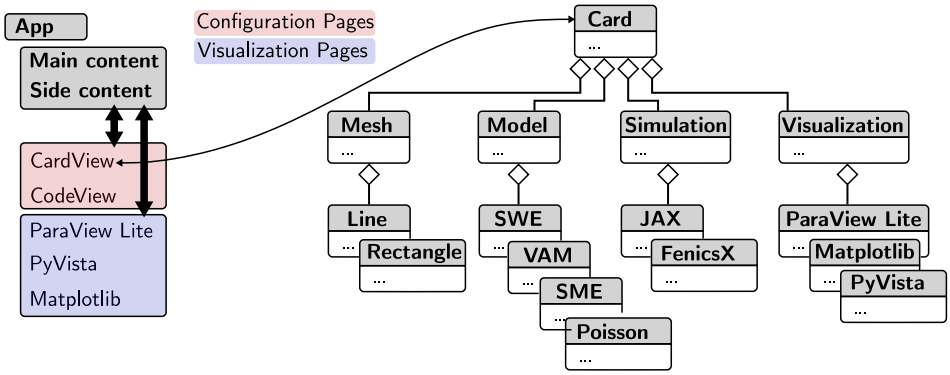

图 5：图形用户界面设计。

图形用户界面还可用作配置器，将测试案例的完整配置生成为 Python 脚本。因此，这些案例可复现，并且能自动兼容命令行工具。

3 建模示例

为了展示前文所介绍的各种应用场景和组件，我们将演示如何使用本软件构建不同的模型。首先，我们将介绍经典的一维和二维浅水方程，并展示瞬态双曲求解器。接着，我们将通过一个求解泊松方程的一维测试案例，展示如何使用牛顿求解器。然后，我们将结合这两个求解器，为 (3a)-(3d) 构建预测-校正格式，以求解非静水 VAM 模型。最后，我们将展示如何实现具有可变多项式次数的静水 SME 模型层次结构。

3.1 一维和二维浅水方程

在此测试案例中，我们展示

- 符号模型定义的工作原理，
- 多维模型的构建，
- 特征值可以通过符号计算得出，
- 符号模型与数值模型之间的桥梁，
- 双曲求解器的正确性，以及
- 如何使用 GMSH 网格。

此测试案例可在 tutorials/swe/simple.ipynb 文件夹中找到。

按照通用偏微分方程模板（2），无底部地形和摩擦的浅水方程为

$$\partial_t \mathbf{Q} + \nabla \cdot \underline{\underline{F}}(\mathbf{Q}) = \mathbf{0} \tag{4}$$

with

$$\underline{\underline{F}}(\mathbf{Q}) = [\mathbf{F}_x(\mathbf{Q}), \mathbf{F}_y(\mathbf{Q})] = \left[\begin{pmatrix} hu \\ hu^2 + gh \\ huv \end{pmatrix}, \begin{pmatrix} hv \\ huv \\ hv^2 + gh \end{pmatrix} \right], \tag{5}$$

其中 h, u, v 分别表示水深以及 x 和 y 方向的速度。g 表示 z 方向的重力加速度。

1 class ShallowWater（Model）： """ 2 3 浅水方程的一维和二维实现 """ 4 5 6 定义通量（自身）： 7 # 构造一个（维度，字段数）矩阵 8 通量 =矩阵（[[0对于 i，范围（自身. n _字段）]对于 j，范围（自身. 维度]）9 h =自身.变量[0]10 #hU = /hu /在一维和 /hu，hv /在二维 11 hU =自身.变量[1: 1+自身.维度]12 I =单位矩阵（自身.维度）13 参数 =自身.参数14 通量[:， 0] =hU 15 通量[:，1:1+自身.维度] =hU *hU.T + 1/2*g*h**2 *I 16 # 输出需要一个由（n_字段数，1）矩阵组成的列表，17 # 例如， /F_x/在一维中，以及 /F_x、F_y/在二维中 18 返回 [通量[i]对于 i，范围（自身.维度）]

为清晰起见，省略构造函数、初始代码和边界代码后，数学模型的实现便已完成。在底层，非保守_矩阵、源项和残差函数均初始化为零。根据通量（F）和非保守_矩阵（NC），可以通过求解广义特征值问题自动计算特征值。

$$\left| \frac{\partial \underline{\underline{F}}}{\partial \mathbf{Q}} \cdot \mathbf{n} + \underline{\underline{\underline{NC}}} \cdot \mathbf{n} + \lambda \underline{\underline{I}} \right| \stackrel{!}{=} 0 \tag{6}$$

其中$\mathbf{n}$ 是一个符号方向向量。

请注意，SymPy 计算出的特征值不一定采用有利于数值离散的形式。在我们的例子中，未经任何额外简化或假设约束的特征值如下：

$$
\begin{aligned}
\lambda_0 &= \frac{h\mathbf{u}\cdot\mathbf{n}}{h} \\
\lambda_{1,2} &= \frac{h\,h\mathbf{u}\cdot\mathbf{n} \pm \sqrt{gh^5}\sqrt{\mathbf{n}^2}}{h^2} \quad .
\end{aligned} \tag{7}
$$

这在数学上是正确的，但对$\mathbf{n}^2 = 1$ 进行简化并降低平方根中的幂次 h^5 会更有利。未来的版本中，我们将实现更多自动简化功能。不过，在代码转换为数值层之前，也可以通过重载现有的特征值函数功能来手动进行简化。

如第 2.2.1 节所述，我们建议实现 `interpolate_3d` 函数，使模型的结果易于与其他自由表面流模型进行比较。对于浅水方程，`z` 方向上的速度分布恒定以及静水压力这两项基本假设，得出了以下提升。

$$
\begin{aligned}
\rho(t,x,y,z) &= \rho_{water} * ((h(t,x,y)^{SWE} - z) \geq 0\,?\,1\,:\,0) \\
u(t,x,y,z) &= u^{SWE}(t,x,y) \\
v(t,x,y,z) &= v^{SWE}(t,x,y) \\
w(t,x,y,z) &= -h\partial_x u^{SWE}(t,x,y) - h\partial_y v^{SWE}(t,x,y) \\
p(t,x,y,z) &= \rho_{water}\, g\, (\max(h(t,x,y)^{SWE} - z, 0))
\end{aligned} \tag{8}
$$

请注意，如果显式考虑底部地形或空气相，上述公式会发生变化。

实现与符号定义密切对应

```
1 def interpolate_3d(self):2 param = self.parameters 3 h = self.variables[0]4 u = self.variables[1]/h 5 dudx = self.auxvariables.dudx_6 v = 07 dvdy = 08 if self.dimension == 2: 9 v = self.variables[2]0 dvdy = self.aux_variables.dvdy 1 rho = param.rho *分段函数((1,h -z),(0,True))
2 w = - h * dudx - h * dvdy 3 p = param.rho *param.g * 分段函数((h-z,
h-z > 0),(0,True))4 return rho,u,v,w,p
```

主要区别在于，空间导数以符号辅助变量的形式表示。

作为数值求解器，我们可以继承 TransientHyperbolic 求解器的实现。唯一需要自定义的是重载 update_qaux 函数，以计算速度场的梯度：

```
class SWESolver(TransientHyperbolic):""" 一维和二维问题的浅水方程求解器 """ def update_qaux(self, Q, Qaux, Qold, Qauxold, 网格, Model, 参数, time, dt): h = Q[0]u = Q[1]/h dudx = compute_derivatives(u, 网格, derivatives_multi_index =([[1, 0]]))[:,0]# 这是用于 Qaux 的 JAX 语法/0/ = dudx Qaux = Qaux.at[0].set(dudx) if Model.dimension == 2: v = Q[2]/h dvdy = compute_derivatives(v, 网格, derivatives_multi_index =([[0,1]]))[:,0]# 这是用于 Qaux 的 JAX 语法/1/ =dvdy Qaux = Qaux.at[1].set(dvdy) return Qaux
```

所有组件就绪后，最终二维仿真的运行器现在如下所示[1]...[2]

```
模型 = ShallowWater（维度 =2, 一个速度场 _fields =['dudx','dvdy'],参数 ={'g':9.81,'rho':1000}, 边界_条件=bcs, 初始_条件=ic, ）主_目录 =os.getenv("Zoomy") 网格 =petscMesh.Mesh.from_gmsh(os.path.join(main_dir,"meshes/quad_2d/mesh_coarse.msh")) 求解器 =SWESolver(计算_dt=时间步进.自适应(库朗数=0.45), 结束时间_.1, )
```

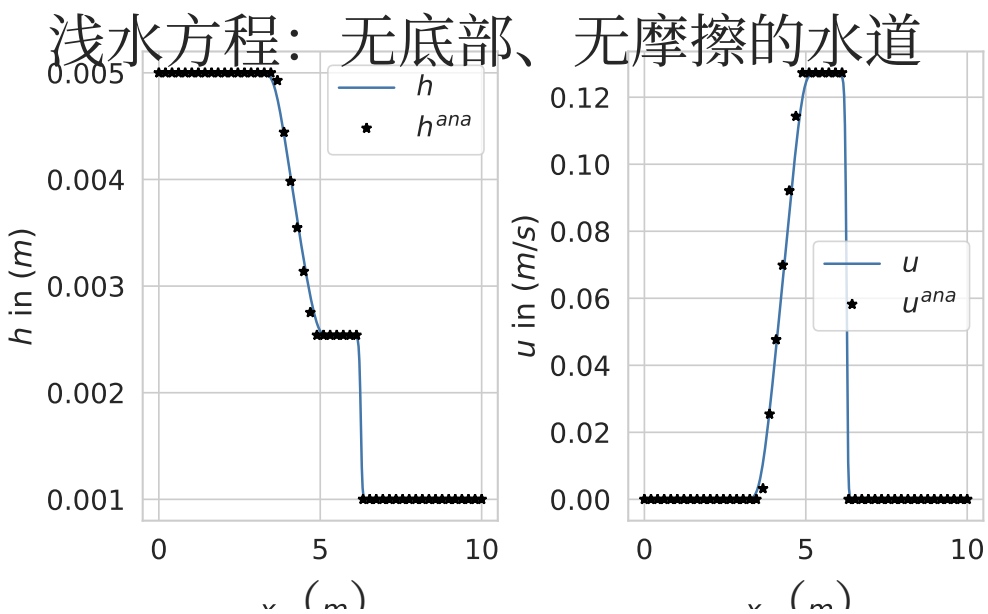

图 6：我们的数值求解与解析解的比较。为进行比较，我们的二维解沿 $y = 1$ 切片。

```
Qnew, Qaux = 求解器.solve(网格, 模型, 设置)

# 生成浅水方程场的 VTK 解io.generate_vtk(settings.output_dir)# 使用插值生成 VTK 解_3dio.generate_lifted_vtk(settings.output_dir)

```

最后，图中展示了数值求解及其与测试案例 无摩擦湿域溃坝 的 [37]比较，见图 6。

3.2 一维稳态泊松方程

由于求解非静水流方程需要求解泊松型系统，我们将通过简单的一维稳态泊松方程介绍这一构建模块。

在此测试案例中，我们展示

- 如何利用残差定义构建椭圆求解器
- SteadyResidual 求解器的正确性

此测试案例可在以下文件夹中找到：`tutorials/poisson/simple_1d.ipynb`。只考虑我们通用偏微分方程模板的（2.2）部分，即泊松方程

$$\begin{cases} \partial_{xx} T(x) = 2 \text{ for } x \in \Omega = [0,1] \\ T(0) = 1 \\ T(1) = 2 \end{cases} \tag{9}$$

在定义域 $x = [0,1]$ 上可将其写成残差形式，并用于定义模型

1 class Poisson (Model) : 2 def residual (self) : 3 R =矩阵 ([0for i in range (self . n _fields)]) 4 T =self . variables [0]5 ddTdxx =self . auxvariables . ddTdxx_6 param =self . parameters 7 8 R[0] = −ddTdxx + 29 return R 以及狄利克雷边界条件 1 bcs = BC. BoundaryConditions ([2 BC. Lambda (physical _tag ='left' , 3 prescribe _fields ={ 0 : lambda t , x , dx , q , qaux , p , n : 1 . } 4) , 5 BC. Lambda (physical _tag ='right' , 6 prescribe _fields ={ 0 : lambda t , x , dx , q , qaux , p , n : 2 . } 7) ,]) 导数 $\partial_{xx} T$ 在辅助变量更新时定义： 1 class PoissonSolver (SteadyResidual) : 2 def update_qaux (self , Q, Qaux , Qold , Qauxold , 3 mesh , model , parameters , time , d t 4) : 5 T =Q[0]6 ddTdxx =计算_导数 (T , mesh , 7 derivatives_multi_index =([[2]]) 8) [: , 0]9 # jax syntax for Qaux$[0]$ = ddTdxx 10 Qaux =Qaux . at [0]. set (ddTdxx) 11 return Qaux

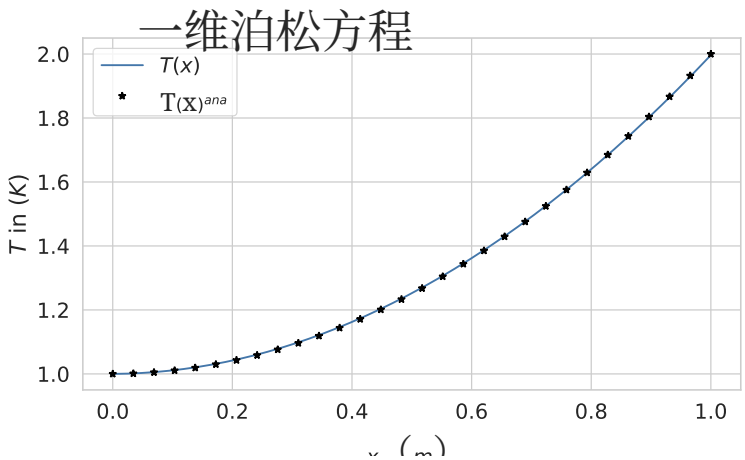

图 7：泊松问题数值结果与解析结果的比较。

```
def 默认_残差 (Q):
    Qaux = self.update_qaux(Q, Qaux, Qold, Qauxold, 网格, Model, 参数, time, dt)
    Q = 边界_算子(time, Q, Qaux, 参数)
    res = Model.残差(Q, Qaux, 参数)
    # JAX 中 res 的语法 [:, mesh.n_内部_单元:] = 0.
    res = res.at[:, mesh.n_内部_单元:].set(0.)
    return res
```

除了编写运行脚本之外，需要做的工作就这些。尤其是，由于内部使用无矩阵 GMRES 线性求解器，因此无需提供有关线性系统组装的额外信息。残差雅可比矩阵的构造无需用户指定，因为雅可比矩阵是通过自动微分构建的。

如果用户未指定自定义的_残差函数，GMRES 求解器将采用默认的_残差实现。

为避免在使用自动微分构造雅可比矩阵时考虑幽灵单元，我们将幽灵单元处的残差设为零（第 8 行）。图 7展示了仿真结果与解析解的对比 $T(x) = x^2+1$（9）

3.3 一维非定常垂向平均动量方程

在这个测试案例中，我们展示

- 如何（三维）-（三维）通过多模型多求解器方法来实现预测-校正法，

- 数值求解器相较于解析解的正确性

此测试案例可在文件夹 tutorials/vam/simple_1d.ipynb 中找到。

VAM 模型推导背后的核心思想是对 RANS 方程进行深度平均。VAM 模型存在不同的推导方法 [12, 16]。本文关注最近在 [16],中推导的版本，因为它采用的方法与 [6] 中 SME 的推导方法几乎相同，下一示例将对此进行演示。两种模型的主要区别在于，VAM 模型纳入了非静水压贡献，因此数值上更为复杂。SME 已被用于探索模型层次结构本身，而这给其实现和分析带来了挑战。

一维 VAM 模型 [16] 假设

- u 为线性,
- w 为二次,
- p为二次

这些函数以勒让德多项式为基底构造。

不考虑摩擦时，VAM 方程可写成以下形式（3a）-（3d），方法是确定

$$
\begin{aligned}
\mathbf{Q} &= \begin{pmatrix} h \\ hu_0 \\ hu_1 \\ hw_0 \\ hw_1 \end{pmatrix} & \mathbf{F} &= \begin{pmatrix} hu_0 \\ hu_0^2 + \frac{1}{3}hu_1^2 \\ 2hu_0u_1 \\ hu_0w_0 + \frac{1}{3}hu_1w_1 \\ hu_0w_1 + u_1(hw_0 + \frac{2}{5}hw_2) \end{pmatrix} \\
\underline{\underline{NC}} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ g\,h & 0 & 0 & 0 & 0 \\ 0 & 0 & -u_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5}w_2 - w_0 & 0 & 0 \end{pmatrix} & \mathbf{S}^P &= \begin{pmatrix} 0 \\ \partial_x(hp_0) + 2p_1\partial_x b \\ -(3p_0 - p_1)\partial_x h - 6(p_0 - p_1)\partial_x b \\ 2p_1 \\ 6(p_0 - p_1) \end{pmatrix}
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
R_0 &= h\partial_x u_0 + \frac{1}{3}\partial_x(hu_1) + \frac{1}{3}u_1\partial_x h + 2(w_0 - u_0\partial_x b) \\
R_1 &= h\partial_x u_0 + u_1\partial_x h + 2(u_1\partial_x b - w_1)
\end{aligned}
\tag{11}
$$

$$
w_2 = -(w_0 + w_1) + (u_0 + u_1)\partial_x b \quad . \tag{12}
$$

确定 $\mathbf{Q}_{aux} = (w_2,\, p_0,\, p_1, \partial_x(h\,p_0)\;\; \partial_x h,\, \partial_x b)^T$ 后，我们可以仿照上述示例定义第一个模型 VAMPredictor。

定义第二个模型 VAMCorrector需要我们将（3c）代入（3b），并进行解析替换，以生成一个泊松型方程。

这可以使用类似以下的代码以符号方式完成

```
1 S =Matrix([[0,2(h *p0).diff(x) + 2*p1*b.diff(x),3(h *p1).diff(x) -(3*p0 -p1)*h.diff(x) - 6*(p0-p1) *b.diff(x),4 - 2*p1,5 6 *(p0-p1)
6 ]]).T 7 Uk =Matrix([[h,h *oldu0,h*oldu1,h*oldw0,h*oldw1]]).T 8 U = Uk - dt *S 9 h =U[0]10 u0 =U[1]/h 11 u1 =U[2]/h 12 w0 =U[3]/h 13 w1 =U[4]/h 14 R1 =h*u0.diff(x) + 1/3 *(h*u1).diff(x) + 1/3 *u1 *h.diff(x) + 2*(w0 -u0 *b.diff(x
)
15 R1 =替换_变量(R1,U)16 R2 =h *u0.diff(x) +u1*h.diff(x) + 2*(u1*b.diff(x) -w1)17 R2 =替换_变量(R2,U)
```

其中，替换_变量会在表达式 R1 和 R2 中代入 U，从而得到一个 Poission 类型方程，该方程取决于未知量$(\partial_{xx}p_0, \partial_{xx}p_1, \partial_x p_0, \partial_x p_1, p0, p1)$ 以及已知变量的系数。

为了对模型进行数值求解，我们现在需要构建两个求解器：基于 HyperbolicSolver 求解器的 VAMPredictorSolver，以及基于 SteadyResidualSolver 求解器的 VamCorrectorSolver。在这两种情况下，只需实现相应的 update_qaux 函数。

最终的求解器现在由这两个求解器组合而成。由于我们需要实现自己的时间循环，因此可以从基类 Solver 继承此求解器。在这个时间循环中，我们执行以下操作：

```
1 ...2 Q =边界_算子_Q（时间，Q，Qaux，参数）3 Qaux =预测器.update_qaux（4 Q，Qaux，Qold，Qauxold，网格，Model，参数，时间，dt5 ）6 双曲型步进_（时间，Q，Qaux，参数，dQ）7 # 复制 Q^*到 Paux 中，使其可供校正器求解器使用8 Paux = Paux.at[0: 6]。设置（Q[0: 6]）9 P =校正器.solve（P）10 # 将 P 复制到 Qaux11 Qaux =Qaux.at[0: 2]。设置（P）12 Q =边界_算子_Q（时间，Q，Qaux，参数）13 Qaux =预测器.update_qaux（14 Q，Qaux，Qold，Qauxold，网格，Model，参数，时间，dt
```

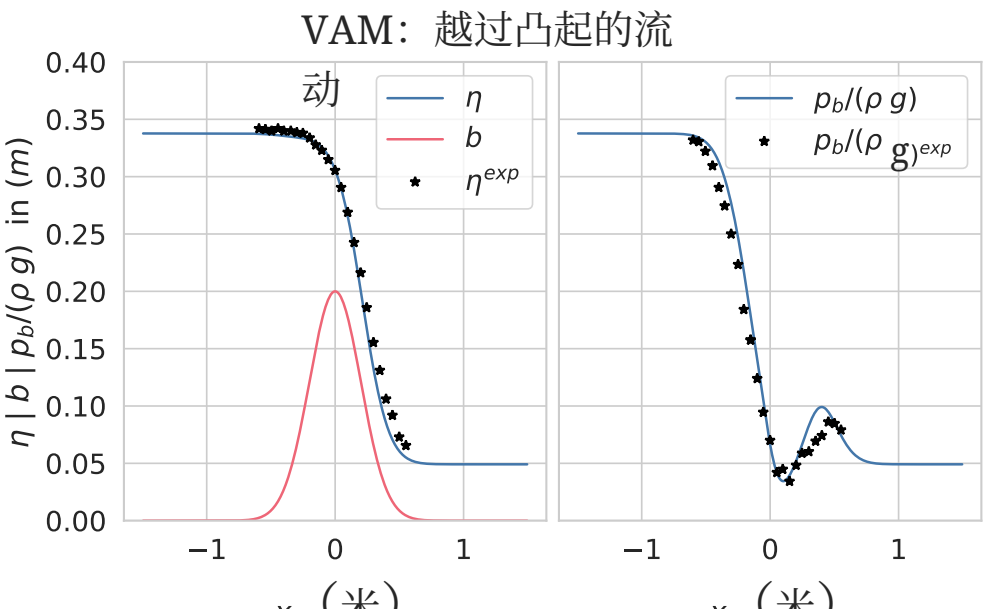

图8：对比 VAM 模型与以下测试案例所述的解析解 [38]。

```
    )
Q = step_source(mesh, model)
```

主要新增内容是从 $\mathbf{Q} \to \mathbf{P}_{aux}$ 传输数据，并从 $\mathbf{P} \to \mathbf{Q}_{aux}$返回。图 8 展示了 [38] 中指定的一个测试案例的数值求解和实验解。

3.4 含摩擦的二维浅水矩方程

在此测试案例中，我们演示

- 模型层次结构的生成与数值求解
- 如何插入材料闭合
- 维度重构的方法。

此测试用例可在文件夹`tutorials/sme/simple_2d.ipynb`中找到。与第3.3节中的 VAM 模型相比，二维 SME 模型 [7] 假设

- $u(t、\ x、\ y、\ \zeta) = \sum i = 0^N \alpha_i(t、\ x、\ y)\phi_i(\zeta)$ 其中 ϕ_i 为最高次数为 N的勒让德多项式，
- w 由质量守恒隐式定义，且
- p为静水压。

偏微分方程组如下

$$\mathbf{Q} = \begin{pmatrix} h \\ h\alpha_k \\ h\beta_k \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix}$$

$$\underline{\underline{F}}(\mathbf{Q}) = \left(\begin{pmatrix} h\alpha_0 \\ \sum_{i,j=0}^{N} h\alpha_i\alpha_j A_{ijk}/M_{kk} + \delta_{k0}\frac{g}{2}h^2 \\ \sum_{i,j=0}^{N} h\alpha_i\beta_j A_{ijk}/M_{kk} \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix}, \begin{pmatrix} h\beta_0 \\ \sum_{i,j=0}^{N} h\beta_i\alpha_j A_{ijk}/M_{kk} \\ \sum_{i,j=0}^{N} h\beta_i\beta_j A_{ijk}/M_{kk} + \delta_{k0}\frac{g}{2}h^2 \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix} \right)$$

$$\underline{\underline{B}}(\mathbf{Q}) = \left(\begin{matrix} \rightarrow 1 & \rightarrow N & \\ \end{matrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & (u_m\delta_{kl} - \alpha_j B_{jlk}/M_{lk}) & 0 \\ 0 & (v_m\delta_{kl} - \beta_j B_{jlk}/M_{lk}) & 0 \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix}, \begin{matrix} \rightarrow N & \rightarrow 1 & \rightarrow N & \rightarrow N \\ \end{matrix} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & (u_m\delta_{kl} - \alpha_j B_{jlk}/M_{lk}) \\ 0 & 0 & (v_m\delta_{kl} - \beta_j B_{jlk}/M_{lk}) \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix} \right)$$

$$\mathbf{S}(\mathbf{Q}) = \begin{pmatrix} 0 \\ -\left(\rho^{-1}\tilde{\sigma}_{xz}\,\phi_k\right)\big|_{\zeta=0}/M_{kk} + \left\langle \rho^{-1}\tilde{\sigma}_{xz}, \phi_k' \right\rangle / M_{kk} \\ -\left(\rho^{-1}\tilde{\sigma}_{yz}\,\phi_k\right)\big|_{\zeta=0}/M_{kk} + \left\langle \rho^{-1}\tilde{\sigma}_{yz}, \phi_k' \right\rangle / M_{kk} \end{pmatrix} \begin{matrix} \downarrow 1 \\ \downarrow N \\ \downarrow N \end{matrix}$$

使用 $k,\ l = 0, \ldots, N$生成矩系统。蓝色箭头表示每个模块中的方程数量。

为简洁起见，我们仅关注源项$\mathbf{S}(\mathbf{Q})$的实现，因为其他示例尚未涉及该项。该源项由底部摩擦的边界闭合项和体摩擦项组成，目前尚未假定任何材料模型。

为展示一组简单的闭合关系，我们分别假设底部摩擦项和体摩擦项满足滑移边界条件，并采用牛顿流体。于是，闭合关系为

$$\begin{aligned} \tilde{\sigma}_{xz}|_{\zeta=0} &= C\ u|_{\zeta=0} \\ \tilde{\sigma}_{yz}|_{\zeta=0} &= C\ v|_{\zeta=0} \\ \tilde{\sigma}_{xz} &= \nu\,(\partial_z u) \\ \tilde{\sigma}_{yz} &= \nu\,(\partial_z v) \quad , \end{aligned} \tag{13}$$

其中 C 为缩放后的滑移长度， ν 为动力黏度。请注意，在体闭合关系中，项 $\partial_x w$ 和 $\partial_y w$ 是基于 [6]中的缩放论证得出的。

底部摩擦项很容易实现，因为底部的速度可以直接获取：

```
    def slip_boundary_condition(self):out =Mat
    rix([0 for i in range(self.n_fields)])offset =self.
    levels +1h =self.variables[0]ha =self.varia
    bles[1: 1 +self.levels + 1]
```

```
hb =self.variables[1+offset: 1+offset +self.levels + 1]p =self.
参数ub = 0vb = 0phi_0 = [self.basismatrices.eval(i,0.0)for i in range(
self.levels + 1)]for i in range(1 +self.levels):ub +=ha[i]*phi_0[i]/
h vb +=hb[i]*phi_0[i]/h for k in range(1, 1 +self.levels):out[1 +k] +=
(-1.0 *p.C/p.rho *ub *phi_0[k]/self.basismatrices.M[k,k]) out[1+offset +k] +=(-1.0 *p.C/p.rho *vb *phi_0[k]/self.basismatrices.M[k,k])
return out
```

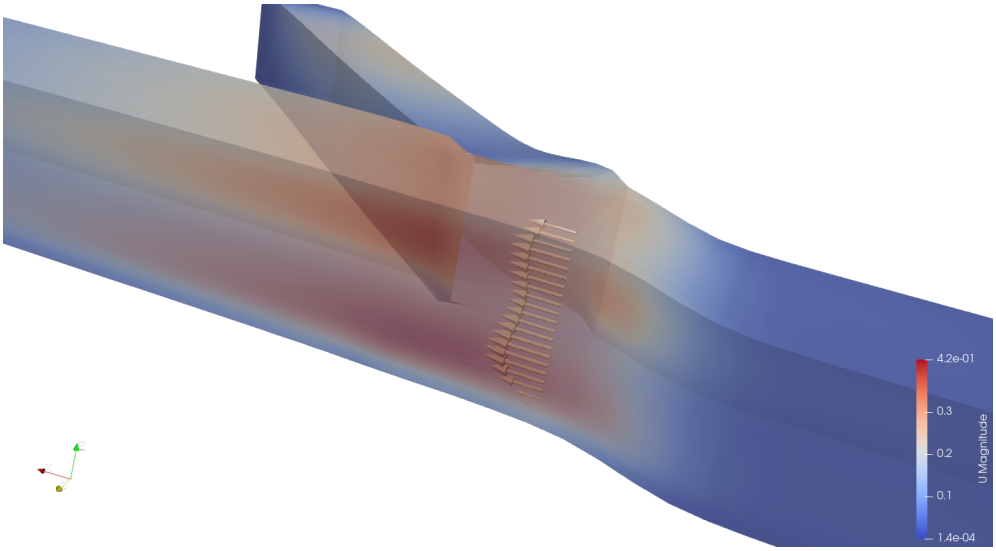

图 9：使用四阶多项式和 SME 对二维仿真进行三维重建。

体摩擦项需要进行符号积分。对于牛顿流体，该项可以简化

$$\begin{aligned}\left\langle \rho^{-1}\tilde{\sigma}_{xz}, \phi_k' \right\rangle / M_{kk} &= \frac{\nu}{\rho} \sum_{i=0}^{N} \alpha_i \left\langle \phi_i, \phi_k' \right\rangle / M_{kk} \\ \left\langle \rho^{-1}\tilde{\sigma}_{yz}, \phi_k' \right\rangle / M_{kk} &= \frac{\nu}{\rho} \sum_{i=0}^{N} \beta_i \left\langle \phi_i, \phi_k' \right\rangle / M_{kk}\end{aligned} \tag{14}$$

而项 $\langle \phi_i, \phi_k' \rangle =: D_{ik}$ 确实可以进行符号积分。

源项中的体摩擦部分为

```
def newtonian(self):
    out = Matrix([0 for i in range(self.n_fields)])
    offset = self.levels + 1
    h = self.variables[0]
    ha = self.variables[1: 1 + self.levels + 1]
    hb = self.variables[1 + offset: 1 + self.levels + 1 + offset]
    p = self.parameters
    for k in range(1 + self.levels):
        for i in range(1 + self.levels):
            out[1 + k] += (
                -p.nu
                / h
                * ha[i]
                / h
                * self.basismatrices.D[i, k]
                / self.basismatrices.M[k, k]
            )
            out[1 + k + offset] += (
                -p.nu
                / h
                * hb[i]
                / h
                * self.basismatrices.D[i, k]
                / self.basismatrices.M[k, k]
            )
    return out
```

合并后，得到如下源项：

```
def source(self):
    out = Matrix([0 for i in range(self.n_fields)])
    out += 牛顿型()
    out += slip_boundary_condition()
    return out
```

以上代码片段表明，分层SME可以实现，材料闭合也可以轻松替换为其他材料

模型。特别要注意的是，与浅水方程不同，SME 和其他层次化方法可以获取垂向速度剖面。这使得我们能够根据速度梯度构建闭合关系，例如牛顿型流体中的闭合关系。

图9展示了通过实现插值_3d得到的三维重建。该测试案例使用 SME 和 4 阶勒让德多项式构建。图中的箭头表示几何体交汇处某个特定位置的垂向速度剖面。表面的高程通过后处理步骤获得。

4 结论

本文介绍了*Zoomy*，这是一款用于分析和模拟深度平均自由表面流模型的软件框架。

速度和压力$(u,\ v,\ w)$ 以及 p 沿深度方向以多项式展开形式描述，且多项式次数可变，由此形成分层模型。由此产生的深度平均和矩投影构成了模型层次结构。

Zoomy 致力于弥补研究空白，系统地描述、分析并求解通过深度平均生成的分层自由表面流模型。具体而言，用户可以

- 高效地表示模型的分层特性，
- 更换底层基函数，
- 以符号方式执行深度积分，
- 灵活定义材料闭合，并
- 在一维和二维非结构网格上数值求解非守恒偏微分方程组。

Zoomy 由一个符号层组成，用于表示和分析模型。代码转换会创建模型的新表示，供数值层用于数值求解偏微分方程组。这些代码转换可生成不同的输出格式，例如 JAX、NumPy 或 C，从而实现符号层与数值层的分离。

基于四个示例，我们介绍了在*Zoomy*中构建、分析和求解模型所需的主要构成模块。这些示例重点展示了如何

- 构建多维模型，
- 表示模型层级，
- 使用符号层 进行分析，并
- 构建基于分裂格式的复杂数值求解器。

目前和未来的工作包括将基于有限体积法、以 JAX 编写的数值后端扩展到大规模并行的块结构自适应网格细化框架*AMReX*。此外，我们旨在创建更易于用户使用的界面，以便通过耦合库*preCICE* [39]将该求解器与*OpenFOAM* 的两相流求解器耦合。

声明

资金支持

Ingo Steldermann 获得了生命、地球与能源数据科学学院（HDS-LEE）的资助。

代码可用性

Zoomy 是根据 GNU 通用公共许可证 v3.0 或更高版本发布的开源软件。有效的源代码仓库位于 https://github.com/mbd-rwth/Zoomy。

本文所述的是 1.0.0 版本，该版本已在源代码仓库中标记为相应的发布版本。

我们通过支持使用 *Conda* 包管理器安装 [30] 或从源代码手动安装，使我们的项目易于获取。此外，我们还支持 Apptainer [31] 和 Docker [32]的预构建容器环境。

致谢

我们感谢 Benjamin Terschanski 提供出色的 GitHub 仓库模板及配套文档，并协助配置持续集成流水线，还为论文初稿提出了宝贵意见。

作者贡献

Ingo Steldermann：写作——初稿、可视化、验证、软件、方法学、调查、正式分析、数据整理、概念化。Julia Kowalski：写作——审阅与编辑、监督、资源、经费获取、概念化。

编辑工具与大型语言模型的使用

作者感谢使用 Grammarly https://grammarly.com 和 LanguageTool https://languagetool.org/编辑个别文本段落。

利益冲突

作者声明不存在利益冲突。

参考文献

[1] Chow, V.T.：《明渠水力学》。Blackburn Press，新泽西州考德威尔（2009年）[2] Pope, S.B.：《湍流》。剑桥大学出版社，剑桥；纽约（2000年）[3] Wilcox, D.C.：《CFD湍流建模》，第3版。DCW Industries，加利福尼亚州拉卡尼亚达（2006年）

[4]Castro-Orgaz, O., Hager, W.H.：《非静水自由表面流》。地球物理与环境力学和数学进展。施普林格，查姆（2017年）。
https://doi.org/10.1007/978-3-319-47971-2 .
http://link.springer.com/10.1007/978-3-319-47971-2[5] Blumberg, A.F., Mellor, G.L.：《海岸海洋数值模型》。载于：Sünder- mann, J., Holz, K.-P.（编）《河口物理数学建模》，第203–219页。施普林格，柏林，海德堡（1980年）。
https://doi.org/10.1007/ 978-3-642-46416-4 16_[6] Kowalski, J., Torrilhon, M.：浅层流的矩近似与模型级联。《计算物理通讯》25(3)（2019年）https://doi.org/10.4208/cicp.OA-2017-0263[7] Steldermann, I.,Torrilhon,M., Kowalski,J.：利用浅层矩捕捉开放弯曲浅水流中的垂向结构。《计算与理论输运杂志》52(7)，475–505（2023年）https://doi.org/10.1080/23324309.2023.2284202[8] Koellermeier, J., Rominger, M.：双曲型浅水矩方程的分析与数值仿真。《计算物理通讯》28，1038–1084（2020年）
https://doi.org/10.4208/cicp.OA-2019-0065[9] Scholz, U., Kowalski, J., Torrilhon, M.：浅层矩方程中的色散。《应用数学与计算通讯》6(4)，2155–2195（2024年）https://doi.org/10.1007/s42967-023-00325-2[10] Serre, F.：对渠道中恒定流与非恒定流研究的贡献。《白煤》39(6)，830–872（1953年）
https://doi.org/10.1051/ lhb/1953058[11] Green, A.E., Naghdi, P.M.：变水深条件下水波传播方程的推导。《Journal of Fluid Mechanics》78(2)，237–246（1976年）https: //doi.org/10.1017/S0022112076002425[12] Khan, A.A., Steffler, P.M.：弯曲床面上流动的垂向平均方程与矩方程模型。《水力工程学报》122(1)，3–9（1996年）https://doi.org/10.1061/(ASCE)0733-9429(1996)122:1(3)[13]Khan, A.A., Steffler, P.M.：利用垂向平均方程与矩方程对跌水进行建模。《水力工程学报》122(7)，397–402（1996年）
https://doi.org/10.1061/(ASCE)0733-9429(1996)122:7(397)[14] Cantero-Chinchilla, F.N., Castro-Orgaz, O., Khan, A.A.：采用加权平均方程对深度积分非静水自由表面流进行建模。《国际流体数值方法杂志》87(1)，27–50（2018年）https: //doi.org/10.1002/fld.4481

[15] Ghamry, H.K.：浅水自由表面流的二维垂向平均方程与矩方程。加拿大国家图书馆 = 加拿大国家图书馆，渥太华（2000年）[16] Escalante, C., Morales De Luna, T., Cantero-Chinchilla, F., Castro-Orgaz, O.：垂向平均方程与矩方程：全新推导、高效数值求解，以及与其他物理近似方法在非静水压自由表面流建模中的比较。计算物理学杂志 504，112882（2024年）
https://doi.org/10.1016/j.jcp.2024.112882[17] Chesnokov, A., Liapidevskii：剪切流的浅水方程，第115卷，第165–179页（2011年）。
https://doi.org/10.1007/978-3-642-17770-5_13。期刊简称：数值流体力学与多学科设计札记[18] Gavrilyuk, S., Ivanova, K., Favrie, N.：多维剪切浅水流：问题与解法。计算物理学杂志 366，252–280（2018年）
https://doi.org/10.1016/j.jcp.2018.04.011[19] Escalante, C., Fernández-Nieto, E.D., Luna, T., Castro, M.J.：一种高效的双层非静水压方法，用于模拟色散水波。科学计算杂志 79(1)，273–320（2019年）
https://doi.org/10.1007/s10915-018-0849-9[20] Fernández-Nieto, E.D., Parisot, M., Penel, Y., Sainte-Marie, J.：用于自由表面流的欧拉方程色散层平均近似层级。数学科学通讯 16(5)，1169–1202（2018年）
https://doi.org/10.4310/CMS.2018.v16.n5.a1[21]Baratta, I.A., Dean, J.P., Dokken, J.S., Habera, M., Hale, J.S., Richardson, C.N., Rognes, M.E., Scroggs, M.W., Sime, N., Wells, G.N.：DOLFINx：新一代 FEniCS 问题求解环境[22] Rathgeber, F., Ham, D.A., Mitchell, L., Lange, M., Luporini, F., Mcrae, A.T.T., Bercea, G.-T., Markall, G.R., Kelly, P.H.J.：Firedrake：通过组合抽象实现有限元方法自动化。ACM数学软件汇刊 43(3)，1–27（2017年）
https://doi.org/10.1145/2998441[23]Dedner, A., Kloefkorn, R., Nolte, M.：DUNE-FEM模块的 Python绑定。Zenodo（2020年）。
https://doi.org/10.5281/ZENODO.3706994[24] Geuzaine, C., Remacle, J.-F.：Gmsh：一种内置前处理和后处理功能的三维有限元网格生成器。国际工程数值方法杂志 79(11)，1309–1331（2009年）https://doi.org/10.1002/nme. 2579
https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2579[25] Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger,

B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cim-rman, R., Scopatz, A.：SymPy：Python中的符号计算。PeerJ 计算机科学 3，103（2017年）https://doi.org/10.7717/peerj-cs.103[26] Dumbser, M., Enaux, C., Toro, E.：用于刚性双曲平衡律的超高精度有限体积格式。计算物理学杂志 227，3971–4001（2008年）https://doi.org/10.1016/j.jcp.2007.12.005[27] Castro, M., Fernández-Nieto, E., Ferreiro, A., García-Rodríguez, J., Madroñal, C.：二维非守恒双曲系统 Roe 格式的高阶扩展。科学计算杂志 39，67–114（2009年）https://doi.org/10.1007/s10915-008-9250-4[28] Castro, M.J., Pardo, A., Parés, C., Toro, E.F.：关于一些用于非守恒系统的快速、平衡良好的一阶求解器。计算数学 79(271)，1427–1472（2009年）https://doi.org/10.1090/S0025-5718-09-02317-5[29] Ayachit, U.：《ParaView指南：并行可视化应用程序》。Kitware, Inc.，美国纽约州克利夫顿帕克（2015年）[30] Anaconda：Anaconda 软件发行版。https://anaconda.com。计算机软件（2016年）[31] Kurtzer, G.M., cclerget, Bauer, M., Kaneshiro, I., Trudgian, D., Godlove, D.：Hpcng/singularity：Singularity 3.7.3。https://doi.org/10.5281/zenodo.4667718。https://doi.org/10.5281/zenodo.4667718[32] Merkel, D.：Docker：用于一致开发与部署的轻量级 Linux 容器。Linux journal 2014(239)，2（2014年）[33] Jupyter, P., Bussonnier, M., Forde, J., Freeman, J., Granger, B., Head, T., Hold-graf, C., Kelley, K., Nalvarte, G., Osheroff, A., Pacer, M., Panda, Y., Perez, F., Ragan-Kelley, B., Willing, C.：Binder 2.0——面向大规模科学研究的可复现、交互式、可共享环境，第113–120页（2018年）。https://doi.org/10.25080/Majora-4af1f417-011[34] Bisong, E.：Google Colaboratory，第59–64页。Apress，伯克利（2019年）。https://doi.org/10.1007/978-1-4842-4470-8_7[35] Sullivan, B., Kaszynski, A.：PyVista：通过简洁的界面使用可视化工具包（VTK）进行三维绘图和网格分析。Journal of Open Source Software 4(37)，1450（2019年）https://doi.org/10.21105/joss.01450[36] Hunter, J.D.：Matplotlib：二维图形环境。《Computing in Science & Engineering》9(3)，90–95（2007年）https://doi.org/10.1109/MCSE.2007.55

[37] Delestre, O., Lucas, C., Ksinant, P.-A., Darboux, F., Laguerre, C., Vo, T.N.T., James, F., Cordier, S.: Swashes：用于水力与环境研究的浅水解析解汇编。International Journal for Numerical Methods in Fluids **72**(3)，269–300(2013) https://doi.org/10.1002/fld.3741 。arXiv:1110.0288 [physics][38] Sivakumaran, N.S., Tingsanchali, T., Hosking, R.J.: 弯曲床面上的稳定浅流。Journal of Fluid Mechanics **128**，469–487(1983) https://doi.org/10.1017/s0022112083000567[39] Chourdakis, G., Davis, K., Rodenberg, B., Schulte, M., Simonis, F., Uekermann, B., Abrams, G., Bungartz, H., Cheung Yau, L., Desai, I., Eder, K., Hertrich, R., Lindner,F., Rusch,A., Sashko,D.,Schneider,D.,Totounferoush,A.,Volland,D., Vollmer, P., Koseomur, O.: preCICE v2：可持续且易于使用的耦合库 [版本 2；同行评审：2 人认可]。Open Research Europe **2**(51) (2022)https://doi.org/10.12688/openreseurope.14445.2