基于巨大肾积水治疗疗效评估的输尿管流量预测模型设计

摘要

巨大肾积水是一种较为罕见的泌尿系统疾病，患者由于尿路梗阻尿液滞留于肾盂内无法排出，其治疗方案的疗效评估往往通过监测输尿管尿流量来进行。本文基于输尿管物理结构分析，运用公式解析和神经网络分别构筑数学模型，为深入研究如何通过输尿管参数预测尿流量提供了合理的参考方案。

针对问题一，首先构建输尿管尿液输运模型，对输尿管流动结构做出简化描述。在模型框架内对输尿管蠕动及管两端压降分别进行模拟，推导两因素对输尿管流量影响的关系式。通过单一因素产生流量与实际值比较结果以及相关性分析，确定两因素的基本耦合形式为线性耦合。将蠕动波视为方波进行分析，引入收缩管径比、收缩长度比两无量纲数作为系数对耦合式进行改造，得到基本的解析式，并确定了两无量纲数对于预测流量的影响模式。随后以流量预测结果与实际结果差值作为目标函数，采取梯度下降法对两无量纲数进行求解。得到最拟合的收缩管径比$\kappa = 0.486485$，收缩长度比$\phi = 0.588144$，从而确立了输尿管流量预测的解析式模型。模型预测值与实际值的相关性系数达到 0.9779，同时对梯度下降法的合理性进行论证，进一步说明了本模型的有效性。

针对问题二，引入物理信息神经网络建立模型。首先根据问题一建立的输尿管流动结构，确立物理约束条件。考虑输入数据间的关联性，采用多层感知机制为基础架构，在已知物理约束下，设计出由物理信息引导的神经网络模型 Kidney Model。模型的损失函数分为数据误差与物理信息误差两项分别进行构建，采用 K-fold 交叉检验方法对模型进行训练，同时使用 Adam 优化器动态调整学习率，优化输入数据间差异。使用建立模型对已知数据进行计算，残差项的统计分布符合标准正态分布特征，验证了模型的精确性与置信度，RMSE、MAE、R^2 指标结果显示本模型的拟合效果较好。

针对问题三，按照题目要求，使用前文建立的解析式模型和 AI 模型对病例进行手术前后输尿管流量的数据进行预测，并与给出的实际数据进行比较。误差分析结果显示，解析式模型具有较高的精确度，AI 模型预测值与实际值相对误差较大，该结果可能源自于其训练集的局限。随后对模型局限性进行分析，解析式模型的前置假设较为简单和理想化，AI 模型的局限性体现在其训练结构设置单薄，对训练集质量也有较高要求，最后提出对应的优化改进方案。

关键词：输尿管流量预测；蠕动泵送；压降驱动；物理信息神经网络

一、 问题重述

1.1 问题背景

巨大肾积水是一种较为罕见的泌尿系统疾病，其在临床上一般表现为肾盂肾盏系统病理性扩张，积水量显著超过常规肾积水阈值（成人单侧积水>1000 mL，或儿童>24 小时尿量），导致肾脏功能严重损害及腹部占位性病变。该疾病的核心病理机制为持续性尿路梗阻，伴随进行性肾实质萎缩与集合系统扩张，最终形成“囊袋样”无功能肾。在尿流动力学中，手术前后对输尿管流量数据进行监测是一种常见的诊疗手段，可为患者的诊断、治疗方法及疗效评定提供客观依据。

1.2 问题要求

附件 1 提供了正常肾脏与病例肾脏两种情况下输尿管的蠕动频率、蠕动幅度、蠕动频率、管径、管长等相关流量数据。附件 2 给出了病例手术前后肾脏造影结果。为评估优化治疗方案，需结合实际数据分析，建立合理有效的数学模型。

针对问题一，使用附件 1 中的正常肾脏输尿管流量数据，结合输尿管蠕动、输尿管几何参数、尿液物性等参数，建立输尿管内流量与压降的解析式方程，描述蠕动对尿流的主动泵送效应，以及压降对流动的阻力效应。

针对问题二，使用附件 1 中的正常肾脏输尿管流量数据，通过 AI 算法建立输尿管流量—压降的数学模型。

针对问题三，基于附件 1 的手术前后数据，使用前两问建立的解析式模型和 AI 模型分别预测输尿管泌尿量。将预测结果与附件 2 提供的实际数据进行比较，分析模型优劣并提出优化改进的建议。

二、 问题分析

2.1 问题一的分析

问题一要求根据已知的正常肾脏输尿管流量数据，探寻输尿管参数与管道流量间的解析式关系。首先简化模型，假设输尿管为等截面长直圆管，尿液为理想流体。根据资料查找结果，将流量影响因素分为压降和蠕动泵送两个方向。针对蠕动泵送，对单个蠕动腔体建立局部坐标系分析流量；针对压降，从连续性方程与纳维-斯托克斯方程对流量情况进行推导。通过单一因素产生流量与实际值比较结果以及相关性分析，确定两因素的耦合形式。将蠕动波视为方波，引入收缩管径比、收缩长度比两无量纲数对解析式进行构建，以流量预测结果与实际结果差值作为目标函数，采取梯度下降法对两无量纲数进行求解。最后对建立的解析式模型与采取的求解方法进行检验分析。

2.2 问题二的分析

问题二要求使用 AI 算法对输尿管参数与管道流量间的关系进行构建。考虑引入物理信息神经网络建立模型。首先根据问题一建立的输尿管流动结构，确立物理约束条件。由于输入数据各部分无时间、空间关系，训练数据量较小，且输出数据与输入数据之间存在解析计算关系，考虑采用多层感知机制为基础的模型

架构，在已知物理约束下，设计出由物理信息引导的神经网络模型 Kidney Model。模型的损失函数分为数据误差与物理信息误差两项分别进行构建，由于训练数据有限，且项与项之间存在单位与数量级差异，采用 K-fold 交叉检验方法对模型进行训练，同时使用 Adam 优化器，动态调整学习率，达到较好的训练效果。

2.3 问题三的分析

问题三要求在实例中对模型进行评估，对于给出病例的输尿管参数，使用前两问建立的解析式模型和 AI 模型对进行手术前后输尿管流量的数据进行预测，并与给出的实际数据进行比较，对误差进行分析。结合误差分析结果，从模型假设的简单和理想化出发对解析式模型的局限性进行阐述，从训练数据集的规模与分布小出发对 AI 模型的局限性进行阐述，最后提出对应的优化改进方案。

三、 模型假设

1. 假设输尿管在无蠕动时为等截面圆直管；
2. 假设蠕动波在整段输尿管上频率和幅度恒定，忽略蠕动波的传播衰减和局部变异；
3. 假设尿液粘度和密度在流动过程中不变，是不可压缩粘性流体，忽略非牛顿特性、温度影响或杂质导致的粘度变化；
4. 假设蠕动波波形可近似简化为方波；
5. 假设每个蠕动腔体内的流动相互独立，忽略腔体间耦合效应；
6. 假设刚性管壁的无滑移边界条件，忽略管壁弹性及生物膜界面效应。

四、 符号说明

表 1 符号说明

符号	说明	单位
L	输尿管长度	m
D	输尿管直径	m
f	蠕动频率	Hz
A	蠕动幅度	m
μ	流体粘度	$\mathrm{Pa\cdot s}$
Q	流量	$\mathrm{m^3/s}$
ΔP	压降	Pa
r	径向坐标	m
z	轴向坐标	m
V_{single}	单个蠕动腔体体积	$\mathrm{m^3}$
Q_{pump}	蠕动泵送流量	$\mathrm{m^3/s}$
K	泵送效率	/
u,v,w	速度场	$\mathrm{m^3/s}$
p	压力场	Pa
α	压力梯度	Pa/m

Q_{press}	压降驱动流量	m^3/s
L_1	收缩段长度	m
d	收缩段管径	m
L_2	舒张段长度	m
κ	收缩半径比	/
ϕ	收缩长度比	/
Q_{total}	总流量	m^3/s
$F(\boldsymbol{\theta};\boldsymbol{x})$	目标函数	m^3/s
γ	压缩系数	/
$\mathscr{L}_{\text{MSE}}$	数据残差项损失函数	/
$\mathscr{L}_{\text{physics}}$	物理信息残差项损失函数	/
Loss	总损失函数	/

五、 模型建立与求解

5.1 问题一的模型建立与求解

5.1.1 肾脏输尿管结构与流动

对于输尿管所处的上尿路系统，其结构如下所示。

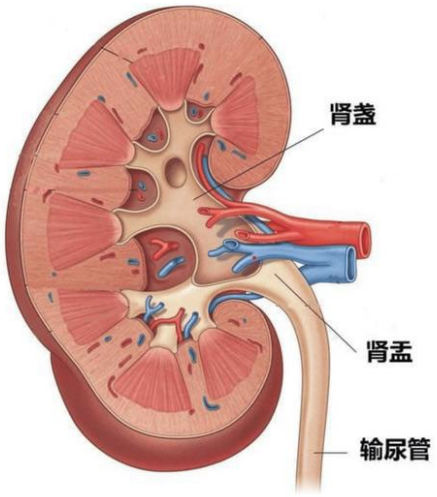

图 1 上尿路结构示意图

可以看到，尿液在肾盏内生成、于肾盂汇聚，经输尿管流往膀胱。这一过程中输尿管对尿液输运起主要作用。经资料查找与文献检索，输运主要动力来源主要有两种，一是输尿管壁平滑肌的节律性蠕动收缩，二是输尿管连接两端的压力差，两者均对输尿管内尿流量大小产生影响，模型的构建基础就建立在这两个因子上。

为方便计算与数值模拟，将输尿管结构近似为等截面的长直圆管，假设尿液为不可压缩、粘度恒定、均质单相的理想流体。综合上述假设约束，构建输尿管尿液输运模型如下：

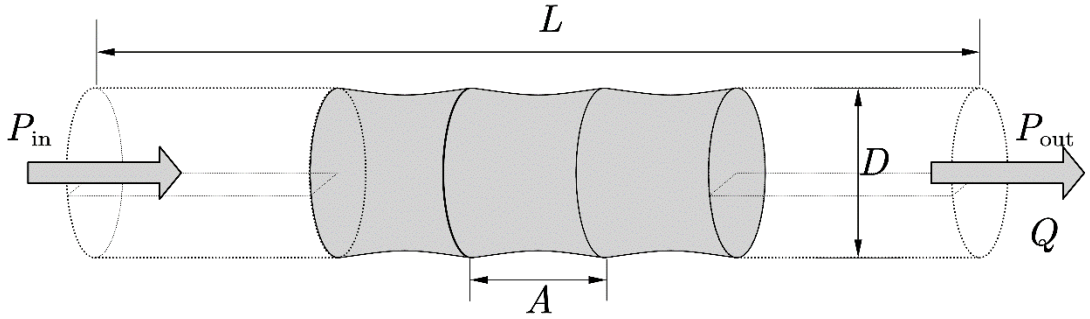

图 2 输尿管流动结构示意图

如图所示，对于一条长度为L，管径为D的输尿管，其时刻以蠕动频率f、蠕动幅度A保持规律性蠕动，流体粘度为μ的尿液在其中流动，流量大小为Q（即单位时间内通过输尿管的尿液体积）。关于输尿管连接两端，P_{in}表示输运初始部位肾盂的腔内压力，P_{out}表示输运末端输尿管出口压力，则压降ΔP可表示为

$$\Delta P = P_{in} - P_{out} \tag{1}$$

5.1.2 蠕动泵送和压降驱动

在建立的模型框架内对**输尿管蠕动**及**管两端压降**分别进行模拟，求取两因素对输尿管流量的影响关系。

针对**输尿管蠕动**行为，对单个蠕动腔体建立局部坐标系。

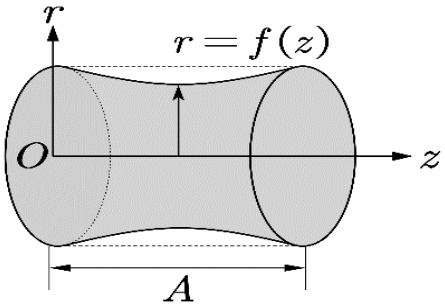

图 3 输尿管蠕动泵送示意图

如图 3 所示，蠕动过程中输尿管的半径随墙体长度z改变，可表示为$r=f(z)$。考虑到管腔始终保持圆截面，截面面积恒为πr^2，得到单个腔体的体积为：

$$V_{\text{single}}=\int_0^A \pi[f(z)]^2\mathrm{d}z \tag{2}$$

在没有压力梯度的情况下，流量完全由蠕动泵送产生，可求得单位时间的流量为：

$$Q_{\text{pump}}=f\cdot V_{\text{single}}=f\cdot\int_0^A \pi[f(z)]^2\mathrm{d}z \tag{3}$$

若蠕动频率f和幅度A不变，泵送流量只与$f(z)$，即腔体的几何形状有关。记泵送效率K为：

$$K=\frac{V_{\text{single}}}{AD^2}=\frac{\int_0^A \pi[f(z)]^2\mathrm{d}z}{AD^2} \tag{4}$$

其物理意义是实际腔体与最大可能腔体（三段边长为A,D,D的长方体）的体积之比，则泵送流量可表示为：

$$Q=K\cdot f\cdot A\cdot D^2 \tag{5}$$

再考虑**压降对流量的影响**：

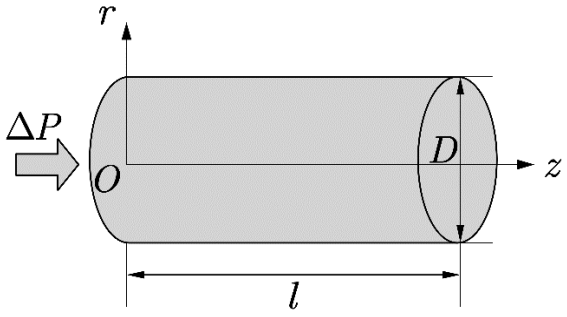

图 4 输尿管压降驱动示意图

由图 4，考虑一个通过直径为D的水平圆直管的流动，圆管长度为l，压降为$\Delta P=P_{\text{in}}-P_{\text{out}}$。

对于不可压缩的均匀流体，其连续性方程为：

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \tag{6}$$

纳维-斯托克斯方程可简化为：

$$\begin{cases}\dfrac{Du}{Dt}=X-\dfrac{1}{\rho}\dfrac{\partial p}{\partial x}+\nu\nabla^2 u\\[2ex]\dfrac{Dv}{Dt}=Y-\dfrac{1}{\rho}\dfrac{\partial p}{\partial y}+\nu\nabla^2 v\\[2ex]\dfrac{Dw}{Dt}=Z-\dfrac{1}{\rho}\dfrac{\partial p}{\partial z}+\nu\nabla^2 w\end{cases} \tag{7}$$

其中$\nu=\mu/\rho$是运动粘性，∇^2是拉普拉斯算子。

对于水平圆直管中的流动，其速度解的形式为：

$$u=u(y,z),v=0,w=0 \tag{8}$$

纳维-斯托克斯方程变为：

$$\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=-\frac{\alpha}{\mu} \tag{9}$$

其中$\alpha=\Delta P/l$为压力梯度。

利用$r^2=x^2+y^2$将上式化为圆柱坐标形式，得到：

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 u}{\partial \theta^2}=-\frac{\alpha}{\mu} \tag{10}$$

根据模型假设，考虑到流动是对称的，得出u仅是r的函数。因此存在关系$\partial^2 u/\partial\theta^2=0$，方程简化为：

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}u}{\mathrm{d}r}\right)=-\frac{\alpha}{\mu} \tag{11}$$

对上式直接积分，可以得到：

$$u=-\frac{\alpha}{\mu}\frac{r^2}{4}+m\log r+n \tag{12}$$

式中m,n均为常数，可以由$r=D/2$处的无滑移条件和对中心线$r=0$的对称性来确定：

$$\begin{aligned}&r=D/2:\ u=0\\&r=0:\frac{\mathrm{d}u}{\mathrm{d}r}=0\end{aligned} \tag{13}$$

最终得到速度解为：

$$u=\frac{\alpha}{4\mu}\left(\frac{D^2}{4}-r^2\right) \tag{14}$$

对上式积分，得到在压力驱动下单位时间通过截面的流量为：

$$Q_{\text{press}}=\int_0^{D/2}2\pi r\cdot u\,\mathrm{d}r=\frac{\pi D^4\Delta P}{128\mu l} \tag{15}$$

为求取二因子在流量模拟中的耦合形式，参照附件 1 内正常条件下输尿管流量数据，分别评估仅存在输尿管蠕动，仅存在管两端压差两种情况下的流量情况，再与表内给出的实际流量情况进行对比，结果如图 5 所示。

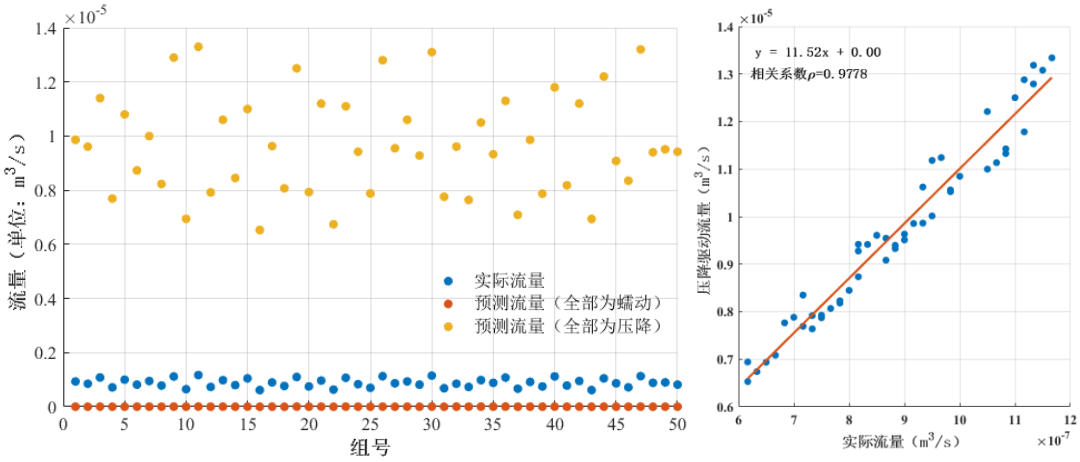

图 5 单一因素与实际流量对比图（左）；压降驱动与实际流量相关性（右）

可以看到，三种情况的流量的离散度存在较大差异，但在分布上存在一致的规律性。如果仅考虑蠕动泵送，则预测流量远低于实际流量；如果仅考虑压降驱动，则预测流量远高于实际流量；同时如图 5 右图所示，两者相关系数$\rho = 0.9778$，压降驱动的流量和实际流量之间存在明显的正相关性。不难得出，蠕动泵送和压力梯度两者之间的耦合形式可能是线性关系，但并非直接叠加。

5.1.3 输尿管流量-压降模型

出于简化模型考虑，结合文献检索结果，本模型将蠕动波简化为方波，即将单次蠕动形成的腔体分为收缩段和舒张段进行分析，定义收缩段长度L_1，管径d，压降Δp_1，舒张段长度L_2，管径D，压降Δp_2。示意图如图 5 所示。

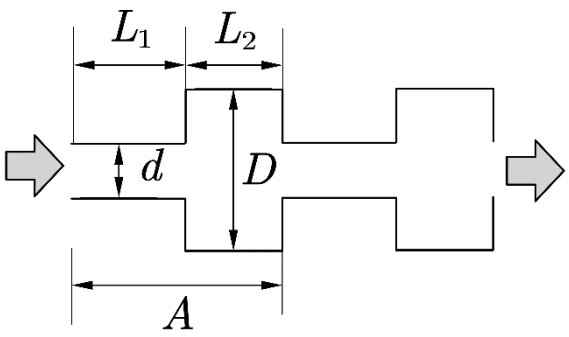

图 6 腔体蠕动收缩示意图

当输尿管以一定频率蠕动时，管径D随之改变。考虑到题目所给数据中的蠕动幅度A与管径D的大小关系，蠕动过程中管截面的变化较大。

对于全输尿管而言，可划分的腔体总数$N = L/A$。假设每段腔体（即每段波长）内压降相同，则每段腔体的出入口压降为：

$$\Delta P_{\text{single}} = \frac{\Delta P}{N} = \frac{A\,\Delta P}{L} \tag{16}$$

由于管道蠕动并不影响截面形状，收缩段和舒张段均满足泊肃叶定律：

$$Q_1 = \frac{\pi d^4 \Delta p_1}{128 \mu L_1},\ Q_2 = \frac{\pi D^4 \Delta p_2}{128 \mu L_2} \tag{17}$$

同时考虑到连续性方程，有：

$$Q_1 = Q_2,\ \Delta p_1 + \Delta p_2 = \Delta P_{\text{single}} \tag{18}$$

得到单个腔体内在压降驱动下的流量为：

$$Q_{\text{press}} = \frac{\pi A \Delta P}{8\mu L} \cdot \frac{1}{\frac{L_1}{d^4} + \frac{L_2}{D^4}} \tag{19}$$

引入两个无量纲数：收缩管径比$\kappa = d/D$，收缩长度比$\phi = L_1/A$，则压降导致的流量公式可记作：

$$Q_{press} = \frac{\pi D^4 \Delta P}{128\mu L\left(\frac{\phi}{\kappa^4} + 1 - \phi\right)} \tag{20}$$

考虑腔体蠕动产生的泵送流量$Q_{\text{pump}} = KfAD^2$，其中泵送效率为

$$K = \frac{\pi d^2 L_1 + \pi D^2 L_2}{AD^2} \tag{21}$$

故蠕动泵送流量为

$$Q_{\text{pump}} = \frac{\pi}{4} fAD^2(\kappa^2\phi + 1 - \phi) \tag{22}$$

总流量可表示为

$$\begin{aligned} Q_{\text{total}} &= Q_{\text{pump}} + Q_{\text{press}} \\ &= \frac{\pi}{4} fAD^2(\kappa^2\phi + 1 - \phi) + \frac{\pi D^4 \Delta P}{128\mu L} \cdot \frac{1}{\frac{\phi}{\kappa^4} + 1 - \phi} \end{aligned} \tag{23}$$

模型建立即是对两无量纲数进行求解。

在极限条件下对公式进行分析。可以看到，当$\kappa \to 1$或$\phi \to 0$时，总流量即是蠕动泵送和压降的直接叠加；当$\kappa \to 0$时，压降效应对流量的影响微乎其微，蠕动效应成为尿液输送的唯一动力来源；当$\phi \to 1$时，考虑到κ的取值范围在$[0,1]$，压降的无量纲系数大于蠕动泵送，即蠕动泵送在尿液输送过程中起主体作用。

为进一步探寻两无量纲系数对模型的影响效应，假设参数如表 2 所示，在该固定工况下，于区间$[0.3, 0.7]$内，以0.1为步长分别对κ和ϕ取值，探寻另一变量与预测流量间的变化关系。

表 2 模拟参数

蠕动频率	蠕动幅度	管径	长度	粘度	压降
0.1Hz	0.001m	0.004m	0.28m	0.001Pa·s	250Pa

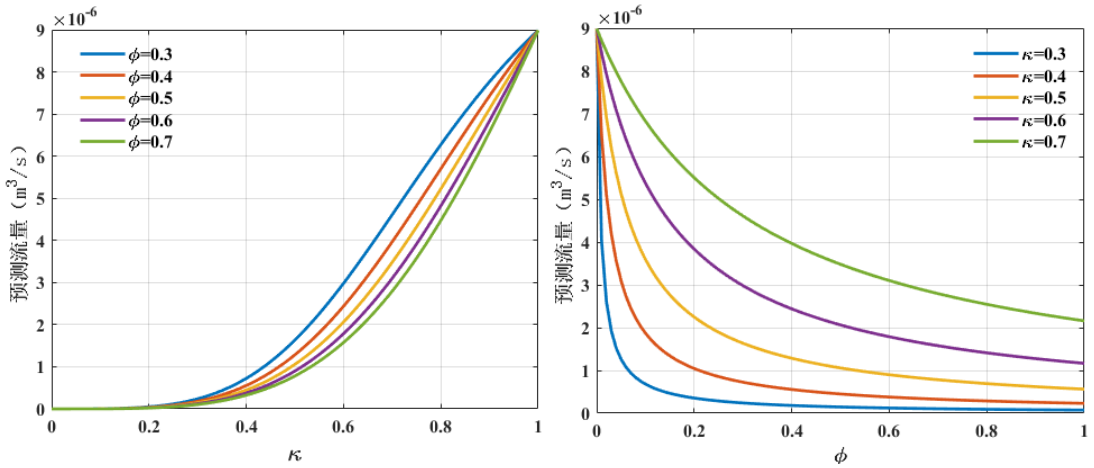

图 7 预测流量随管径收缩比/收缩长度比的变化曲线图

如图 7 所示，预测流量随管径收缩比κ的增大而增大，这一变化在κ值较小时并不明显，随$\kappa \to 1$变化速率逐步增长；预测流量随收缩长度比ϕ的增大而减小，变化速率在$\phi = 0$附近较大，在$\phi = 1$附近较小，κ值较小时这一规律尤为显著。

5.1.4 基于梯度下降的模型求解

出于模型实用性考虑，预测值要求尽可能贴近真实数据。将预测流量与实际流量差值作为优化对象，基于 **5.1.3** 中总流量公式，定义函数：

$$\begin{aligned} & f(\kappa,\phi;Q_{\text{real}},f,\lambda,D,\Delta P,\mu,L) \\ & = Q_{total} - Q_{real} = Q_{pump} + Q_{press} - Q_{real} \\ & = \frac{\pi}{4} f\lambda D^2(\kappa^2\phi + (1-\phi)) + \frac{\pi D^4 \Delta P}{128\mu L\left(\frac{\phi}{\kappa^4}+1-\phi\right)} - Q_{\text{real}} \end{aligned} \tag{24}$$

模型求解即转化为双变量回归问题：

$$\begin{aligned} \min.\ F(\kappa,\phi;Q,f,\lambda,D,\Delta P,\mu,L) \ &= \ \frac{1}{2}\sum_{k=1}^{n_{dataset}} f\ (\kappa,\phi;Q,f,\lambda,D,\Delta P,\mu,L)^{\,2} \\ s.t.\ \kappa \in (0,1),\ \phi \in (0,1) & \end{aligned} \tag{25}$$

可推得目标函数$F(\boldsymbol{\theta};\boldsymbol{x})$与两待优化参数间的偏导数为：

$$\begin{aligned} \frac{\partial F}{\partial \kappa} &= \frac{\pi}{2} f\lambda D^2 \kappa\theta + \frac{\pi D^4 \Delta P}{128\mu L} \frac{4\theta}{\kappa^5 (\theta/\kappa^4 + 1 - \theta)^2} \\ \frac{\partial F}{\partial \theta} &= \frac{\pi}{4} f\lambda D^2(\kappa^2 - 1) - \frac{\pi D^4 \Delta P}{128\mu L} \frac{1/\kappa^4 - 1}{(\theta/\kappa^4 + 1 - \theta)^2} \end{aligned} \tag{26}$$

由问题形式与问题性质，考虑使用梯度下降法对问题进行求解。

梯度下降是一种广泛用于求解线性和非线性模型最优解的迭代算法，它的中心思想在于通过迭代次数的递增，调整权重使得损失函数最小化，从而不断逼近最优结果。

在本题的求解中，先将目标函数$F(\theta;x)$的初始参数值设定为θ_0，在每次迭代内计算当前参数下的梯度向量$\nabla F(\theta;x)$与动态衰减因子，并用衰减后梯度更新参数，直至梯度向量 g的所有分量均小于阈值 ϵ，则跳出循环。

以伪代码的形式呈现如下：

Input	目标函数$F(\boldsymbol{\theta};\boldsymbol{x})$, 梯度$\nabla F(\boldsymbol{\theta};\boldsymbol{x})$, 初始参数$\boldsymbol{\theta_o}$, 学习率$\boldsymbol{\eta}$ 最大迭代次数n_{iters}, 梯度阈值ϵ
Output	$\boldsymbol{\theta}$, $F(\boldsymbol{\theta};\boldsymbol{x})$
	$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta_o}$ **For** $i = 1$ to n_{iters} **do**: Let $\boldsymbol{g} = \nabla F(\boldsymbol{\theta},\boldsymbol{x})$, $decay = 1 - \frac{i}{n_{iters}}$ **If** $all\ \boldsymbol{g} < \epsilon$: **break;** $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \boldsymbol{\eta} \otimes \boldsymbol{g} \cdot decay$ ($\otimes$: *Hadamard Product*) **End**

由于方波在$\kappa=0.5,\phi=0.5$的条件下最接近实际蠕动波的正弦波形，经过分析，设置初始参数表 3 所示。

表 3 梯度下降算法的初始参数设置

n_{iters}	1900
$\boldsymbol{\theta}_0=(\kappa,\ \phi)^T$	$(0.5,\ 0.5)^T$
$\boldsymbol{\eta}\ =\ (\eta_\kappa,\ \eta_\phi)^T$	$(1e10,\ 1e12)^T$
ε	$1e-6$

根据以上过程，得到结果：

$$\kappa=0.486485,\phi=0.588144 \tag{27}$$

故最终解析式模型的**总流量公式**为

$$Q_{\text{total}}=0.4327\cdot fAD^2+0.0913\cdot\frac{\pi D^4\Delta P}{128\mu L} \tag{28}$$

将求取结果回代入式（22），参照附件 1 给出的正常肾脏输尿管数据对尿液流量进行计算，并与实际值进行比较。

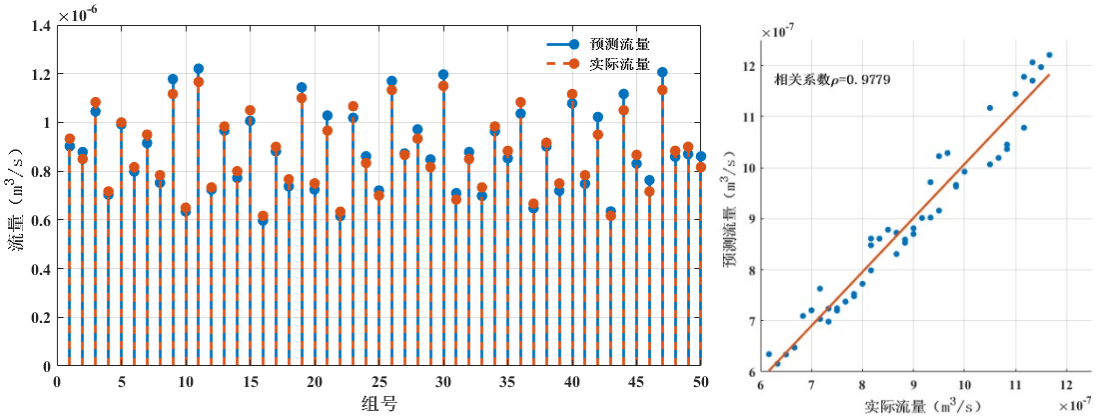

图 8 解析式模型预测流量与实际流量对比

如图 8 所示，可以看到，模型预测流量结果与实际值具有极高的相似度，仅在极小范围内浮动。同时经相关度检验，两者相关性系数达到$\rho=0.9779$，这一结果很好的反映了模型的合理性。

为验证梯度下降法的合理性，选取不同初始参数值重复，结果如图 9 所示。

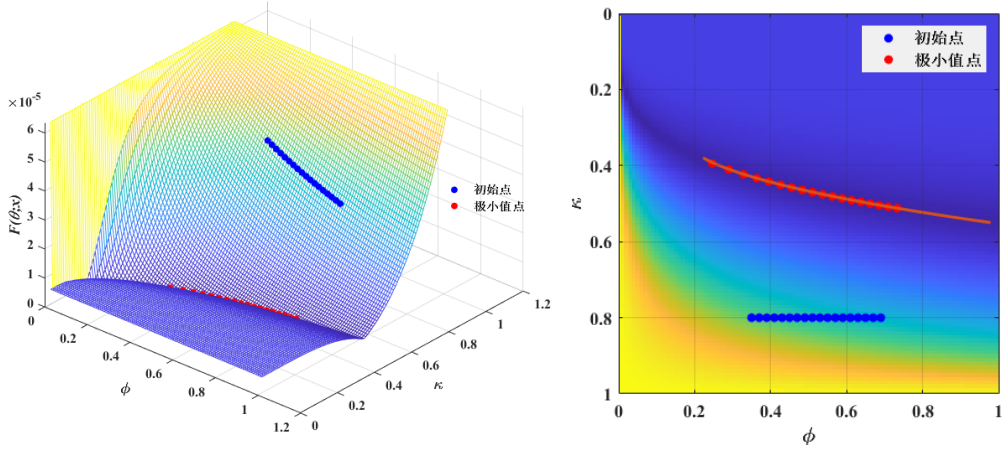

图 9 不同初始参数的梯度下降结果

图中蓝色点集为不同初始参数，红色点集为对应的梯度下降结果，通过观察可以发现，对于任意初始参数，其经梯度下降后的结果均会落在同一条平面抛物线上。若定义压缩系数$\gamma=\frac{\phi}{\kappa^2}=\frac{L_1D^2}{Ad^2}$，即单次蠕动腔体中管腔收缩段对流量的压缩能力。而在同一输尿管内该能力无疑是恒定的。这一结论证明，抛物线作为梯度下降结果的集合，其存在是符合物理规律的，进而验证了本方法的合理性。

5.2 问题二的模型建立与求解

人工智能正以前所未有的态势深度融入医疗领域，为医疗行业带来诸多变革。从监管到药物器械研发，再到医疗数据管理与服务，AI 的身影无处不在。

基于题目要求，考虑使用**物理信息神经网络**（Physics-Informed Neural Networks，简称 PINN）建立模型。这是一种结合了深度学习和物理学知识的 AI 算法。与传统的数据驱动的神经网络不同，PINN 在学习过程中利用物理法则对模型进行指导，将物理规律嵌入神经网络训练的框架，从而提高模型泛化能力。其在解决数据匮乏、数据获取成本较高或是噪声较大的情况下的问题时尤有价值。

5.2.1 物理信息引导的神经网络模型

根据 5.1 建立的输尿管流动结构，可得到以下物理约束条件：

- **轴向控制方程约束**：建立了速度场与压力场之间的关系

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)=\frac{1}{\mu}\frac{\partial p}{\partial z} \tag{29}$$

- **管壁运动边界条件约束**：假设管壁半径随蠕动按照正弦波变化

$$R(z)=\frac{D}{4}\left(2-\phi\cos\left(2\pi\frac{z}{A}\right)-\phi\right) \tag{30}$$

其中D为管径，A为蠕动幅度，ϕ为收缩比。

- **对称轴边界条件约束**：中心轴$r=0$处速度梯度为0

$$\left.\frac{\partial u}{\partial r}\right|_{r=0}=0 \tag{31}$$

- **压力梯度约束**：单个蠕动腔体两端压降为常数：

$$p(0)-p(A)=\frac{A}{L}\Delta P \tag{32}$$

- **流量守恒约束**：不同截面的流量相等

$$Q(z_1)=Q(z_2) \tag{33}$$

- **流量与速度场积分约束**：

$$Q=\int_0^{R(z)}2\pi r\cdot u(r)\mathrm{d}r \tag{34}$$

本题中输入数据各部分无时间、空间关系，训练数据量较小，且输出数据与输入数据之间存在解析计算关系，流量Q主要由被输入数据决定的速度场u与压力场p决定，而速度场与压力场的相互关系被公式(29)约束，故考虑采用多层感知机制为基础的(MLP-based)模型架构。

5.2.2 模型架构设计

在已知物理约束下，设计出由**物理信息引导的神经网络模型Kidney Model**，模型架构图如图 10 所示。

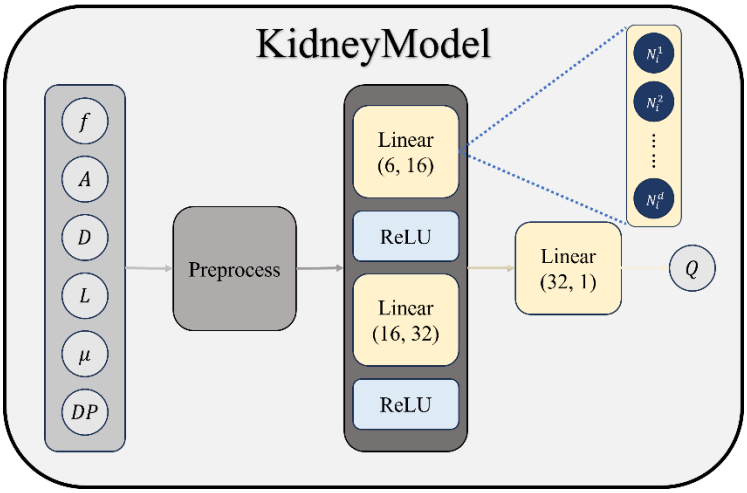

图 10 物理信息引导的神经网络模型架构示意图

模型输入值为$f, A, D, L, \mu, \Delta P$。最终输出流量$Q$。

数据预处理部分(Preprocess)，考查输入数据，发现数据之间数量级差异较大。为避免模型权重不平衡，导致梯度爆炸、消失等问题，设计如下映射对输入数据进行预处理：

$$Preprocess(\boldsymbol{x}) = [1,\ 10^2,\ 10^2,\ 1,\ 10^2,\ 10^{-3},\ 2\times10^{-3}]^T \otimes \boldsymbol{x} \tag{35}$$

值得注意的是，由于流量与压降的强相关性，直接处理较为困难。考虑在训练时将流量与压降做相同数量级处理，在推理时将得到结果进行反变换：

$$\begin{gathered}\text{Train: } \hat{y} \leftarrow y \times 2 \times 10^{-3} \\ \text{Inference: result} \leftarrow \text{output} \times 5 \times 10^{2}\end{gathered} \tag{36}$$

特征提取部分，采用双线性层（Linear）+激活函数对预处理后的输入数据进行特征提取，模拟模型对速度场与压力场特征的提取情况。采用线性整流单元（ReLU）作为激活函数，以模拟运算中的非线性变换。线性整流单元定义为

$$ReLU(\boldsymbol{x}) = \begin{cases} x, & x \geqslant 0 \\ 0, & x < 0 \end{cases} \tag{37}$$

数据输出部分，考虑到训练数据有限，为避免过拟合，采用单线性层，模拟对速度场与压力场共同作用下的流量的计算。

5.2.3 损失函数设计

模型的损失函数由两部分构成。第一部分用于衡量网络预测输出与实际观测数据之间的差异，目的是使网络能够尽可能拟合数据。使用均方误差 MSE 损失：

$$\mathscr{L}_{MSE} = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \hat{y}_i\right)^2 \tag{38}$$

其中，m为每次梯度反向传播的数据量。

第二部分为物理信息残差项，用于衡量网络预测结果是否满足物理定律，将网络预测的物理量代入相应的物理定律中计算得到残差，从而确保了物理一致性。

对每项物理约束所产生的残差进行分析，再统一正则化，以约束模型的参数值域，具体操作如下：

- **轴向控制方程损失函数：**

$$L_{\text{flow}} = \frac{1}{N}\sum_{1}^{N}\left\| \frac{1}{\mu}\frac{\partial p}{\partial z} - \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right)\right\|^2 \tag{39}$$

- **管壁运动边界条件损失函数：**即无滑移条件

$$L_{wall} = \frac{1}{N}\sum_{1}^{N}\| u(R(z),z)\|^2 \tag{40}$$

- **对称轴边界条件损失函数：**

$$L_{\text{symmetry}} = \frac{1}{N}\sum_{1}^{N}\left\| \frac{\partial u}{\partial r}\Big|_{r=0}\right\|^2 \tag{41}$$

- **压力梯度损失函数：**

$$L_{\text{pressure}} = \left\| p(0) - p(A) - \frac{A}{L}\Delta P\right\|^2 \tag{42}$$

- **流量守恒损失函数：**

$$L_{\text{conserve}} = \frac{1}{N}\sum_{i=1}^{N}\| Q(z_{1,i}) - Q(z_{2,i})\|^2 \tag{43}$$

- **流量与速度场积分损失函数：**

$$L_{\text{intergral}} = \frac{1}{N}\sum_{i=1}^{N}\left\| Q_i - \int_0^{R(z_i)} 2\pi r\cdot u(r)\,\mathrm{d}r\right\|^2 \tag{44}$$

将以上损失函数结合，形成**物理信息残差项的总损失函数：**

$$\begin{aligned} L_{\text{physics}} = {} & \lambda_1 L_{\text{flow}} + \lambda_2 L_{\text{wall}} + \lambda_3 L_{\text{symmetry}} \\ & + \lambda_4 L_{\text{pressure}} + \lambda_5 L_{\text{conserve}} + \lambda_6 L_{\text{intergral}} \end{aligned} \tag{45}$$

$L_{physics}$的各项均表现在输出层中，在模型中将其以正则化项的形式进行表示。

$$\mathscr{L}_{L_{physics}} = \sum_{i=1}^{m} {y_i}^2 \tag{46}$$

综上，模型的总损失函数如下所示。

$$\text{Loss} = \alpha\mathscr{L}_{MSE} + \beta\mathscr{L}_{physics} \tag{47}$$

其中α, β为可学习参数，由机器学习过程产生。

5.2.4 训练设计

由于训练数据有限，为充分利用数据，模型训练采用 K-fold 交叉检验方法，具体操作如下：

STEP1: 按互不重叠、比例近似相等的原则将数据集划分为K份；

STEP2: 选取其中$K-1$份作为训练数据，1份作为验证数据，记录每次训练模型的性能；

STEP3: 重复K次训练；

STEP4: 用所有数据进行整体训练，最终的性能度量由K次验证集上的结果按样本数加权平均得到。

由于不同物理量之间存在单位与数量级差异，考虑采用 Adam 优化器，动态调整学习率，达到较好的训练效果。

在外层迭代i的循环内设置内层循环，对模型参数合集 **Params** 中的每个参数θ独立执行。计算当前参数下的梯度g，通过计算指数加权移动平均的梯度动量m更新一阶矩估计β_1，通过计算梯度平方的指数加权平均值v更新二阶矩估计β_2，修正初始零值的偏差后，应用自适应学习率规则更新参数θ。

Adam 优化器运行流程的伪代码如下：

Input	模型M，模型参数**Params**，学习率α 一阶矩估计β_1，二阶矩估计β_2，最大迭代次数n_{iters}
Output	**Params**
	For $i=1$ to n_{iters}: For θ in **Params**: $g=\nabla M(\theta)$ $m = \beta_1 \times m + (1-\beta_1) \times g$ $v = \beta_2 \times v + (1-\beta_2) \times g^2$ $\hat{m}=\frac{m}{1-\beta_1^t}$, $\hat{v}=\frac{v}{1-\beta_2^t}$ $\theta -= a \times \frac{\hat{m}}{\sqrt{\hat{v}}+10^{-6}}$

5.2.5 结果分析

根据以上模型，对已知的正常肾脏输尿管流量数据展开分析，得到结果如图 11 所示。可以看到，本模型计算的流量数据与实际流量数据基本吻合。虽在部分组上存在一定误差，但这一现象在回归模型中可以被认为是合理的。

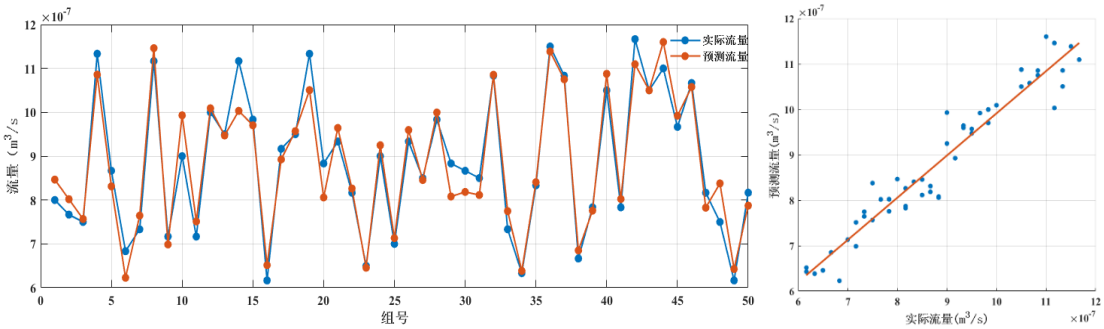

图 11 AI 模型预测流量与实际流量对比

为进一步探寻模型的有效性，对每一组数据的残差进行统计分析，结果如图 12 所示。左图反映了残差项值的统计分布，发现该分布大致符合标准正态分布的形态特征。为进一步判断两分布之间的相似度，绘制两分布间的分位数-分位数图，即图 12 右图。可以看到样本分布与理论分布相似，即残差项值的统计分布符合标准正态分布规律。这一结论说明预测流量的残差具有相当的随机性，该

模型具有较高的置信度，模型结果精确率高。

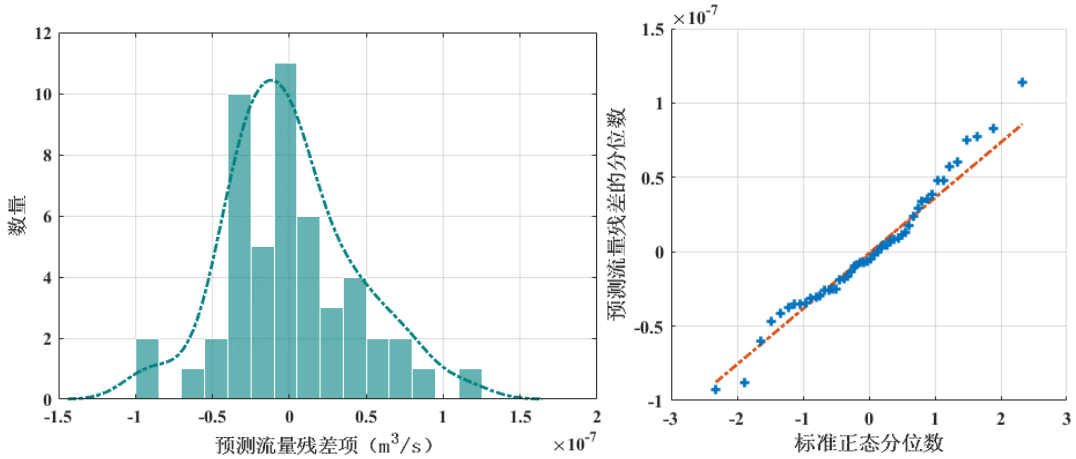

图 12 预测流量残差项直方图（左）；分位数-分位数图（右）

选取部分指标对本模型拟合效果进行评估，各指标值计算结果如表 4 所示。

表 4 神经网络模型指标

样本数量	RMSE(m^3/s)	MAE(m^3/s)	R^2
50	3.939×10^{-8}	3.101×10^{-8}	0.9371

RMSE 值和MAE 值反映了预测值偏离真实值的平均幅度，本模型的RMSE 值和MAE 值均小于实际流量的数量级，说明计算结果与实际值的偏差较小，本模型拟合误差较小；决定系数R^2衡量了模型对数据的拟合程度，本模型的R^2值为0.9371，说明模型拟合效果较好。

5.3 问题三的模型建立与求解

根据题目要求，基于附件 1 给出的患者手术前后输尿管参数，使用 5.1 中构建的解析式模型与 5.2 中构建的 AI 模型分别对输尿管流量进行预测。已知样本数据如下：

表 5 手术前后流量数据

	手术前数据	手术后数据	单位
蠕动频率	0.05	0.12	Hz
蠕动幅度	0.0002	0.0008	m
输尿管直径	0.001	0.004	m
输尿管长度	0.28	0.28	m
尿液粘度	0.0012	0.0012	Pa·s
压降	3066	130	Pa
实际流量	1.21	13.11	ml/min

使用已建立模型对输尿管流量进行计算，并将预测数据与附件 2 给出的实际数据进行比较，结果如下：

表 6 解析式模型和 AI 模型预测流量对比

		手术前数据	手术后数据
实际流量m^3/s		2.017×10^{-8}	2.185×10^{-7}
解析式模型	预测流量m^3/s	2.045×10^{-8}	2.226×10^{-7}
	相对误差	1.4%	1.9%
AI 模型	预测流量m^3/s	5.167×10^{-7}	7.167×10^{-7}
	相对误差	2460%	230%

可以看到，解析式模型预测流量与实际流量基本已知，相对误差较小。AI 模型预测流量与实际值相对误差较大，经评估，可能原因主要有两点：其一，模型的训练数据集质量较低。数据集只有 50 条数据，且数据的分布集中，不具代表性，这导致模型在学习过程中无法学习到具有普适性的深层信息。其二，模型架构简单。受限于较小的数据集，模型的架构较为简单。尽管充分考虑物理信息的引导，但较少的层数与维度使模型无法充分学习物理规律。

结合模型结构分析与文献资料查找，在模型预测结果的基础上对解析式模型和 AI 模型的**局限性**进行讨论。

对于解析式模型的建立过程，我们的假设是一个较为简单理想的流动结构，并未考虑实际情况下病理因素导致的流动结构的动态变化。模型假设了输尿管蠕动均匀协调（频率f和幅度A恒定），但在梗阻、渗漏等病理状态下，蠕动波可能存在失调或消失现象，肾盂收缩所提供的初始动力在假设中同样被忽视，实际上该值可能会因病理因素减小。此外，模型假设了蠕动波的传递形式为方波，这与真实情况的正弦蠕动也存在一定差异。

对于 AI 模型，除去与解析式模型相同的前置模型假设局限，其在数据与泛化方面同样存在缺失，首先是训练数据偏差： 由于本模型仅使用 50 条正常肾脏数据作为训练集，缺乏大规模样本以及异常状态样本（如低流量、高压力下的数据），这也导致了本模型对对术前状态预测失效。受限于训练数据集的规模，特征提取层与输出层的设置也较为单薄，使得模型无法充分学习。

针对以上不足，对两模型分别提出可行的**优化改进方案**：

针对解析式模型：

1. 添加衰减因子,（如蠕动效率衰减系数$\eta(F,\Delta P)$），根据病例描述的病理阶段（急性期蠕动增强，慢性期消失）随时对因子进行动态调整；
2. 将蠕动波建模为真实蠕动情况的正弦波。

针对 AI 模型：

1. 使用更多高质量数据扩充数据集。可通过人工采集，人工标注方式，获取各个情况下的数据，以提高数据集的覆盖率，提升模型的泛化性；
2. 引入更为直接的医工学先验知识。目前模型考虑的物理知识均为力学分析结果，难以直接在模型中进行表征。可以加入更多医工学先验知识，显性的引导模型学习；
3. 设计数据生成流水线，通过模拟手段，生成批量数据集，供神经网络学习。

六、 灵敏度分析

6.1 解析式模型对收缩半径比κ的灵敏度

在上文的解析式模型中，得到收缩半径比κ的参数为0.486485。考虑实际生理情况，为探究收缩半径比改变对流量的影响，我们使参数κ在$[0.44, 0.52]$范围内，以0.2为步长进行调整，并按照解析式模型进行流量预测，如图 13 所示。

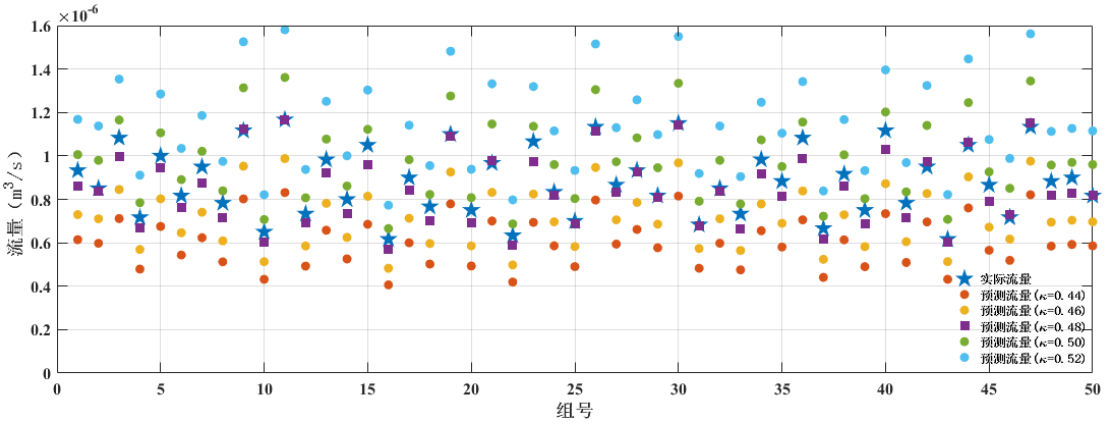

图 13 不同收缩半径比κ下预测流量与实际流量对比图

可以看到，预测流量随收缩半径比κ的增大显著上涨，κ值 0.02 的改变量能带来约 20%的增长收益，且调整后预测流量的总体分布依然符合实际流量的分布特征。这表明解析式模型对收缩半径比较敏感，同时也具有一定的稳健性。

6.2 解析式模型对收缩长度比ϕ的灵敏度

在上文的解析式模型中，得到收缩长度比ϕ的参数为0.588144。考虑实际生理情况，为探究收缩长度比改变对流量的影响，我们使参数ϕ在$[0.54, 0.62]$范围内，以0.2为步长进行调整，并按照解析式模型进行流量预测，如图 14 所示。

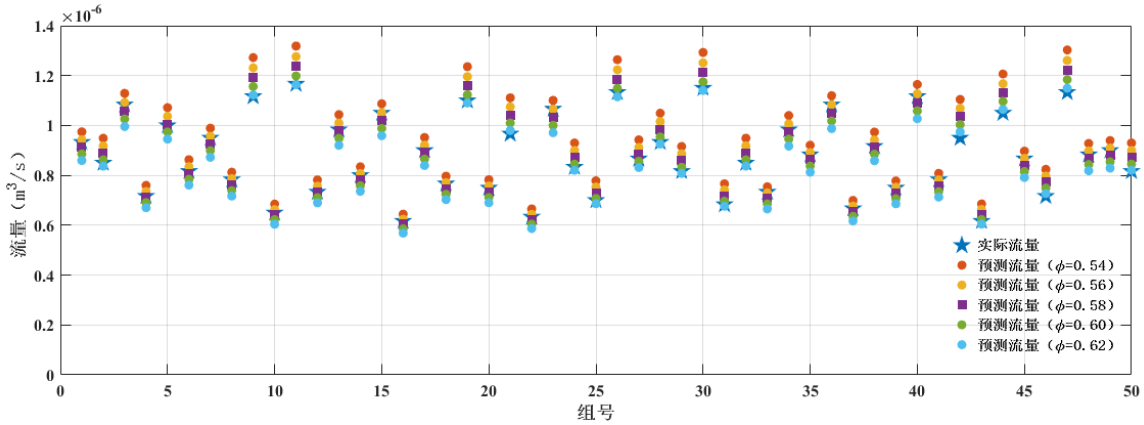

图 14 不同收缩长度比ϕ下预测流量与实际流量对比图

可以看到，预测流量随收缩长度比ϕ的增大小幅下降，调整后预测流量的总体分布也依然符合实际流量的分布特征。这表明解析式模型对收缩长度比不敏感，同时也再一次验证了模型的稳健性。

6.3 神经网络模型对激活函数形式的灵敏度

在上文的神经网络模型中，采用性整流单元 ReLU 作为激活函数，以模拟运算中的非线性变换。为探究不同激活函数形式对预测流量效果的影响，采用 GELU、Sigmoid、Tanh、Mish 作为激活函数进行训练，函数形式和训练效果如表 7 所示。

表 7 不同激活函数形式的训练效果

激活函数	形式	$\mathrm{RMSE}(\mathrm{m}^3/\mathrm{s})$	R^2
ReLU	$ReLU(x)=\begin{cases} x, & x\geqslant 0 \\ 0, & x<0 \end{cases}$	3.939×10^{-8}	0.9371
GELU	$\mathrm{GELU}(x)=0.5x(1+\\+\tanh\left(\sqrt{\frac{2}{\pi}}(x+0.044715x^3)\right))$	5.022×10^{-8}	0.8977
Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	5.524×10^{-8}	0.8762
Tanh	$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	4.468×10^{-8}	0.9190
Mish	$f(x)=x\cdot\tanh(\ln(1+e^{x}))$	5.286×10^{-8}	0.8866
	无激活函数	5.156×10^{-8}	0.8921

可以看到，ReLU 依然具有最好的激活性能。换用不同激活函数后，模型的拟合效果均有不同程度的小幅下降，具体表现为结果中 RMSE 值的小幅上升及 R^2 值的小幅下降，下降幅度均在可接受内。

七、 模型评价与推广

7.1 模型的优点

针对解析式模型：

- 引入收缩管径比、收缩长度比两无量纲数，对输尿管输送尿液的结构和机理进行了精确的数学描述，详细解释了蠕动泵送和压降对于尿流的推动作用；
- 采取梯度下降法对模型进行求解，通过迭代得到最优参数，流量预测结果与实际值误差小，具有极高的精确度。

针对 AI 模型：

- KidneyModel 在训练数据集上展现出了较好的性能。其推理速度快，可拓展性强，在省去复杂计算的同时在实际物理意义上具有较强可解释性；
- 针对训练数据集样本较少，采用 K-fold 交叉检验，对训练集进行充分利用。针对不同物理量之间单位与数量级的差异，采用 Adam 优化器，动态调整学习率，达到较好的训练效果。

7.2 模型的缺点

- 模型假设是一种简单理想的流动结构，但实际情况下梗阻、渗漏等病理因素

可能会对管道情况产生影响，比如管径变小、尿液粘度上升、蠕动波传播衰减等，模型缺乏对这类复杂情况的兼容能力；

- 处于简化模型考虑，模型将蠕动波视为方波进行考虑，这与实际情况的正弦波形存在一定偏差；
- 受限于训练集数据分布窄、规模小，AI 模型的训练架构较为简单，在对于训练集分布外数据的处理上能力不足。

7.3 模型的推广

针对解析式模型：

- 添加衰减因子，根据病例描述的病理阶段与病灶发生位置，评估该病灶对于输尿管各参数的影响情况，随时对因子进行动态调整；
- 将蠕动波重新建模为正弦波形进行分析，使模型更贴近真实蠕动情况。

针对 AI 模型：

- 可通过人工采集标注，获取更多高质量数据扩充数据集，以提高数据集的分布覆盖，提升模型的泛化性；
- 目前模型考虑的物理知识均为力学分析结果，难以直接在模型中进行表征。考虑加入更多医工学先验知识，显性的引导模型学习；
- 结合实际病例的尿动力学分析，设计数据生成流水线，通过模拟手段，生成批量数据集，供神经网络学习。

参考文献

[1] Karnak, I., N. Büyükpamukçu, and F. C. Tanyel. "The effects of flow rate, length and external pressure upon the pressure required for fluid to flow through a ureter." BJU international 88.4 (2001): 335-338.

[2] Vahidi, Bahman, et al. "A mathematical simulation of the ureter: effects of the model parameters on ureteral pressure/flow relations." (2011): 031004.

[3] Griffiths, D. J. "Flow of urine through the ureter: a collapsible, muscular tube undergoing peristalsis." (1989): 206-211.

[4] Jimenez Lozano, Joel N. Peristaltic flow with application to ureteral biomechanics. Diss. University of Notre Dame, 2009.

附录

附录一 支撑材料列表

程序一：Q1.m
程序二：Q3.m
程序三：utils.py
程序四：Problem_1.py
程序五：Problem_2.py
程序六：Problem_3.py

附录二 程序代码

程序一：Q1.m

```
clc,clear
data=readmatrix('data_new.csv');
f=data(:,1); % 频率
A=data(:,2); % 幅度
D=data(:,3); % 管径 D
L=data(:,4); % 长度 L
mu=data(:,5); % 黏度 μ
Dp=data(:,6); % 压降
Qm_real=data(:,8)/60/1000/1000; % 流量 单位 m^3/s
Q_fore=320.*f.*A.*D.*D;
X1=f.*A.*D.*D;
X2=pi*D.^4.*Dp./mu./L/128;

%%
Q_fang=zeros(50,1);
result=zeros(101,101);
for kappa=0:0.01:1
    for phi=0:0.01:1
        Q_fang=pi/4.*f.*A.*D.*D*((kappa^2)*phi+1-
phi)+(D.^4).*pi.*Dp./(128*mu.*L*(phi/(kappa^4)+1-phi));
        %Distance=1 - (sum((Q_fang- Qm_real).^2) / sum((Qm_real
- mean(Qm_real)).^2));
        Distance=sqrt(sum((Q_fang(:)-Qm_real(:)).^2));
        result(round(kappa*100+1),round(phi*100+1))=Distance;
    end
end
[X, Y] = meshgrid(1:101, 1:101);
X = X(:);
Y = Y(:);
for i=1
```

```
Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_origin(i)^2)*theta_origin(
i)+1-
theta_origin(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_origin(i)/(
kappa_origin(i)^4)+1-theta_origin(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));

scatter3(theta_origin(i),kappa_origin(i),Distance_new,'blue','
filled')
    hold on
end
for i=1

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_end(i)^2)*theta_end(i)+1-
theta_end(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_end(i)/(kappa_
end(i)^4)+1-theta_end(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));

scatter3(theta_end(i),kappa_end(i),Distance_new,'red','filled'
)
end
for i=21:20:length(kappa_origin)

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_origin(i)^2)*theta_origin(
i)+1-
theta_origin(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_origin(i)/(
kappa_origin(i)^4)+1-theta_origin(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));

scatter3(theta_origin(i),kappa_origin(i),Distance_new,'blue','
filled')
end
for i=21:20:length(kappa_end)

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_end(i)^2)*theta_end(i)+1-
theta_end(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_end(i)/(kappa_
end(i)^4)+1-theta_end(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));

scatter3(theta_end(i),kappa_end(i),Distance_new,'red','filled'
)
end
imagesc(0:0.01:1,0:0.01:1,result)
xlabel('\kappa', 'FontSize', 12);
```

```
ylabel('\phi', 'FontSize', 12);
axis equal
%%
Q_fang=zeros(50,1);
result=zeros(101,101);
for kappa=0:0.01:1
    for phi=0:0.01:1
        Q_fang=pi/4.*f.*A.*D.*D*((kappa^2)*phi+1-
phi)+(D.^4).*pi.*Dp./(128*mu.*L*(phi/(kappa^4)+1-phi));
        %Distance=1 - (sum((Q_fang- Qm_real).^2) / sum((Qm_real
- mean(Qm_real)).^2));
        Distance=sqrt(sum((Q_fang(:)-Qm_real(:)).^2));
        result(round(kappa*100+1),round(phi*100+1))=Distance;
    end
end
[X, Y] = meshgrid(1:101, 1:101);
% X = X(:);
% Y = Y(:);
imagesc(0:0.01:1,0:0.01:1,result)
hold on
for i=1

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_origin(i)^2)*theta_origin(
i)+1-
theta_origin(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_origin(i)/(
kappa_origin(i)^4)+1-theta_origin(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));
    scatter(theta_origin(i),kappa_origin(i),'blue','filled')
    hold on
end
for i=1

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_end(i)^2)*theta_end(i)+1-
theta_end(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_end(i)/(kappa_
end(i)^4)+1-theta_end(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));
    scatter(theta_end(i),kappa_end(i),'red','filled')
end
for i=21:20:length(kappa_origin)

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_origin(i)^2)*theta_origin(
i)+1-
theta_origin(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_origin(i)/(
kappa_origin(i)^4)+1-theta_origin(i)));
```

```
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));
    scatter(theta_origin(i),kappa_origin(i),'blue','filled')
end
for i=21:20:length(kappa_end)

Q_fang_new=pi/4.*f.*A.*D.*D*((kappa_end(i)^2)*theta_end(i)+1-
theta_end(i))+(D.^4).*pi.*Dp./(128*mu.*L*(theta_end(i)/(kappa_
end(i)^4)+1-theta_end(i)));
    Distance_new=sqrt(sum((Q_fang_new(:)-Qm_real(:)).^2));
    scatter(theta_end(i),kappa_end(i),'red','filled')
end
phi_nihe=0.38:0.01:0.55;
kappa_nihe=15.4553*phi_nihe.*phi_nihe-9.92*phi_nihe+1.76;
hold on
plot(kappa_nihe',phi_nihe')
xlabel('\phi', 'FontSize', 12);
ylabel('\kappa', 'FontSize', 12);
axis equal
%%
scatter(1:50,data(:,7),'filled');
hold on
scatter(1:50,data(:,10),'filled');
scatter(1:50,data(:,11),'filled');
%%
R = corrcoef(Qm_real, X2);       % 计算相关系数矩阵
correlation = R(1, 2);    % 提取 Pearson 相关系数
figure;
scatter(Qm_real, X2, 80, 'filled', 'MarkerFaceColor', [0.2,
0.6, 0.9]); % 绘制散点图
hold on;

% 添加线性拟合线
p = polyfit(Qm_real, X2, 1);          % 一元线性拟合
y_fit = polyval(p, Qm_real);         % 计算拟合值
plot(Qm_real, y_fit, 'r-', 'LineWidth', 2); % 绘制拟合线

% 设置图形样式
grid on;
xlabel('X 数据', 'FontSize', 12);
ylabel('Y 数据', 'FontSize', 12);
title('两组数据相关性分析', 'FontSize', 14);

% 在图中标注相关系数
text(0.6, 0.9, sprintf('相关系数 r = %.4f', correlation), ...
```

```
    'Units', 'normalized', 'FontSize', 12, ...
    'BackgroundColor', [1, 1, 0.8], ...
    'EdgeColor', [0.3, 0.3, 0.3]);

% 添加数据标签示例
text(0.05, 0.95, ['y = ', num2str(p(1), '%.2f'), 'x + ',
num2str(p(2), '%.2f')], ...
    'Units', 'normalized', 'FontSize', 10, 'Color', 'red');

hold off;
disp(['Pearson 相关系数: ', num2str(correlation)]);
%%
R = corrcoef(Qm_real, Q_fang);       % 计算相关系数矩阵
correlation = R(1, 2);    % 提取 Pearson 相关系数
figure;
scatter(Qm_real, Q_fang, 80, 'filled', 'MarkerFaceColor',
[0.2, 0.6, 0.9]); % 绘制散点图
hold on;

% 添加线性拟合线
p = polyfit(Qm_real, Q_fang, 1);          % 一元线性拟合
y_fit = polyval(p, Qm_real);         % 计算拟合值
plot(Qm_real, y_fit, 'r-', 'LineWidth', 2); % 绘制拟合线

% 设置图形样式
grid on;
xlabel('X 数据', 'FontSize', 12);
ylabel('Y 数据', 'FontSize', 12);
title('两组数据相关性分析', 'FontSize', 14);

% 在图中标注相关系数
text(0.6, 0.9, sprintf('相关系数 r = %.4f', correlation), ...
    'Units', 'normalized', 'FontSize', 12, ...
    'BackgroundColor', [1, 1, 0.8], ...
    'EdgeColor', [0.3, 0.3, 0.3]);

% 添加数据标签示例
text(0.05, 0.95, ['y = ', num2str(p(1), '%.2f'), 'x + ',
num2str(p(2), '%.2f')], ...
    'Units', 'normalized', 'FontSize', 10, 'Color', 'red');

hold off;
disp(['Pearson 相关系数: ', num2str(correlation)]);
```

```
%%
kappa=0.486;
phi=0.588;
Q_fang=pi/4.*f.*A.*D.*D*((kappa^2)*phi+1-
phi)+(D.^4).*pi.*Dp./(128*mu.*L*(phi/(kappa^4)+1-phi));
Distance=sqrt(sum((Q_fang(:)-Qm_real(:)).^2));
R2=1 - (sum((Q_fang- Qm_real).^2) / sum((Qm_real -
mean(Qm_real)).^2));
%scatter(Q_fang,Qm_real)
%axis equal
stem(1:50,Q_fang)
hold on
stem(1:50,Qm_real)
%plot([min(Qm_real), max(Qm_real)], [min(Qm_real),
max(Qm_real)], 'r--');
%%
phi=0:0.01:1;
for kappa=0.3:0.1:0.7
Q_kappa=pi/4*0.1*0.001*0.0045*0.0045*((kappa.^2).*phi+1-
phi)+(0.0045^4).*pi*250./(128*0.001*0.28*(phi./(kappa.^4)+1-
phi));
plot(phi,Q_kappa)
hold on
end
```

程序二： Q3.m

```
clc,clear
data=readmatrix('data_new.csv');
f=data(:,1); % 频率
A=data(:,2); % 幅度
D=data(:,3); % 管径 D
L=data(:,4); % 长度 L
mu=data(:,5); % 黏度 μ
Dp=data(:,6); % 压降
Qm_real=data(:,8)/60/1000/1000; % 流量 单位 m^3/s
%%
stem(1:50,Qm_real)
hold on
for kappa=0.44:0.02:0.52
for phi=0.588
Q_fang=pi/4.*f.*A.*D.*D*((kappa^2)*phi+1-
phi)+(D.^4).*pi.*Dp./(128*mu.*L*(phi/(kappa^4)+1-phi));
Distance=sqrt(sum((Q_fang(:)-Qm_real(:)).^2));
```

```
R2=1 - (sum((Q_fang- Qm_real).^2) / sum((Qm_real -
mean(Qm_real)).^2));
%scatter(Q_fang,Qm_real)
%axis equal
stem(1:50,Q_fang)
hold on
%plot([min(Qm_real), max(Qm_real)], [min(Qm_real),
max(Qm_real)], 'r--');
end
end

%%
stem(1:50,Qm_real)
hold on
for kappa=0.486
for phi=0.54:0.02:0.62
Q_fang=pi/4.*f.*A.*D.*D*((kappa^2)*phi+1-
phi)+(D.^4).*pi.*Dp./(128*mu.*L*(phi/(kappa^4)+1-phi));
Distance=sqrt(sum((Q_fang(:)-Qm_real(:)).^2));
R2=1 - (sum((Q_fang- Qm_real).^2) / sum((Qm_real -
mean(Qm_real)).^2));
%scatter(Q_fang,Qm_real)
%axis equal
stem(1:50,Q_fang)
hold on
%plot([min(Qm_real), max(Qm_real)], [min(Qm_real),
max(Qm_real)], 'r--');
end
end
```

程序三： **utils.py**

```
from __future__ import annotations
import math
import random
import warnings
from pathlib import Path
from typing import List, Union, Sequence

import numpy as np
import torch
import torch.nn as nn
from torch.autograd import Variable
from torch.utils.data import Dataset, DataLoader, TensorDataset
from sklearn.model_selection import KFold
```

```
from openpyxl import load_workbook
from tqdm import tqdm
import seaborn as sns
import matplotlib.pyplot as plt

from rich.console import Console
from rich.table import Table
from rich import box

console = Console()
sns.set_style("whitegrid")

class Item:
    __slots__ = ("index", "f", "A", "D", "L", "mu", "DP", "Q1",
"Q2")

    def __init__(self, row: List[Union[int, float, str]]) -> No
ne:
        def _to_float(x):
            try:
                return float(x)
            except (ValueError, TypeError):
                return float("nan")
        if len(row) < 9:
            raise ValueError(f"Row has {len(row)} < 9 columns.")
        (
            self.index,
            self.f,
            self.A,
            self.D,
            self.L,
            self.mu,
            self.DP,
            self.Q1,
            self.Q2,
        ) = map(_to_float, row)

    def __repr__(self) -> str:
        return f"Item(index={self.index:6f}\nf={self.f:6f}\tA=
{self.A:6f}\tD={self.D:6f}\tL={self.L:6f}\nmu={self.mu:6f}\tDP
={self.DP:6f}\tQ1={self.Q1:6f}\tQ2={self.Q2:6f})"

def _iter_excel_rows(file_path: str, sheet_index: int = 0):
    file_path = Path(file_path).expanduser()
```

```
    if not file_path.is_file():
        raise FileNotFoundError(f"Excel file not found: {file_pa
th}")
    wb = load_workbook(file_path, read_only=True, data_only=Tru
e)
    ws = wb.worksheets[sheet_index]
    rows_iter = ws.iter_rows(values_only=True)
    next(rows_iter, None)
    for row in rows_iter:
        if row is None or all(v is None or str(v).strip() == ""
for v in row):
            continue
        yield list(row)

def standard(item):
    if isinstance(item, Item):
        return (
            [item.f, item.A * 100, item.D * 100, item.L, item.mu
 * 100, item.DP / 1000],
            [item.Q2 / 500],
        )
    else:
        return (
            [item[0], item[1] * 100, item[2] * 100, item[3], ite
m[4] * 100, item[5] / 1000],
            [item[6] / 500],
        )

def restan(output):
    return 500 * output

class KidneyDataset(Dataset):
    def __init__(self, file_path: str = "./Problem_A/data_new.x
lsx", sheet_index: int = 1) -> None:
        super().__init__()
        self.items: List[Item] = []
        for row in _iter_excel_rows(file_path, sheet_index):
            try:
                self.items.append(Item(row))
            except Exception as e:
                warnings.warn(f"Skipping invalid row {row}: {e}
")
        if len(self.items) == 0:
            raise RuntimeError("No valid data found in Excel!")
```

```
    def __len__(self) -> int:
        return len(self.items)

    def __getitem__(self, idx: int) -> dict[str, torch.Tensor]:
        item = self.items[idx]
        x, y = standard(item)
        return {"x": torch.tensor(x, dtype=torch.float32), "y":
torch.tensor(y, dtype=torch.float32)}

class KidneyModel(nn.Module):
    def __init__(self, in_dim: int = 6, hidden: int = 16, dropou
t: float = 0.0) -> None:
        super().__init__()
        self.feature = nn.Sequential(
            nn.Linear(in_dim, hidden),
            nn.ReLU(),
            nn.Linear(hidden, 2*hidden),
            nn.ReLU(),
            nn.Dropout(dropout),
        )
        self.output = nn.Linear(2*hidden,1)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.feature(x)
        out = self.output(x)
        return torch.squeeze(out)

class KidneyCriterion(nn.Module):
    def __init__(self, a, b, c):
        super(KidneyCriterion, self).__init__()
        self.a = a
        self.b = b
        self.c = c

    def forward(self, out, label):
        loss = 0
        Q = out
        loss += self.a*torch.nn.functional.mse_loss(Q, label)
        loss += self.c*torch.sum(Q[:]**2)
        return loss

def train_once(model, train_loader, criterion, optimizer, devic
e, epochs=200):
```

```
    model.train()
    for _ in tqdm(range(epochs), desc="Training", leave=False):
        for x, y in train_loader:
            x, y = x.to(device), y.to(device)
            optimizer.zero_grad()
            loss = criterion(model(x), y)
            loss.backward()
            optimizer.step()
    model.eval()
    return model

@torch.no_grad()
def inference(model, x, device):
    if isinstance(x, (list, tuple)):
        x = torch.tensor(x, dtype=torch.float32)
    x = x.to(device)
    if x.dim() == 1:
        x = x.unsqueeze(0)
    model.eval()
    return restan(model(x).item())

def load_model(weight_path: Path, device: torch.device):
    model = KidneyModel()
    model.load_state_dict(torch.load(weight_path, map_location=
device))
    model.to(device)
    model.eval()
    return model

def predict_single(model, features, device):
    x, _ = standard(features)
    x = torch.tensor([x], dtype=torch.float32, device=device)
    with torch.no_grad():
        result = model(x)
        return restan(result)
```

程序四：Problem_1.py

```
from utils import *
from matplotlib import pyplot as plt
from scipy.io import savemat

def Func(Q, f, a, D, DP, mu, L, kappa, theta):
    tmp1 = (np.pi/4)* f * a * D**2 * (kappa**2 * theta + (1-thet
a))
```

```
    tmp2 = (np.pi * D**4 * DP) / (128 * mu * L * ((theta / (kapp
a**4) + 1 - theta)))
    return tmp1 + tmp2 - Q

def Grad_kappa(Q, f, a, D, DP, mu, L, kappa, theta):
    tmp1 = (np.pi / 2) * f * a * D**2 * kappa * theta
    tmp2 = (np.pi * D**4 * DP) / (128 * mu * L) * (4*theta/(kapp
a**5)) / ((theta/(kappa**4) + 1 - theta)**2)
    return tmp1 + tmp2

def Grad_theta(Q, f, a, D, DP, mu, L, kappa, theta):
    tmp1 = (np.pi/4) * f * a * D**2 * (kappa**2 - 1)
    tmp2 = (np.pi*D**4 * DP) / (128 * mu * L) * (1/((kappa**4) -
 1)) / ((theta/(kappa**4) + 1 - theta)**2)
    return tmp1 - tmp2

def main():
    dataset = KidneyDataset(file_path="./Problem_A/data_new.xls
x").items
    L = len(dataset)
    n_epoch = 1000
    lr_kappa = 1e10
    lr_theta = 1e12
    kappa = 0.47
    theta = 0.57
    result = []
    for kk in range(250, 600):
        kappa = 0.47
        kappa0 = 0.47
        theta = float(kk)/1000+0.1
        theta0 = float(kk)/1000+0.1
        for i in range(n_epoch):
            loss = 0
            grad_kappa = 0
            grad_theta = 0
            for it in dataset:
                loss += Func(it.Q1, it.f, it.A, it.D, it.DP, it.m
u, it.L, kappa, theta)/len(dataset)
                grad_kappa += Grad_kappa(it.Q1, it.f, it.A, it.
D, it.DP, it.mu, it.L, kappa, theta) *loss * lr_kappa * (1 - i/n
_epoch)
                grad_theta += Grad_theta(it.Q1, it.f, it.A, it.
D, it.DP, it.mu, it.L, kappa, theta) *loss * lr_theta * (1 - i/n
_epoch)
```

```
            kappa -= grad_kappa/len(dataset)
            theta -= grad_theta/len(dataset)
            # if(i%100 == 0):
            #     print(f"""kappa = {kappa:5f}, theta = {theta:5
f}""")
        print([kappa, theta])
        result.append([kappa0, theta0, kappa, theta, loss])

    result = np.array(result)
    data = {
        'kappa_origin': result[:, 0],
        'theta_origin': result[:, 1],
        'kappa_end': result[:, 2],
        'theta_end': result[:, 3],
    }
    savemat('problem_1.mat', data)

if __name__ == '__main__':
    main()
```

程序五：**Problem_2.py**

```
from __future__ import annotations

import math
import random
from pathlib import Path
from typing import List

import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import torch
from scipy import stats
from sklearn.model_selection import KFold
from torch.utils.data import DataLoader, TensorDataset
from utils import KidneyDataset, KidneyModel, KidneyCriterion,
inference, restan, train_once

from rich import box
from rich.table import Table
from rich.console import Console

console = Console()
```

```python
def plot_results(y_true: np.ndarray, y_pred: np.ndarray, save_d
ir: Path) -> None:
    save_dir.mkdir(parents=True, exist_ok=True)
    residual = y_true - y_pred

    plt.figure(figsize=(6, 5))
    sns.scatterplot(x=y_true, y=y_pred, s=70)
    sns.regplot(x=y_true, y=y_pred, scatter=False, color="red")
    plt.xlabel("True Q2 (mL/min)")
    plt.ylabel("Predicted Q2 (mL/min)")
    plt.title("True vs Predicted")
    plt.tight_layout()
    plt.savefig(save_dir / "scatter.png", dpi=300)
    plt.close()

    plt.figure(figsize=(6, 4))
    sns.histplot(residual, kde=True, bins=15, color="teal")
    plt.xlabel("Residual (mL/min)")
    plt.title("Residual Distribution")
    plt.tight_layout()
    plt.savefig(save_dir / "residual_hist.png", dpi=300)
    plt.close()

    plt.figure(figsize=(6, 4))
    stats.probplot(residual, dist="norm", plot=plt)
    plt.title("Q-Q plot of Residuals")
    plt.tight_layout()
    plt.savefig(save_dir / "qq_plot.png", dpi=300)
    plt.close()

    plt.figure(figsize=(10, 4))
    idx = np.arange(min(50, len(y_true)))
    plt.plot(idx, y_true[idx], label="True", marker="o")
    plt.plot(idx, y_pred[idx], label="Predicted", marker="x")
    plt.xlabel("Sample Index")
    plt.ylabel("Q2 (mL/min)")
    plt.title("True vs Predicted (First 50)")
    plt.legend()
    plt.tight_layout()
    plt.savefig(save_dir / "line_compare.png", dpi=300)
    plt.close()
```

```
def main() -> None:
    device = torch.device("cuda" if torch.cuda.is_available() e
lse "cpu")
    dataset = KidneyDataset(file_path="./Problem_A/data_new.xls
x", sheet_index=1)
    x_all = torch.stack([d["x"] for d in dataset])
    y_all = torch.cat([d["y"] for d in dataset])
    n_samples = len(dataset)
    n_splits = 5

    kf = KFold(n_splits=n_splits, shuffle=True, random_state=4
2)
    preds, targets = [], []

    epochs = 200
    lr = 1e-3
    criterion = KidneyCriterion(1, 0.00, 0.00)

    for fold, (train_idx, val_idx) in enumerate(kf.split(x_al
l), 1):
        console.print(f"\n[bold magenta]===== Fold {fold}/{n_sp
lits} =====[/bold magenta]")
        x_tr, y_tr = x_all[train_idx], y_all[train_idx]
        x_val, y_val = x_all[val_idx], y_all[val_idx]

        train_ds = TensorDataset(x_tr, y_tr)
        train_loader = DataLoader(train_ds, batch_size=2, shuffl
e=True)
        model = KidneyModel().to(device)
        optimizer = torch.optim.Adam(model.parameters(), lr=lr)
        model = train_once(model, train_loader, criterion, optim
izer, device, epochs)

        with torch.no_grad():
            result = model(x_val.to(device))
            pred = restan(result).cpu()
            print(pred)
        preds.extend(pred.tolist())
        targets.extend(restan(y_val).tolist())


    preds = np.array(preds)
    targets = np.array(targets)
```

```
    np.save("preds.npy",preds)
    np.save("targets.npy", targets)

    mse = float(np.mean((preds - targets) ** 2))
    rmse = math.sqrt(mse)
    mae = float(np.mean(np.abs(preds - targets)))
    ss_res = np.sum((preds - targets) ** 2)
    ss_tot = np.sum((targets - np.mean(targets)) ** 2)
    r2 = 1 - ss_res / ss_tot

    table = Table(
        title="5-Fold Cross-Validation Results",
        box=box.SIMPLE_HEAVY,
        show_header=True,
        header_style="bold magenta",
    )
    table.add_column("Metric", justify="left", style="cyan")
    table.add_column("Value", justify="right", style="green")
    table.add_row("Samples", str(n_samples))
    table.add_row("RMSE (mL/min)", f"{rmse:.4f}")
    table.add_row("MAE (mL/min)", f"{mae:.4f}")
    table.add_row("R²", f"{r2:.4f}")
    console.print(table)

    save_dir = Path("results")
    plot_results(targets, preds, save_dir)
    console.print(f"\n[bold blue]Visualizations saved to {save_
dir.resolve()}[/bold blue]")

    full_ds = TensorDataset(x_all, y_all)
    full_loader = DataLoader(full_ds, batch_size=4, shuffle=Tru
e)
    final_model = KidneyModel().to(device)
    optimizer = torch.optim.Adam(final_model.parameters(), lr=l
r)
    final_model = train_once(final_model, full_loader, criterio
n, optimizer, device, epochs)

    save_path = Path("./simulate/kidney_q2_5fold_GeLU.pth")
    # torch.save(final_model.state_dict(), save_path)
    console.print(f"\n[bold blue]Model saved to {save_path.reso
lve()}[/bold blue]")

    sample = random.choice(dataset)
```

```
    x_raw = sample["x"].unsqueeze(0)
    y_true = restan(sample["y"].item())
    y_pred = inference(final_model, x_raw, device)
    console.print("\n[bold yellow]Random inference sample[/bold
 yellow]")
    console.print(f"  True:    [bold green]{y_true:7.2f}[/bold g
reen] mL/min")
    console.print(f"  Predict: [bold red]{y_pred:7.2f}[/bold re
d] mL/min")
    console.print(f"  |Error|: [bold white]{abs(y_pred - y_tru
e):7.2f}[/bold white] mL/min")


if __name__ == "__main__":
    main()
```

程序六：Problem_3.py

```
from __future__ import annotations
from typing import List
import numpy as np
from utils import *
def main():
    model = load_model(weight_path="./kidney_q2_5fold_best.pth", devi
ce='cuda')

    features_1 = [0.05, 0.0002, 0.001, 0.28, 0.0012, 3066, 1.21]
    features_2 = [0.12, 0.0008, 0.004, 0.28, 0.0012, 190, 13.11]

    print(predict_single(model, features_1, 'cuda'), features_1[-1])
    print(predict_single(model, features_2, 'cuda'), features_2[-1])

if __name__ == "__main__":
    main()
```