上海大学
SHANGHAI UNIVERSITY

本科毕业论文（设计）

UNDERGRADUATE THESIS (PROJECT)

题　　目：非线性耦合方程的无网格方法

学　　院：理学院

专　　业：数学与应用数学

学　　号：20123175

学生姓名：俞梦泽

指导教师：李新祥

起讫日期：2024.1.8 - 2024.5.31

姓　　名：俞梦泽　　学号：20123175

论文题目：非线性耦合方程的无网格方法

原 创 性 声 明

本人声明：所呈交的论文是本人在指导教师指导下进行的研究工作。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：俞梦泽　日 期：2024.5.24

本论文使用授权说明

本人完全了解上海大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（保密的论文在解密后应遵守此规定）

签 名：俞梦泽　指导教师签名：李张泽　日期：2024.5.24

摘　要

随着数学学科的不断发展，越来越多的自然规律，可以用数学建模来描述。其中，非线性耦合方程，作为一种偏微分方程，可以用来描述一大类的自然现象。然而，精确求解非线性耦合方程一直是非常困难的。基于网格的有限元法在处理高维高阶的非线性耦合方程时，存在前处理过于复杂，难以处理复杂定义域等问题。因此，本文将介绍一种利用无网格方法简便求解非线性耦合方程近似解的方法。文中使用径向基函数构建近似函数，并利用半隐式差分法处理时间微分项。同时，文章还对捕食者-猎物模型求解，分析了这种方法的精度，以及参数和其改变给计算结果带来的实际精度改变。结果表明此类无网格法运行速度较快，且具备一定精度。但存在对诸如径向基函数参数等系数较敏感的问题。

关键词：非线性耦合方程；无网格法；径向基函数；半隐式差分法。

ABSTRACT

With the continuous development of mathematics, more and more natural laws can be described by using mathematical modeling. Among them, nonlinear coupled equations, as a type of partial differential equation, can be used to describe this large class of natural phenomena. However, accurately solving nonlinear coupled equations has always been very difficult. The grid based finite element method faces problems such as overly complex preprocessing and difficulty in handling complex domains when dealing with high-dimensional and high-order nonlinear coupled equations. Thus，this article will introduce a simple way of using meshless methods to solve approximate solutions of nonlinear coupled equations. In the article, radial basis functions are used to construct approximate functions, and semi-implicit difference methods are used to handle time differential terms. At the same time, the article also analyzes the accuracy of the predator-prey model solution, as well as the actual accuracy changes brought by the parameters and their changes to the calculation results. The results indicate that this type of meshless method runs faster and has a certain degree of accuracy. But there are issues with sensitivity to coefficients such as radial basis function parameters.

Keywords: nonlinear coupled equations; meshless methods; radial basis functions; semi-implicit difference methods.

目　录

1 引言

当下，在科技、科研与生活生产的各个领域都充斥着以非线性耦合方程为表现形式的数学模型。然而非线性耦合方程作为一种较为特殊的微分方程，长期以来，一直存在着精确解求解困难，近似解求解过程繁琐，精度较低的问题。因此，本文将某类特殊的非线性耦合方程组作为主要研究对象，通过无网格法对这类非线性耦合方程组进行近似求解，以找到一种精度比较合适，求解过程尽量简单，又在某些领域普遍适用的求解方法。本文以无网格法中的径向基函数方法拟合出系数待定的近似函数，代入方程。再结合非线性耦合方程的半隐式表示方式，将近似后的新的非线性的耦合方程处理成更方便运算的形式，以此简化方程求解的过程。

1.1 非线性耦合方程

非线性耦合方程作为本文的研究对象，是一种常见的微分方程组。其中非线性的含义是在构成该微分方程组的方程中，存在某个或某几个方程的部分微分项不是一次的。换言之，即在方程组的某些项中，存在关于未知函数或其各阶导数的对数关系、指数关系或三角函数关系等非线性关系。譬如著名的浅水波相关的 Kdv 方程（某种一阶形式）：

$$\frac{\partial u}{\partial t}+\frac{1}{2}\left(\frac{\partial u}{\partial x}\right)^{2}-u=cos(xt)\,. \tag{1.1.1}$$

就是一个非常典型的非线性方程，其中，u 关于 x 的一阶导数呈二次形式，这种非线性性能够很好地描述真实物理量之间的关系[1]。

而耦合指的则是，在方程组的各个方程中，变量之间相互影响，相互产生联系，相互制约，譬如有物理学家将波动方程：

$$\frac{1}{c^{2}(z)}\frac{\partial^{2}p}{\partial t^{2}}-\frac{\partial^{2}p}{\partial z^{2}}=-\left(\frac{\partial}{\partial z}+\frac{1}{c(z)}\frac{\partial}{\partial t}\right)\cdot\left(\frac{\partial}{\partial z}-\frac{1}{c(z)}\frac{\partial}{\partial t}\right)p+\frac{c_{z}}{c^{2}(z)}\frac{\partial p}{\partial t}=0. \tag{1.1.2}$$

写为上行波与下行波：

$$U(t,z)=\frac{1}{2}\left(\frac{1}{c(z)}\frac{\partial}{\partial t}+\frac{\partial}{\partial z}\right)p(t,x). \tag{1.1.3}$$

$$D(t,z)=\frac{1}{2}\left(\frac{1}{c(z)}\frac{\partial}{\partial t}-\frac{\partial}{\partial z}\right)p(t,x). \tag{1.1.4}$$

结合 1.1.2、1.1.3 和 1.1.4 式得到如下方程组：

$$\begin{cases}\left(\frac{1}{c}\frac{\partial}{\partial t}+\frac{\partial}{\partial z}\right)D(t,z)=-\frac{c_z}{2c(z)}(D+U),\\ \left(\frac{1}{c}\frac{\partial}{\partial t}-\frac{\partial}{\partial z}\right)U(t,z)=\frac{c_z}{2c(z)}(D+U).\end{cases} \tag{1.1.5}$$

就是一个非常典型的使用耦合方程组的案例[2]，利用两波互相制约构建方程。

非线性耦合方程组在科研建模和日常生产中都极其重要。因为相当大量的物理模型、生物模型、经济模型乃至生产模型都依赖于非线性耦合方程。譬如，生物学中，由 Holling 提出的某类捕食者与猎物生态模型具有如下形式[3]：

$$\begin{cases}\dfrac{\partial u}{\partial t}-\Delta u=a_1u-p(u)-a_2f(u)\mu, & (X,t)\in\Omega\times(0,T)\\ \dfrac{\partial \mu}{\partial t}-\Delta \mu=-b_1\mu+g(\mu)+b_2f(u)\mu, & (X,t)\in\Omega\times(0,T)\\ u(\boldsymbol{X},0)=g_1(\boldsymbol{X}),\quad \mu(\boldsymbol{X},t)=g_2(\boldsymbol{X}), & X\in\Omega\\ u(\boldsymbol{X},t)=0,\quad \mu(\boldsymbol{X},t)=0. & X\in\partial\Omega\end{cases} \tag{1.1.6}$$

这是一组经典的非线性耦合偏微分方程组（式中拉普拉斯算子Δ仅表示对$\boldsymbol{X}$的二阶全微分）。其中，$\boldsymbol{X}$是对生物数量有影响的生态环境的外在因素的集合，t表示时间。u和μ均是未知的关于$\boldsymbol{X}$和t的函数，分别代表生态环境中某时刻，受环境影响下，某类捕食者和对应猎物的数量。a_1，b_1分别表示捕食者和对应猎物数量的自然增长率。$p(u)$，$g(\mu)$则分别代表了已知的捕食者和对应猎物数量的死亡率。$f(u)$是 Holling 提出的，某个已知的捕食者数量受猎物数量影响的反馈函数，一般情况下，其呈现非线性。由于，在自然界中，捕食者数量和猎物数量息息相关，呈现一种动态平衡，正是一种相互影响，互相制约的非线性耦合状态，因而，上述模型能极大程度上反应真实世界的生态环境。求解这类方程也具有重要的现实意义。

本文也将以这种捕食者与猎物模型的某种变种形式，作为研究的主要对象。为确保其有精确解可以作为近似解的误差参考，文章中对这种方程组进行了一定的简化与变形。但是，本文的方法依然可以被运用于上述的原始形式以及其他类似方程当中，作为这一类生物或物理、金融模型的求解方法。

1.2 无网格法

无网格法，作为有限元法（基于网格）的一种延伸，在当下求解微分方程等领域是最重要的求解方法之一。无网格法是指，在数值计算的方法中，不使用网格，而是直接使用坐标点作为插值函数构造的依据。不同于有限元法，无网格法的近似函数是建立在一系列离散点上的，不需要借助于网格，克服了有限元法对网格的依赖性，在涉及网格畸变、网格移动等问题中显示出明显的优势，同时无网格法的前处理过程也比有限元法更为简单[4]。

1.2.1 无网格法发展历史

对无网格法的研究可以追溯到 20 世纪 70 年代对非规则网格有限差分法的研究[5][6]。但由于当时，有限元法刚刚收获巨大成功，名声大噪，成为主流，这类研究，并没有吸引太多学者的注意。无网格法，首次进入大众视野是在 1977 年，Lucyt 提出了光滑质点流体动力学方法（Smoothed Particle Hydrodynamics，简称 SPH）[7][8]。并且，在后续的十数年中，数学家及物理学家们，根据 Lucyt 的研究，不断修正，克服其不稳定性，提高其精度，将其积极运用于动能、材料等领域，使得无网格法，逐渐成为数值计算中一个具有一定可行性，值得研究的新方向。

无网格法的真正火热，则是在 1992 年。该年，Nayroles 将最小二乘法作为近似方法加入了无网格法的研究，提出了散射元法（Diffuse Element Method，简称 DEM）。尽管会耗费更多算力，这一新方法，却能够显著提高 SPH 方法的协调性与稳定性。并且由于，最小二乘近似构造出的函数，可以拥有C^1连续性。数学家与物理学家们还考虑将这一方法与边界元法和有限元法结合，一时间解决了大量积分问题。这也是无网格法第一次显现出，其较之于有限元法的巨大优势：不需要频繁地重复画网格。在此之后，数学家们还将这种最小二乘构造限制在局部域中，使其完全由离散点控制，彻底摆脱了背景网格的限制，使无网格法得以独立于有限元法，成为全新的数值计算方法。

但是，早期的最小二乘法也具有其局限性，由于其集中在摆脱网格限制，提高计算效率上，其全局解的精度无法得到有效的保证。直到张雄等数学家提出了加权最小二乘方法，才终于使最小二乘近似成为一种成熟的运算手段[9]。

早期的无网格法，除了精度问题，对于处理边界条件也存在一定局限性。由于近似函数本身不具有插值特性，使得处理边界条件时，需要引入其他方法辅助。Belytschko、Atluri、Mukherjee、Liu 和张雄等都对本质边界条件的处理进行了深入研究，分别相继提出了直接配点法、拉格朗日乘子法、修正变分原理、罚函数法、与有限元耦合法、容许近似法、达朗伯原理、修正配点法和位移约束方程法等方法[4]。

受到最小二乘近似的成功的启发，数学家们又相继提出了多种函数近似方法，譬如，单位分解近似，核函数近似，以及本文使用的径向基函数。他们都有各自的优点与不足，可以处理各种实际运用场景，共同构建了无网格法在近似函数时的基础。

1.2.2 径向基函数

径向基函数是一种较新的函数近似方法，经过演化，成为无网格法中一种非常简单高效的近似方法。

如前文，在最初的无网格研究期间，最流行的方法是最小二乘方法。其近似形式如下[10]：

$$\hat{u}(\boldsymbol{x}) = \sum_{i=1}^{M} p_i(\boldsymbol{x}) \cdot a_i(\boldsymbol{x}) . \tag{1.2.1}$$

其中$p_i(\boldsymbol{x})$为基函数，$a_i(\boldsymbol{x})$为待解的基函数对应的系数。由于，基函数与系数均为关于$\boldsymbol{x}$的函数，因此被称为最小二乘法，但也正因如此，最小二乘法的近似与求解过程会相对比较繁琐，对于算力分配的要求较高。

后来，便有学者基于有限差分法的部分理论，提出了单位分解近似，其方法如下：

先定义一些覆盖求解区域$\Omega \in R^n (n = 1,2,3)$的子域$\Omega_I \coloneqq \{\boldsymbol{y} \in R^n : \|\boldsymbol{x}_I - \boldsymbol{y}\|_{R^n} < h_I\}$，即对每个节点$\boldsymbol{x}_I$，以其为中心，半径为$h_I$的域。随后在每个子域$\Omega_I$上定义一个满足单位分解条件且仅在该子域上非零的函数$\Phi(\boldsymbol{x})$令：$\sum_I \Phi(\boldsymbol{x}) = 1$，于是有了近似函数[11]：

$$\hat{u}(\boldsymbol{x}) = \sum_I [\Phi(\boldsymbol{x}) \cdot u_I + \sum_{i=1}^{m} b_{iI} \cdot \Phi(\boldsymbol{x}) \cdot q_i(\boldsymbol{x})] . \tag{1.2.2}$$

其中$q_i(\boldsymbol{x})$是基函数，u_I和b_{iI}是待求的系数。这类方法的优势是节点所对应的

未知数可以不受自由度数限制，具有很强的自适应性。但其缺点也很明显，在高维环境下，这种方法并不适用。

同一时期的核函数法也具有类似的优缺点。核函数法利用积分变化取近似函数[12]：

$$\hat{u}(\boldsymbol{x}) = \int w(\boldsymbol{x}-\boldsymbol{y},h)u(\boldsymbol{y})\,d\Omega_y\,. \tag{1.2.3}$$

其中，$w(\boldsymbol{x}-\boldsymbol{y},h)$称为核函数。因其具有紧支性，所以可以大大提高方法的精度和灵活性，然而其与单位分解法一样，并没有解决高维问题域难以处理的问题[13]。

直到径向基函数的出现，提供了一种十分简便且能保证一定精度的近似形式[14]。径向基函数先找到一种基函数$\boldsymbol{\Phi}(\|\boldsymbol{x}-\boldsymbol{x}_i\|)$，其自变量为点到节点的欧氏距离。随后用这组基函数的线性组合，来近似函数[15]：

$$\hat{u}(\boldsymbol{x}) = \sum_{i=1}^{N} v_i \cdot \boldsymbol{\Phi}(\|\boldsymbol{x}-\boldsymbol{x}_i\|). \tag{1.2.4}$$

从而，避免了高维域难处理的困境。虽然它的精度与灵活性不如前两种方法，但其牺牲了少量精度，换取了便捷性，节省了算力，使其在大型应用场景中，十分适用。本文也将采用径向基函数作为近似方法进行推导和数值实验。

1.2.3 半隐式差分法

在确定近似函数之后，需要进行的就是求解近似函数中的待定系数，如式1.2.4中的$v_i(i=1,2,\dots,N)$。在这一过程中，如果求解方程或方程组是线性的，则可以用简单的系数矩阵求逆进行求解。然而当方程中出现非线性项时，则需要一些简单的处理，使运算过程简便。

其中，当方程组中含有对时间变量的微分时，对时间进行差分是一种非常经典有效的方式。因为时间这一物理量具有实际意义，且适合分段观察。所以，我们可以用$(Y(t+\Delta t)-Y(t))/\Delta t$来替代$\partial y/\partial t$。随后递推得出不同时间段的$Y(t)$。而在古典时间差分方式中对于$\partial y/\partial t$以外的剩余项的表达，一般有显式和隐式两种处理方式。其中，显示差分格式如下：

$$Y(t+\Delta t) = F\big(Y(t)\big). \tag{1.2.5}$$

隐示差分格式如下：

$$G\big(Y(t),Y(t+\Delta t)\big)=0. \tag{1.2.6}$$

不难看出，显示格式在递推过程中，保留了所有剩余项，令他们不随Δt变化。这使得，整体方程，十分便于求解。但是显示方法也要求Δt足够小，否则会损失稳定性与收敛性，使得求解的误差较大。而隐式格式则对所有的剩余项作了差分处理，取其在经过Δt迭代前后的均值。这使得隐式格式具有很强的稳定性与收敛性，但在求解时，却往往需要联立方程，较为复杂。

于是，在 1961 年，气象学家曾存志想到将显示格式与隐式格式结合，同时运用，保留各自的优点，提出了半隐式差分法[16]。具体地说，当我们遇到线性项时，我们可以采用隐式表示。因为线性项可以经过简单的移项处理将Δt迭代前后的各项分离，保持方程可显式求解的特征，同时提高解的稳定性和收敛性。而面对非线性项时，则直接使用显式表示，避免求解困难的情况。因而，半隐式差分法作为一种既便于求解，又具有较好的稳定性，收敛性的方法，在气象学，热学等多种方向都被广泛使用，认可。

2 无网格法求解捕食者-猎物方程

本章节将详细讲述基于径向基函数与半隐式差分法的无网格法的具体操作过程。其中，主要包含了如何创建近似函数，以及如何求解该近似函数，以达到求解效果。

在正式讲述方法之前，本文基于式 1.1.6，写出如下方程组：

$$\begin{cases} \dfrac{\partial u}{\partial t}=\Delta u-a_1 f(u)\mu+F_1(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ \dfrac{\partial \mu}{\partial t}=\Delta \mu-a_2 f(u)\mu+F_2(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ u(\boldsymbol{X},0)=g_1(\boldsymbol{X}),\quad \mu(\boldsymbol{X},0)=g_2(\boldsymbol{X}), & \boldsymbol{X}\in\Omega \\ u(\boldsymbol{X},t)=0,\quad \mu(\boldsymbol{X},t)=0. & (\boldsymbol{X},t)\in\partial\Omega\times(0,T) \end{cases} \tag{2.0.1}$$

其中，F_1，F_2是两个已知的关于$\boldsymbol{X}$和$\boldsymbol{t}$的函数，用以确保非线性耦合方程组有精确解，以便后续进行误差分析。$f(u)$为 Holling 提出的已知函数，具有非线性性。本文将上述方程组确定为主要研究对象。通过无网格法求解该方程组，并将近似函数与精确解比较，分析无网格方法的优劣。

2.1 半隐式差分法预处理方程

如前文所示，为后续求解便捷，需要使用半隐式差分法将原方程对时间差分。具体到上述方程 2.0.1，我们对线性部分Δu，$\Delta\mu$，μ以及$F_1(\mathbf{X},t)$，$F_2(\mathbf{X},t)$进行隐式表示。而对非线性项$f(u)$作显示表示。得到如下方程组：

$$\begin{cases}\dfrac{u^{n+1}-u^n}{\tau}=\dfrac{\Delta u^n+\Delta u^{n+1}}{2}-a_1 f(u^n)\dfrac{\mu^n+\mu^{n+1}}{2}+\dfrac{{F_1}^n+{F_1}^{n+1}}{2}, & \boldsymbol{X}\in\Omega\\ \dfrac{\mu^{n+1}-\mu^n}{\tau}=\dfrac{\Delta\mu^n+\Delta\mu^{n+1}}{2}-a_1 f(u^n)\dfrac{\mu^n+\mu^{n+1}}{2}+\dfrac{{F_2}^n+{F_2}^{n+1}}{2}, & \boldsymbol{X}\in\Omega\\ u^0(\boldsymbol{X})=g_1(\boldsymbol{X}),\quad \mu^0(\boldsymbol{X})=g_2(\boldsymbol{X}), & \boldsymbol{X}\in\Omega\\ u^{n+1}(\boldsymbol{X})=0,\quad \mu^{n+1}(\boldsymbol{X})=0. & \boldsymbol{X}\in\partial\Omega\end{cases}\tag{2.1.1}$$

上式中，τ表示每次递进时的步长，u^n表示当第n次递推时，即$t=n\cdot\tau$时的函数u。所以，u^{n}是一个仅关于变量$\boldsymbol{X}$的函数。同理，$u^{\mathrm{n+1}}$则表示当第$n+1$次递推时，即$t=(n+1)\cdot\tau$时的函数u，也是一个仅关于变量$\boldsymbol{X}$的函数。上式中的μ^{n}，$\mu^{\mathrm{n+1}}$，Δu^{n}，$\Delta u^{\mathrm{n+1}}$，$\Delta\mu^{\mathrm{n}}$，$\Delta\mu^{\mathrm{n+1}}$，${F_1}^{\mathrm{n}}$，${F_1}^{\mathrm{n+1}}$，${F_2}^{\mathrm{n}}$，${F_2}^{\mathrm{n+1}}$均采用这种表示方法，并且，他们都变成了仅关于变量$\boldsymbol{X}$的函数。同时，因为原方程组具有初值条件：$u(\boldsymbol{X},0)=\mathrm{g}_1(\boldsymbol{X})$，$\mu(\boldsymbol{X},0)=\mathrm{g}_2(\boldsymbol{X})$，可以从中得到$t=0$时的递推起点$u^0(\boldsymbol{X})$和$\mu^0(\boldsymbol{X})$。又由于$f$是已知函数，且$u^{\mathrm{n}}$在递推过程中也是确定的。因此，可以将$f(u^n)$视为一个常系数。于是，该方程组便被转化为了一种非常易于递推求解的线性耦合方程组[17]。

在求解过程中，可以将$f(u^n)$视为常系数，${F_1}^n$和${F_1}^{n+1}$视为常数项，随后将其余项中的第$n+1$次递推得到的项放在等式左侧，第n次递推得到的项放在等式右侧，得到如下方程组：

$$\begin{cases}2u^{n+1}-\tau\cdot\Delta u^{n+1}+\tau\cdot f(u^n)\cdot\mu^{n+1}=2u^n+\tau\cdot\Delta u^n\\ -\tau\cdot f(u^n)\cdot\mu^n+\tau\cdot\left({F_1}^n+{F_1}^{n+1}\right), & \boldsymbol{X}\in\Omega\\ \left(2+\tau\cdot f(u^n)\right)\cdot\mu^{n+1}-\tau\cdot\Delta\mu^{n+1}=\left(2-\tau\cdot f(u^n)\right)\cdot\mu^n\\ +\tau\cdot\Delta\mu^n+\tau\cdot\left({F_2}^n+{F_2}^{n+1}\right), & \boldsymbol{X}\in\Omega\\ u^0(\boldsymbol{X})=g_1(\boldsymbol{X}),\quad \mu^0(\boldsymbol{X})=g_2(\boldsymbol{X}), & \boldsymbol{X}\in\Omega\\ u^{n+1}(\boldsymbol{X})=0,\quad \mu^{n+1}(\boldsymbol{X})=0. & \boldsymbol{X}\in\partial\Omega\end{cases}\tag{2.1.2}$$

至此，本文将原方程关于时间差分的内容就已完成，剩余部分将叙述如何将近似函数带入方程组 2.1.1 并求解。

2.2 径向基函数

在上一节中，u^n与μ^n都已经被处理成了仅关于变量$\boldsymbol{X}$的函数，式 2.0.1 也被转化成了可递推的线性耦合形式。因此，在本章剩余的内容中，将展示如何对节点构造与求解径向基函数，从而达到求取每个t时刻下，对应的式 2.0.1 的解，即u^n与μ^n。

2.2.1 构造径向基函数

如前文所示，径向基函数属于无网格法，是一种基于节点构造函数的方法。所以，使用径向基函数前需要先确定节点。不过，由于无网格法的特性，节点的选取方式可以相对自由，甚至可以不受原方程定义域的限制，考虑一个更大的包含原定义域的域Ω'[18]。因此，可以有许多不同的节点选取方法，部分选取方法在大型问题和定义域不规则的问题中具有很大的优势。在本文中，假定定义域Ω是规则的。只考虑最简单的，在定义域Ω上均匀的取点。

需要选取如下的节点：

在规则区域Ω上均匀地取N个节点$\boldsymbol{Y}$，称为虚拟点，是构造近似函数的重要组成部分。

随后，令$\boldsymbol{X}=\boldsymbol{Y}$， 称为测试点，用于求解待定系数。其中，由于初始条件限制，需要根据$\mathbf{X}$内的点是否在边界上，分为边界点$\boldsymbol{X}_B(\boldsymbol{X}_B\in\partial\Omega)$，和内部点$\boldsymbol{X}_I$ $(\boldsymbol{X}_I=\boldsymbol{X}/\boldsymbol{X}_B)$。令边界点$\boldsymbol{X}_B$和内部点$\boldsymbol{X}_I$的数量分别为$N_B$和$N_I$ 。

选取完节点后，需要选取近似时使用的径向基函数，一般采用带有常系数的指数函数或二次函数。在本文中选取$\varphi(r)=\sqrt{c^2+r^2}$作为径向基函数，其中$c$是常系数，$r$代表某个测试点和某个虚拟点之间的欧氏距离[19]。

即在$\mathbf{X}$一维情况下：

$$\varphi\left(r_{ij}\right)=\sqrt{c^2+{r_{ij}}^2}=\sqrt{c^2+(\boldsymbol{X}_i-\boldsymbol{Y}_j)^2}. \tag{2.2.1}$$

由此可推出$\varphi\left(r_{ij}\right)$的一阶与二阶微分形式：

$$\nabla\varphi\left(r_{ij}\right)=\frac{\partial\sqrt{c^2+{r_{ij}}^2}}{\partial\boldsymbol{X}_i}=\left(\boldsymbol{X}_i-\boldsymbol{Y}_j\right)\left(c^2+(\boldsymbol{X}_i-\boldsymbol{Y}_j)^2\right)^{-\frac{1}{2}}. \tag{2.2.2}$$

$$\Delta\varphi(r_{ij}) = \left(c^2 + (\boldsymbol{X}_i - \boldsymbol{Y}_j)^2\right)^{-\frac{1}{2}} - \left(\boldsymbol{X}_i - \boldsymbol{Y}_j\right)^2\left(c^2 + (\boldsymbol{X}_i - \boldsymbol{Y}_j)^2\right)^{-\frac{3}{2}}. \tag{2.2.3}$$

同理，在$\boldsymbol{X}$二维情况下：

$$\varphi(r_{ij}) = \sqrt{c^2 + {r_{ij}}^2} = \sqrt{c^2 + (\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j})^2 + (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})^2}. \tag{2.2.4}$$

$$\nabla\varphi(r_{ij}) = \left(\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j}\right)\left(c^2 + (\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j})^2 + (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})^2\right)^{-\frac{1}{2}}$$

$$+\left(\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j}\right)\left(c^2 + (\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j})^2 + (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})^2\right)^{-\frac{1}{2}}. \tag{2.2.5}$$

$$\Delta\varphi(r_{ij}) = 2 \cdot \left(c^2 + (\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j})^2 + (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})^2\right)^{-\frac{1}{2}}$$

$$-((\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j})^2 + (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})^2 + \boldsymbol{2} \cdot (\boldsymbol{X}_{1i} - \boldsymbol{Y}_{1j}) \cdot (\boldsymbol{X}_{2i} - \boldsymbol{Y}_{2j})$$

$$\cdot \left(c^2 + (\boldsymbol{X}_i - \boldsymbol{Y}_j)^2\right)^{-\frac{3}{2}}. \tag{2.2.6}$$

综上，如前文中式 1.2.4，近似函数$\hat{u}$与$\hat{\mu}$写作如下形式：

$$\hat{u}^n(\boldsymbol{X}_i) = \sum_{j=1}^{N} {v_j}^n \cdot \varphi(r_{ij}) = \sum_{j=1}^{N} {v_j}^n \cdot \varphi(\|\boldsymbol{X}_i - \boldsymbol{Y}_j\|), \tag{2.2.7}$$

$$\hat{\mu}^n(\boldsymbol{X}_i) = \sum_{j=N+1}^{2N} {v_j}^n \cdot \varphi(r_{ij}) = \sum_{j=N+1}^{2N} {v_j}^n \cdot \varphi(\|\boldsymbol{X}_i - \boldsymbol{Y}_j\|). \tag{2.2.8}$$

显然，只需求解出第n次迭代的系数${v_j}^{\mathrm{n}}$，即可得到在$t = n \cdot \tau$时的函数u^n的值。

2.2.2 求解径向基函数

本节主要讨论如何对上一节中的近似函数求解：

首先考虑式 2.1.2 中的t初值问题，即：

$$u^0(\boldsymbol{X}) = g_1(\boldsymbol{X}), \quad \mu^0(\boldsymbol{X}) = g_2(\boldsymbol{X}). \quad \boldsymbol{X} \in \Omega \tag{2.2.9}$$

将式 2.2.7 和式 2.2.8 分别代入该方程组，用$\varphi_j(\boldsymbol{X}_i)$表示 $\varphi(r_{ij})$得到：

$$\begin{cases} \hat{u}^0(\boldsymbol{X}_i) = \sum_{j=1}^{N} v_j{}^0 \cdot \varphi_j(\boldsymbol{X}_i) = g_1(\boldsymbol{X}_i), & \boldsymbol{X}_i \in \boldsymbol{X} \\ \hat{\mu}^0(\boldsymbol{X}_i) = \sum_{j=N+1}^{2N} v_j{}^0 \cdot \varphi_j(\boldsymbol{X}_i) = g_2(\boldsymbol{X}_i). & \boldsymbol{X}_i \in \boldsymbol{X} \end{cases} \tag{2.2.10}$$

由于$\boldsymbol{X}$中有 *N* 个元素，所以共可以写出 *N* 个上述方程组，即 *2N* 个方程，刚好可以求解元素数量为 *2N* 的待定系数v^0。具体可以用如下矩阵求解：

$$A^0 \cdot v^0{}_{2N\times 1} = b^0. \tag{2.2.11}$$

其中：

$$A^0 = \begin{bmatrix} \varphi_j(\boldsymbol{X})_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & \varphi_j(\boldsymbol{X})_{N\times N} \end{bmatrix}, \qquad b^0 = [g_1(\boldsymbol{X}), g_2(\boldsymbol{X})]^T. \tag{2.2.12}$$

于是可以求得$v^0 = A^{0^{-1}} \cdot b^0$.

有了初始值v^0之后，可以进行迭代，将式 2.2.7 与式 2.2.8 代入差分完成的式 2.1.2 后，可得如下方程：

$$\begin{cases} \sum_{j=1}^{N} v_j{}^{n+1} \cdot (2 \cdot \varphi_j(\boldsymbol{X}_i) - \tau \cdot \Delta\varphi_j(\boldsymbol{X}_i)) + \sum_{j=N+1}^{2N} v_j{}^{n+1} \cdot \tau \cdot f(u^n) \cdot \varphi_j(\boldsymbol{X}_i) \\ \quad = \sum_{j=1}^{N} v_j{}^n \cdot (2 \cdot \varphi_j(\boldsymbol{X}_i) + \tau \cdot \Delta\varphi_j(\boldsymbol{X}_i)) - \sum_{j=N+1}^{2N} v_j{}^n \cdot \tau \cdot f(u^n) \cdot \varphi_j(\boldsymbol{X}_i) \\ \quad +\tau \cdot \left(F_1{}^n + F_1{}^{n+1}\right), & \boldsymbol{X}_i \in \boldsymbol{X_I} \\ \sum_{j=N+1}^{2N} v_j{}^{n+1} \cdot (2 + \tau \cdot f(u^n) \cdot \varphi_j(\boldsymbol{X}_i) - \tau \cdot \Delta\varphi_j(\boldsymbol{X}_i)) \\ \quad = \sum_{j=N+1}^{2N} v_j{}^n \cdot \left(2 - \tau \cdot f(u^n) \cdot \varphi_j(\boldsymbol{X}_i) + \tau \cdot \Delta\varphi_j(\boldsymbol{X}_i)\right) \\ \quad +\tau \cdot \left(F_2{}^n + F_2{}^{n+1}\right), & \boldsymbol{X}_i \in \boldsymbol{X_I} \\ \sum_{j=1}^{N} v_j{}^{n+1} \cdot \varphi_j(\boldsymbol{X}_b) = 0, \quad \sum_{j=N+1}^{2N} v_j{}^{n+1} \cdot \varphi_j(\boldsymbol{X}_b) = 0. & \boldsymbol{X}_b \in \boldsymbol{X_B} \end{cases} \tag{2.2.13}$$

注意上式中对测试点中的内部点与边界点进行了不同的处理。将上式转化为矩阵相乘形式，得如下方程：

$$A^n \cdot v^{n+1}{}_{2N\times 1} = B^n \cdot v^n{}_{2N\times 1} + b^n. \tag{2.2.14}$$

其中：

$$A^n=\begin{bmatrix} A_{b_{N_b\times N}} & 0_{N_b\times N} \\ 0_{N_b\times N} & A_{b_{N_b\times N}} \\ A_{1_{N_i\times N}} & A_{2_{N_i\times N}} \\ A_{3_{N_i\times N}} & A_{4_{N_i\times N}} \end{bmatrix},\qquad B^n=\begin{bmatrix} 0_{N_b\times N} & 0_{N_b\times N} \\ 0_{N_b\times N} & 0_{N_b\times N} \\ B_{1_{N_i\times N}} & B_{2_{N_i\times N}} \\ B_{3_{N_i\times N}} & B_{4_{N_i\times N}} \end{bmatrix},$$

$$b^n=\left[0_{2N_b\times 1},\tau\cdot\left({F_1}^n(\boldsymbol{X_I})+{F_1}^{n+1}(\boldsymbol{X_I})\right),\tau\cdot\left({F_2}^n(\boldsymbol{X_I})+{F_2}^{n+1}(\boldsymbol{X_I})\right)\right]^T. \quad (2.2.15)$$

$$A_{b_{N_b\times N}}=\varphi_j(\boldsymbol{X}_b),$$

$$A_{1_{N_i\times N}}=2\cdot\varphi_j(\boldsymbol{X}_i)-\tau\cdot\Delta\varphi_j(\boldsymbol{X}_i),\qquad A_{2_{N_i\times N}}=\tau\cdot f(u^n)\cdot\varphi_j(\boldsymbol{X}_i),$$

$$A_{3_{N_i\times N}}=0_{N_i\times N},\qquad A_{4_{N_i\times N}}=\left(2+\tau\cdot f(u^n)\right)\cdot\varphi_j(\boldsymbol{X}_i)-\tau\cdot\Delta\varphi_j(\boldsymbol{X}_i),$$

$$B_{1_{N_i\times N}}=2\cdot\varphi_j(\boldsymbol{X}_i)+\tau\cdot\Delta\varphi_j(\boldsymbol{X}_i),\qquad B_{2_{N_i\times N}}=-\tau\cdot f(u^n)\cdot\varphi_j(\boldsymbol{X}_i),$$

$$B_{3_{N_i\times N}}=0_{N_i\times N},\qquad B_{4_{N_i\times N}}=\left(2-\tau\cdot f(u^n)\right)\cdot\varphi_j(\boldsymbol{X}_i)+\tau\cdot\Delta\varphi_j(\boldsymbol{X}_i). \quad (2.2.16)$$

因此，结合初值条件，即可从每一步v^{n}，推导出下一步$v^{\mathrm{n+1}}$，从而，得到，每一时刻近似的$\hat{u}^n(\boldsymbol{X}_i)=\sum_{j=1}^{N}{v_j}^n\cdot\varphi_j(\boldsymbol{X}_i)$.

2.2.3 近似结果分析

得到近似函数$\hat{u}^n(\boldsymbol{X})$后，需要对其精度进行分析。因此本节将定义误差的计算公式，并猜测在近似过程中，有哪些参数的改变会对精度产生影响，用以在下一章的数值实验中进行验证。

首先,需定义每个$\hat{u}^n(\boldsymbol{X})$的误差，取其在定义域内各点与真实值的插值的最大值，即：

$$E_u^n=\|\hat{u}^n(\boldsymbol{x})-u^n(\boldsymbol{x})\|_\infty.\qquad \boldsymbol{x}\in\boldsymbol{X} \quad (2.2.17)$$

然后，取所有E^n中的最大值作为整个近似方法对$\hat{u}(\boldsymbol{X})$的误差：

$$E_u=\|E_u^n\|_\infty.\qquad n=0,1,2,\dots,\left\lfloor\frac{T}{\tau}\right\rfloor \quad (2.2.18)$$

同理，得到每个$\hat{\mu}^n(\boldsymbol{X})$的误差和整个近似方法对$\hat{\mu}(\boldsymbol{X})$的误差：

$$E_u^n=\|\hat{\mu}^n(\boldsymbol{x})-\mu^n(\boldsymbol{x})\|_\infty,\qquad \boldsymbol{x}\in\boldsymbol{X} \quad (2.2.19)$$

$$E_\mu=\left\|E_\mu^n\right\|_\infty.\qquad n=0,1,2,\dots,\left\lfloor\frac{T}{\tau}\right\rfloor \quad (2.2.20)$$

以上是误差分析思路，当E_u和E_μ越小时，表明近似函数越接近精确解，也就

表明方法的精度越高。

根据前文的内容，本文猜测，以下变量有可能会影响近似函数的精度：

径向基函数φ中的参数c，按时间差分过程中的步长τ以及构造近似函数时的虚拟点数量N。因此，在下一章的数值实验中，将对上述各变量进行不同取值，比较精度。

3 数值实验

3.1 一维不含拉普拉斯项的非线性耦合方程组

首先，从最简单的非线性耦合方程组开始。

考虑如下不含拉普拉斯项偏微分方程组：

$$\begin{cases} \dfrac{\partial u}{\partial t} = u - a_1 f(u)\mu + F_1(\boldsymbol{X},t), & (\boldsymbol{X},t) \in \Omega \times (0,T) \\ \dfrac{\partial \mu}{\partial t} = \mu - a_2 f(u)\mu + F_2(\boldsymbol{X},t), & (\boldsymbol{X},t) \in \Omega \times (0,T) \\ u(\boldsymbol{X},0) = g_1(\boldsymbol{X}), \quad \mu(\boldsymbol{X},0) = g_2(\boldsymbol{X}), & \boldsymbol{X} \in \Omega \\ u(\boldsymbol{X},t) = 0, \quad \mu(\boldsymbol{X},t) = 0. & \boldsymbol{X} \in \partial\Omega \end{cases} \tag{3.1.1}$$

其中，$\boldsymbol{X}$是一维的，$a_1 = a_2 = 1$，$f(u) = \frac{u}{u+1}$，$g_1(\boldsymbol{X}) = g_2(\boldsymbol{X}) = sin\,(\pi \boldsymbol{X})$，$\Omega = (0,1)$，$T = 1$。则$\partial\Omega = [0,1]$。

令$F_1(\boldsymbol{X},t) = F_2(\boldsymbol{X},t) = \frac{e^{2t} \cdot sin^2(\pi x)}{e^t \cdot sin(\pi x)+1}$，则对应的精确解：

$$u(\boldsymbol{X},t) = \mu(\boldsymbol{X},t) = e^{\mathrm{t}} \cdot sin(\pi x).$$

随后，按式 2.2.1 构造径向基函数，取$c = 1$，即$\varphi(r) = \sqrt{1+r^2}$。令$\tau = 0.01$，于是，$n = 0,1,\ldots,100$。差分后的方程组与式 2.1.2 稍有不同，用矩阵形式表示如下：

$$A^0 \cdot {v^0}_{2N\times 1} = b^0. \tag{3.1.2}$$

其中：

$$A^0 = \begin{bmatrix} \varphi_j(\boldsymbol{X})_{N\times N} & 0_{N\times N} \\ 0_{N\times N} & \varphi_j(\boldsymbol{X})_{N\times N} \end{bmatrix}, \qquad b^0 = [g_1(\boldsymbol{X}), g_2(\boldsymbol{X})\,]^T,$$

$$A^n \cdot {v^{n+1}}_{2N\times 1} = B^n \cdot {v^n}_{2N\times 1} + b^n. \tag{3.1.3}$$

其中：

$$A^n=\begin{bmatrix} A_{b_{N_b\times N}} & 0_{N_b\times N} \\ 0_{N_b\times N} & A_{b_{N_b\times N}} \\ A_{1_{N_i\times N}} & A_{2_{N_i\times N}} \\ A_{3_{N_i\times N}} & A_{4_{N_i\times N}} \end{bmatrix},\qquad B^n=\begin{bmatrix} 0_{N_b\times N} & 0_{N_b\times N} \\ 0_{N_b\times N} & 0_{N_b\times N} \\ B_{1_{N_i\times N}} & B_{2_{N_i\times N}} \\ B_{3_{N_i\times N}} & B_{4_{N_i\times N}} \end{bmatrix},$$

$$b^n=\left[0_{2N_b\times 1},\tau\cdot\left({F_1}^n(\boldsymbol{X_I})+{F_1}^{n+1}(\boldsymbol{X_I})\right),\tau\cdot\left({F_2}^n(\boldsymbol{X_I})+{F_2}^{n+1}(\boldsymbol{X_I})\right)\right]^T. \quad (3.1.4)$$

$$A_{b_{N_b\times N}}=\varphi_j(\boldsymbol{X}_b),$$

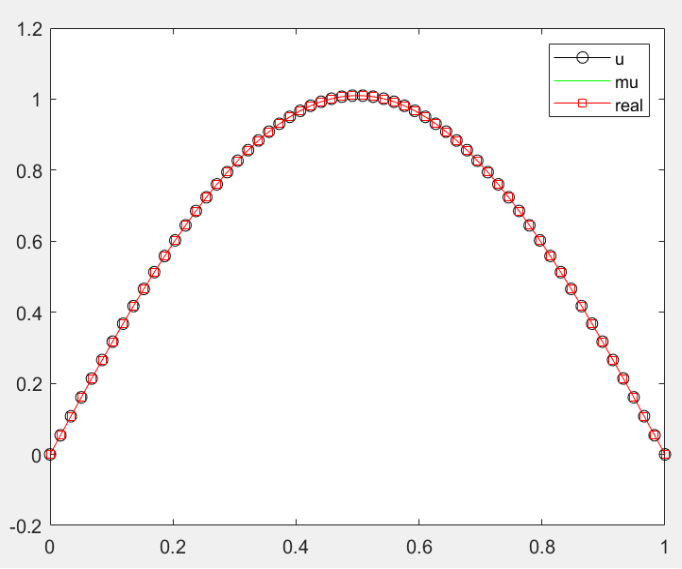
$$A_{1_{N_i\times N}}=(2-\tau)\cdot\varphi_j(\boldsymbol{X}_i),\qquad A_{2_{N_i\times N}}=\tau\cdot f(u^n)\cdot\varphi_j(\boldsymbol{X}_i),$$

$$A_{3_{N_i\times N}}=0_{N_i\times N},\qquad A_{4_{N_i\times N}}=(2+\tau\cdot f(u^n)-\tau)\cdot\varphi_j(\boldsymbol{X}_i),$$

$$B_{1_{N_i\times N}}=(2+\tau)\cdot\varphi_j(\boldsymbol{X}_i),\qquad B_{2_{N_i\times N}}=-\tau\cdot f(u^n)\cdot\varphi_j(\boldsymbol{X}_i),$$

$$B_{3_{N_i\times N}}=0_{N_i\times N},\qquad B_{4_{N_i\times N}}=(2-\tau\cdot f(u^n)+\tau)\cdot\varphi_j(\boldsymbol{X}_i). \quad (3.1.5)$$

对于上述方法，在$\Omega=(0,1)$内均匀地取$N=60$个虚拟点，即$N_i=58$，$N_b=2$，开始进行无网格法处理，得到如下函数图像：



x

图 3.1.1 t=0.01 时，近似函数与精确值的图像

首先，由图 3.1.1 给出第一次迭代，即$t=\tau=0.01$时，$\hat{u}^1(x)$，$\hat{\mu}^1(x)$，$u^1(x)$，$\mu^1(x)$关于x的函数图像。（图中曲线 u 表示$\hat{u}^1(x)$，曲线 mu 表示$\hat{\mu}^1(x)$，由于$u^1(x)=\mu^1(x)$，所以在图中用同一条曲线 real 表示）

其中，最大的误差$E_u^1=8.708\times10^{-6}$，$E_\mu^1=1.219\times10^{-5}$。

随后，由图 3.1.2 给出最后一次迭代时，即$t=T=1$，时，$\hat{u}^{100}(x)$，$\hat{\mu}^{100}(x)$，

$u^{100}(x)$，$\mu^{100}(x)$关于x的函数图像。（与上相同，图中曲线 u 表示$\hat{u}^{100}(x)$，曲线 mu 表示$\hat{\mu}^{100}(x)$，由于$u^{100}(x) = \mu^{100}(x)$，所以在图中用同一条曲线 real 表示）。

其中，最大的误差$E_u^{100} = 2.521 \times 10^{-3}$，$E_\mu^{100} = 2.227 \times 10^{-3}$。

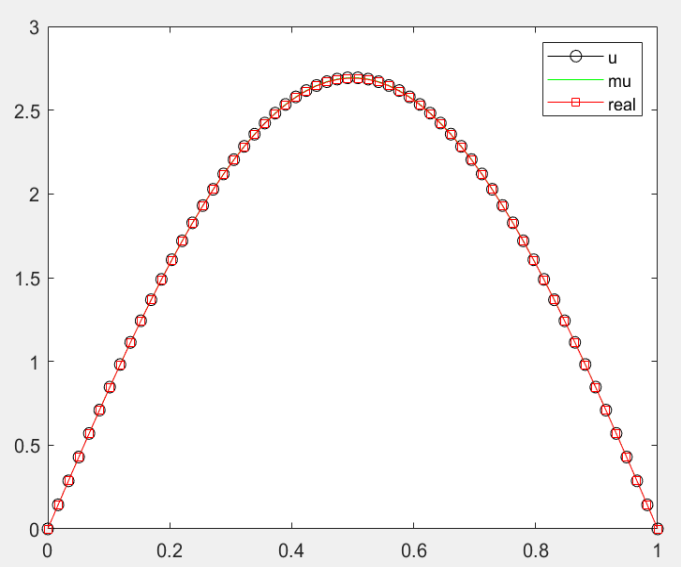


图 3.1.2 t=1 时，近似函数与精确值的图像

由图3.1.3给出$t = 0, 0.25, 0.5, 0.75, 1$时，$\hat{u}^n(x)$与$u^n(x)$的图像。其中图3.1.3(a)为各时间下，$\hat{u}^n(x)$关于$x$的函数图像。图 3.1.3(b) 为各时间下，$u^n(x)$关于$x$的函数图像。

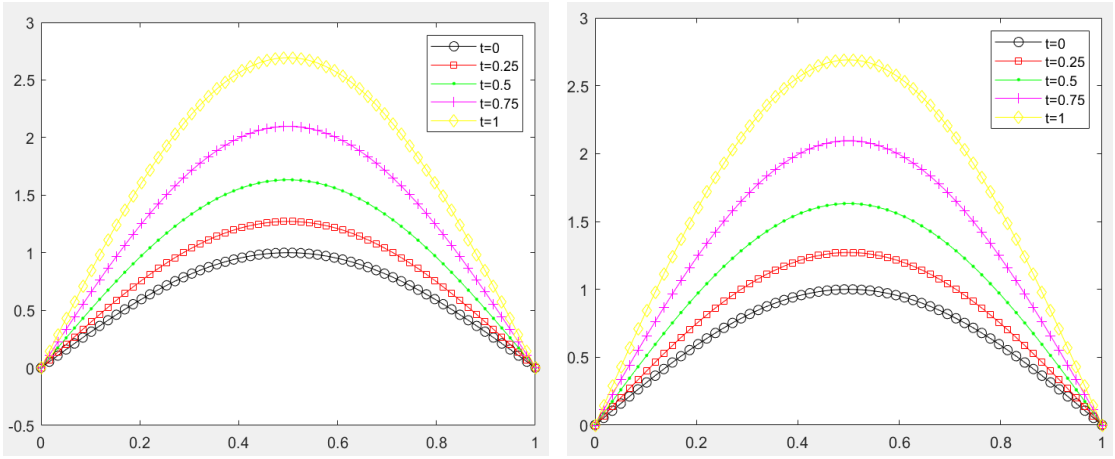
其中，最大误差$E_u = 2.521 \times 10^{-3}$。

由图3.1.4给出$t = 0, 0.25, 0.5, 0.75, 1$时，$\hat{\mu}^n(x)$与$\mu^n(x)$的图像。其中图3.1.4(a)为各时间下，$\hat{\mu}^n(x)$关于$x$的函数图像。图 3.1.4(b) 为各时间下，$\mu^n(x)$关于$x$的函数图像。

其中，最大误差$E_\mu = 2.227 \times 10^{-3}$。

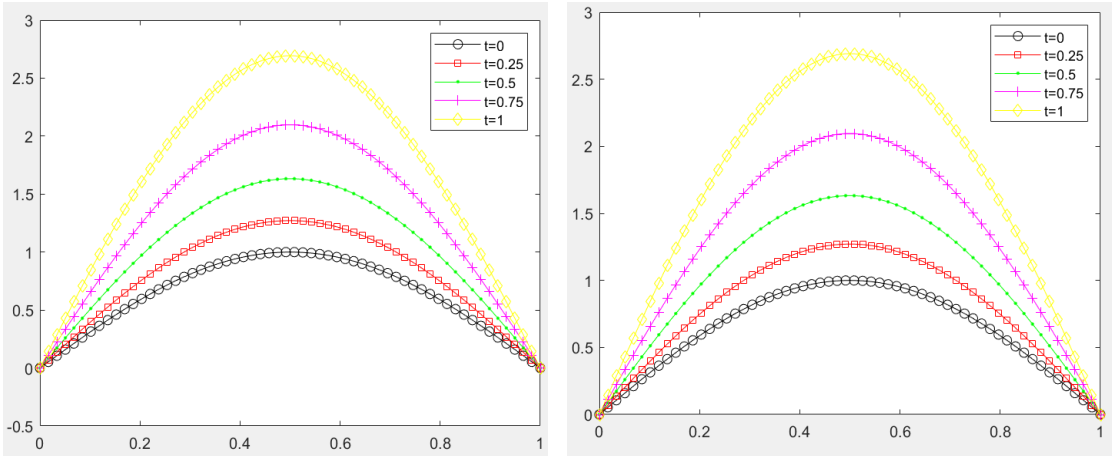
综合上述图像与误差，本文使用的无网格法整体是一种精度尚可，运算简便的方法。

图 3.1.5 表示了在每次递推时，E_u^n和E_μ^n随 n 改变的图像。其中，图 3.1.5(a)为 E_u^n随 n 改变的图像，图 3.1.5(b)为E_μ^n随 n 改变的图像。根据图像可以发现，E_u^n和E_μ^n总体随 n 增大，呈逐渐增大的趋势。推测为每次随时间差分时，由于使用半隐式方法，其中包含显式差分项，使得误差随递推过程，逐渐放大，因此当T过于大时，可能会出现近似效果不佳的现象。



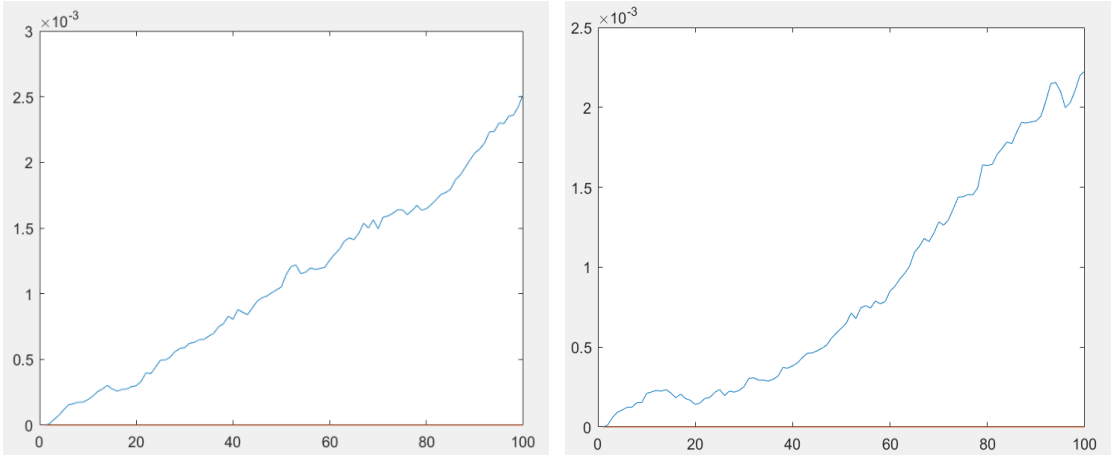
(a) **(b)**

图 3.1.3 各时间下，$\hat{u}$与u的图像



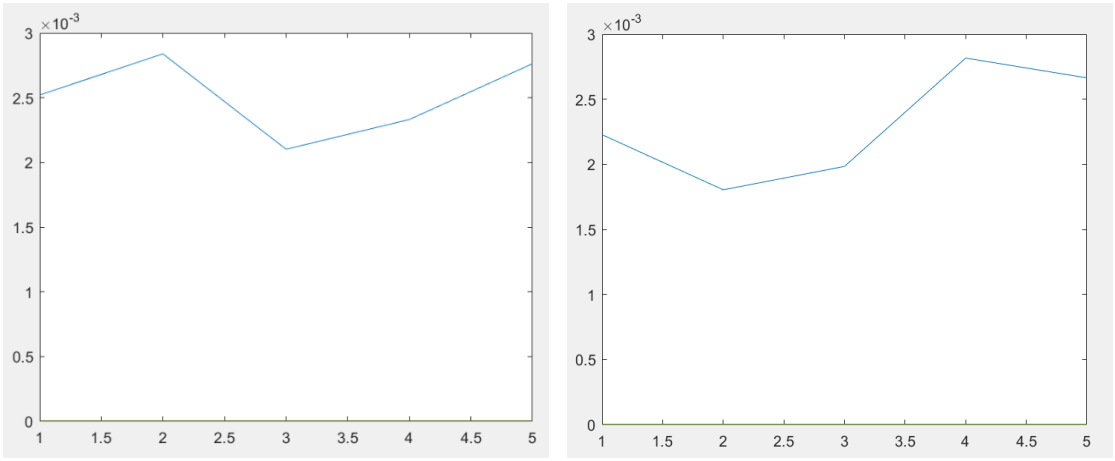
(a) **(b)**

图 3.1.4 各时间下，$\hat{\mu}$与μ的图像



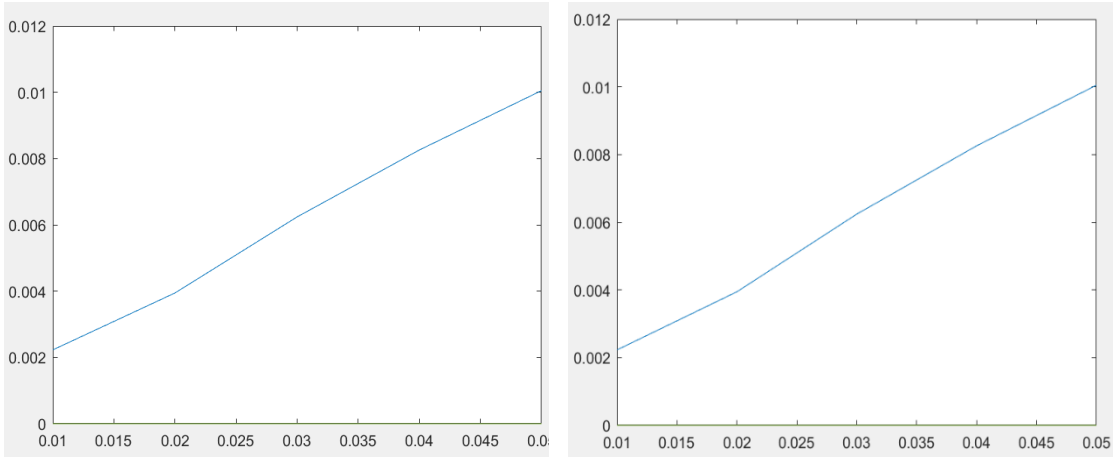
(a) **(b)**

图 3.1.5 E_u^n和E_μ^n随 n 改变的图像



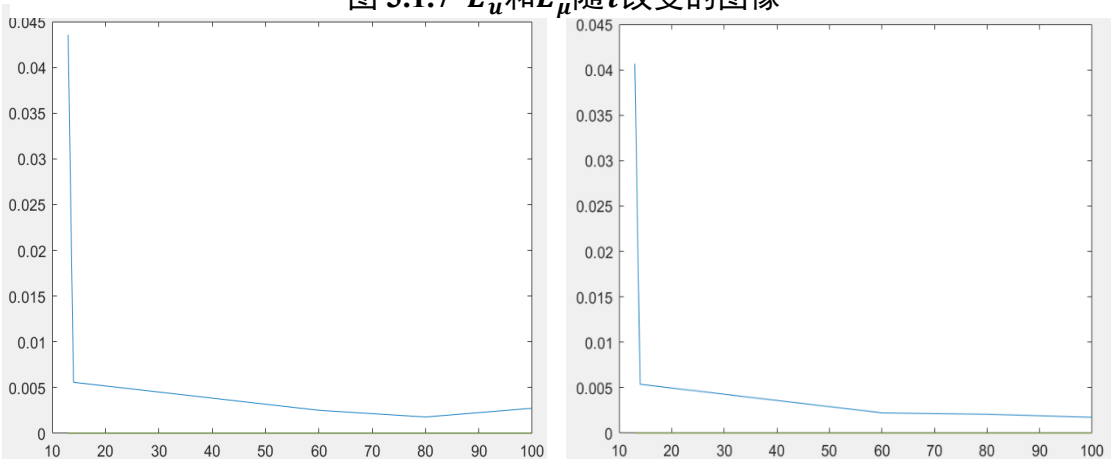
(a) **(b)**

图 3.1.6 E_u^c和E_μ^c随 c 改变的图像



(a) **(b)**

图 3.1.7 E_u^τ和E_μ^τ随τ改变的图像



(a) **(b)**

图 3.1.8 E_u^N和E_μ^N随 N 改变的图像

结合上述步骤，本小结将进一步分析各参数对方法精度产生的影响。首先考虑，径向基函数中的参数 c 对方法精度的影响。

图 3.1.6 与表 3.1.1 分别给出了参数$c = 1,2,3,4,5$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^c$和$E_\mu^c$。其中图 3.1.6(a)为$E_u^c$关于 c 的变化的曲线，图 3.1.6(b)为E_μ^c关于 c 的变化的曲线。

总体，可以发现E_u^c与E_μ^c随 c 的改变总体可控，但非常地不规律，需要多次尝试，寻找合适的参数 c。

表 3.1.1 不同参数 c 下的无网格法误差

c	E_u^c（保留四位有效数字）	E_μ^c（保留四位有效数字）
1	2.521×10^{-3}	2.227×10^{-3}
2	2.839×10^{-3}	1.804×10^{-3}
3	2.102×10^{-3}	1.984×10^{-3}
4	2.331×10^{-3}	2.816×10^{-3}
5	2.763×10^{-3}	2.664×10^{-3}

随后考虑，时间差分法中步长τ对方法精度的影响。

图 3.1.7 与表 3.1.2 分别给出了步长$\tau = 0.01,0.02,0.03,0.04,0.05$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^\tau$和$E_\mu^\tau$。其中图 3.1.7(a)为$E_u^\tau$关于$\tau$的变化的曲线，图 3.1.7(b)为 E_μ^τ关于τ的变化的曲线。

总体，可以发现E_u^c与E_μ^c随τ的变大而变大，这表明，当步长增大时，误差也会显著提升。猜测这是因为迭代次数变少，导致每次迭代时的误差显著增大。因此，在迭代时，应选择尽可能小的步长，获得更高的精度。

表 3.1.2 不同步长$\boldsymbol{\tau}$下的无网格法误差

τ	E_u^τ（保留四位有效数字）	E_μ^τ（保留四位有效数字）
0.01	2.521×10^{-3}	2.227×10^{-3}
0.02	4.350×10^{-3}	3.945×10^{-3}
0.03	5.942×10^{-3}	6.240×10^{-3}
0.04	8.386×10^{-3}	8.259×10^{-3}
0.05	1.000×10^{-2}	1.005×10^{-2}

最后考虑，构造近似函数时，选取的虚拟点个数 N 对方法精度的影响。

图 3.1.8 与表 3.1.3 分别给出了虚拟点个数$N = 13,14,60,80,100$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^N$和$E_\mu^N$。其中图 3.1.8(a)为$E_u^N$关于 N 的变化的曲线，图 3.1.8(b)为 E_μ^N关于 N 的变化的曲线。

总体，可以发现，无网格法需要足够的虚拟点作为基础，当虚拟点个数过少时，方法精度极低。而当虚拟点个数足够时，精度变化不会过大，相对稳定。但是E_u^N与E_μ^N并不一味地随N的变大而变大，这表明，在选取虚拟点时，并非越多越好，也应合理选择。

表 3.1.3 不同虚拟点个数 *N* 下的无网格法误差

N	E_u^N（保留四位有效数字）	E_μ^N（保留四位有效数字）
13	4.357×10^{-2}	4.066×10^{-2}
14	5.581×10^{-3}	5.372×10^{-3}
60	2.521×10^{-3}	2.227×10^{-3}
80	1.788×10^{-3}	2.073×10^{-3}
100	2.745×10^{-3}	1.740×10^{-3}

3.2 一维含拉普拉斯项的非线性耦合方程组

考虑上一章推导的方程组 2.0.1 的一维情况：

$$\begin{cases} \dfrac{\partial u}{\partial t}=\Delta u-a_1 f(u)\mu+F_1(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ \dfrac{\partial \mu}{\partial t}=\Delta \mu-a_2 f(u)\mu+F_2(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ u(\boldsymbol{X},0)=g_1(\boldsymbol{X}),\quad \mu(\boldsymbol{X},0)=g_2(\boldsymbol{X}), & \boldsymbol{X}\in\Omega \\ u(\boldsymbol{X},t)=0,\quad \mu(\boldsymbol{X},t)=0. & (\boldsymbol{X},t)\in\partial\Omega\times(0,T) \end{cases} \tag{3.2.1}$$

令其中，$\boldsymbol{X}$是一维的，$a_1=a_2=1$，$f(u)=\frac{u}{u+1}$，$g_1(\boldsymbol{X})=g_2(\boldsymbol{X})=sin\,(\pi\boldsymbol{X})$，$\Omega=(0,1)$，$T=1$。则$\partial\Omega=[0,1]$。

令$F_1(\boldsymbol{X},t)=F_2(\boldsymbol{X},t)=e^t\cdot sin(\pi x)+\pi\cdot e^t\cdot sin(\pi x)+\frac{e^{2t}\cdot sin^2(\pi x)}{e^t\cdot sin(\pi x)+1}$，则对应的精确解：

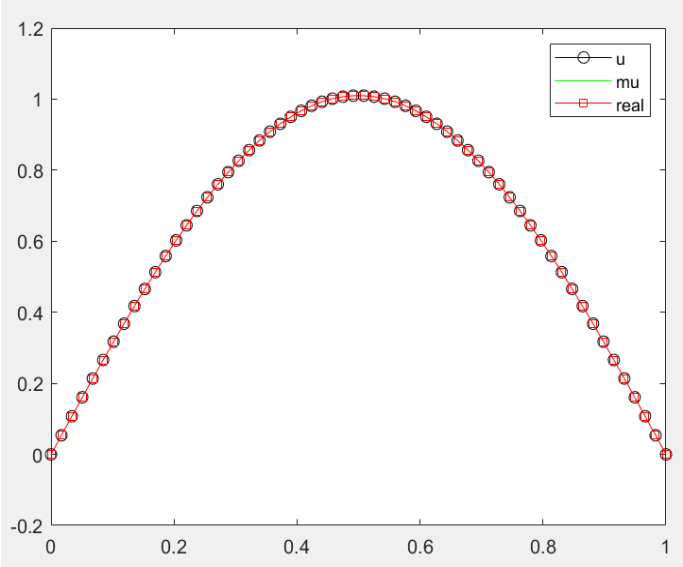
$$u(\boldsymbol{X},t)=\mu(\boldsymbol{X},t)=\mathrm{e}^{\mathrm{t}}\cdot\sin(\pi\mathrm{x}).$$

解上述方程的无网格法可参考第二章中的式 2.2.14，式 2.2.15 与式 2.2.16。

对于上述方法，在$\Omega=(0,1)$内均匀地取$N=60$个虚拟点，即$N_i=58$，$N_b=2$，开始进行无网格法处理，得到如下函数图像：

由图 3.2.1 给出第一次迭代，即$t=\tau=0.01$时，$\hat{u}^1(x)$，$\hat{\mu}^1(x)$，$u^1(x)$，$\mu^1(x)$关于x的函数图像。（图中曲线 u 表示$\hat{u}^1(x)$，曲线 mu 表示$\hat{\mu}^1(x)$，由于$u^1(x)=\mu^1(x)$，所以在图中用同一条曲线 real 表示）。

其中，最大的误差$E_u^1=2.302\times10^{-5}$，$E_\mu^1=3.807\times10^{-5}$。

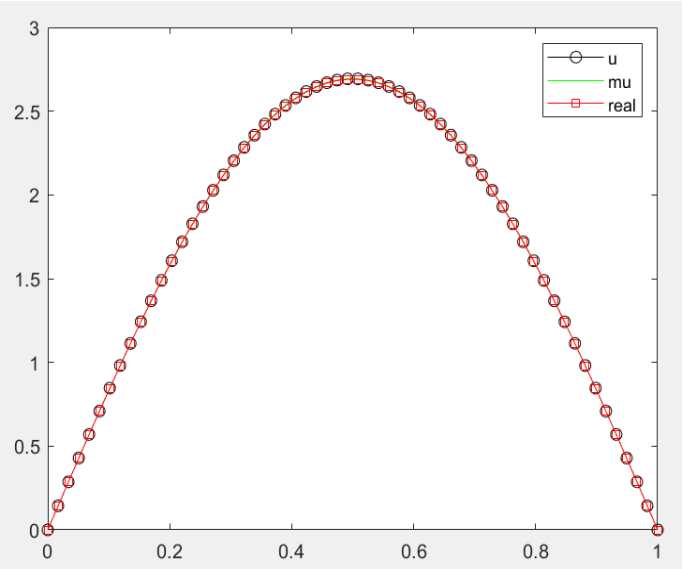


x

图 3.2.1 t=0.01 时，近似函数与精确值的图像

随后，由图 3.2.2 给出最后一次迭代时，即$t = T = 1$，时，$\hat{u}^{100}(x)$，$\hat{\mu}^{100}(x)$，$u^{100}(x)$，$\mu^{100}(x)$关于x的函数图像。（与上相同，图中曲线 u 表示$\hat{u}^{100}(x)$，曲线 mu 表示$\hat{\mu}^{100}(x)$，由于$u^{100}(x) = \mu^{100}(x)$，所以在图中用同一条曲线 real 表示）。

其中，最大的误差$E_u^{100} = 2.560 \times 10^{-4}$，$E_\mu^{100} = 2.955 \times 10^{-4}$。

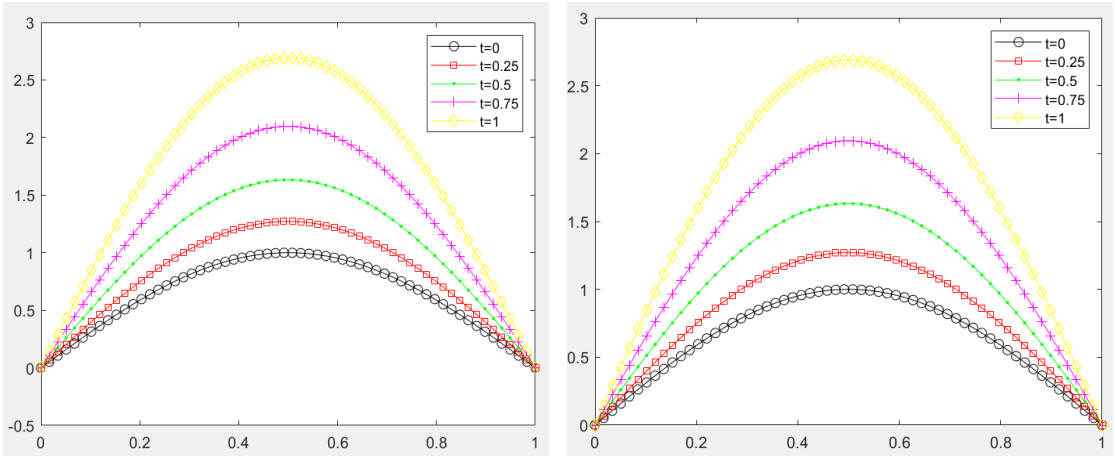


x

图 3.2.2 t=1 时，近似函数与精确值的图像

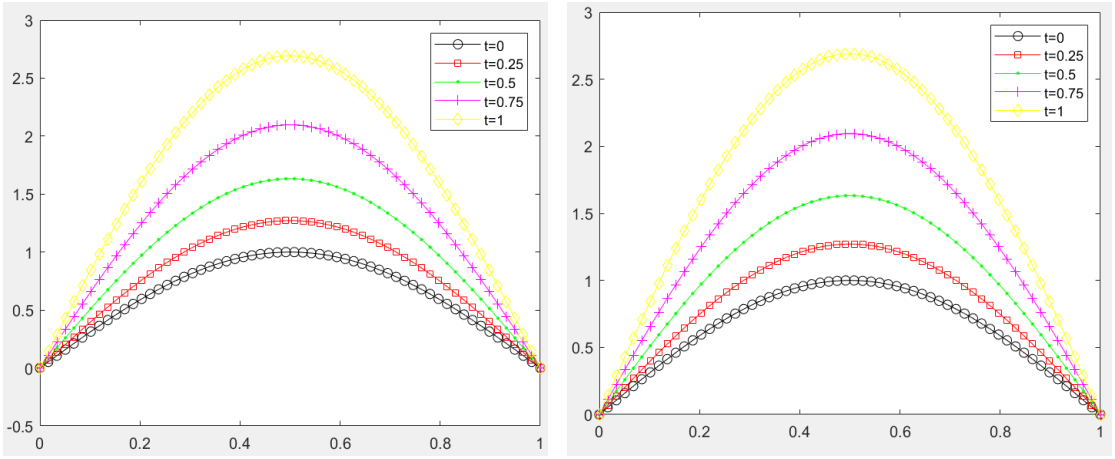
由图3.2.3 给出$t = 0, 0.25, 0.5, 0.75, 1$时，$\hat{u}^n(x)$与$u^n(x)$的图像。其中图3.2.3(a)为各时间下，$\hat{u}^n(x)$关于$x$的函数图像。图 3.2.3(b) 为各时间下，$u^n(x)$关于$x$的函数图像。

其中，最大误差$E_u = 3.628 \times 10^{-4}$。



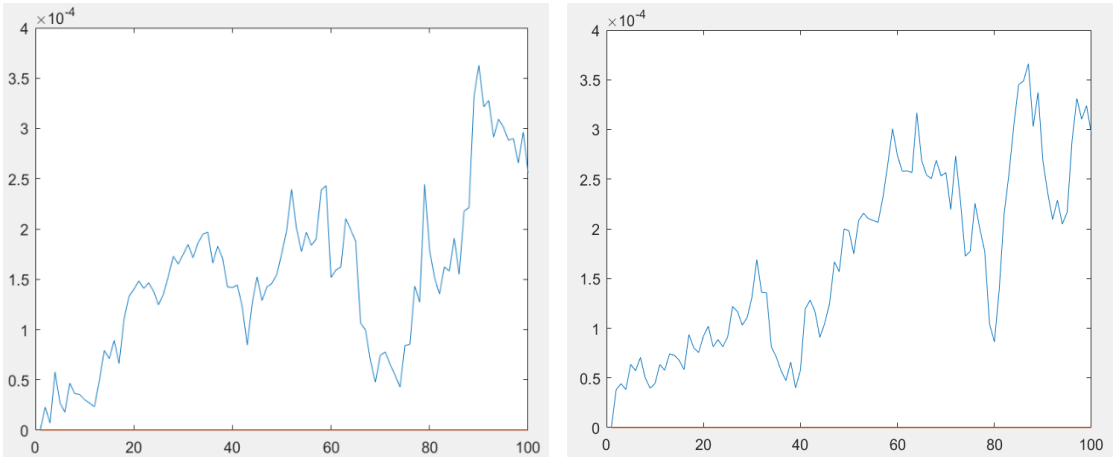
(a) **(b)**

图 3.2.3 各时间下，$\hat{u}$与u的图像



(a) **(b)**

图 3.2.4 各时间下，$\hat{\mu}$与μ的图像



(a) **(b)**

图 3.2.5 E_u^n和E_μ^n随 n 改变的图像

由图3.2.4给出$t=0,0.25,0.5,0.75,1$时，$\hat{\mu}^n(x)$与$\mu^n(x)$的图像。其中图3.2.4(a)为各时间下，$\hat{\mu}^n(x)$关于x的函数图像。图3.2.4(b) 为各时间下，$\mu^n(x)$关于x的函数图像。

其中，最大误差$E_\mu=3.662\times10^{-4}$。

综合上述图像与误差，本文使用的无网格法对于含高阶导数的微分方程组仍是一种精度尚可，运算简便的方法。

图3.2.5表示了在每次递推时，E_u^n和E_μ^n随n改变的图像。其中，图3.2.5(a)为E_u^n随n改变的图像，图3.2.5(b)为E_μ^n随n改变的图像。根据图像可以发现，当含有拉普拉斯项时，E_u^n和E_μ^n总体随n增大的趋势更不明显，但是从宏观上来看，依旧存在。因此依旧需要注意，当T过于大时，可能会出现的近似效果不佳的现象。

与上文相同，本小节将还进一步分析各参数对方法精度产生的影响，已验证是否与前文猜想一致。

首先考虑，径向基函数中的参数c对方法精度的影响。

图3.2.6与表3.2.1分别给出了参数$c=1,2,3,4,5$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^c$和$E_\mu^c$。其中图3.2.6(a)为$E_u^c$关于$c$的变化的曲线，图3.2.6(b)为$E_\mu^c$关于$c$的变化的曲线。

总体，可以发现，当c取1,2,3,4时，与前文类似，E_u^c与E_μ^c随c的改变总体可控，但非常地不规律。而当$c=5$时，会出现误差的巨大增加，这表明，当参数c选择不当时，会使方法效果下降明显，需要警惕。

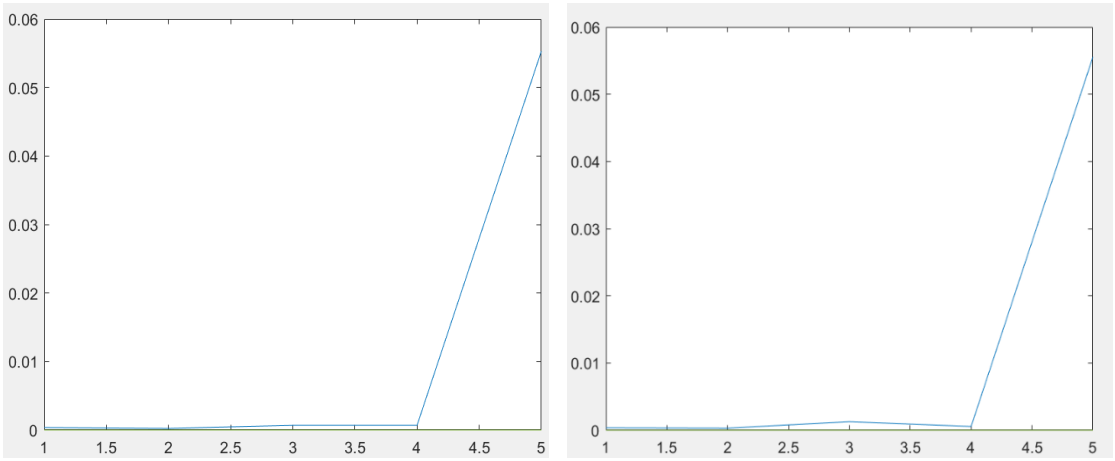
表3.2.1 不同参数c下的无网格法误差

c	E_u^c（保留四位有效数字）	E_μ^c（保留四位有效数字）
1	3.628×10^{-4}	3.662×10^{-4}
2	2.339×10^{-4}	3.094×10^{-4}
3	6.808×10^{-4}	1.269×10^{-3}
4	6.893×10^{-4}	5.574×10^{-4}
5	5.531×10^{-2}	5.553×10^{-2}

随后考虑，时间差分法中步长τ对方法精度的影响。

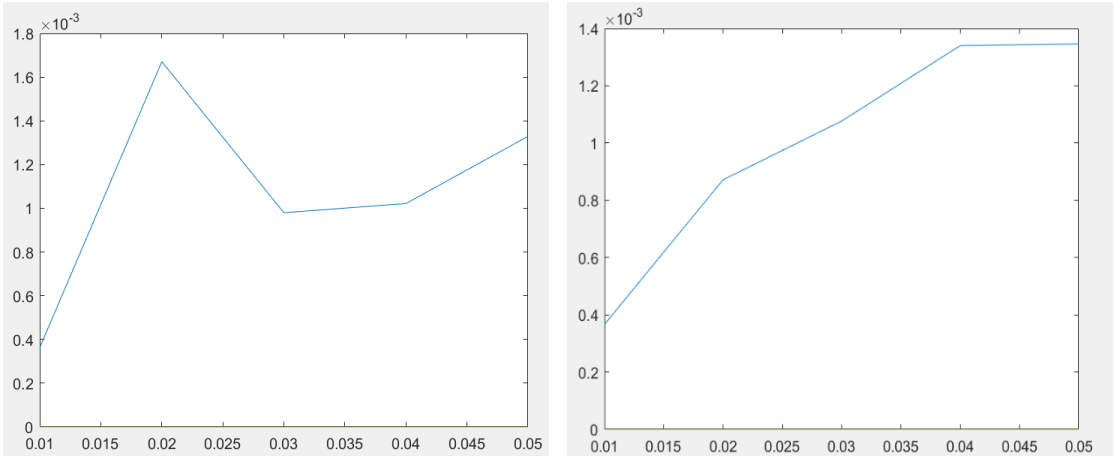
图3.2.7与表3.2.2分别给出了步长$\tau=0.01,0.02,0.03,0.04,0.05$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^\tau$和$E_\mu^\tau$。其中图3.2.7(a)为$E_u^\tau$关于$\tau$的变化的曲线，图3.2.7(b)为 E_μ^τ关于τ的变化的曲线。

与上文类似，总体上，E_u^c与E_μ^c随τ的变大而变大，因此尽量小的步长，对于方法来说更为合适。



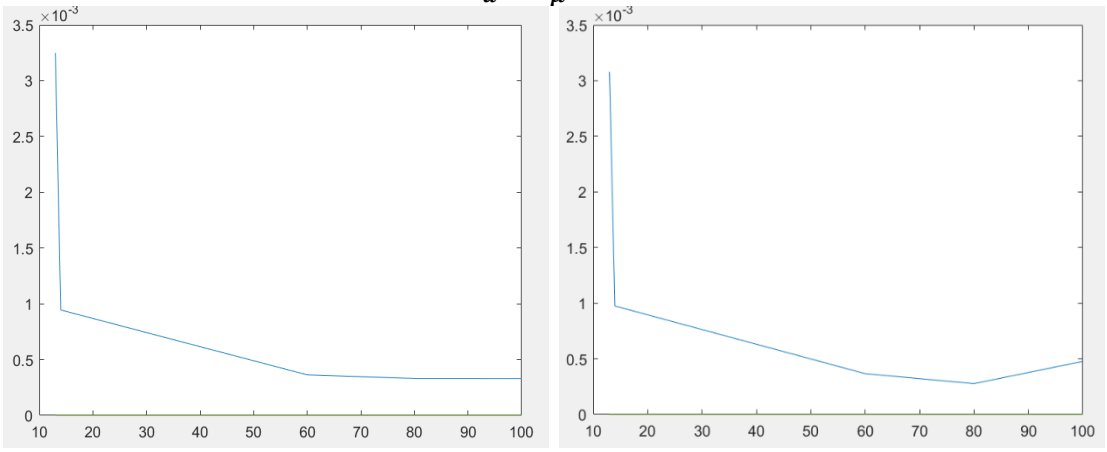
(a) **(b)**

图 3.2.6 E_u^c和E_μ^c随 c 改变的图像



(a) **(b)**

图 3.2.7 E_u^τ和E_μ^τ随τ改变的图像



(a) **(b)**

图 3.2.8 E_u^N和E_μ^N随 N 改变的图像

表 3.2.2 不同步长τ下的无网格法误差

τ	E_u^τ（保留四位有效数字）	E_μ^τ（保留四位有效数字）
0.01	3.628×10^{-4}	3.662×10^{-4}
0.02	1.670×10^{-3}	8.715×10^{-4}
0.03	9.798×10^{-4}	1.077×10^{-3}
0.04	1.022×10^{-3}	1.340×10^{-3}
0.05	1.329×10^{-3}	1.346×10^{-3}

最后考虑，构造近似函数时，选取的虚拟点个数 N 对方法精度的影响。

图 3.2.8 与表 3.2.3 分别给出了虚拟点个数$N=13,14,60,80,100$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^N$和$E_\mu^N$。其中图 3.2.8(a)为$E_u^N$关于 N 的变化的曲线，图 3.2.8(b)为 E_μ^N关于 N 的变化的曲线。

总体，可以发现，与上文相同，当虚拟点个数过少时，方法精度极低。而当虚拟点个数足够时，精度变化不会过大，相对稳定。且E_u^N与E_μ^N并不一味地随N的变大而变大。

表 3.2.3 不同虚拟点个数 N 下的无网格法误差

N	E_u^N（保留四位有效数字）	E_μ^N（保留四位有效数字）
13	3.250×10^{-3}	3.079×10^{-3}
14	9.445×10^{-4}	9.747×10^{-4}
60	3.628×10^{-4}	3.662×10^{-4}
80	3.306×10^{-4}	2.766×10^{-4}
100	3.293×10^{-4}	4.766×10^{-4}

3.3 二维含拉普拉斯项的非线性耦合方程组

考虑上一章推导的方程组 2.0.1 的二维情况：

$$\begin{cases} \dfrac{\partial u}{\partial t}=\Delta u-a_1 f(u)\mu+F_1(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ \dfrac{\partial \mu}{\partial t}=\Delta \mu-a_2 f(u)\mu+F_2(\boldsymbol{X},t), & (\boldsymbol{X},t)\in\Omega\times(0,T) \\ u(\boldsymbol{X},0)=g_1(\boldsymbol{X}), \quad \mu(\boldsymbol{X},0)=g_2(\boldsymbol{X}), & \boldsymbol{X}\in\Omega \\ u(\boldsymbol{X},t)=0, \quad \mu(\boldsymbol{X},t)=0. & (\boldsymbol{X},t)\in\partial\Omega\times(0,T) \end{cases} \tag{3.2.2}$$

令其中，$\boldsymbol{X}$是二维的，包含(x_1,x_2)，$a_1=a_2=1$，$f(u)=\frac{u}{u+1}$，$g_1(\boldsymbol{X})=g_2(\boldsymbol{X})=sin\,(\pi x_1)\cdot sin\,(\pi x_2)$，$\Omega=(0,1)\times(0,1)$，$T=1$。则$\partial\Omega=[0,1]\times(0,1)\cup(0,1)\times[0,1]$。

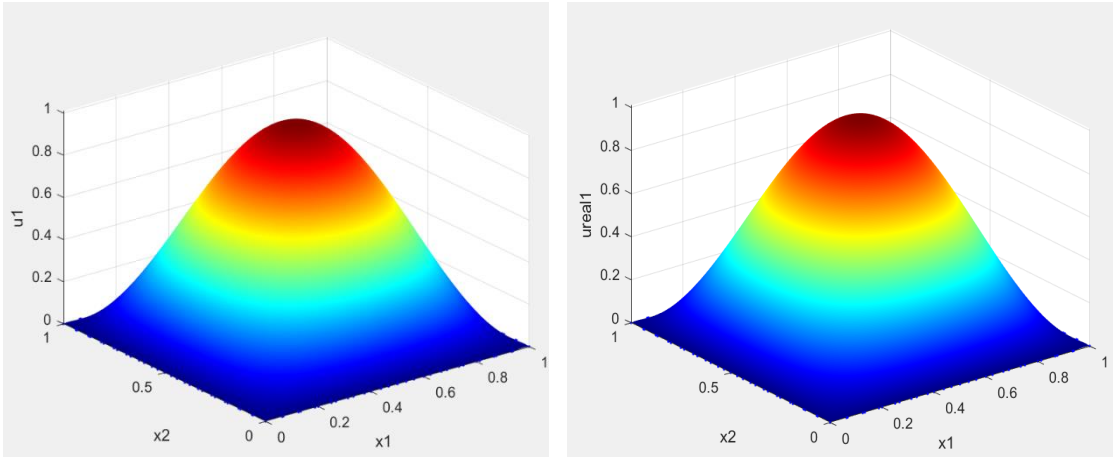
令$F_1(\boldsymbol{X},t)=F_2(\boldsymbol{X},t)=\frac{e^{2t}\cdot sin^2(\pi x_1)\cdot sin^2(\pi x_2)}{e^t\cdot sin(\pi x_1)\cdot sin(\pi x_2)+1}+(2\pi^2+1)\cdot e^t\cdot sin(\pi x_1)\cdot sin(\pi x_2)-2\pi^2\cdot e^t\cdot cos(\pi x_1)\cdot cos(\pi x_2)$，

则对应的精确解：

$$u(\boldsymbol{X},t)=\mu(\boldsymbol{X},t)=e^t\cdot sin(\pi x_1)\cdot sin(\pi x_2).$$

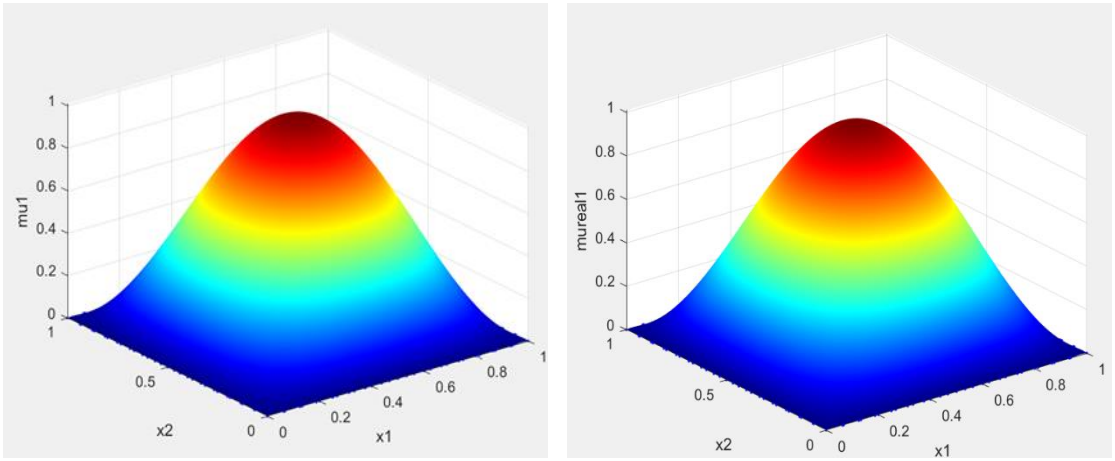
解上述方程的无网格法依然可以参考第二章中的式 2.2.14，式 2.2.15 与式 2.2.16，其中关于二维径向基函数的拉普拉斯导数已由式 2.2.6 推导。

对于上述方法，在$\Omega=(0,1)\times(0,1)$内均匀地取$N=18\times18=324$个虚拟点，即$N_i=16\times16=256$，$N_b=4\times17=68$，开始进行无网格法处理，得到如下函数图像：



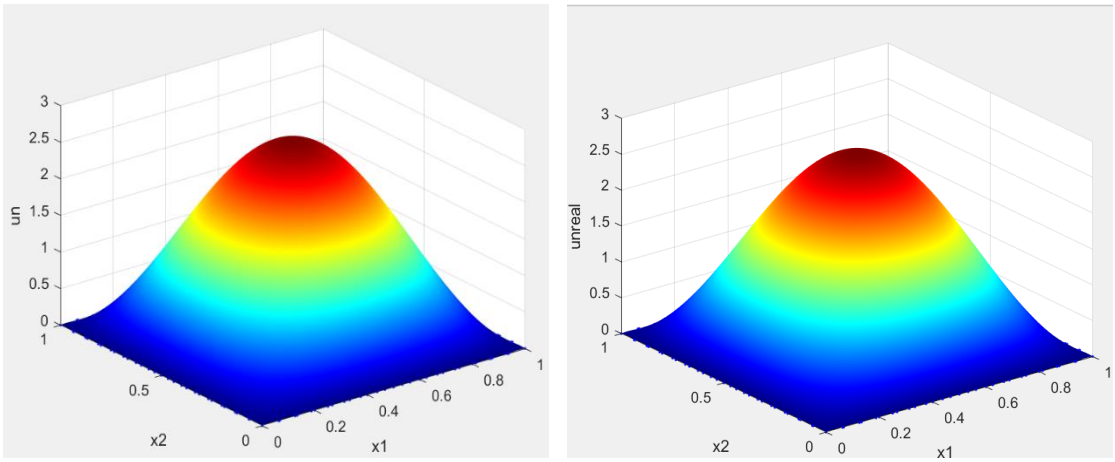
(a) **(b)**

图 3.3.1 当$t=\tau=0.01$时，$\hat{u}$与u的图像



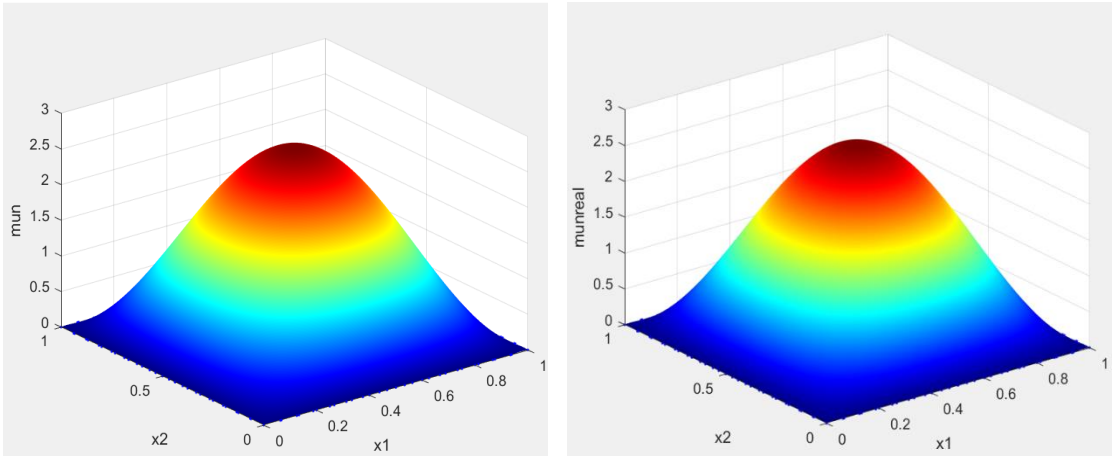
(a) **(b)**

图 3.3.2 当$t=\tau=0.01$时，$\hat{\mu}$与μ的图像



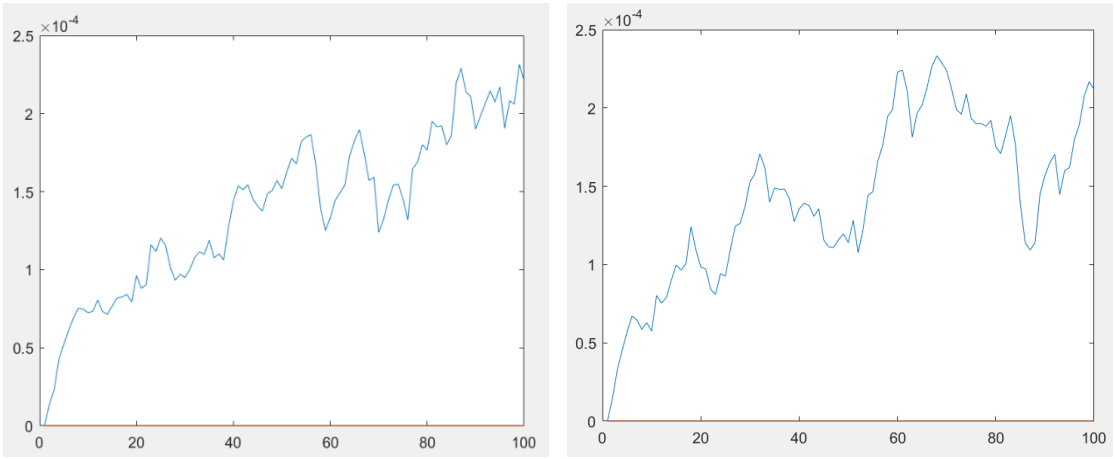
(a) **(b)**

图 3.3.3 当$t = T = 1$时，$\hat{u}$与u的图像



(a) **(b)**

图 3.3.4 当$t = T = 1$时，$\hat{\mu}$与μ的图像



(a) **(b)**

图 3.3.5 E_u^n和E_μ^n随 n 改变的图像

首先，由图 3.3.1 给出第一次迭代，即$t=\tau=0.01$时，$\hat{u}^1(\boldsymbol{X})$，$u^1(\boldsymbol{X})$关于$x_1$，$x_2$的函数图像。（其中图 3.3.1(a)表示$\hat{u}^1(\boldsymbol{X})$，图 3.3.1(b)表示$u^1(\boldsymbol{X})$）

其中，最大的误差$E_u^1=1.353\times10^{-5}$。

首先，由图 3.3.2 给出第一次迭代，即$t=\tau=0.01$时，$\hat{\mu}^1(\boldsymbol{X})$，$\mu^1(\boldsymbol{X})$关于$x_1$，$x_2$的函数图像。（其中图 3.3.2(a)表示$\hat{\mu}^1(\boldsymbol{X})$，图 3.3.2(b)表示$\mu^1(\boldsymbol{X})$）

其中，最大的误差$E_\mu^1=1.412\times10^{-5}$。

随后，由图 3.3.3 给出最后一次迭代时，即$t=T=1$，时，$\hat{u}^{100}(\boldsymbol{X})$，$u^{100}(\boldsymbol{X})$关于$x_1$，$x_2$的函数图像。（其中图 3.3.3(a)表示$\hat{u}^{100}(\boldsymbol{X})$，图 3.3.3(b)表示$u^{100}(\boldsymbol{X})$）

其中，最大的误差$E_u^{100}=2.209\times10^{-4}$。

由图 3.3.4 给出最后一次迭代时，即$t=T=1$，时，$\hat{\mu}^{100}(\boldsymbol{X})$，$\mu^{100}(\boldsymbol{X})$关于$x_1$，$x_2$的函数图像。（其中图 3.3.4(a)表示$\hat{\mu}^{100}(\boldsymbol{X})$，图 3.3.4(b)表示$\mu^{100}(\boldsymbol{X})$）

其中，最大的误差$E_\mu^{100}=2.117\times10^{-4}$。

整体方法的最大误差$E_u=2.316\times10^{-4}$，$E_\mu=2.333\times10^{-4}$。

图 3.3.5 表示了在每次递推时，E_u^n和E_μ^n随 n 改变的图像。其中，图 3.3.5(a)为 E_u^n随 n 改变的图像，图 3.3.5(b)为E_μ^n随 n 改变的图像。图像基本与一维含拉普拉斯项的情况一致。E_u^n和E_μ^n总体随 n 改变不规律，但是从宏观上来看，依旧在逐渐增大。

与上文相同，本小节剩余部分将分析各参数对方法精度产生的影响。

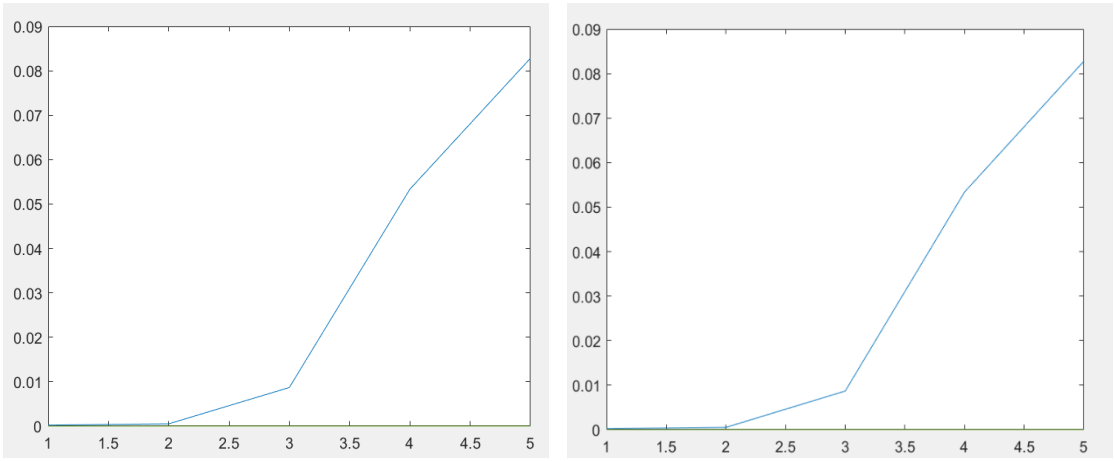
首先考虑，径向基函数中的参数 c 对方法精度的影响。

图 3.3.6 与表 3.3.1 分别给出了参数$c=1,2,3,4,5$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^c$和$E_\mu^c$。其中图 3.3.6(a)为$E_u^c$关于 c 的变化的曲线，图 3.3.6(b)为E_μ^c关于 c 的变化的曲线。

总体，与前文不同，当参数 c 增大时，无网格法的误差呈现显著增大，这表明在高维复杂情况下，方法对于 c 的敏感度更高，c 的选择不合适，将导致更严重的误差。但这并不意味着当 c 越小时，方法精度越高，事实上，当 c 过小时，依然会出现误差过大的情况。

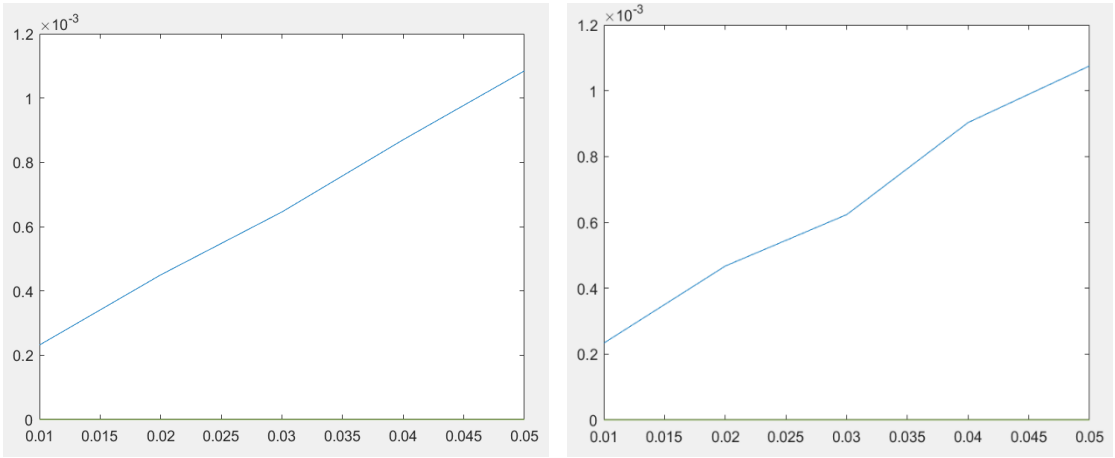
表 3.3.1 不同参数 c 下的无网格法误差

c	E_u^c（保留四位有效数字）	E_μ^c（保留四位有效数字）
1	2.316×10^{-4}	2.333×10^{-4}
2	5.144×10^{-4}	5.157×10^{-4}
3	8.690×10^{-3}	8.677×10^{-3}
4	5.334×10^{-2}	5.338×10^{-2}
5	8.281×10^{-2}	8.277×10^{-2}



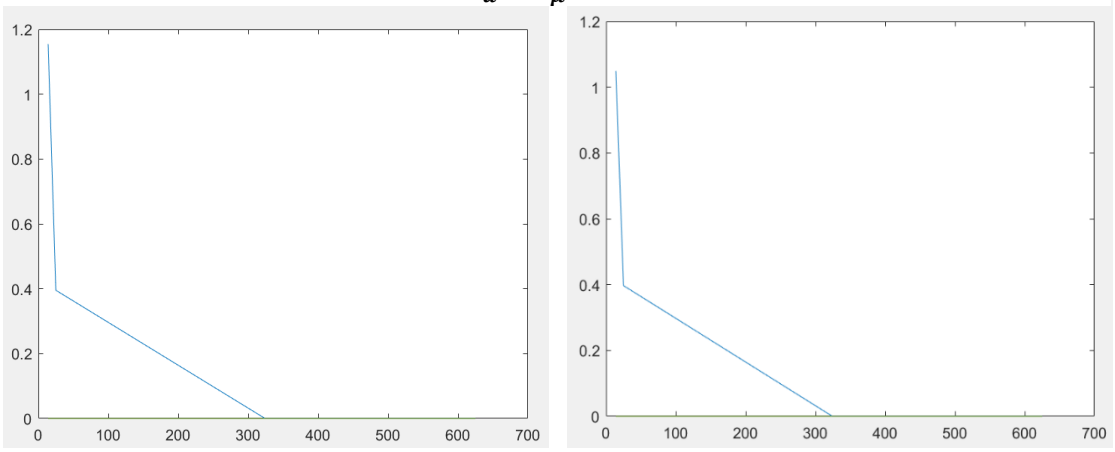
(a) **(b)**

图 3.3.6 E_u^c和E_μ^c随 c 改变的图像



(a) **(b)**

图 3.3.7 E_u^τ和E_μ^τ随τ改变的图像



(a) **(b)**

图 3.3.8 E_u^N和E_μ^N随 N 改变的图像

随后考虑，时间差分法中步长τ对方法精度的影响。

图 3.3.7 与表 3.3.2 分别给出了步长$\tau = 0.01, 0.02, 0.03, 0.04, 0.05$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^\tau$和$E_\mu^\tau$。其中图 3.3.7(a)为$E_u^\tau$关于$\tau$的变化的曲线，图 3.3.7(b)为 E_μ^τ关于τ的变化的曲线。

可以很明显地观察到误差随步长的变大，即迭代次数的减少而增大，因此由于前文结论一致，尽量小的步长，对于方法来说更为合适。

表 3.3.2 不同步长τ下的无网格法误差

τ	E_u^τ（保留四位有效数字）	E_μ^τ（保留四位有效数字）
0.01	2.316×10^{-4}	2.333×10^{-4}
0.02	4.499×10^{-4}	4.673×10^{-4}
0.03	6.457×10^{-4}	6.236×10^{-4}
0.04	8.704×10^{-4}	9.033×10^{-4}
0.05	1.085×10^{-3}	1.075×10^{-3}

最后考虑，构造近似函数时，选取的虚拟点个数 N 对方法精度的影响。

图 3.3.8 与表 3.3.3 分别给出了虚拟点个数$N = 16, 25, 324, 400, 625$时，方法总体对于$\hat{u}(\boldsymbol{X})$和$\hat{\mu}(\boldsymbol{X})$的最大误差$E_u^N$和$E_\mu^N$。其中图 3.3.8(a)为$E_u^N$关于 N 的变化的曲线，图 3.3.8(b)为 E_μ^N关于 N 的变化的曲线。

结论与上文相似，当虚拟点个数过少时，方法精度极低。而当虚拟点个数足够时，精度变化不会过大，相对稳定。

表 3.3.3 不同虚拟点个数 N 下的无网格法误差

N	E_u^N（保留四位有效数字）	E_μ^N（保留四位有效数字）
16	1.155	1.049
25	3.954×10^{-1}	3.969×10^{-1}
324	2.316×10^{-4}	2.333×10^{-4}
400	2.632×10^{-4}	2.489×10^{-4}
625	2.352×10^{-4}	2.247×10^{-4}

结论

综上，本文针对生物学中某类典型的捕食者与猎物模型进行了无网格法尝试，以验证无网格法求解非线性耦合方程时的效果。在推导过程中，本文先利用了半隐式差分法对方程组进行线性化处理，使其易于求解，再利用径向基函数构造近似函数，代入方程组，求解，得到近似解。实验结果表明此类无网格法有着不错的精度，同时十分易于求解，求解过程耗时很短，是一种可靠的方案。因此，本文中使用的方法可以被广泛应用于各种包含非线性耦合方程的自然科学模型中。尤其是对于部分含有时间作为变量的方程，具有很强的现实意义，是一种实用的方法。

同时本文也对无网格法对参数的敏感度作了测试，发现方法对于部分参数，包括径向基函数系数、时间差分步长以及虚拟点数量较为敏感。其中，其精度随径向基函数中的参数 c 会不规则地改变这一问题较为显著。对于此问题，可以考虑使用某种增强型径向基函数。即在原径向基函数的基础上加入多项式，该方法可以一定程度上减少 c 的变化带来的影响[20]。另外，本方法中的半隐式差分法，在时间宽度较大的情况下，会不断放大误差，使结果精确度下降。为避免这种因为递推过程中产生放大的误差，可以在非线性项作时间差分过程中，用推导后的非线性项再次代入原本的显式差分项中，作隐式差分，使结果更精确。

参考文献

[1] 寿媛，张辉.利用拟插值方法求解 KdV 方程[J].洛阳师范学院学报，2021，40(08)：1-3.

[2] 张关泉.波动方程的上行波和下行波的耦合方程组[J].应用数学学报，1993，(02)：251-263.

[3] Dimitrov, Dobromir T., and Hristo V. Kojouharov. Nonstandard Finite-Difference Methods for Predator-Prey Models with General Functional Response[J]. Mathematics and computers in simulation, 2008, 78(1): 1-11.

[4] 张雄，宋康祖，陆明万.无网格法研究进展及其应用[J].计算力学学报，2003，(06)：730-742.

[5] Liszka T, Orkisz J. Finite Difference Method for Arbitrary Irregular Meshes in Nonlinear Problems of Applied Mechanics [M]．San Francisco: IVS MiRt，1977.

[6] Perrone N，Kao R．A general finite difference method for arbitrary meshes [J]．Compute Struct, 1975, 5: 45-58．

[7] Lucy LB．A numerical approach to the testing of the fission hypothesis[J]．The Astron J, 1977, 8(12): 1013-1024．

[8] Gingold R A, Moraghan J J．Smoothed particle hydrodynamics： theory and applications to non-spherical stars[J]．Mon Not Roy Astrou Soc，1977，18：375-389．

[9] 张雄，胡炜，潘小飞，等.加权最小二乘无网格法[J].力学学报，2003，(04)：425-431.

[10] 龙述尧，刘凯远，胡德安.移动最小二乘近似函数中样条权函数的研究[J].湖南大学学报(自然科学版)，2003，(06):10-13+18.

[11] 马文涛，师俊平，李宁.模拟裂纹扩展的单位分解扩展无网格法[J].计算力学学报，2013，30(01)：28-33.

[12] 白泽刚，杨元明，钱勤.一种新型核函数下的无单元法及应用[J].陕西工学院学报，2001，(01)：54-58.

[13] 高政国，刘光廷.数值计算中的一种无网格方法研究[J].工程力学，2003，(06)：28-33.

[14] 柴杰，江青茵，曹志凯.RBF 神经网络的函数逼近能力及其算法[J].模式识别与人工智能，2002，15(03)：310-316.

[15] 朱明星，张德龙.RBF 网络基函数中心选取算法的研究[J].安徽大学学报(自然科学版)，2000，(01)：72-78.

[16] 郑永骏，金之雁，陈德辉.半隐式半拉格朗日动力框架的动能谱分析[J].气象学报，2008，(02)：143-157.

[17] 曹婉容，刘明珠.随机延迟微分方程半隐式 Milstein 数值方法的稳定性[J].哈尔滨工业大学学报，2005，(04)：446-448.

[18] 胡明皓，王莉华.基于拉格朗日插值的无网格直接配点法和稳定配点法[J].力学学报，2023，55(07)：1526-1536.

[19] 缪报，陈发来.径向基函数神经网络在散乱数据插值中的应用[J].中国科学技术大学学报，2001，(02)：12-19.

[20] Cao Dingding, Li Xinxiang, and Zhu Huiqing. A Polynomial-Augmented RBF Collocation Method for Fourth-Order Boundary Value Problems. Computers & mathematics with applications[J]. 2023, 133: 1-11.

附　录

附录 A、求解一维不含拉普拉斯项非线性耦合方程代码

```
ni=58; nb=2; nc=60; nt=20;%N=60， 边界 2， 内部 58
tau=0.01;c=1;
u0= @(x) sin(pi*x);
mu0= @(x) sin(pi*x);
ureal= @(x,t) exp(t).*sin(pi*x);
mureal= @(x,t) exp(t).*sin(pi*x);
F_1= @(x,t) exp(2*t).*(sin(pi*x).^2)./(exp(t).*sin(pi*x)+1);
F_2= @(x,t) exp(2*t).*(sin(pi*x).^2)./(exp(t).*sin(pi*x)+1);
phi = @(r,c) sqrt(c.^2.+r.^2);
xi=linspace(0,1,ni+2)';
xi=xi(2:ni+1);%内部点 Xi
xb=[0,1]';%边界点 Xb
xy=[xb;xi];
xc=linspace(0,1,nc)';%中心点 y
xt=linspace(0,1,nc)';
DM1=DistanceMatrix(xi,xc);%计算内部 r
DM2=DistanceMatrix(xb,xc);%计算边界 r
DM3=DistanceMatrix(xt,xc);
m1=repmat(xi,1,nc);m2=repmat(xc',ni,1);%size：58*60
m3=repmat(xt,1,nc);m4=repmat(xc',nc,1);%size：60*60
m5=repmat(xb,1,nc);m6=repmat(xc',nb,1);%size：2*60
mi1=m1-m2;%（xi-xc'）
mc1=m3-m4;%（xt-xc'）
mb1=m5-m6;
Phi=phi(DM1,c);
un=zeros(nc,10000);%解的矩阵
mun=zeros(nc,10000);
un(:,1)= u0(xt);
uni(:,1)= un((2:ni+1),1);
mun(:,1)= mu0(xt);
Ab=phi(DM2,c);          Ob=zeros(nb,nc); Obv=zeros(nb,1);
A1=(2-tau)*phi(DM1,c);          A2=tau*(uni(:,1)./(uni(:,1)+1)).*phi(DM1,c);
A3=zeros(ni,nc);                A4=(2-tau+tau*(uni(:,1)./(uni(:,1)+1))).*phi(DM1,c);
B1v=(tau+2)*u0(xi);             B2v=-tau*(uni(:,1)./(uni(:,1)+1)).*mu0(xi);
B3v=zeros(ni,1);                B4v=(-tau*(uni(:,1)./(uni(:,1)+1))+tau+2).*mu0(xi);
t_0=0*ones(ni);
t_1=tau*ones(ni);
F1=(F_1(xi,t_0)+F_1(xi,t_1))/2;
F2=(F_2(xi,t_0)+F_2(xi,t_1))/2;
```

```
bn=[zeros(nb,1);zeros(nb,1);2*tau*F1(:,1);2*tau*F2(:,1)];
A=[Ab Ob;Ob Ab;A1 A2;A3 A4];%A
tol=det(A);
Bv=[Obv;Obv;B1v+B2v;B3v+B4v];%B0
v=zeros(2*nc,5000);
v(:,2)=pinv(A)*(Bv+bn);%求伪逆效果更好  得到 v1
u_0=u0(xt);
u1=phi(DM3,c)*v((1:nc),2);
un(:,2)= phi(DM3,c)*v((1:nc),2);
uni(:,2)= un(2:ni+1,2);
mun(:,2)= phi(DM3,c)*v((nc+1:2*nc),2);

tend=1;
for i=2:floor(tend/tau)

Ab=phi(DM2,c);              Ob=zeros(nb,nc);
A1=(2-tau)*phi(DM1,c);          A2=tau*(uni(:,i)./(uni(:,i)+1)).*phi(DM1,c);
A3=zeros(ni,nc);                A4=(2-tau+tau*(uni(:,i)./(uni(:,i)+1))).*phi(DM1,c);
B1=(tau+2)*phi(DM1,c);          B2=-tau*(uni(:,i)./(uni(:,i)+1)).*phi(DM1,c);
B3=zeros(ni,nc);                B4=(-tau*(uni(:,i)./(uni(:,i)+1))+tau+2).*phi(DM1,c);
A=[Ab Ob;Ob Ab;A1 A2;A3 A4];
B=[Ob Ob;Ob Ob;B1 B2;B3 B4];

t_n=tau*(i-1)*ones(ni);
t_n1=tau*i*ones(ni);
F1=(F_1(xi,t_n)+F_1(xi,t_n1))/2;
F2=(F_2(xi,t_n)+F_2(xi,t_n1))/2;
bn=[zeros(nb,1);zeros(nb,1);2*tau*F1(:,1);2*tau*F2(:,1)];
v(:,i+1)=pinv(A)*(B*v(:,i)+bn);
un(:,i+1)=phi(DM3,c)*v((1:nc),i+1);
mun(:,i+1)=phi(DM3,c)*v((nc+1:2*nc),i+1);
uni(:,i+1)=un(2:ni+1,i+1);
end

unreal= zeros(nc,10000);
munreal= zeros(nc,10000);
for i=1:floor(tend/tau)
unreal(:,i)= ureal(xt,tau*(i-1));
munreal(:,i)= mureal(xt,tau*(i-1));
end

E_1= zeros(floor(tend/tau));
E_2= zeros(floor(tend/tau));
```

```
for i=2:floor(tend/tau)
E_1(i)= norm(un(:,i)-unreal(:,i),inf);
E_2(i)= norm(mun(:,i)-munreal(:,i),inf);
end

fprintf('Eu1 = %8.3e\n', E_1(2));
fprintf('Emu1 = %8.3e\n', E_2(2));
fprintf('Eun = %8.3e\n', E_1(floor(tend/tau)));
fprintf('Emun = %8.3e\n', E_2(floor(tend/tau)));
Eu_1=norm(E_1(1:floor(tend/tau)),inf);
fprintf('Eu = %8.3e\n', Eu_1);
Eu_2=norm(E_2(1:floor(tend/tau)),inf);
fprintf('Emu = %8.3e\n', Eu_2);
Ec=norm((un(:,(floor(tend/tau)))- un(:,(floor(tend/tau)-1))),inf)/norm(un(:,(floor(tend/tau))),inf);
fprintf('Ec = %8.3e\n', Ec);
plot(xt,un(:,1),'-ok');
hold on
plot(xt,un(:,(floor(0.25/tau))),'-sr');
hold on
plot(xt,un(:,(floor(0.5/tau))),'-g.');
hold on
plot(xt,un(:,(floor(0.75/tau))),'-+m');
hold on
plot(xt,un(:,(floor(tend/tau))),'-dy');
legend('t=0','t=0.25','t=0.5','t=0.75','t=1')
figure;

plot(xt,mun(:,1),'-ok');
hold on
plot(xt,mun(:,(floor(0.25/tau))),'-sr');
hold on
plot(xt,mun(:,(floor(0.5/tau))),'-g.');
hold on
plot(xt,mun(:,(floor(0.75/tau))),'-+m');
hold on
plot(xt,mun(:,(floor(tend/tau))),'-dy');
legend('t=0','t=0.25','t=0.5','t=0.75','t=1')
figure;

plot(xt,unreal(:,1),'-ok');
hold on
plot(xt,unreal(:,(floor(0.25/tau))),'-sr');
hold on
```

```
plot(xt,unreal(:,(floor(0.5/tau))),'-g.');
hold on
plot(xt,unreal(:,(floor(0.75/tau))),'-+m');
hold on
plot(xt,unreal(:,(floor(tend/tau))),'-dy');
legend('t=0','t=0.25','t=0.5','t=0.75','t=1')
figure;

plot(xt,munreal(:,1),'-ok');
hold on
plot(xt,munreal(:,(floor(0.25/tau))),'-sr');
hold on
plot(xt,munreal(:,(floor(0.5/tau))),'-g.');
hold on
plot(xt,munreal(:,(floor(0.75/tau))),'-+m');
hold on
plot(xt,munreal(:,(floor(tend/tau))),'-dy');
legend('t=0','t=0.25','t=0.5','t=0.75','t=1')
figure;

plot(xt,un(:,2),'-ok');
hold on
plot(xt,mun(:,2),'-g');
hold on
plot(xt,unreal(:,2),'-sr');
legend('u','mu','real')
figure;

plot(xt,un(:,(floor(tend/tau))),'-ok');
hold on
plot(xt,mun(:,(floor(tend/tau))),'-g');
hold on
plot(xt,unreal(:,(floor(tend/tau))),'-sr');
legend('u','mu','real')
figure;

N= linspace(1,100,100)';
plot(N,E_1);
figure;
plot(N,E_2)
```

附录 B、求解二维含拉普拉斯项非线性耦合方程代码

```
tau=0.01;c=1;
```

```
u0= @(x,y) sin(pi*x).*sin(pi*y);
u0_1= @(x,y) pi.*cos(pi*x).*sin(pi*y)+pi.*sin(pi*x).*cos(pi*y);
u0_2= @(x,y) -2*(pi^2).*sin(pi*x).*sin(pi*y)+2*(pi^2).*cos(pi*x).*cos(pi*y);
mu0= @(x,y) sin(pi*x).*sin(pi*y);
mu0_1= @(x,y) pi.*cos(pi*x).*sin(pi*y)+pi.*sin(pi*x).*cos(pi*y);
mu0_2= @(x,y) -2*(pi^2).*sin(pi*x).*sin(pi*y)+2*(pi^2).*cos(pi*x).*cos(pi*y);
ureal= @(x,y,t) exp(t).*sin(pi*x).*sin(pi*y);
mureal= @(x,y,t) exp(t).*sin(pi*x).*sin(pi*y);
F_1= @(x,y,t)
exp(2*t).*(sin(pi*x).^2).*(sin(pi*y).^2)./(exp(t).*sin(pi*x).*sin(pi*y)+1)+(2*pi^2+1).*exp(t).*sin(pi*x
).*sin(pi*y)-(2*pi^2).*exp(t).*cos(pi*x).*cos(pi*y);
F_2= @(x,y,t)
exp(2*t).*(sin(pi*x).^2).*(sin(pi*y).^2)./(exp(t).*sin(pi*x).*sin(pi*y)+1)+(2*pi^2+1).*exp(t).*sin(pi*x
).*sin(pi*y)-(2*pi^2).*exp(t).*cos(pi*x).*cos(pi*y);
phi = @(r,c) (c.^2.+r.^2).^(1/2);
phi_r = @(r,c) ((c.*c+r.*r).^(-1/2));%部分
phi_r1=@(r,c) ((c.*c+r.*r).^(-1/2));
phi_r2=@(r,c) ((c.*c+r.*r).^(-3/2));

% 边界点
c1=zeros(nb/4,1);c2=linspace(0,1,nb/4)';
c3=ones(nb/4,1);c4=linspace(0,1,nb/4)';
r1=linspace(0,1,nb/4)';r2=zeros(nb/4,1);
r3=linspace(0,1,nb/4)';r4=ones(nb/4,1);
xb=[c1;c2;c3;c4];
yb=[r1;r2;r3;r4];

%内部点
nig=sqrt(ni);
xi1=linspace(0,1,nig);yi11=linspace(0,1,nig)';%yi12=linspace(1,0,nt)';
xi2=repmat(xi1,nig,1);yi2=repmat(yi11,1,nig);
xi=reshape(xi2,[ni,1]);yi=reshape(yi2,[ni,1]);

%中心点
xc=[xb;xi];yc=[yb;yi];

DM1=DistanceMatrix([xi yi],[xc yc]);% 内点与中心点的距离
DM2=DistanceMatrix([xb yb],[xc yc]);% 边界点与中心点的距离
DM3=DistanceMatrix([xc yc],[xc yc]);% 测试点与中心点的距离

m11=repmat(xi,1,nc);m12=repmat(xc',ni,1);
m21=repmat(yi,1,nc);m22=repmat(yc',ni,1);
m31=repmat(xc,1,nc);m32=repmat(xc',nc,1);
```

```
m41=repmat(yc,1,nc);m42=repmat(yc',nc,1);
m51=repmat(xb,1,nc);m52=repmat(xc',nb,1);
m61=repmat(yb,1,nc);m62=repmat(yc',nb,1);
mi1=m11-m12;mi2=m21-m22;
mc1=m31-m32;mc2=m41-m42;
mb1=m51-m52;mb2=m61-m62;

Phi=phi(DM1,c);
Phi_rr=(2.*phi_r1(DM1,c)-(mi1.^2).*phi_r2(DM1,c)-(mi2.^2).*phi_r2(DM1,c)-
2*(mi1.*mi2).*phi_r2(DM1,c));
un=zeros(nc,10000);%解的矩阵
mun=zeros(nc,10000);
un(:,1)= u0(xc,yc);
uni(:,1)= un((nb+1:nc),1);
mun(:,1)= mu0(xc,yc);
Ab=phi(DM2,c);              Ob=zeros(nb,nc); Obv=zeros(nb,1);
A1=2*phi(DM1,c)-tau*Phi_rr;                  A2=tau*(uni(:,1)./(uni(:,1)+1)).*phi(DM1,c);
A3=zeros(ni,nc);                             A4=(2+tau*(uni(:,1)./(uni(:,1)+1))).*phi(DM1,c)-
tau*Phi_rr;
B1v=2*u0(xi,yi)+tau*u0_2(xi,yi);             B2v=-tau*(uni(:,1)./(uni(:,1)+1)).*mu0(xi,yi);
B3v=zeros(ni,1);                             B4v=(-
tau*(uni(:,1)./(uni(:,1)+1))+2).*mu0(xi,yi)+tau*mu0_2(xi,yi);
t_0=0;
t_1=tau;
F1=(F_1(xi,yi,t_0)+F_1(xi,yi,t_1))/2;
F2=(F_2(xi,yi,t_0)+F_2(xi,yi,t_1))/2;
bn=[zeros(nb,1);zeros(nb,1);2*tau*F1(:,1);2*tau*F2(:,1)];
A=[Ab Ob;Ob Ab;A1 A2;A3 A4];%A
tol=det(A);
Bv=[Obv;Obv;B1v+B2v;B3v+B4v];%B0
v=zeros(2*nc,5000);
v(:,2)=pinv(A)*(Bv+bn);%求伪逆效果更好  得到 v1
u_0=u0(xc,yc);
u1=phi(DM3,c)*v((1:nc),2);
un(:,2)= phi(DM3,c)*v((1:nc),2);
uni(:,2)= un(nb+1:nc,2);
mun(:,2)= phi(DM3,c)*v((nc+1:2*nc),2);


tend=1;
for i=2:floor(tend/tau)

Ab=phi(DM2,c);              Ob=zeros(nb,nc);
```

```
A1=2*phi(DM1,c)-tau*Phi_rr;                    A2=tau*(uni(:,i)./(uni(:,i)+1)).*phi(DM1,c);
A3=zeros(ni,nc);                               A4=(2+tau*(uni(:,i)./(uni(:,i)+1))).*phi(DM1,c)-
tau*Phi_rr;
B1=2*phi(DM1,c)+tau*Phi_rr;                    B2=-tau*(uni(:,i)./(uni(:,i)+1)).*phi(DM1,c);
B3=zeros(ni,nc);                               B4=(-
tau*(uni(:,i)./(uni(:,i)+1))+2).*phi(DM1,c)+tau*Phi_rr;
A=[Ab Ob;Ob Ab;A1 A2;A3 A4];
B=[Ob Ob;Ob Ob;B1 B2;B3 B4];

t_n=tau*(i-1);
t_n1=tau*i;
F1=(F_1(xi,yi,t_n)+F_1(xi,yi,t_n1))/2;
F2=(F_2(xi,yi,t_n)+F_2(xi,yi,t_n1))/2;
bn=[zeros(nb,1);zeros(nb,1);2*tau*F1(:,1);2*tau*F2(:,1)];
v(:,i+1)=pinv(A)*(B*v(:,i)+bn);
un(:,i+1)=phi(DM3,c)*v((1:nc),i+1);
mun(:,i+1)=phi(DM3,c)*v((nc+1:2*nc),i+1);
uni(:,i+1)=un(nb+1:nc,i+1);
end

unreal= zeros(nc,10000);
munreal= zeros(nc,10000);
for i=1:floor(tend/tau)
unreal(:,i)= ureal(xc,yc,tau*(i-1));
munreal(:,i)= mureal(xc,yc,tau*(i-1));
end

E_1= zeros(floor(tend/tau));
E_2= zeros(floor(tend/tau));
for i=2:floor(tend/tau)
E_1(i)= norm(un(:,i)-unreal(:,i),inf);
E_2(i)= norm(mun(:,i)-munreal(:,i),inf);
end

fprintf('Eu1 = %8.3e\n', E_1(2));
fprintf('Eu2 = %8.3e\n', E_2(2));
fprintf('Eu1 = %8.3e\n', E_1(tend/tau));
fprintf('Eu2 = %8.3e\n', E_2(tend/tau));
Eu_1=norm(E_1(1:floor(tend/tau)),inf);
fprintf('Eu = %8.3e\n', Eu_1);
Eu_2=norm(E_2(1:floor(tend/tau)),inf);
fprintf('Emu = %8.3e\n', Eu_2);
Ec=norm((un(:,(floor(tend/tau)))- un(:,(floor(tend/tau)-1))),inf)/norm(un(:,(floor(tend/tau))),inf);
```

```
fprintf('Ec = %8.3e\n', Ec);

[X,Y,Z]=griddata(xc,yc, un(:,2),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('u1');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc, mun(:,2),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('mu1');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc, unreal(:,2),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('ureal1');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc, munreal(:,2),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('mureal1');
hold on
plot(xc,yc,'.b','MarkerSize',4);
```

```
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc,
un(:,(floor(tend/tau))),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('un');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc,
mun(:,(floor(tend/tau))),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('mun');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc,
unreal(:,(floor(tend/tau))),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
xlabel('x1');
ylabel('x2');
zlabel('unreal');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

[X,Y,Z]=griddata(xc,yc,
munreal(:,(floor(tend/tau))),linspace(min(xc),max(xc))',linspace(min(yc),max(yc)),'v4');
surf(X,Y,Z);
shading interp;
```

```
xlabel('x1');
ylabel('x2');
zlabel('munreal');
hold on
plot(xc,yc,'.b','MarkerSize',4);
colormap jet
figure;

N= linspace(1,100,100)';
plot(N,E_1);
figure;
plot(N,E_2)
```

致　谢

首先，十分感谢指导教师李新祥老师为我完成本片论文作出的巨大贡献。从课程选题，到相关文献查找，到完成设计项目，再到最后的论文写作，李新祥老师都给了我很多指导和帮助。非常感谢，老师牺牲业余时间为我解答文献中的难点，帮我寻找程序设计的缺陷。另外，我也非常感谢上海大学理学院数学系的各位指导老师，帮助我在上海大学的这四年学习过程中积累了知识与能力。同时，我也感谢我的父母，在情感与经济上一直支撑着我，使我能够完成大学的学业。最后我要感谢我的好友们，在我焦虑烦闷的时候陪伴我，帮助我。

俞梦泽

上海大学

2024 年 5 月 10 日


上海大学
SHANGHAI UNIVERSITY