國立臺灣大學理學院應用數學科學研究所

碩士論文

Institute of Applied Mathematical Sciences

College of Science

National Taiwan University

Master Thesis

一種用於三維不可壓縮 Navier-Stokes 方程潛在爆破解的網格細化方法

A Refinement Method for Potential Blowup Solutions to the 3D incompressible Navier-Stokes equations

吳家豪

Jia Hao Wu

指導教授: 阮文先 博士

Advisor: Van Tien Nguyen Ph.D.

中華民國 113 年 7 月

July, 2024

doi:10.6342/NTU202402879

國立臺灣大學碩士學位論文
口試委員會審定書

MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

一種用於三維不可壓縮 Navier Stokes 方程潛在爆破解的網格細化方法

A Refinement Method for Potential Blowup Solutions to the 3D incompressible Navier-Stokes equations

本論文係___吳家豪__(姓名)___R11246010_（學號）在國立臺灣大學____應數____(系/所/學位學程)完成之碩士學位論文，於民國 113 年_7_月_22_日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department / Graduate Institute of ____Applied Mathematics____ on __22__(date)__July__ (month)__2024__ (year) have examined a Master's Thesis entitled above presented by_____Jia-Hao Wu_____(name)_____R11246010_____(student ID) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

Van Tien Nguyen　陳信全　沈俊嚴

（指導教授 Advisor）

_____　_____　_____

系（所、學位學程）主管 Director: _____

doi:10.6342/NTU202402879

Acknowledgements

首先，我想要感謝我自己，感謝自己在這段研究過程中所付出的努力與堅持。完成這個研究課題對我來說是一個重大的里程碑，程式從零開始到最後完成時有的一千多行，這是一開始的我完全無法想像的，無數次的 bug 以及靠著自己一行一行慢慢檢查，至今忘不了每次找出程式錯誤的部分並修改正確的那份喜悅。

其次，我要感謝我的父母和家人。你們無條件的支持和鼓勵，讓我能夠安心地專注於學業。沒有你們的支持，我無法在這條學術道路上走到這裡。

最後，我要感謝在台大認識的每一位同學和老師。你們的幫助和指導對我的成長至關重要。特別感謝我的指導教授，謝謝您一年的悉心指導和耐心教誨，您的專業知識和鼓勵讓我得以完成這項研究。另外，那些在課業上給予我寶貴意見和建議的老師們，以及那些與我一同努力的同學們，感謝你們的陪伴和支持。

doi:10.6342/NTU202402879

doi:10.6342/NTU202402879

摘要

不可壓縮 Navier-Stokes 方程是否能夠從平滑的初始數據展現出有限時間奇異行爲，是非線性偏微分方程中的一個具有挑戰性的問題之一。本文提出了一些數值證據，針對 Thomas Y. Hou 使用的平滑初始數據的軸對稱 Navier-Stokes 方程進行數值模擬，得到其可能在原點發展出潛在的奇異行爲。我們的方法遵循了 Berger 和 Kohn 的結果，這套方法是基於方程的尺度不變性。與 Thomas Y. Hou 的情況不同，我們固定黏性項進行數值模擬，並且最後呈現不同的結果。由於這種基於方程自身性質的方法，它可以在迭代過程中保持結構。

關鍵字：網格細化方法、尺度不變性、軸對稱 Navier-Stokes 方程、潛在爆破

doi:10.6342/NTU202402879

doi:10.6342/NTU202402879

Abstract

Whether the 3D incompressible Navier-Stokes equation can exhibit finite-time singularity from smooth initial data is one of the challenging problems in nonlinear PDEs. In this paper, we present numerical evidence that the axisymmetric Navier-Stokes equations, with Thomas Y. Hou's smooth initial data, can develop potential singular behavior at the origin. Our method follows the approach of Berger and Kohn's study, which is based on the scaling invariance of the equation. Different from Thomas Y. Hou's situation, we fix the viscosity and then obtain some different results. Due to the method's reliance on the self-similarity properties of the equation, it can preserve the structure throughout the iteration process.

Keywords: refinemenft method, scaling invariance, axisymmetry Navier-Stokes equations, potentially blow-up

doi:10.6342/NTU202402879

doi:10.6342/NTU202402879

Contents

Page

Verification Letter from the Oral Examination Committee **i**

Acknowledgements **iii**

摘要 **v**

Abstract **vii**

Contents **ix**

Chapter 1 Introduction **1**

Chapter 2 Preliminaries **7**

Chapter 3 A Refinement Method **11**

3.1 Refinement Process in One Dimension 12

3.2 Extend the coarser region 20

3.3 Application 1: the Energy-Subcritical Semilinear Heat Equation . . . 23

3.3.1 Convergence of rescaled solutions 24

3.3.2 One Dimensional Case 27

3.3.3 Two Dimensional Case 33

3.4 Application 2: A random path 41

doi:10.6342/NTU202402879

Chapter 4 Numerical Results of 3D Navier-Stokes equations **45**

Chapter 5 Conclusion **53**

References **55**

doi:10.6342/NTU202402879

Chapter 1 Introduction

In the realm of fluid dynamics, the three-dimensional (3D) incompressible Navier-Stokes equations serve as the foundational mathematical framework dictating the behavior of viscous, incompressible fluid movements. Stemming from fundamental physical principles and predicated on a linear correlation between stress and the rate of strain within the fluid, their relevance to practical scenarios remains uncontested [13]. For example, they have been used to model ocean currents, weather patterns, and other fluid-related phenomena. The question about the global regularity of the 3D Navier-Stokes equations, given that the initial data is smooth and of finite energy, is one of the most important and famous fundamental questions in nonlinear partial differential equations. Moreover, among the seven Millennium Problems posted by the Clay Mathematics Institute, the Navier-Stokes equation is one of them [6]. The main difficulty in this problem is that the nonlinearity due to vortex stretching is super-critical.

In this paper, we focus on the 3D incompressible axisymmetric Navier-Stokes equations and present numerical evidence that they seem to evolve potentially singular solutions at the origin with a smooth initial data of finite energy. Specifically, we observe that the maximum value increases and the maximum location moves toward the origin over time. However, in such blow-up problems, we should implement an adaptive mesh method to describe the location of the singularity in more detail. Our method is a re-

doi:10.6342/NTU202402879

finement based on Berger and Kohn's results [2], with some adjustments. In the refining process, the key point comes from the scaling invariance of the equations. We only refine the region of interest.

In this paper, the problem's conditions are the same as in Hou's result [11]. Under a periodic cylindrical domain, we numerically solve the 3D axisymmetric Navier-Stokes equations. For the boundary condition, we impose a no-slip no-flow condition at $r = 1$. On the other hand, we use a periodic boundary condition along the $z-$axis with period 1. Suppose that u_θ , ω_θ, and ψ_θ be the angular components of the velocity, the vorticity, and the vector stream function, respectively. In [12], Hou and Li established the following variables from the u_θ, ω_θ, and ψ_θ:

$$\tilde{u} = \frac{u_\theta}{r},\ \tilde{\omega} = \frac{\omega_\theta}{r},\ \tilde{\psi} = \frac{\psi_\theta}{r},$$

and, moreover, converted the 3D incompressible Navier-Stokes equations into the following form:

$$\begin{cases} \tilde{u}_t + u^r \tilde{u}_r + u^z \tilde{u}_z & = 2\tilde{u}\tilde{\psi}_z + \nu\left(\tilde{u}_{rr} + \frac{3}{r}\tilde{u}_r + \tilde{u}_{zz}\right), \\ \tilde{\omega}_t + u^r \tilde{\omega}_r + u^z \tilde{\omega}_z & = 2\tilde{u}\tilde{u}_z + \nu\left(\tilde{\omega}_{rr} + \frac{3}{r}\tilde{\omega}_r + \tilde{\omega}_{zz}\right), \\ -\left(\partial_{rr} + \frac{3}{r}\partial r + \partial_{zz}\right)\tilde{\psi} & = \tilde{\omega}, \end{cases} \tag{1}$$

where $u^r = -r\tilde{\psi}_z$, $u^z = 2\tilde{\psi} + r\tilde{\psi}_r$, and ν is a constant viscosity. If we directly simulate (1) and plot the results using only finite difference methods, it seems to develop potentially singular solutions at the origin. The initial data we used is the same as in Hou's study:

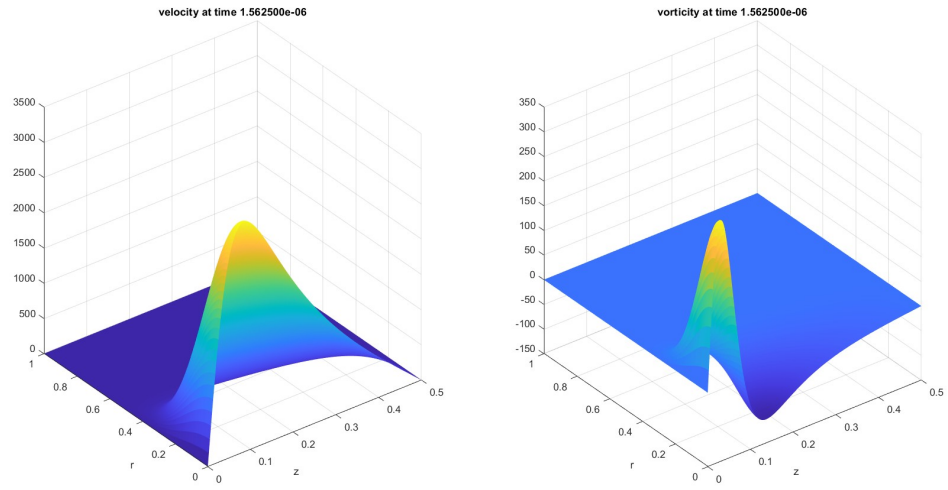
$$\tilde{u}_0(z,r) = g(r)\,h(z),\ \tilde{\omega}_0(z,r) = 0,$$

doi:10.6342/NTU202402879

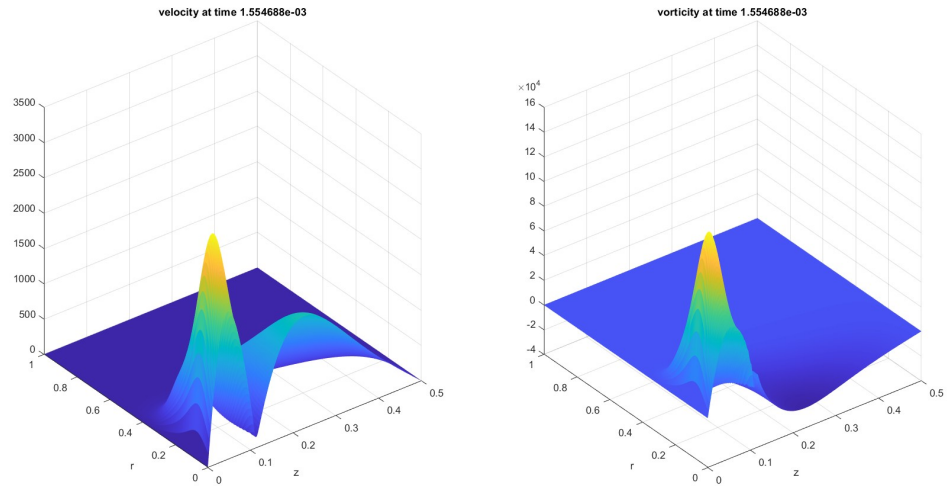
where

$$h(z) = \frac{\sin(2\pi z)}{1+12.5(\sin(\pi z))^2},$$
$$g(r) = 12000\left(1-r^2\right)^{18}.$$

In Fig.1.2, we present the smooth initial data of finite energy along with the results after a short time. Additionally, in Figs.1.4, we plot the maximum locations of $\tilde{u}$ and $\tilde{\omega}$ after a short period. It appears that they may develop potentially singular solutions at the origin.



((a)) After solving once, the profiles of $\tilde{u}$ and $\tilde{\omega}$.

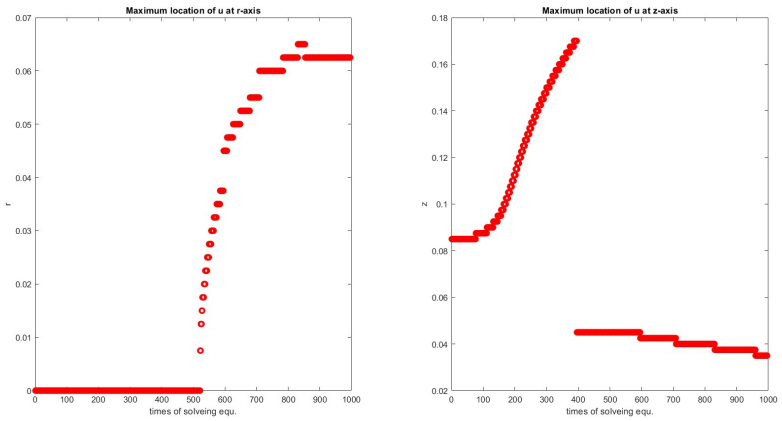


((b)) At a short time, the profiles of $\tilde{u}$ and $\tilde{\omega}$

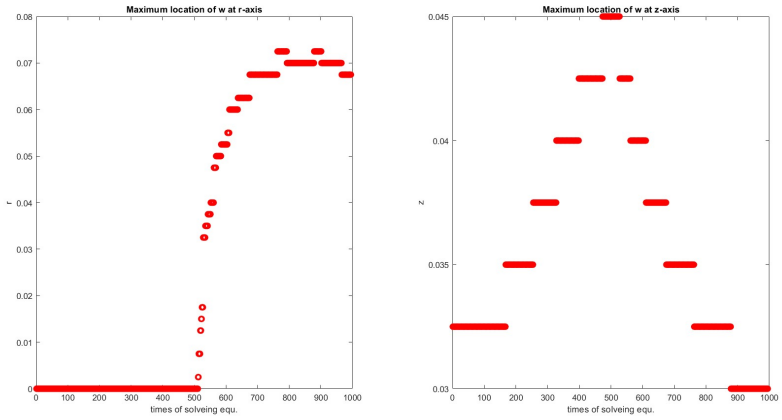
Figure 1.2: The evolution of profiles of $\tilde{u}$ and $\tilde{\omega}$ with initial data $\tilde{u}_0(z,r) = g(r)\,h(z)$, $\tilde{\omega}_0(z,r) = 0$

We employ a second-order finite difference method to discretize the spatial deriva-

doi:10.6342/NTU202402879



((a)) In first case, maximum locations r (left) , z (right) of $\tilde{u}$ w.r.t the numbers of solving equations of the first case.



((b)) In first case, maximum locations r (left) , z (right) of $\tilde{\omega}$ w.r.t the numbers of solving equations of the first case.

Figure 1.4: In the computing process, we record the maximum locations of $\tilde{u}$, $\tilde{\omega}$ and plot z, r separately corresponding to time. The left-hand side (LHS) pertains to the result of $\tilde{u}$, while the right-hand side (RHS) pertains to the result of $\tilde{\omega}$.

doi:10.6342/NTU202402879

tives and a first-order forward method for time discretization. An adaptive time-step size is utilized based on the standard stability constraint for time-stepping. Eventually, after several refinement steps, the smallest time-step size reaches orders of $O(10^{-13})$. The overall method achieves second-order accuracy.

doi:10.6342/NTU202402879

doi:10.6342/NTU202402879

Chapter 2 Preliminaries

In this section, we introduce the 3D axisymmetric incompressible Navier-Stokes equations along with the boundary conditions, which constitute the main focus of this paper. First, we define some related variables. Let $\mathbf{u}(x,y,z) \in \mathbb{R}^3$ represent the velocity field, and define $\omega = \nabla \times \mathbf{u}$ as the 3D vorticity vector. To establish the 3D axisymmetric Navier-Stokes equations, we decompose the radially symmetric velocity field as follows:

$$\mathbf{u}(t,z,r) = u^r(t,z,r)\, e_r + u^\theta(t,z,r)\, e_\theta + u^z(t,z,r)\, e_z,$$

where $e_r,\ e_\theta,\ e_z$ are unit vectors in cylindrical coordinates and defined by

$$e_r = \frac{1}{r}(x,y,0)^T,\ e_\theta = \frac{1}{r}(-y,x,0)^T,\ e_z = (0,0,1)^T.$$

Due to this definition, we can express the vorticity in cylindrical coordinates as follows:

$$\omega(t,z,r) = -\left(u^\theta(t,z,r)\right)_z e_r + \omega^\theta(t,z,r)\, e_\theta + \frac{1}{r}\left(r u^\theta(t,z,r)\right)_r e_z.$$

Let ψ be the stream function, where ψ^θ be the angular part. Through a change of variables, we define:

$$\tilde{u} = \frac{u^\theta}{r},\ \tilde{\omega} = \frac{\omega^\theta}{r},\ \tilde{\psi} = \frac{\psi^\theta}{r},$$

doi:10.6342/NTU202402879

Hou and Li in [12] derived the following equivalent axisymmetric Navier-Stokes equations:

$$\begin{cases} \tilde{u}_t + u^r \tilde{u}_r + u^z \tilde{u}_z & = 2\tilde{u}\tilde{\psi}_z + \nu\left(\tilde{u}_{rr} + \frac{3}{r}\tilde{u}_r + \tilde{u}_{zz}\right), \\ \tilde{\omega}_t + u^r \tilde{\omega}_r + u^z \tilde{\omega}_z & = 2\tilde{u}\tilde{u}_z + \nu\left(\tilde{\omega}_{rr} + \frac{3}{r}\tilde{\omega}_r + \tilde{\omega}_{zz}\right), \\ -\left(\partial_{rr} + \frac{3}{r}\partial r + \partial_{zz}\right)\tilde{\psi} & = \tilde{\omega}, \\ u^r = -r\tilde{\psi}_z,\ u^z = 2\tilde{\psi} + r\tilde{\psi}_r \end{cases} \tag{2}$$

One of the benefits of this revised form is that it has eliminated the singularity $\frac{1}{r}$ from the cylindrical coordinates.

Next, we set the initial conditions. Our smooth initial condition is the same as in Hou's study, which is given below:

$$\tilde{u}_0(z,r) = g(r)\,h(z),\ \tilde{\omega}_0(z,r) = 0,$$

where

$$\begin{aligned} h(z) &= \tfrac{\sin(2\pi z)}{1+12.5(\sin(\pi z))^2}, \\ g(r) &= 12000\,(1-r^2)^{18}. \end{aligned}$$

The main feature of this case is that the initial maximum location is at $r = 0$. Therefore, in the 3D dynamics, it initially moves away from $r = 0$ and then moves toward the origin. Upon observing the initial profile, its mass is concentrated near the origin. Moreover, the flow is initially entirely driven by a large swirl. The other two velocity components are initially set to zero. If we examine whether the solution exhibits blow-up behavior in the very early stages, these initial data do not seem to lead to potentially singular behavior. In particular, the maximum of $\tilde{u}$ actually decreases in the initial stages. However, after a short time, the profiles dynamically evolve beneficial structures, and we observe a strong nonlinear alignment of vortex stretching. Beyond this initial stage, we observe that the

doi:10.6342/NTU202402879

maximum vorticity rapidly increases throughout the computation.

Last but not least, following Hou's result, we impose similar boundary conditions. We apply a periodic boundary condition in the $z-$axis with period 1 and the odd symmetry. Given that u^θ, ω^θ, ψ^θ are odd functions of r, $\tilde{u}$, $\tilde{\omega}$, $\tilde{\psi}$ are even functions. Thus, we impose the following conditions at the $r = 0$:

$$\tilde{u}_r(t, z, 0) = \tilde{\omega}_r(t, z, 0) = \tilde{\psi}_r(t, z, 0) = 0.$$

On the solid boundary at $r = 1$, we enforce a no-slip no-flow boundary condition. Specifically, the no-flow boundary condition is given by

$$\tilde{\psi}(t, z, 1) = 0,$$

and the no-slip boundary condition is given by $u^\theta(t, 1, z) = u^z(t, 1, z) = 0$ for all z. From equations (2.1d) and (2.4), this can further imply $\tilde{\psi}_r(t, 1, z) = 0$. Therefore, under these new variables $\tilde{u}$, $\tilde{\omega}$, $\tilde{\psi}$ the no-slip boundary condition translates to:

$$\tilde{u}(t, z, 1) = 0,\ \tilde{\omega}(t, z, 1) = -\tilde{\psi}_{rr}(t, z, 1).$$

Moreover, through discretizing $\tilde{\omega}(t, z, 1) = -\tilde{\psi}_{rr}(t, z, 1)$ and imposing $\tilde{\psi}_r(t, 1, z) = 0$, we enforce the no-slip boundary condition for $\tilde{\omega}$ as a vorticity boundary condition. Lastly, choosing a suitable simulation domain involves considering the periodicity and odd symmetry of the solution. Therefore, we only need to consider equations (2.1) in the half-period domain:

$$\mathbf{D} = \{(z, r) : 0 \le r \le 1,\ 0 \le z \le 0.5\},$$

doi:10.6342/NTU202402879

Furthermore, u^r and u^z satisfy the following conditions

$$u^r = -r\tilde{\psi}_z = 0 \text{ on } r = 0, 1 \text{ and } u^z = 2\tilde{\psi} + r\tilde{\psi}_r = 0 \text{ on } z = 0, 1/2.$$

Thus, under these conditions, we can envision the boundaries of **D** acting like 'solid barriers' or 'impermeable boundaries'.

To numerically capture potential singularity profiles of the equations (2), we employ a numerical method inspired by Burger and Kohn's result [2]. In the next section, we delve into this numerical process in greater detail. This method can indeed be generalized for other equations exhibiting scaling invariance. Additionally, we explore various applications of this method in the subsequent section.

doi:10.6342/NTU202402879

Chapter 3 A Refinement Method

In addressing the numerical challenge of blow-up problems, there are two main types of commonly used mesh adaptation techniques. One approach, akin to the Berger-Kohn algorithm [2], involves mesh refinement. Here, grid points are added as the mesh size decreases, aligning with the principle of rescaled invariance. In contrast, the other approach involves mesh movement [3, 5]. This method adjusts a fixed number of grid points to concentrate finer mesh sizes near potential blow-up points. In our focus on mesh refinement in this paper, different refinement processes are explored. For instance, in [1, 9], adaptive time step generation is guided by L^2-norm or $L^\infty-$norm,considerations, respectively. Alternatively, in [4], a time-adaptive mesh is determined using a posteriori error estimation.

Unlike the aforementioned approaches, our adaptive time-step method is directly derived from the rescaled invariance described in the Berger-Kohn algorithm [2]. While the previous results primarily concentrate on refining the temporal mesh, we also incorporate a space-adaptive mesh approach based on scaling invariance. Despite the computational challenges posed by increasing values of u, this method enables a more detailed and specific characterization of the solution profile.

doi:10.6342/NTU202402879

3.1 Refinement Process in One Dimension

In this section, we delve into the refinement process in greater detail. The fundamental concept is rooted in the scaling invariance of (2), akin to the Berger-Kohn algorithm [2]. The refinement method is crucial as it provides a meaningful approach to determining adaptively-defined temporal mesh sizes. By leveraging scaling invariance, this method preserves the underlying structure.

As mentioned earlier, the foundation of our method relies on the following observation: if $\tilde{u}(t,z,r)$, $\tilde{\omega}(t,z,r)$, $\tilde{\psi}(t,z,r)$ are the solutions to (2), then so are

$$\begin{cases} \tilde{u}_\lambda(s,k,l) &= \lambda^2 \tilde{u}(\lambda^2 s, \lambda k, \lambda l), \\ \tilde{\omega}_\lambda(s,k,l) &= \lambda^3 \tilde{\omega}(\lambda^2 s, \lambda k, \lambda l), \\ \tilde{\psi}_\lambda(s,k,l) &= \lambda \tilde{\psi}(\lambda^2 s, \lambda k, \lambda l), \end{cases} \tag{3}$$

for any $\lambda > 0$ with $t = \lambda^2 s,\ z = \lambda k,\ r = \lambda l$. This formula can be verified by substituting (3) into (2) directly.

Following the Burger-Kohn algorithm, we control the rescaled solutions $\tilde{u}_\lambda$, $\tilde{\omega}_\lambda$, $\tilde{\psi}_\lambda$ by factors of $\lambda^2,\ \lambda^3,\ \lambda$, respectively. This means choosing a small λ when $\tilde{u},\ \tilde{\omega},\ \tilde{\psi}$ have large values. Subsequently, we adjust the coefficients $\lambda^2,\ \lambda,\ \lambda$ corresponding to $t,\ z,\ r$ to refine the spatial and temporal mesh sizes, respectively. This refinement process is crucial. Specifically, among the three equations in (3), we focus on refining the mesh based on $\tilde{\omega}$. The scaling parameter for $\tilde{\omega}$ is theoretically the largest, allowing it to span a broader range during growth. Consequently, fewer iterations are needed to reach higher values, as will be discussed later. However, due to rapid initial growth, numerical control measures are necessary. Conversely, $\tilde{u}$ initially exhibits a decreasing trend.

doi:10.6342/NTU202402879

Next, we introduce the refinement method based on $\tilde{\omega}$ to generate a new mesh. For convenience, we consider a one-dimensional spatial domain, taking $\tilde{\omega}_\lambda(s,k) = \lambda^3\tilde{\omega}(\lambda^2 s, \lambda k)$ as an example. For the two-dimensional case, the same process is applied to the second axis. After refining the mesh based on $\tilde{\omega}$, we will redefine $\tilde{u}$ and $\tilde{\psi}$ on this new mesh. This method has an iterative structure, so in k-iterations, there are k different mesh sizes and rescaled solution data $\tilde{\omega}_k$ corresponding to the rescaling parameters $p = \lambda, \lambda^2, \ldots, \lambda^k$, where λ is fixed initially. To avoid excessively high computational costs, we only refine the region near the mass center. Specifically, the region where $\tilde{\omega}_k$ exceeds a certain threshold will be refined. Therefore, besides λ, other parameters must be set. In conclusion, we need to set the following parameters:

$$\lambda = \text{ the scaling parameter and } \lambda < 1$$

$$M_0 = \text{ the initial threshold before first refinement}$$

In general, we should set suitable conditions for the region we want to refine. Following Berger and Kohn [2], we primarily refine the region where $\{\alpha M \leq \tilde{\omega} \leq M\}$ for some initially fixed positive $\alpha < 1$ due to the increasing value in blow-up problems. For the 3D axisymmetric Navier-Stokes equations, we use the same setting for refinement. To meaningfully determine the criterion, we define M_0 based on the initial data $\tilde{\omega}_0$. due to the initial condition $\tilde{\omega} = 0$, we set this threshold by solving for $\tilde{\omega}$ once or twice to obtain the value. For convenience, we use $\tilde{\omega}_0$ to represent

$$M_0 = \lambda^a \left|\left|\tilde{\omega}_0\right|\right|_{L^\infty}, \tag{4}$$

doi:10.6342/NTU202402879

for some $a > 0$. For the convenience of the discussion, we define the following symbols:

$\tilde{\omega}_{k_{\text{entire}}}$: the entire solution with k different mesh size

$\tilde{\omega}_k$: the k-th rescaled solution

$Z_{k_{\text{entire}}}$: the entire grid points of z with k different mesh size

Z_k : the k-th refined grid points of z

M_k : the k-th maximum value before refinement.

d_k : the k-th spatial mesh size

dt_k : the k-th temporal mesh size.

Note that $k = 0$ means that we have not done any refinement in the process.

At the first, we directly update the solution from the initial data using the Euler finite difference scheme. However, we need to ensure that dt_0 is sufficiently small. In most cases, the blowup time T is quite small. As we start updating the solution data, after some time steps, we will encounter:

$$||\tilde{\omega}(t_{0_m}, \cdot, \cdot)||_{L^\infty} > M_0$$

where t_{n_m} means time τ_n plus m time step dt_n, i.e. $t_{n_m} := \tau_n + mdt_n$. The value τ_n is the time when the maximum value of $\tilde{\omega}$ equals to the previous threshold M_{n-1}, with $\tau_0 = 0$. Hence, when $\tilde{\omega} > M_0$, we need to use interpolation to find τ_1 such that:

$$||\tilde{\omega}(\tau_1, \cdot, \cdot)||_{L^\infty} = M_0 \text{ with } (m-1)\, dt_0 \leq \tau_1 \leq mdt_0.$$

doi:10.6342/NTU202402879

At this time, we also perform interpolation on $\tilde{u}$ and $\tilde{\psi}$ sto find their values at time τ_1. The next remark outlines the necessary steps in the program. This is crucial because, in computations, if the interval size is too small, we may not perform the interpolation within it.

Remark 1. *Suppose in the n-th process the maximum value is greater than M_n at time t_{n_m} with step size dt_n. Before the $(n+1)$-th refining process, we initially perform the translation to apply the linear interpolation on the time interval $[0, dt_n]$ and then preform the translation again to find the suitable time τ_n in the interval $[(m-1)\, dt_n, mdt_n]$.*

Secondly, we begin the process of refining the mesh using $\alpha,\ M_0$ to initially determine the region that requires refinement. We denote $\mathcal{C}_{\text{NS}}$ as the set of grids that satisfy the condition awaiting refinement:

$$\mathcal{R}_{z_0} = \{z \in \mathcal{C}_{\text{NS}}\} := \left\{ z_{0_{refine1_1}}, \ldots, z_{0_{refine_{end}}} \right\}. \tag{6}$$

Based on the (3), if we set $s,\ k$ are the spatial and temporal parameters of $\tilde{\omega}$, respectively, then $t = \lambda^2 s,\ z = \lambda k$. From $\mathcal{R}_{z_0}$, we can deduce that k should be restricted on

$$\mathcal{D}_{z_1} = \left\{ \lambda^{-1} z_{0_{refine_1}}, \ldots, \lambda^{-1} z_{0_{refined_{end}}} \right\}.$$

To maintain accuracy and stability, we ensure that the mesh size in $\mathcal{D}_{z_1}$ remains consistent with the original setting, i.e. $\Delta s = \Delta t$, and $\Delta k = \Delta z$. Here's how we achieve this refinement:

1. Inserting New Grid Points: Initially, we have grid points $\{z_j\}$ used to determine $\tilde{\omega}$, but they determine the scaling solution only on the grid points with mesh size $\lambda^{-1} d_0$. Hence, the $\lambda^{-1} - 1$ new points should be inserted between each pair of the

doi:10.6342/NTU202402879

original points of $z-$axis.

2. Creating a Finer Domain: This insertion of new points results in a finer domain $\mathcal{D}'_{z_1} := \mathcal{D}_{z_1} \cup \{\text{new points}\}$.These new points reduce the mesh size on the $z-$axis from d_0 to $d_1 = \lambda d_0$.

3. Example with $\lambda = \frac{1}{4}$: If $\lambda = \frac{1}{4}$, then $4 - 1 = 3$ new points should be added to each interval of $z-$axis. This refinement decreases the mesh size on the Δz to $\frac{\Delta z}{4}$.

4. Choosing λ^{-1}: Initially, we choose λ^{-1} as a positive constant to control the level of refinement. This ensures that the refined mesh remains structured and aligned with the original grid spacing in z.

By following these steps, we effectively refine the mesh while maintaining the necessary accuracy and stability required for the numerical solution of the problem.

After completing the refinement process, we follow these steps to determine the solution data:

1. Interpolation at New Points: Perform interpolation to compute the solution data at the newly inserted grid points within $\mathcal{D}'_{z_1}$.

2. Constructing the rescaled solution $\tilde{\omega}_1$ on $\mathcal{D}'_{z_1}$: Obtain the first rescaled solution $\tilde{\omega}_1$ over the refined domain $\mathcal{D}'_{z_1}$.

doi:10.6342/NTU202402879

3. Mapping Back to Original Domain: Transform $\tilde{\omega}_1$ back to the original domain $\mathcal{R}_{z_0}$. Although the solution exists in $\mathcal{R}_{z_0}$, the mesh size is now $d_1 = \lambda d_0$.

4. Collecting Finer Grid Points: Define Z_1 as the grid points on the $z-$axis transformed from $\mathcal{D}'_{z_1}$. These points Z_1 represent the finer resolution grid compared to Z_0.

5. Entire Solution $\tilde{\omega}_{1_{\text{entire}}}$: The new solution $\tilde{\omega}_{1_{\text{entire}}}$ spans the entire domain $Z_{1_{\text{entire}}} = ((Z_0 \setminus \mathcal{R}_{z_0}) \cup Z_1)$. This domain is characterized by mesh sizes $d_0,\ d_1$.

6. Interpolation for $\tilde{u}$ and $\tilde{\psi}$: Ensure $\tilde{u}$ and $\tilde{\psi}$ are appropriately defined over $Z_{1_{\text{entire}}}$ using interpolation techniques.

By following these steps, we effectively refine the mesh and obtain the first rescaled solution $\tilde{\omega}_1$, ensuring that the solution data remains accurate and stable throughout the entire domain.

Before proceeding with updating of the data $\tilde{\omega}_{1_{\text{entire}}}$, we introduce the new parameters M_1 and dt_1

$$M_1 = \lambda^{-3} M_0 \tag{7}$$
$$dt_1 = \lambda^2 dt_0.$$

Next, we replace dt_0 as dt_1 and start the update from time τ_0. Once $\tilde{\omega}_{1_{\text{entire}}}$ reaches its maximum at M_1, we will begin the mesh refinement process again and obtain a new rescaled solution $\tilde{\omega}_2$. It is important to note that the maximum of $\tilde{\omega}_2$ will be maintained at M_0 due

doi:10.6342/NTU202402879

to the λ^3 factor in (4). Similarly, we employ linear interpolation to find τ_2 such that the maximum of $\tilde{\omega}_{1_{\text{entire}}}(\tau_2, \cdot) = M_1$. Then, use again interpolation to find the data of $\tilde{u}_{1_{\text{entire}}}$ and $\tilde{\psi}_{1_{\text{entire}}}$ at time τ_1.

On the other hand, we will encounter points where the mesh sizes on the left-hand side and right-hand side are different, which can be seen as a kind of boundary data issue. These cases require additional handling using interpolation to find auxiliary points. For instance, at the interface where the mesh size changes from d_0 to d_1, the solution data needs to be interpolated to ensure continuity and accuracy across the mesh boundaries. By introducing auxiliary points through interpolation, we can create a smooth transition between regions with different mesh sizes. This ensures that the numerical solution remains stable and accurate even at the boundaries where mesh refinement occurs.

Last, we continue the process outlined above, updating the data in the entire domain. Before the k-th iteration, we have k different mesh sizes in the whole domain with $d_n = \lambda^n d_0$, temporal mesh size with $dt_{k-1} = \lambda^{2(k-1)} dt_0$, the solution data in the entire domain $\tilde{\omega}_{k-1_{\text{entire}}}$, and the sequence of $k-1$ rescaled solutions $\tilde{\omega}_1, \tilde{\omega}_2, \ldots, \tilde{\omega}_{k-1}$. Note that the maximum criterion is $M_{k-1} = \lambda^{-3(k-1)} M_0$. As time progresses, when the maximum value of $\tilde{\omega}_{k-1_{\text{entire}}}$ reaches M_{k-1}, the k-th refinement method follows. Similarly, we employ linear interpolation to determine the time τ_k such that $||\tilde{\omega}_{k-1_{\text{entire}}}(\tau_k, \cdot)||_{L^\infty} = M_{k-1}$, then we also store the data of the rescaled solution data $\tilde{\omega}_{k-1}(\tau_{k-1}, \cdot)$ with $||\tilde{\omega}_{k-1}(\tau_k, \cdot)||_{L^\infty} = M_0$. After multiple iterations, the domain becomes increasingly refined, especially near the mass center. Consequently, we have a sequence of times:

$$\tau_0,\ \tau_1, \ldots,\ \tau_{k-1},\ \tau_k, \ldots$$

In this framework, τ_k is our approximated blow-up time. For any τ_k, it corresponds to the

doi:10.6342/NTU202402879

entire domain data such that:

$$||\tilde{\omega}_{k_{\text{entire}}}(\tau_k, \cdot)||_{L^\infty} = M_{k-1} = \lambda^{-3(k-1)} M_0$$

Given that $\lambda < 1$ and $M_0 > 0$ are positive and fixed constants, we find that $||\tilde{\omega}_{k_{\text{entire}}}(\tau_k, \cdot)||_{L^\infty} \to +\infty$ as $k \to +\infty$.

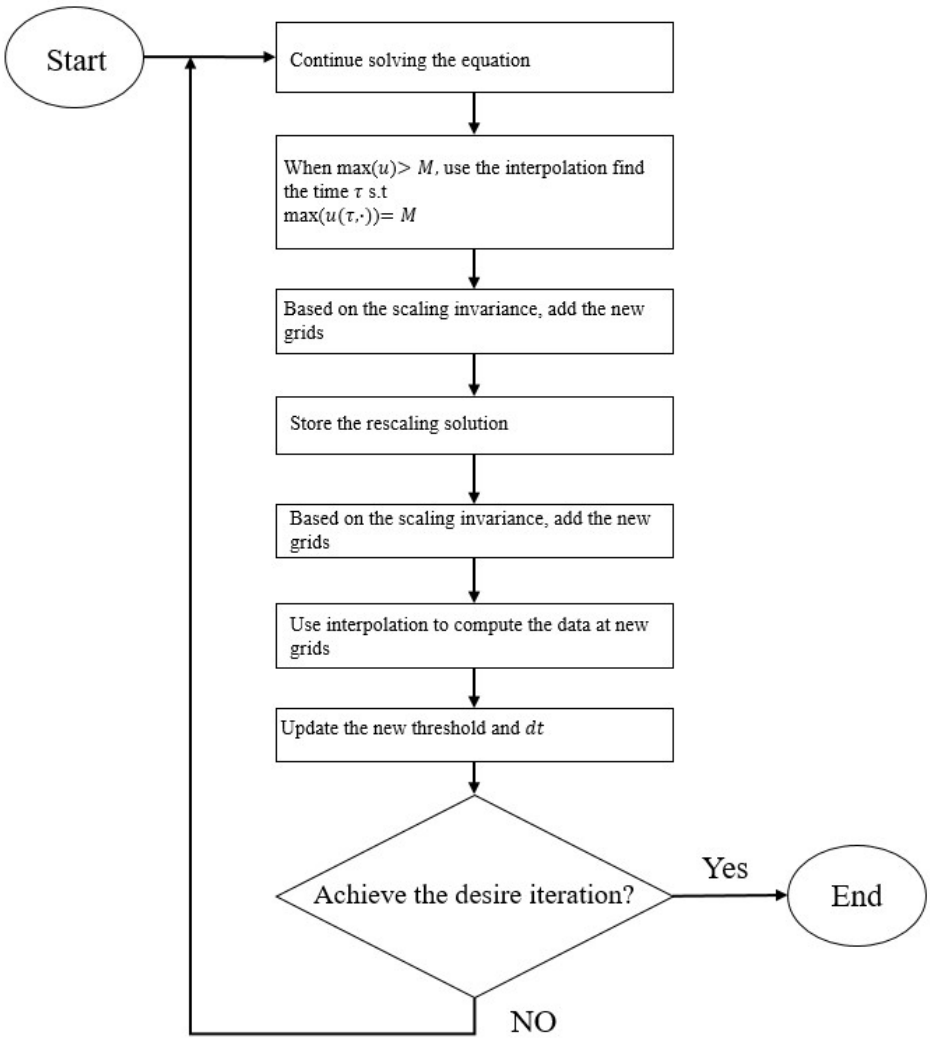


Figure 3.1: The diagram of the refinement process

doi:10.6342/NTU202402879

3.2 Extend the coarser region

When we operate the refining process, we may encounter a situation where different mesh sizes exist within the region we want to refine. To handle this, we first adjust these regions to have the minimum mesh size present. Then, we proceed to refine this newly adjusted region. However, to maintain the structure of the problem, we aim to arrange these refined meshes layer by layer. Specifically, the region with mesh size d_k should be positioned between the regions with mesh sizes d_{k-1} and d_{k+1}.

To achieve this, we design a suitable program. In the computational process, we store the boundary values of each refinement layer for z and r. After completing $n-$th refinement process, we have the latest boundary values $z_{\text{bdd}_{n1}}$, $z_{\text{bdd}_{n2}}$, $r_{\text{bdd}_{n1}}$, $r_{\text{bdd}_{n2}}$. These boundary values are stored in two new matrices:

$$Z_{\text{boundary}_n} = \begin{bmatrix} z_{\text{bdd}_{01}} & z_{\text{bdd}_{02}} \\ z_{\text{bdd}_{11}} & z_{\text{bdd}_{12}} \\ \vdots & \vdots \\ z_{\text{bdd}_{n1}} & z_{\text{bdd}_{n2}} \end{bmatrix}, \quad R_{\text{boundary}_n} = \begin{bmatrix} r_{\text{bdd}_{01}} & r_{\text{bdd}_{02}} \\ r_{\text{bdd}_{11}} & r_{\text{bdd}_{12}} \\ \vdots & \vdots \\ r_{\text{bdd}_{n1}} & r_{\text{bdd}_{n2}} \end{bmatrix},$$

These matrices store the boundary data for each layer of refinement, ensuring that the mesh sizes are arranged appropriately and the structure of the problem is maintained throughout the refinement process. At this stage, the previous $n-1$ rows contain old data. We now update them and check whether some regions should be extended. Starting from the $n-1$ row, we check if:

$$z_{\text{bdd}_{n-1,1}} > z_{\text{bdd}_{n1}}, \text{ and } z_{\text{bdd}_{n-1,2}} < z_{\text{bdd}_{n2}}.$$

doi:10.6342/NTU202402879

If this condition is true, it indicates the need to extend this region such that:

$$z_{\text{bdd}_{n-1,1}} < z_{\text{bdd}_{n1}}, \text{ and } z_{\text{bdd}_{n-1,2}} > z_{\text{bdd}_{n2}}.$$

Continuing this process, for the i-row, we check if

$$z_{\text{bdd}_{i,1}} > z_{\text{bdd}_{i+1,1}}, \text{ and } z_{\text{bdd}_{i,2}} < z_{\text{bdd}_{i+1,2}}.$$

If this condition is true, it indicates the need to extend the region such that:

$$z_{\text{bdd}_{i,1}} < z_{\text{bdd}_{i+1,1}}, \text{ and } z_{\text{bdd}_{i,2}} > z_{\text{bdd}_{i+1,2}}.$$

This iterative process ensures that each layer is appropriately extended to maintain the necessary structure and alignment with adjacent layers. By doing so, we achieve a refined and well-structured computational grid.

Certainly, here's a summary of the 2D programming process:

1. Set Parameters and Initial Data:Begin with suitable parameters and initial data $\tilde{u}_0$, $\tilde{\omega}_0$.

2. First Maximum Reached: When $||\tilde{\omega}_0||_{L^\infty} \geq M_0$, use interpolation to find the time τ_1 and determine $\tilde{u}_0$, $\tilde{\omega}_0$, $\tilde{\psi}_0$ at τ_1.

3. Start Refining Process: Initiate the refinement process.

4. After Refining: Use interpolation to combine the solution data into the entire solu-

doi:10.6342/NTU202402879

tion $\tilde{\omega}_{1_{\text{entire}}}$:

$$\tilde{\omega}_{1_{\text{entire}}} = \tilde{\omega}_{10_{\text{non-refined}}} \cup \tilde{\omega}_{11_{\text{refined}}}$$

where

(a) $\tilde{\omega}_{10_{\text{non-refined}}}$is defined on $\{(z,r) : \tilde{\omega}_0 \leq \alpha M_0\}$ with the mesh size d_0

(b) $\tilde{\omega}_{11_{\text{refined}}}$is defined on $\{(z,r) : \alpha M_0 \leq \tilde{\omega}_0 \leq M_0\}$ with the mesh size d_1.

so as $\tilde{u}_{1_{\text{entire}}}$ and $\tilde{\psi}_{1_{\text{entire}}}$.

5. Store Rescaled solution: Obtain the resacled solution $\tilde{\omega}_1$ and store.

6. Continue Updating: Update the entire solution data using the new temporal mesh size dt_1 and new threshold M_1, until $||\tilde{\omega}_{1_{\text{entire}}}||_{L^\infty} \geq M_1$.

7. Apply Interpolation: Use linear interpolation to find τ_2 such that

$$||\tilde{\omega}_{1_{\text{entire}}}(\tau_2, \cdot, \cdot)||_{L^\infty} = M_1$$

and find $\tilde{u}_{1_{\text{entire}}}$ and $\tilde{\psi}_{1_{\text{entire}}}$ at τ_2.

8. Refine the Region: Refine the region further and ensure that every finer region is layered appropriately.

doi:10.6342/NTU202402879

9. Iteration: At the n-th iteration, obtain the sequence of n rescaled solutions $\tilde{\omega}_1, \tilde{\omega}_2, \ldots, \tilde{\omega}_n$ and the entire solution data

$$\tilde{\omega}_n = \left(\bigcup_{i \in \{1,2,\ldots,n-1\}} \tilde{\omega}_{ni_{\text{no-refined}}} \right) \cup \tilde{\omega}_{ni_{\text{refined}}}$$

where

(a) $\tilde{\omega}_{ni_{\text{non-refined}}}$ is defined on $\{(z,r) : \alpha M_{n-1} \leq \tilde{\omega}_{n-1_{\text{entire}}} \leq \alpha M_n\}$.

(b) $\tilde{\omega}_{nn_{\text{refined}}}$ is defined on $\{(z,r) : \alpha M_{n-1} \leq \tilde{\omega}_{n-1_{\text{entire}}} \leq M_n\}$.

3.3 Application 1: the Energy-Subcritical Semilinear Heat Equation

Here, we are similar to Berger and Kohn's result and show the first two examples, the 1D and 2D energy-subcritical semilinear heat equation,

$$u_t = \Delta u + u^p, \text{ where } p = 5, \ x \in \mathbb{R}^d, \ d = 1, 2.$$

where the scaling invariance is

$$u_r(y,s) = r^{\frac{2}{p-1}} u\left(ry, r^2 s\right). \tag{8}$$

These two examples are different from the 3D Navier-Stokes equations due to the fixed blowup point at the origin. Firstly, we introduce a lemma about checking the semilinear heat equations. For this lemma, we revise Berger and Kohn's content by Herrero and

doi:10.6342/NTU202402879

Velazquez's result in [15][10]:

$$\lim_{t\to T} u\left(x\left(\left(T-t\right)\left|\log\left(T-t\right)\right|\right)^{\frac{1}{2}},t\right)\left(T-t\right)^{\frac{1}{p-1}}=f\left(x\right) \tag{9}$$

where

$$f\left(x\right)=\kappa\left(1+\frac{\left(p-1\right)x^{2}}{4p}\right)^{\frac{-1}{p-1}},\ \text{where}\ \kappa=\left(p-1\right)^{\frac{-1}{p-1}}. \tag{10}$$

This behavior shows that there would be a free-boundary moving in (x,t)-coordinates at the rate $\left(\left(T-t\right)\left|\log\left(T-t\right)\right|\right)^{\frac{1}{2}}$. Moreover, Filippas and Liu[7] or Velazquez[14] showed the same results in multiple dimension.

3.3.1 Convergence of rescaled solutions

If we store the rescaling solutions $u_k\left(x_k,\tau_k\right)$ which is defined on the region

$$\left[-\lambda^{-k}x_{k-1_{refined1}},\lambda^{-k}x_{k-1_{refined2}}\right],$$

we can show that these profiles converge to a predicted one. To compare the profiles for different k, it is natural to rescale each $u_k\left(y_k,\tau_k\right)$ such that they are defined on a fixed interval $z\to\left[-1,1\right]$. Note that this rescaling affects the space alone, not the value of u_k. Following the similar argument in Berger-Kohn[2], we can explain the following lemma, which states why all profiles of the rescaled solutions u_k, whose maximum reaches M_0, will converge to the predicted profile. In the following, $o\left(1\right)$ means a term which tends to 0 as $k\to\infty$.

Lemma 1. *If we consider rescaled profiles*

$$z\to u_k\left(z\lambda^{-k}x_k,\tau_k\right),\ \ -1\le z\le 1,$$

doi:10.6342/NTU202402879

then these profiles will converge to the profile

$$z \to M_0 \left[1 + z^2 \lambda^{-2} \left(\alpha^{1-p} - 1\right)\right]^{\frac{1}{1-p}}. \tag{11}$$

Proof. Coming form the result of [15][10], we know the actual profile f which is introduced in (10). Following (8), we first have

$$u_k\left(x_k, \tau_k\right) = \lambda^{\frac{2k}{p-1}} u\left(\lambda^k x_k, \tau_k\right) \tag{12}$$

Based on a fact in [8]

$$\lim_{t \to T} (T-t)^{\frac{-1}{p-1}} u\left(x\sqrt{T-t}, t\right) = 0 \text{ or } \pm\kappa \tag{13}$$

where T is the blow-up time and $\kappa = \frac{1}{p-1}^{\frac{1}{p-1}}$, there is an estimate

$$\left(T - \tau_k\right)^{\lambda^{-2k}} = M_0^{1-p} \frac{1}{p-1} + o\left(1\right). \tag{14}$$

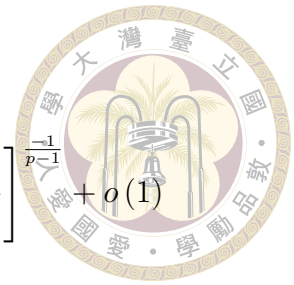
Combined with (12) (13), we can derive the following computation:

$$\begin{aligned} u_k\left(x_k, \tau_k\right) &= \left[M_0^{p-1}\left(T - \tau_k\right)(p-1)\right]^{\frac{1}{p-1}} u\left(M_0^{\frac{p-1}{2}} \sqrt{T-\tau_k}\,(p-1)^{\frac{1}{2}} x_k, \tau_k\right) + o\left(1\right) \\ &= M_0 \left(T - \tau_k\right)^{\frac{1}{p-1}} (p-1)^{\frac{1}{p-1}} \left[\left(T-\tau_k\right)^{\frac{-1}{p-1}} f\left(\frac{M_0^{\frac{p-1}{2}} (p-1)^{\frac{1}{2}} |x_k|}{\sqrt{|\log\left(T-\tau_k\right)|}}\right)\right] + o\left(1\right) \\ &= M_0 \left[1 + \frac{p-1}{4p} \frac{M_0^{p-1}(p-1)\, x_k^2}{|\log\left(T-\tau_k\right)|}\right]^{\frac{-1}{p-1}} + o\left(1\right) \end{aligned} \tag{15}$$

doi:10.6342/NTU202402879

That is,

$$u_k\left(z\lambda^{-1}x_{k-1},\tau_k\right)=M_0\left[1+\frac{p-1}{4p}\frac{M_0^{p-1}\left(p-1\right)z^2\lambda^{-2}x_{k-1}^2}{\left|\log\left(T-\tau_k\right)\right|}\right]^{\frac{-1}{p-1}}+o\left(1\right)$$



Following the definition of the refined region and the radial symmetry, we can get

$$u_{k-1}\left(x_{k_{refined_1}},\tau_{k-1}\right)=u_{k-1}\left(x_{k_{refined_{end}}},\tau_{k-1}\right)=\alpha M_0 \tag{15}$$

Hence, from (15) (15), we have

$$\alpha M_0=M_0\left[1+\frac{p-1}{4p}\frac{M_0^{p-1}\left(p-1\right)x_{k-1}^2}{\left|\log\left(T-\tau_{k-1}\right)\right|}\right]^{\frac{-1}{p-1}}+o\left(1\right),$$

so that $\frac{x_{k-1}}{\left|\log(T-\tau_{k-1})\right|^{\frac{1}{2}}}$ tends as $k\to\infty$ to the root ζ,

$$\zeta=\left(\alpha^{1-p}-1\right)^{\frac{1}{2}}\sqrt{\frac{4p}{\left(p-1\right)^2}M_0^{1-p}} \tag{16}$$

Finally, we can conclude that the rescaled profile u_k is asymptotically

$$z\to M_0\left[1+z^2\lambda^{-2}\left(\alpha^{1-p}-1\right)\right]^{\frac{1}{1-p}}.$$

□

The argument for the two-dimensional case is similar to the one-dimensional case described above, but in (11) , we should replace z with $|z|$ due to the radial property of the solution. Moreover, to observe the 2D rescaled profile, we select the cross-section at $x=0$ and $y=0$ for visualization purposes. This adjustment accounts for the symmetry and ensures that the comparison of profiles is consistent with the radial nature of the problem.

doi:10.6342/NTU202402879

3.3.2 One Dimensional Case

In 1D simulation, we use the following setting:

1. Initial data with domain:

$$u_0(x) = 1 + \cos(\pi x), \ x \in [-1, 1]$$

2. The numbers of grid points and the initial mesh size:

$$n = 100, \ dx_0 = \frac{2}{100} = 0.02, \ dt_0 = (dx_0)^2 / 4$$

3. Initial threshold:

$$M_0 = 3, \ \alpha = \frac{1.8}{M_0}$$

4. Scaling parameter:

$$\lambda = \frac{1}{2}$$

5. The number of iterations:

$$N = 40, \ 80.$$

Next, let's quickly and sequentially explain why we chose these parameters. Initially, the initial data satisfies the radial property. Consequently, in the subsequent results, we can observe that the refined region is symmetric about the origin. This symmetry is crucial for simplifying the analysis and ensuring consistency in the refinement process.

Next, we choose a sufficiently small mesh size to maintain computational accuracy. The specific choice of mesh size is somewhat flexible, provided it is small enough to cap-

doi:10.6342/NTU202402879

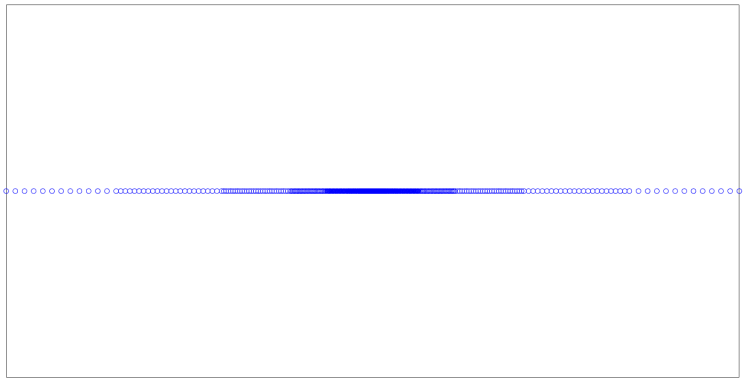


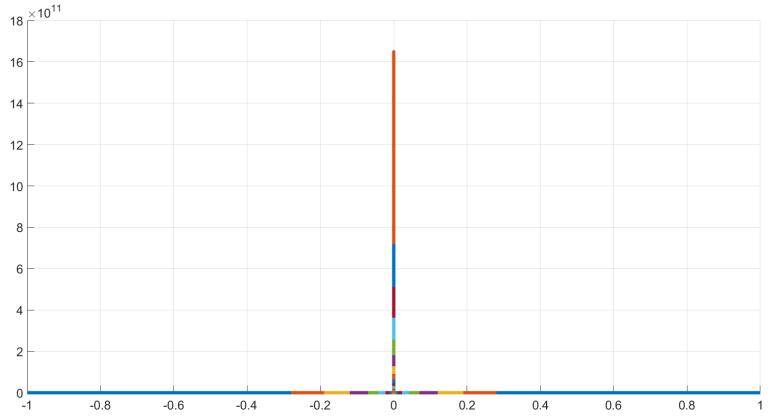
Figure 3.2: The final grid points of 1D after 80 iterations.

ture the necessary details of the solution. This ensures that the numerical solution remains accurate while balancing computational efficiency. On the other hand, the value of α can be chosen arbitrarily large or small. However, to manage computational memory efficiently, we select a suitable value based on the problem's requirements. Here, we refer to the Berger-Kohn results and choose a value that balances accuracy and memory usage. Typically, we opt for selecting positive integers as reciprocals of λ. Although λ can be chosen arbitrarily, selecting an appropriate value is important to manage memory usage effectively. Again, referring to the Berger-Kohn results, we choose a suitable λ to ensure the accuracy and feasibility of the computations. Finally, as noted in Berger-Kohn, the more iterations we use, the more accurate the result is. To demonstrate this, we compare the results of 80 iterations with those of 40 iterations. This comparison highlights the improvements in accuracy achieved through increased iterations.

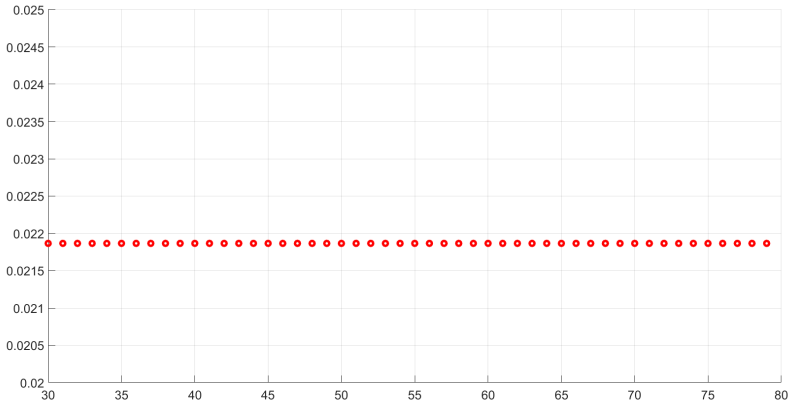
In Figure 3.2, we illustrate the final mesh size after 80 iterations. For better visualization, we focus on a small region, specifically $[-0.1, 0.1]$, rather than the entire domain. The blue points in the figure represent the grid points. From this zoomed-in view, it is evident that the mesh size becomes increasingly fine near the center region. Additionally, we can observe that the grid is nearly symmetrical around the center point, demonstrating

doi:10.6342/NTU202402879

the consistency and accuracy of our mesh refinement process.



((a)) The entire solution with different color corresponding to different mesh size after 80 iterations.

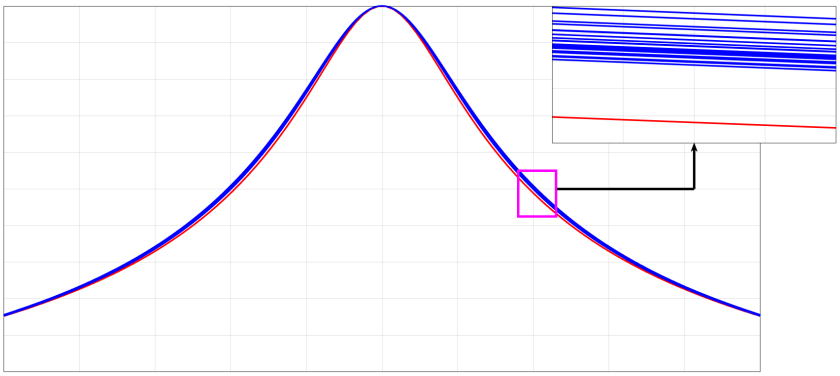


((b)) The approximated blow-up time. The x-axis is the numbers of refined mesh. The y-axis is the approximated time. In final stage, due to they rounding, the approximated time almost maintain near 0.022.

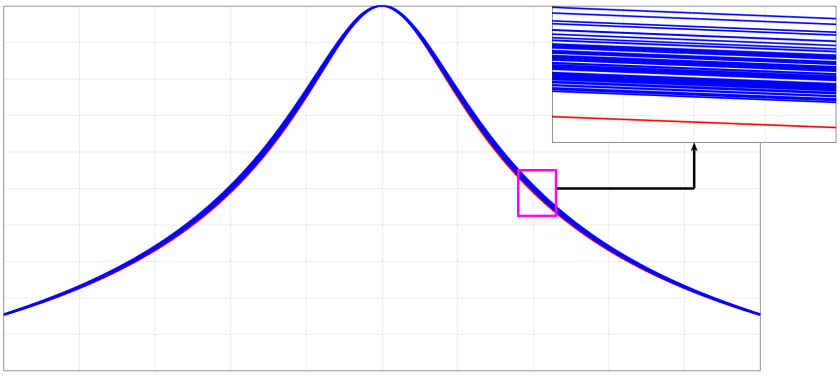
Figure 3.4: The result after 80 iterations

Based on the 80 iterations, we plot the entire solution, and the rescaling solutions when the maximum reaches the M, and numerically verify the blow-up rate. In Figure 3.3(a), this plot shows the entire solution data after 80 iterations. Each color corresponds to a different mesh size used in the computation. We observe that over time (iterations), the mass center becomes progressively finer, indicating increasing resolution in critical areas. In Figure 3.3(b), This figure displays the approximated blow-up time. As the number of iterations k increases, the computed blow-up time appears to approach a certain value.

doi:10.6342/NTU202402879



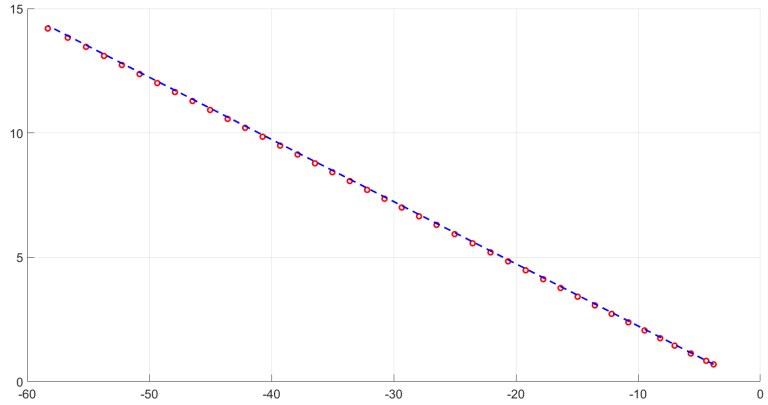
((a)) The rescaled profiles from iteration 20 to 40. The red one is the predict profile ploted by (11).



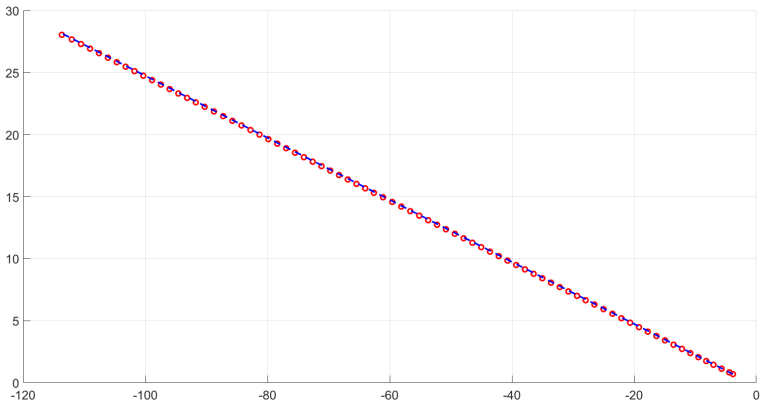
((b)) The rescaled profiles from iteration 20 to 80. The red one is the predict profile ploted by (11).

Figure 3.6: After 80 iterations, observe the rescaled profiles. Compare the results from 40 and 80 iterations by zooming in on the same small region, the pink box, near $x = 0.4$ and $u = 2$.

doi:10.6342/NTU202402879



((a)) The blow-up rate based on 40 iterations, and the final point approaches approximately -68 on the x-axis and 14 on the y-axis.



((b)) The blow-up rate based on 80 iterations, and the final point approaches approximately -115 on the x-axis and 27 on the y-axis.

Figure 3.8: Compute the blow up rate based on 40 and 80 iterations, respectively. The x-axis is $\log(T-t)$ and the y-axis is $\log||u(t)||$. The red points is computed from the iterating process and the blue line is the predicted result, which the slope is $\frac{-1}{4}$

doi:10.6342/NTU202402879

This trend suggests that with more iterations, the time step size dt_k, decreases gradually. However, it's important to note that numerical calculations might be affected by rounding errors inherent in computational tools like MATLAB, where very small numbers can be rounded off.

In Figure 3.6, we can compare the rescaled profiles from 20 to 40 iterations with from 20 to 80 iterations. For comparing the difference, we restrict the domain in the $[-1, 1]$ and plot first the predicted profile, the red one. With the iteration increasing, the rescaling profiles, blue lines, seem to be more and more closed to the red in two results. This is what we want. To further confirm whether these rescaled profiles converge to the predicted profile, we zoom in on the positive position of $u = 2$ cross-section and show 40 result in the figure 3.5(a) and 80 result in the figure 3.5(b). As the iteration count increases, the rescaled profiles progressively approach the predicted profile from the right side towards the center. The proximity between the rescaled profiles and the predicted profile becomes increasingly accurate with higher iteration counts.

To determine if the solution exhibits Type I behavior, where the blow-up rate conforms to the ODE rate,

$$\limsup_{t \to T} (T - t)^{\frac{1}{p-1}} ||u(\cdot, t)||_{L^\infty(\mathbb{R}^n)} < +\infty,$$

we analyze the blow-up rates plotted in Figure 3.8. Figures 3.7(a) and 3.7(b) respectively depict analyses based on the first 40 and 80 iterations. The x-axis represents $\log(T - t)$, where T is the estimated blow-up time derived from the iteration count. The y-axis represents $\log(||u(t)||_{L^\infty})$. The red points is the data from the computation and The blue line originates from the same starting point and follows the slope $-\frac{1}{p-1}$. Ideally, the red points should align closely with or be parallel to the blue line, indicating consistency with the expected blow-up rate. Note that not all data points are plotted; only select points are

doi:10.6342/NTU202402879

chosen for clarity, as initial iterations may exhibit slight oscillations due to certain unfavorable points. Finally, the result is almost same as out expectation. Note the blow-up rate is main in the time closed to the blow-up time. In fact, we are more concerned about the tail part, specifically the top-left position. In Figure 3.7(a), the final point approaches approximately -68 on the x-axis and 14 on the y-axis. In Figure 3.7(b), it approaches approximately -115 on the x-axis and 27 on the y-axis. When we want to compute the blow-up rate, the rounding error is crucial. Direct subtraction of computed time from the estimated blow-up time may yield infinity due to rounding. To mitigate this, dt_k values used in each iteration are stored in a matrix. When computing the blow-up rate, $T - t$ is obtained by summing this matrix backwards from the last time point to ensure accuracy.

3.3.3 Two Dimensional Case

Computing the 2D problem poses greater challenges compared to the 1D case, primarily due to increased computational demands. We present two examples with different initial data sets. For the first example, results are based on iterations up to 40, 60 respectively. Due to the extremely fine and dense nature of the final entire solution profile in 2D, visualizing the entire solution is impractical. In the second example, we focus solely on the 60 iterations result. The initial data is constructed following the behavior described in (9). This allows us to compare the approximated blow-up time with the predicted blow-up behavior.

Example 1

doi:10.6342/NTU202402879

1. Initial data with domain:

$$u_0(x,y) = (0.002)^{\frac{-1}{4}} \exp\left(-\left(\frac{\sqrt{x^2+y^2}}{0.002\,|\ln(0.002)|}\right)^2\right) \times \phi(r)$$

$$\text{with } (x,y) \in [-1,1] \times [-1,1]$$

where $r = \sqrt{x^2+y^2}$ and $\phi(\cdot)$ is a cut-off function defined by

$$\phi(z) = \begin{cases} 1, & \text{if } z \leq 0.25 \\ -8(z-0.25)^2 + 1, & \text{if } 0.25 < z \leq 0.5 \\ 8(z-0.75)^2, & \text{if } 0.5 < z \leq 0.75 \\ 0, & \text{if } z > 0.75 \end{cases} \tag{17}$$

2. The numbers of grid points of each axis and the initial mesh size:

$$n = 400,\ dx_0 = dy_0 = \frac{1}{200} = 0.005,\ dt_0 = (dx_0)^2/4$$

3. Initial threshold:

$$M_0 = \lambda^{\frac{-2}{p-1}} ||u_0||_{L^\infty} \sim 5.62,\ \alpha = \frac{3.6}{M_0}$$

4. Scaling parameter:

$$\lambda = \frac{1}{2}$$

5. The number of iterations:

$$N = 40,\ 60.$$

Let's revisit the setup described earlier. To ensure the radial property, our initial data is chosen based on the final behavior characteristics. Additionally, to accommodate Dirichlet boundary conditions, a cutoff function is applied to enforce the initial data to

doi:10.6342/NTU202402879

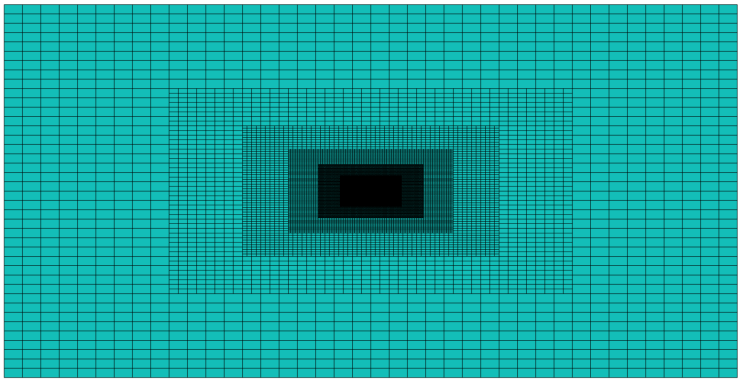


Figure 3.9: The final mesh size of 2D after 60 iterations.

be zero on the boundary. In our computational experiments focusing on accuracy and stability in the 2D case, we employ a grid size of 500 points along both the x-axis, y-axis. This choice ensures sufficient resolution while maintaining computational feasibility. Similarly, the scaling parameter remains consistent across the setup.

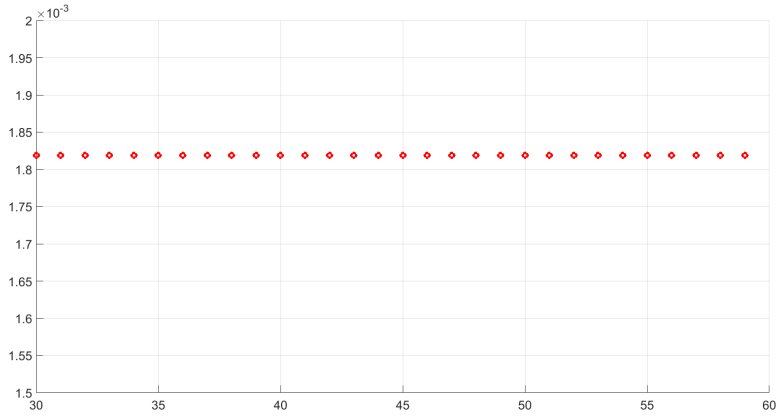
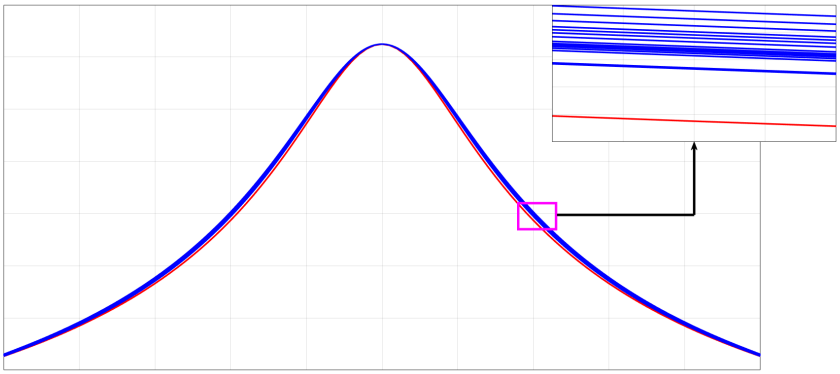


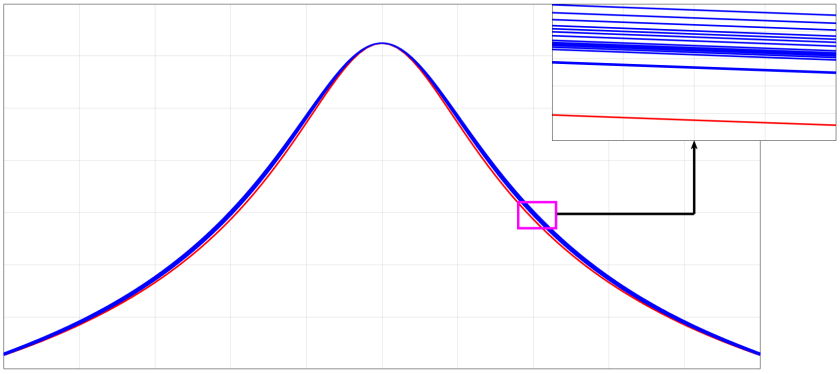
Figure 3.10: The approximated blow-up time based on different numbers of refined mesh. Concerning rounding error, it's almost 0.0018.

In Fig.3.9, This figure displays the final mesh configuration after 40 iterations. To enhance clarity, the domain is limited to $[-0.1, 0.1] \times [-0.1, 0.1]$. Here, the color intensity increases as the mesh grid converges towards the origin, reflecting finer resolution. By a similar manner, we plot the 60 iterations blow-up time in Figure 3.10. The plot reveals a nearly horizontal final phase, attributable to rounding effects in the computations.

doi:10.6342/NTU202402879



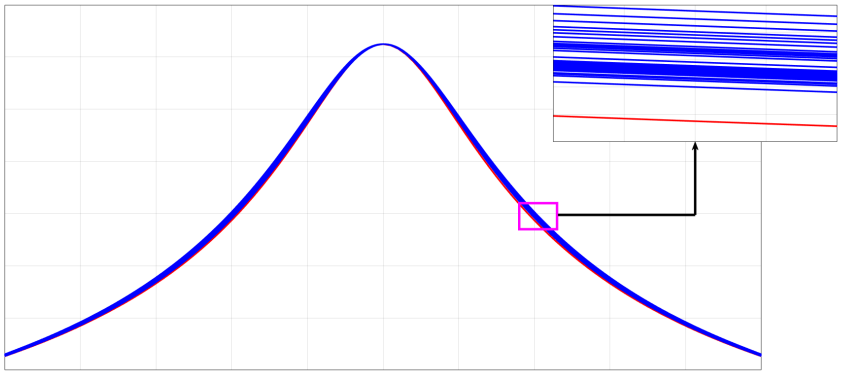
((a)) The cross-section at $x = 0$ here, and zoom in near $y = 0.4$.



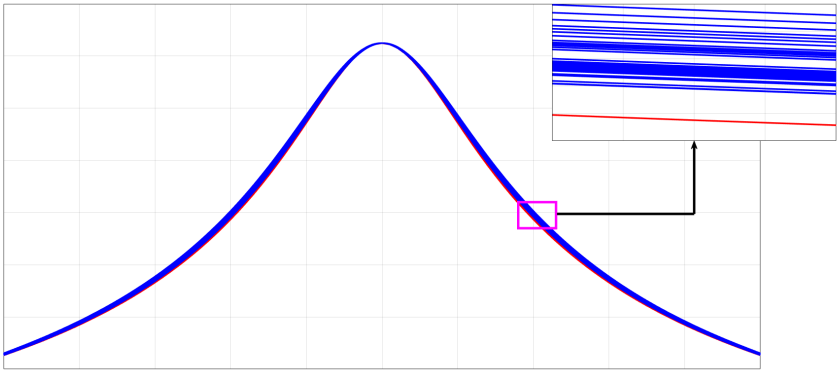
((b)) The cross-section at $y = 0$ here, and zoom in near $x = 0.4$.

Figure 3.12: The rescaled profiles from 20 to 40 iterations, and the predicted profile, the red one. For checking the convergence, zoom in on a small region, the pink box, near $x, y = 0.4$ and $u = 4$. 3.11(a),3.11(b) correspond to different section, $x = 0$, $y = 0$.

doi:10.6342/NTU202402879



((a)) The cross-section at $x = 0$ here, and zoom in near $y = 0.4$.



((b)) The cross-section at $y = 0$ here, and zoom in near $x = 0.4$.

Figure 3.14: The rescaled profiles from 20 to 60 iterations, and the predicted profile, the red one. For checking the convergence, zoom in on a small region, the pink box, near $x, y = 0.4$ and $u = 4$. 3.13(a) and 3.13(b) correspond to different section, $x = 0$, $y = 0$.

doi:10.6342/NTU202402879

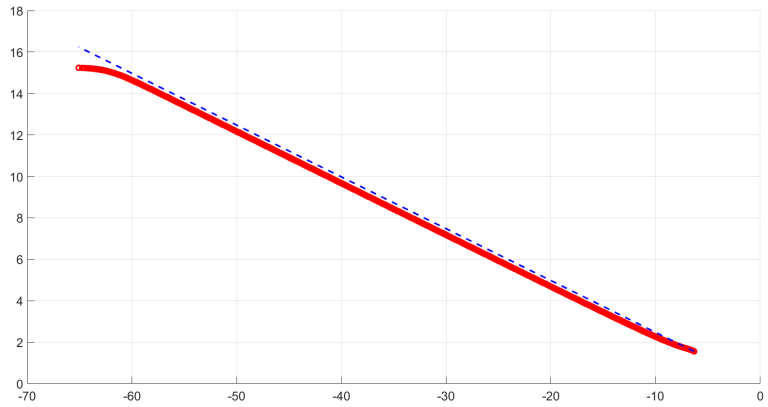
Next, we plot the rescaled profiles based on 40, 60 iterations and show them in Figure 3.12, 3.14, respectively. We observe the $x = 0, y = 0$ cross-section for visualization. First of all, we consider 40 iterations result in the figure 3.11(a), 3.11(b), where the profiles of the cross-sections at $x = 0$ and $y = 0$ are depicted. The red line represents the predicted profile. Here, we consider the result from 20 to 40 iterations. However, over subsequent iterations, the rescaled profiles gradually become smoother, converging towards the predicted profile. For checking such closer examination, we focused on the positive region with $u = 5.5$. Similarly, we plot, and zoom in the result of 60 iterations in the figure 3.14. Comparing the figure 3.12 with the figure 3.14, especially in the small region, it's evident that with an increasing number of iterations, the final profile closely approximates the predicted profile.

Finally, we can attempt to approximate the blow-up rate to investigate whether it follows a Type I behavior. Again, plot the relation between $\log(T-t)$ and $\log||u(t)||_{L^\infty}$. In Figure 3.16, we show the result of 40 iterations and 60 iterations, and the x-axis presents $\log(T-t)$ while the y-axis presents $\log||u(t)||_{L^\infty}$, in a similar manner. Despite the initial iterations displaying some peculiar points that clearly surpass the blue line—where we anticipated the data to align—the tail end nearly maintains a slope of $\frac{-1}{p-1}$. In Figure 3.15(a), $\log||u(t)||_{L^\infty} \sim 16$ with $\log(T-t) \sim -60$. The other figure 3.15(b), $\log||u(t)||_{L^\infty} \sim 25$ with $\log(T-t) \sim -100$.

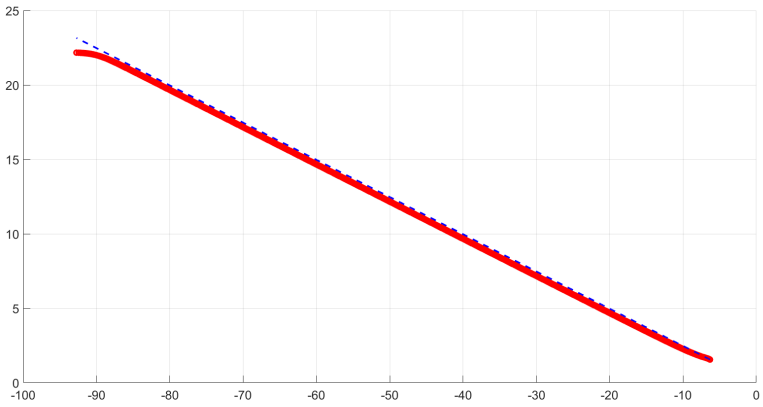
Based on such simulation result, we can briefly conclude that based on our setting, initial data and the nonlinear exponent $p = 5$, the solution will blow-up in finite time. Moreover, following the Berger-Kohn result, the rescaling profile will also converge to the profile (11).

Example 2

doi:10.6342/NTU202402879



((a)) The blow-up rate of 2D base on 40 iterations and the final point approaches approximately -61 on the x-axis and 16 on the y-axis.



((b)) The blow-up rate of 2D base on 60 iterations and the final point approaches approximately -90 on the x-axis and 24 on the y-axis.

Figure 3.16: Compute the blow-up rate based on 40, 60 iterations, respectively. The x-axis is $\log(T-t)$ and the y-axis is $\log||u(t)||$. The red points is computed from the iterating process and the blue line is the predicted result, which the slope is $\frac{-1}{4}$.

doi:10.6342/NTU202402879

1. Initial data with domain:

$$u_0(x,y) = T^{\frac{-1}{4}} f\left(\frac{r}{\sqrt{T\left|\log T\right|}}\right) \times \phi(r),\ (x,y) \in [-1,1] \times [-1,1]$$

where $T = .002$, $r = \sqrt{x^2+y^2}$, f is introduced in (10), and $\phi(\cdot)$ is the same cut-off function defined in (17).

2. The numbers of grid points of each axis and the initial mesh size:

$$n = 200,\ dx_0 = dy_0 = \frac{1}{100} = 0.001,\ dt_0 = (dx_0)^2/4$$

3. Initial threshold:

$$M_0 = \lambda^{\frac{-2}{p-1}} ||u_0||_{L^\infty} \sim 3.98,\ \alpha = \frac{2.5}{M_0}$$

4. Scaling parameter:

$$\lambda = \frac{1}{2}$$

5. The number of iterations:

$$N = 60.$$

Here we show another example which the initial data is constructed by (9). We quickly introduce how to construct. This behavior depends on the the value of the blow-up time. That is, we can first assume a blow-up time T_{theory} then put it into the (10). Hence, we can get a final behavior which depends on T_{theory}. Next, if we want to get a initial data which should blow up in finite time T_{theory}, we put $t = 0$ in (9) and use the change of variable $x = \frac{z}{\sqrt{T_{\text{theory}}\left|\log T_{\text{theory}}\right|}}$. Finally, we can get an initial data

$$u_0(z) = T_{\text{theory}}^{\frac{-1}{p-1}} f\left(\frac{z}{\sqrt{T_{\text{theory}}\left|\log T_{\text{theory}}\right|}}\right)$$

doi:10.6342/NTU202402879

where f is defined in (10). In fact, we can test more different examples. However, there are some advantage in this case. One of them is that we can roughly predict the blow up time based on the initial data. Here, it should be near $T_{\text{theory}} = 0.002$. On the other hand, this initial data provides the radial property so that the solutions is symmetry to the origin. However, due to the influence of the outer part, the final approximated blow-up time will not exactly equal to T_{theory}. Therefore, there will be an error. In the figure 3.17, we can see that the approximated time is near $T_{\text{approx}} = 0.0023$, that is $|T_{\text{approx}} - T_{\text{theory}}| = O(10^{-4})$. This kind of error is acceptable. Nevertheless, in the figure 3.19, 3.20, we can see that the rescaled profiles converging to the predicted one and the blow-up rate, respectively. These results is based on the 60 iterations.

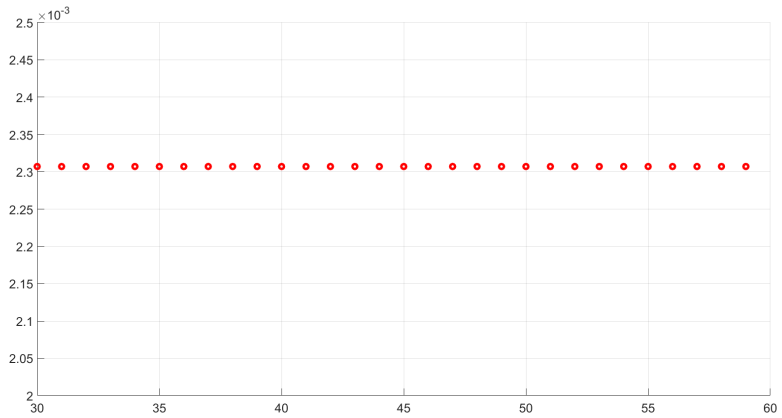
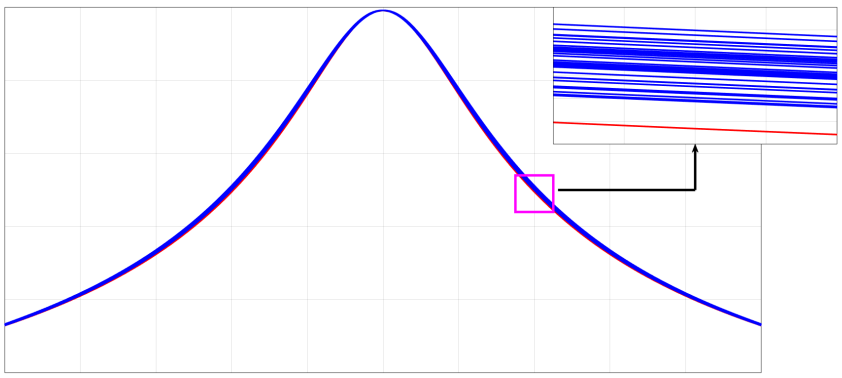


Figure 3.17: The approximated blow-up time based on different numbers of refined mesh. Concerning rounding error, it's almost 0.0023.

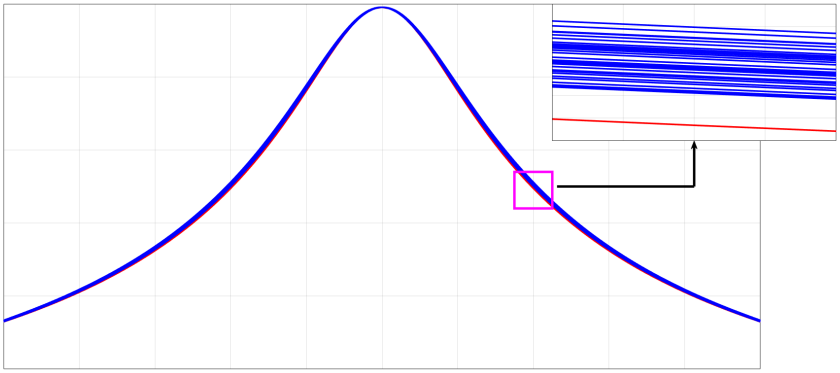
3.4 Application 2: A random path

To address a scenario involving a moving blowup point, our initial objective is to implement a method that accommodates its random trajectory. This involves beginning with an assumed starting point, which we treat as the blowup point, and allowing it to move randomly. Our goal is to dynamically refine the computational domain to accommodate

doi:10.6342/NTU202402879



((a)) The cross-section at $x = 0$ here, and zoom in near $y = 0.4$.



((b)) The cross-section at $y = 0$ here, and zoom in near $x = 0.4$.

Figure 3.19: The rescaled profiles from 20 to 60 iterations, and the predicted profile, the red one. For checking the convergence, zoom in on a small region, the pink box, near $x, y = 0.4$ and $u = 4$. 3.18(a) and 3.18(b) correspond to different section, $x = 0$, $y = 0$.

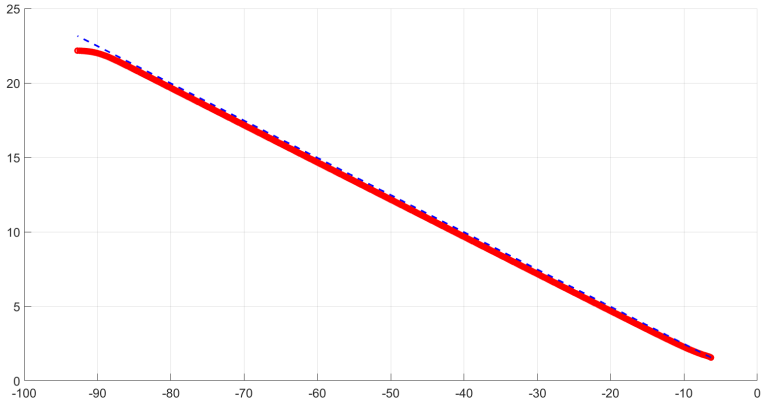


Figure 3.20: The blow-up rate of 2D base on 60 iterations and the final point approaches approximately -90 on the x-axis and 24 on the y-axis.

doi:10.6342/NTU202402879

this movement. However, there are specific constraints that need to be considered in this context:

1. Move slowly,

2. Move toward to the origin.

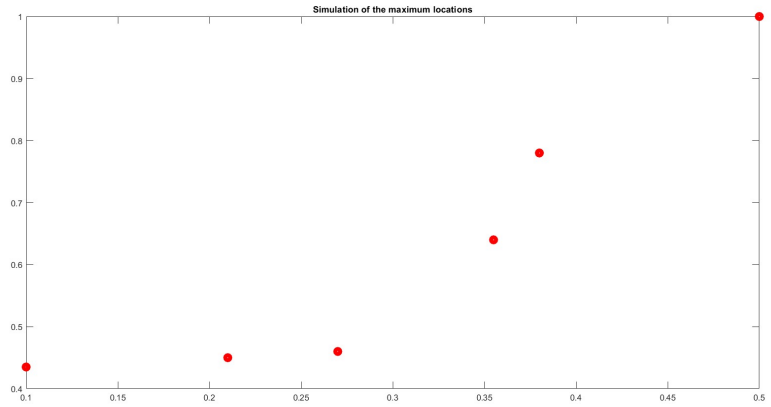
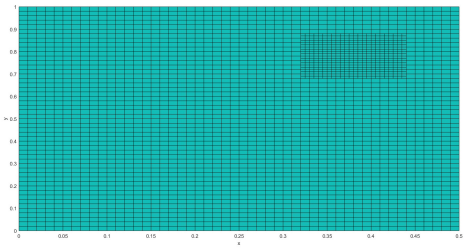


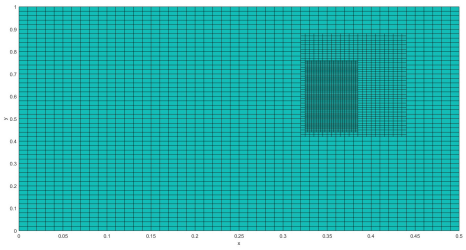
Figure 3.21: Simulation of the maximum locations under this random path

Therefore, it may be necessary to extend the computational region to allow for layer-by-layer arrangement of these refined meshes. In Fig. 3.23, we present results after 5 iterations. Additionally, in Fig. 3.21, we illustrate the random trajectory of this point.

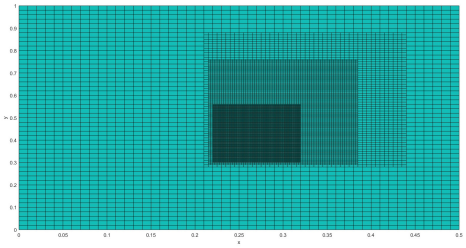
doi:10.6342/NTU202402879



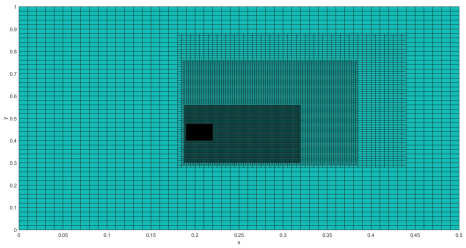
((a)) 1-st refinement under a random path



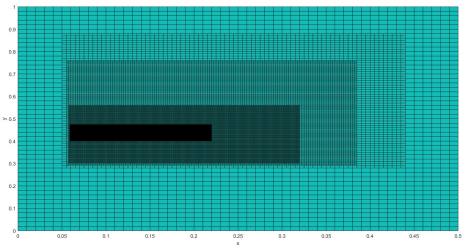
((b)) 2-nd refinement under a random path.



((c)) 3-nd refinement under a random path



((d)) 4-nd refinement under a random path



((e)) 5-nd refinement under a random path

Figure 3.23: 5th refinement under a random path

doi:10.6342/NTU202402879

Chapter 4 Numerical Results of 3D Navier-Stokes equations

Finally, we present some numerical results for the 3D axisymmetric Navier-Stokes equations:

$$\begin{cases} \tilde{u}_t + u^r \tilde{u}_r + u^z \tilde{u}_z & = 2\tilde{u}\tilde{\psi}_z + \nu\left(\tilde{u}_{rr} + \frac{3}{r}\tilde{u}_r + \tilde{u}_{zz}\right), \\ \tilde{\omega}_t + u^r \tilde{\omega}_r + u^z \tilde{\omega}_z & = 2\tilde{u}\tilde{u}_z + \nu\left(\tilde{\omega}_{rr} + \frac{3}{r}\tilde{\omega}_r + \tilde{\omega}_{zz}\right), \\ -\left(\partial_{rr} + \frac{3}{r}\partial r + \partial_{zz}\right)\tilde{\psi} & = \tilde{\omega}, \\ u^r = -r\tilde{\psi}_z,\ u^z = 2\tilde{\psi} + r\tilde{\psi}_r, & \end{cases}$$

where the refinement method is based on the scaling invariance:

$$\begin{cases} \tilde{u}_\lambda(s,k,l) & = \lambda^2 \tilde{u}\left(\lambda^2 s, \lambda k, \lambda l\right), \\ \tilde{\omega}_\lambda(s,k,l) & = \lambda^3 \tilde{\omega}\left(\lambda^2 s, \lambda k, \lambda l\right), \\ \tilde{\psi}_\lambda(s,k,l) & = \lambda \tilde{\psi}\left(\lambda^2 s, \lambda k, \lambda l\right). \end{cases}$$

For the constant viscosity ν, we set $\nu = 5 \times 10^{-4}$ which is same as Hou's initial setting. As discussed in Chapter 3.1, we define the refinement region as $\mathcal{C}_{\text{NS}} = \{\alpha M \le \tilde{\omega} \le M\} := [z_{\text{NS}_1}, z_{\text{NS}_2}] \times [r_{\text{NS}_1}, r_{\text{NS}_2}]$, with $\alpha = 0.6$ and the initial threshold $M_0 \approx 2.7 \times 10^3$. However, in our experiment, we find if we want to make the computing

doi:10.6342/NTU202402879

stability, we have to refine the region extend to the origin. That is, in every refinement, we have to refine $\mathcal{C}'_{\text{NS}} = [0, z_{\text{NS}_2}] \times [0, r_{\text{NS}_2}]$. Considering the mesh size, we choose 200 grids and 400 grids on $z-$ and $r-$ axis, respectively, so that the mesh size of z is same as the mesh size of r. Finally, for convenience, we choose $\lambda = 1/2$.

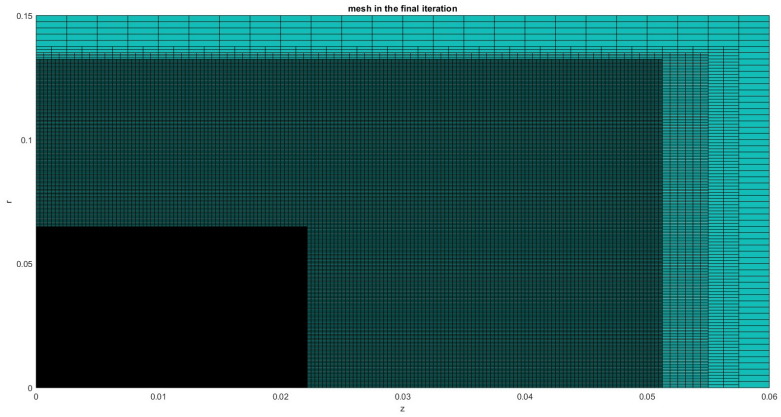
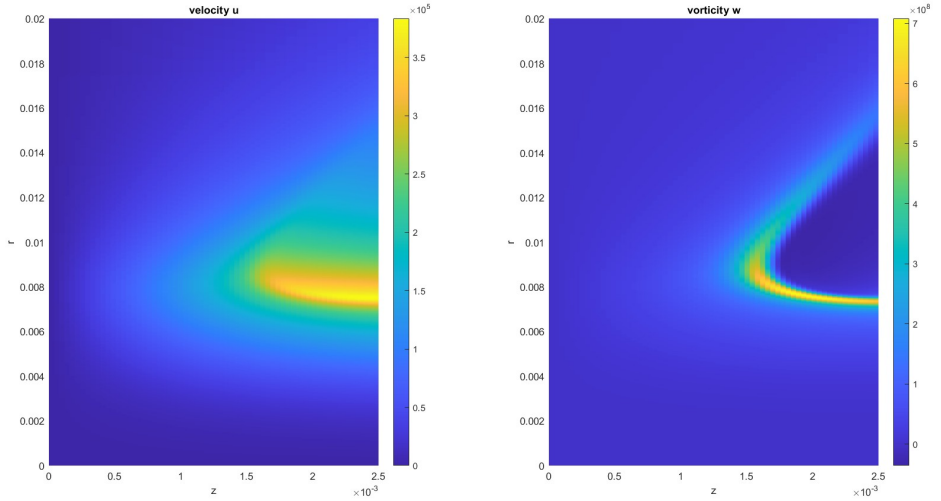


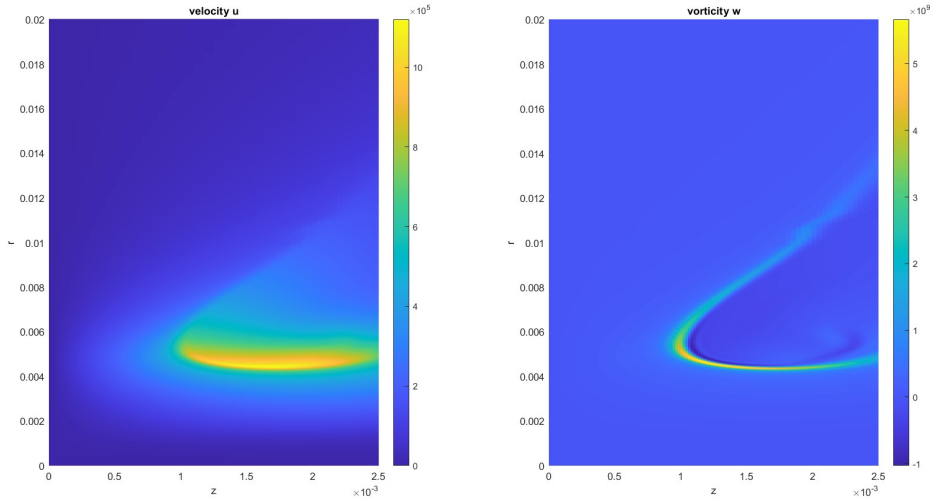
Figure 4.1: The final mesh of the domain

First of all, we check the final mesh of the domain and display it in Fig. 4.1. The final mesh satisfies our requirements, with the mesh arranged layer by layer as intended. In Fig.4.3, we display the velocity and vorticity results for the 7th, 8th, and 9th iterations, respectively. After the 9th iteration, the maximum of $\tilde{\omega}$ reaches 10^{10}. For visualization, the view is from the top. To facilitate comparison, the plots are constrained to the domain $(z, r) \in [0, 2.5 \times 10^{-3}] \times [0, 2 \times 10^{-2}]$. Additionally, the mass center becomes progressively thinner and more concentrated. These figures illustrate that the solution's mass center moves toward the origin as hypothesized. However, especially in the 9th iteration result, we can observe the appearance of turbulence. In the next stage, this turbulence may influence the profiles of velocity and vorticity. In Fig. 4.5, we display the 10th, 11th and 12th iteration results, where the maximum value achieve to 10^{11}, 10^{12}, 10^{13}, respectively. For convenience, we restrict the visualization to the small region $[0, 1 \times 10^{-3}] \times [0, 5 \times 10^{-3}]$. From these results, we can infer that the solution to

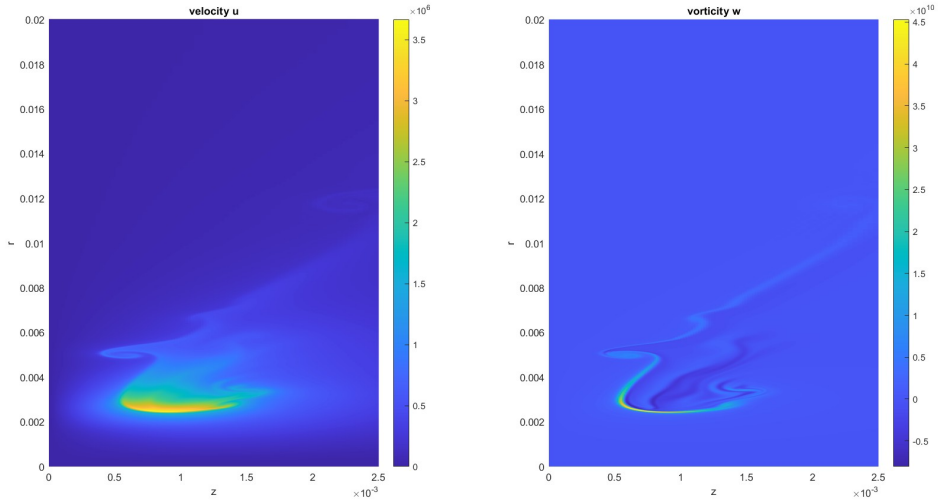
doi:10.6342/NTU202402879



((a)) 7-th iteration result. (L): velocity, (R):vorticity



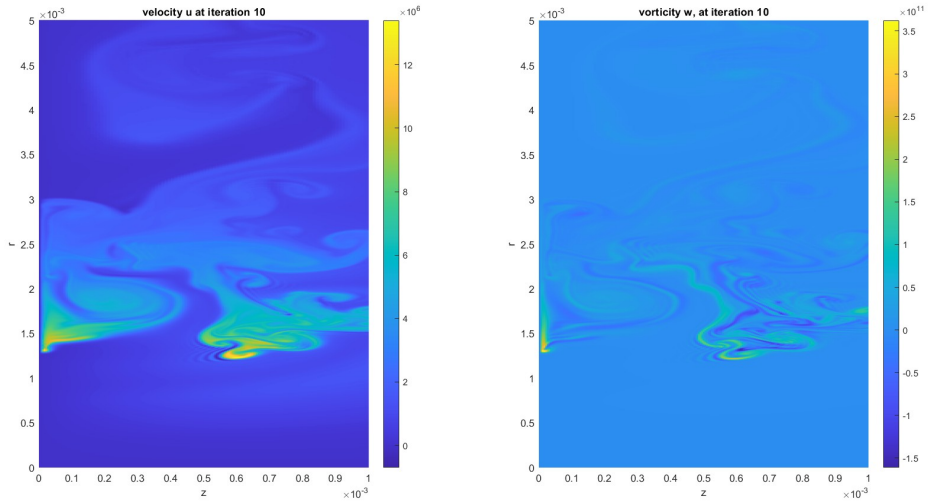
((b)) 8-th iteration result. (L): velocity, (R):vorticity



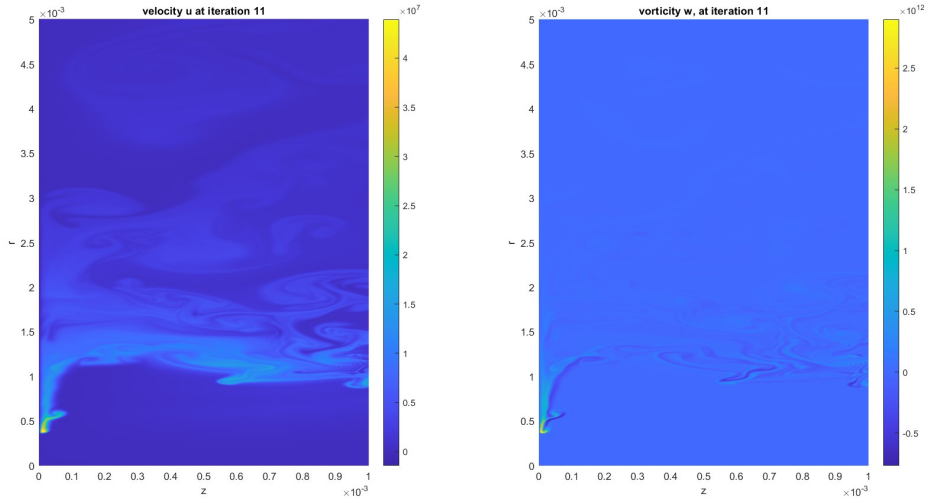
((c)) 9-th iteration result. (L): velocity, (R):vorticity

Figure 4.3: 7th, 8th, 9th iteration results in $[0, 2.5 \times 10^{-3}] \times [0, 2 \times 10^{-2}]$

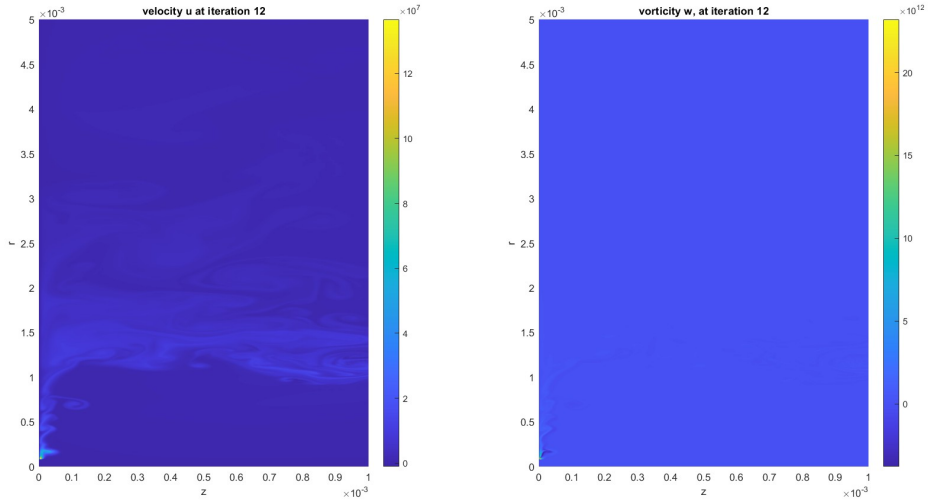
doi:10.6342/NTU202402879



((a)) 10-th iteration result. (L): velocity, (R):vorticity



((b)) 11-th iteration result. (L): velocity, (R):vorticity



((c)) 12-th iteration result. (L): velocity, (R):vorticity

Figure 4.5: 10th, 11th, 12th iteration results $[0, 1 \times 10^{-3}] \times [0, 5 \times 10^{-3}]$

doi:10.6342/NTU202402879

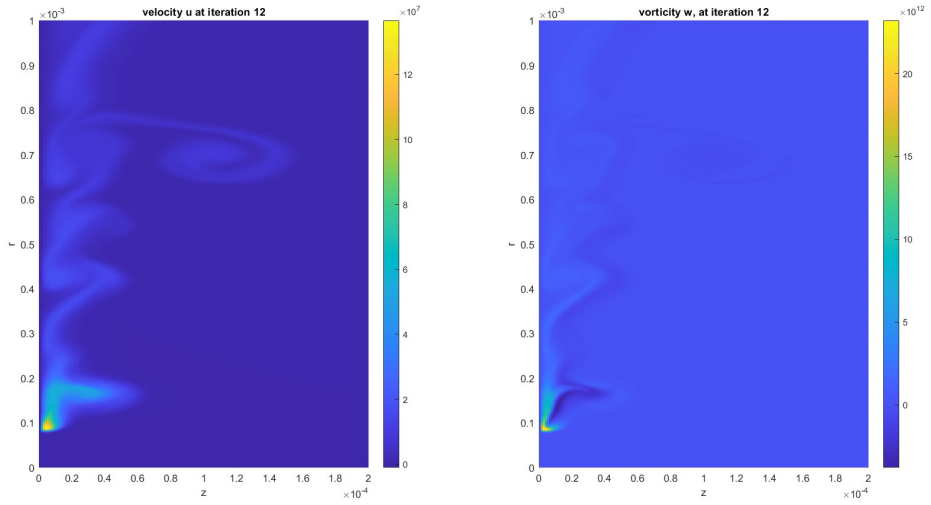


Figure 4.6: 12th iteration result at $[0, 2 \times 10^{-4}] \times [0, 1 \times 10^{-3}]$

the Navier-Stokes equations is unstable at this stage. However, the instability observed in the 11th iteration appears to stabilize in the 12th iteration. In Fig. 4.6, we display the 12th iteration result in the smaller region, $[0, 2 \times 10^{-4}] \times [0, 1 \times 10^{-3}]$. This result shows more stability compared to the 10th and 11th iterations. Throughout the process, the values continue to increase, and the maximum location moves towards the origin. These results suggest a more natural progression, which differs slightly from Hou's results.

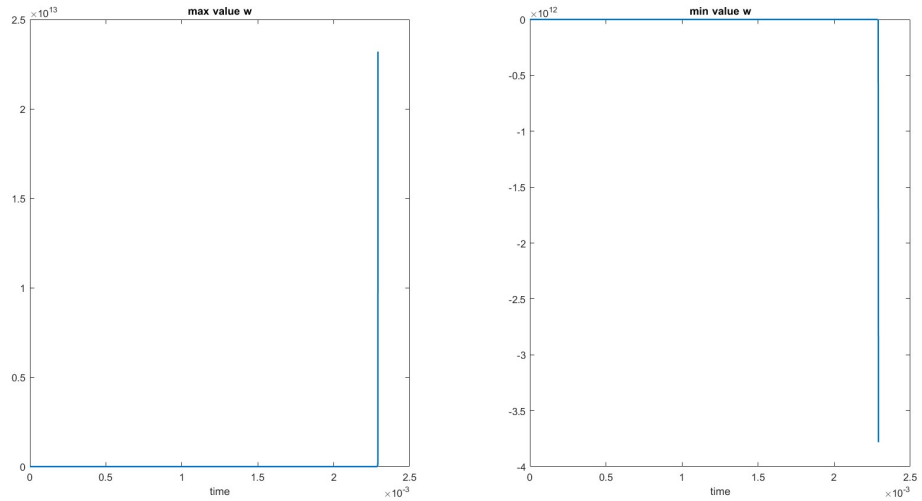


Figure 4.7: In $9-$th iteration, the maximum and minimum value of vorticity

doi:10.6342/NTU202402879

If we consider the changes in vorticity over the first 12th iterations, where the maximum value reaches 10^{13} and the minimum drops to -10^{12}, as shown in Fig.4.7, We can observe a clear trend: the maximum value of vorticity becomes increasingly positive, while the minimum value becomes increasingly negative. This indicates a significant amplification in the range of vorticity values. Despite this divergence, the difference between these extremes consistently remains at least 10.

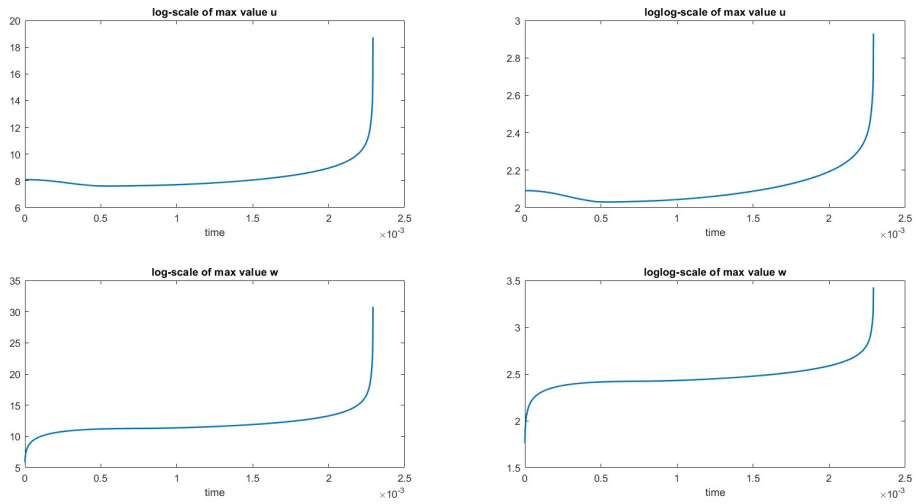


Figure 4.8: Plot of $log-$ and $loglog-$ scale of maximum value of velocity and vorticity. The first row is about velocity, and the second row is about vorticity.

In Fig.4.8, we provide more detailed information about the changes in velocity and vorticity. We take the log and $loglog$ of the changes in both velocity and vorticity. As mentioned earlier, velocity decreases in the first stage. Simultaneously, vorticity rapidly increases and then almost maintains a certain level. After this stage, both components begin to increase steadily. This detailed analysis highlights the distinct phases in the evolution of velocity and vorticity, emphasizing the transition from initial rapid changes to a more stable growth pattern.

We show the relation between the maximum location of vorticity and the time. In Fig. 4.9, we separately plot the location z and r corresponding to the time. This shows that they

doi:10.6342/NTU202402879

really move toward to the origin. Moreover, in the simulation process, they first move away and then move toward the origin. Interestingly, the stage moving away from the origin seems to be related to the stage when velocity is decreasing and the change of value of vorticity is almost stop. It's worth to note that when the maximum location moves away from the origin, the maximum of velocity decreases and of vorticity is almost stop.

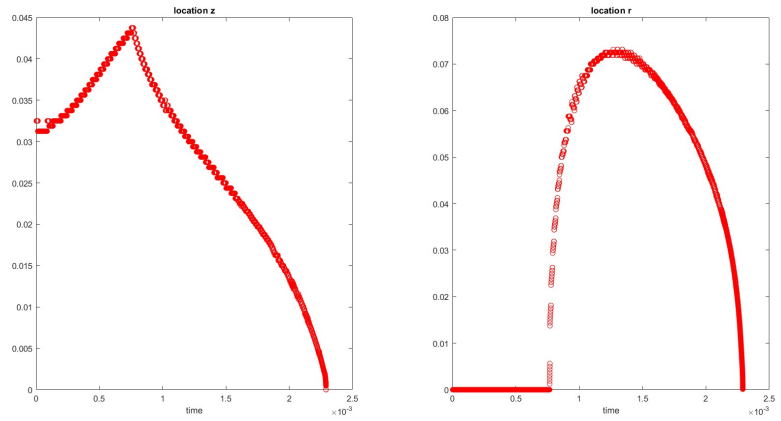


Figure 4.9: Vorticity maximum location of z and r. The LHS is the location of z w.r.t time. The RHS is the location of r w.r.t time.

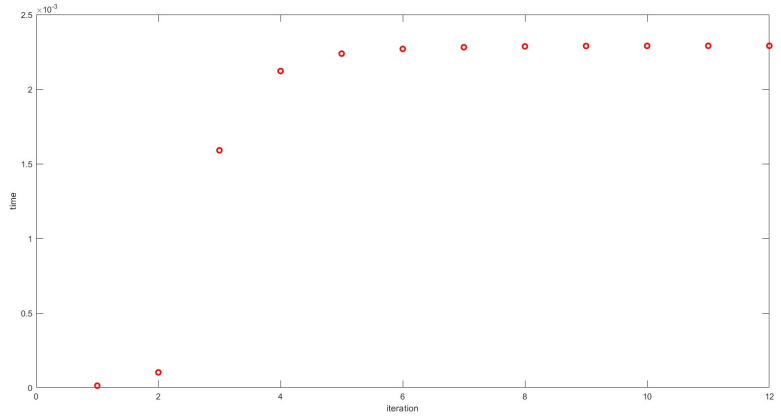


Figure 4.10: Approximated blow up time of 3D Navier Stokes equations

Using this refinement method, we can approximate the blow-up time through a sequence of values $\tau_0, \tau_1, \tau_2, \ldots$. We display the approximated blow-up time in Fig. 4.10. The result indicates an approximate blow-up time of $T_{\text{blowup}} \sim 2.3 \times 10^{-3}$. Finally, in Fig. 4.11, we show the rescaled profiles from the 7th to the 12th iteration. These profiles appear irregular, and based on the numerical results, we cannot guarantee that the rescaled

doi:10.6342/NTU202402879

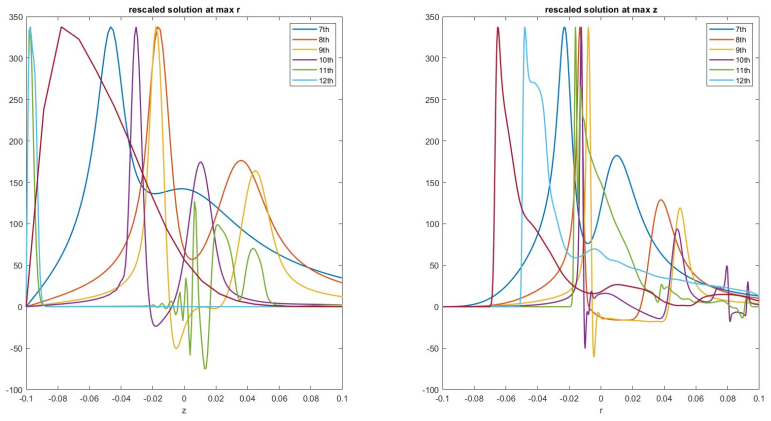


Figure 4.11: ross section of scaled solutions from 7th to 12th. The LHS is cross at $r =$ Maxlocation. The RHS is cross at $z =$ Maxlocation.

profiles will converge to a predicted profile. This irregularity may indicate the presence of underlying complexities in the solution dynamics, suggesting that further investigation and refinement of the computational methods may be necessary to fully understand the convergence behavior.

doi:10.6342/NTU202402879

Chapter 5 Conclusion

Utilizing the invariance scaling formula, we refine both the spatial and temporal mesh, resulting in a non-uniform mesh that provides increasingly detailed information about the equation. However, choosing the same mesh as the final one would result in unnecessary storage and computational wastage. That is, we only need to refine the crucial region, namely, the mass center. Compared with the Berger-Kohn, our method is also concerning the entire domain situation and can be easily used in others case. Moreover, in sec.3.2, we provide a programming techniques. Hence, even for an example with a moving blow-up point, we can also numerically solve it by this method. In blow-up problems, the values tend to increase significantly, potentially leading to numerically unstable scenarios. Consequently, ensuring accuracy across the entire solution becomes challenging. On the other hand, although this can describe the blow-up profile in more detail, due to more and more added grids, the computation becomes more and more large. Therefore, it is also a careful problem how to reduce the computation and designing an efficient program.

In this work, we primarily present the results of applying this method to the 3D axisymmetric Navier-Stokes equations. These numerical results exhibit some differences from the original hypothesis. During the first 9 iterations, the solution concentrates and becomes progressively thinner. However, starting from the 10th iteration, turbulence begins

doi:10.6342/NTU202402879

to appear, indicating that the solution to the 3D Navier-Stokes equations becomes unstable. Compared with Hou's results, our findings are based on a fixed constant viscosity. In some dynamic situations, our results differ from Hou's and appear more natural. Unfortunately, due to time limitations, I was unable to test other cases or different viscosities. For example, we could adjust the initial data such that

$$\tilde{u}_0(z,r) = g(r)\,h(z)\,,\ \tilde{\omega}_0(z,r) = 0,$$

where

$$\begin{aligned} h(z) &= \tfrac{\sin(2\pi z)}{1+12.5(\sin(\pi z))^2}, \\ g(r) &= 1200\,(1-r^2+0.2r)^{18}\,. \end{aligned}$$

The notable difference in this initial data is that the initial maximum location is not at the origin. Additionally, both Hou's and our results consider vorticity to be initially zero. Therefore, choosing a nonzero initial vorticity is a feasible direction for future work. Lastly, we could also test different viscosities to observe whether viscosity influences the blow-up problem.

In theory, although how to analysis or prove whether the 3D Navier-Stokes equations is still a unknown and challenging problem, we can numerically prove it and display some evidence. In numerical world, while the results suggest some positives, there remain specific computational intricacies that need optimization, especially storing data and linear interpolation. Presently, we consider utilizing the matrix form to solve the equation as essential. In solving the Poisson equation in the equations, it is worth to think how to optimize and reduce the computation.

doi:10.6342/NTU202402879

References

[1] G. Acosta, R. G. Durán, and J. D. Rossi. An adaptive time step procedure for a parabolic problem with blow-up. Computing, 68(4):343–373, 2002.

[2] M. Berger and R. V. Kohn. A rescaling algorithm for the numerical calculation of blowing-up solutions. Comm. Pure Appl. Math., 41(6):841–863, 1988.

[3] C. J. Budd, W. Huang, and R. D. Russell. Moving mesh methods for problems with blow-up. SIAM J. Sci. Comput., 17(2):305–327, 1996.

[4] A. Cangiani, E. H. Georgoulis, I. Kyza, and S. Metcalfe. Adaptivity and blow-up detection for nonlinear evolution problems. SIAM J. Sci. Comput., 38(6):A3833–A3856, 2016.

[5] I. Dimov, I. Faragó, and L. Vulkov, editors. Numerical analysis and its applications, volume 10187 of Lecture Notes in Computer Science. Springer, Cham, 2017. Revised selected papers of the 6th International Conference (NAA 2016) held in Lozenetz, June 15–22, 2016.

[6] C. L. Fefferman. Existence and smoothness of the Navier-Stokes equation. In The millennium prize problems, pages 57–67. Clay Math. Inst., Cambridge, MA, 2006.

doi:10.6342/NTU202402879

[7] S. Filippas and W. X. Liu. On the blowup of multidimensional semilinear heat equations. Ann. Inst. H. Poincaré C Anal. Non Linéaire, 10(3):313–344, 1993.

[8] Y. Giga and R. V. Kohn. Asymptotically self-similar blow-up of semilinear heat equations. Comm. Pure Appl. Math., 38(3):297–319, 1985.

[9] P. Groisman. Totally discrete explicit and semi-implicit Euler methods for a blow-up problem in several space dimensions. Computing, 76(3-4):325–352, 2006.

[10] M. A. Herrero and J. J. L. Velázquez. Blow-up behaviour of one-dimensional semilinear parabolic equations. Ann. Inst. H. Poincaré C Anal. Non Linéaire, 10(2):131–189, 1993.

[11] T. Y. Hou. Potentially singular behavior of the 3D Navier-Stokes equations. Found. Comput. Math., 23(6):2251–2299, 2023.

[12] T. Y. Hou and C. Li. Dynamic stability of the three-dimensional axisymmetric Navier-Stokes equations with swirl. Comm. Pure Appl. Math., 61(5):661–697, 2008.

[13] J. C. Robinson, J. L. Rodrigo, and W. Sadowski. The three-dimensional Navier-Stokes equations, volume 157 of Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 2016. Classical theory.

[14] J. J. L. Velázquez. Higher-dimensional blow up for semilinear parabolic equations. Comm. Partial Differential Equations, 17(9-10):1567–1596, 1992.

[15] J. J. L. Velázquez, V. A. Galaktionov, and M. A. Herrero. The space structure near a blow-up point for semilinear heat equations: a formal approach. Zh. Vychisl. Mat. i Mat. Fiz., 31(3):399–411, 1991.

doi:10.6342/NTU202402879