

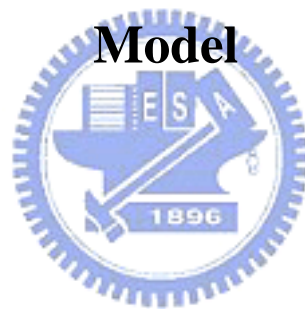
國立交通大學

應用數學系

碩士論文

計算火焰片模型的數值方法

Numerical Methods for Computing Flame Sheet



研究生：林宗澤

指導老師：葉立明 教授

中華民國九十三年六月

計算火焰片模型的數值方法

Numerical Methods for Computing Flame Sheet Model

研究生：林宗澤

Student：Tzong-Tzer Lin

指導教授：葉立明

Advisor：Li-Ming Yeh



Submitted to Department of Applied Mathematics

College of Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Applied Mathematics

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

計算火焰片模型的數值方法

學生: 林宗澤

指導教授: 葉立明

國立交通大學應用數學系 (研究所) 碩士班

摘要

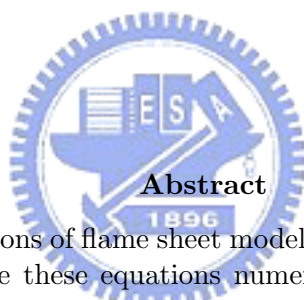
火焰片 (flame sheet) 模型的微分方程式是個非常非線性並且彼此互相相關的系統。在數值上為了解出這些方程式, 我們使用了抑制牛頓法 (damped Newton's method) 並且結合單項網格法 (one way multigrid) 和 Krylov 子空間法 (Krylov subspace methods)。本篇論文的目的是研究對此模型下較有效率的 Krylov 子空間法, 例如, 雙共軛梯度穩定法 (biconjugate gradient stabilized method), 廣義最小剩餘法 (generalized minimum residual method) 和無轉置的擬最小剩餘法 (transpose-free QMR method)。我們已經為此火焰片模型發展了 C 語言程式碼, 並於文章內展示數值結果並且加以討論。

Numerical Methods for Computing Flame Sheet Model

Student: Tzong-Tzer Lin

Advisor: Dr. Li-Ming Yeh

*Department of Applied Mathematics
National Chiao Tung University*



Abstract

The differential equations of flame sheet model are highly nonlinear and strongly coupled system. To solve these equations numerically, we use damped Newton's method combining with one way multigrid method and Krylov subspace methods. The purpose of this thesis is to survey effective Krylov subspace methods for this model, for example, biconjugate gradient stabilized method (BICGSTAB), generalized minimum residual method (GMRES), and transpose-free QMR method (TFQMR). A code for the flame sheet model in *C* language is developed and numerical results will be presented and discussed in this work.

誌 謝

這是我的第一篇論文，因此在寫作的過程中難免會遇到許多的挫折和難題，很幸運地，都能讓我一一地克服了。現在回頭看看過往，內心充滿了許多的感激。本論文得以完成，主要仰賴於我的指導教授 葉立明博士，葉教授對於我一開始鬆散且不正確的為學態度，都能不厭其煩地指正和引導，讓我能唸研究所的過程中學習到做學問該有的嚴謹態度和方法，這真是一份很難得的體驗和經驗，著實讓我受益良多，在此我要再次感謝葉教授的指導。

其次，對於我在研究所唸書時一路相伴的同學們說聲謝謝，如果沒有你們，我想在這裡求學的過程中絕對是孤獨無趣多了。也再此預祝大家鵬程萬里。

最後，也是最重要的，我要特別感激我的父母。對身為長子的我到現在還能置身事外，並沒有被加諸太多責任和要求，使我能再無後顧之憂的情況下追求我的理想，如果沒有父母的包容和關愛，我是無法順利完成這個學業的。



Contents

Abstract (in Chinese)	i
Abstract (in English)	ii
Acknowledgement	iii
Contents	iv
List of Figures	v
1 Introduction	1
1.1 The Governing Equations for Flame Sheet Model	2
1.2 Discretized Form for Flame Sheet Model	3
2 Krylov Subspace Methods	14
2.1 Projection Methods	14
2.2 Arnoldi's Method	18
2.3 FOM, IOM, and DIOM	21
2.3.1 FOM	21
2.3.2 IOM	22
2.3.3 DIOM	23
3 Biconjugate Gradient Stabilized Method	25
3.1 Conjugate Gradient Method	25
3.2 Lanczos Biorthogonalization	30
3.2.1 Two-Sided Lanczos	30
3.2.2 BCG	32
3.3 Transpose-Free Variants	34
3.3.1 CGS	34
3.3.2 BICGSTAB	37
4 Transpose-Free QMR Method	40
4.1 GMRES, QGMRES, and DQGMRES	40
4.1.1 GMRES	40
4.1.2 QGMRES	41
4.1.3 DQGMRES	42
4.2 Quasi-Minimal Residual	46
4.3 TFQMR	48
5 Algorithm and Numerical Results	53
5.1 General Solution Algorithm	53
5.2 Numerical Results and Discussion	54
References	60

List of Figures

Figure		Page
1	Physical configuration for diffusion flame model (not in scale)	10
2	Contour plot of the temperature.....	58
3	Residual norm of Newton step during the time stepping phase.....	59



1 Introduction

Primitive formulation for laminar diffusion flame in three dimensional rectangular coordinates can be transformed into the vorticity-velocity formulation by using two dimensional cylindrical coordinates. A detailed derivation of the vorticity-velocity formulation can be found in [4, 5]. Usually a flame sheet model [2] is used to initialize multidimensional diffusion flames. The governing equations for flame sheet model can be derived from equations for finite rate diffusion flame model in vorticity-velocity formulation. The flame sheet model is based on the assumptions that the chemical reaction in a laminar diffusion flame is a one-step irreversible reaction and that the conversion of reactant into stable product is infinitely fast. In the reaction zone, fuel and oxidizer are separated; fuel and oxidizer react in stoichiometric proportion. With these assumptions, no fuel appears on the oxidizer side and vice versa. Because the differential equations of flame sheet model are highly nonlinear and strongly coupled system, it is extremely difficult to analyze mathematically. Newton's method is a standard method in solving highly nonlinear and strongly coupled system of partial differential equations. Combining with multigrid method and Krylov subspace methods, we can solve the flame sheet model numerically.

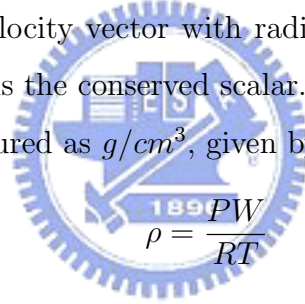
Purpose of this thesis is to survey efficient Krylov subspace methods and apply these methods to solve the flame sheet model. We shall concentrate on three Krylov subspace methods, namely biconjugate gradient stabilized method (BICGSTAB), generalized minimum residual method (GMRES), and transpose-free QMR method (TFQMR). This work is organized in the following ways: In the first chapter, we discretize the governing equation for flame sheet model by using centered difference scheme. The convective terms are evaluated by using upwind differenced scheme to preserve monotonicity. In the second chapter, we introduce the projection method and use Arnoldi's method to construct the Krylov subspace. In the third and fourth chapters, we will introduce BICGSTAB, GMRES, and TFQMR methods in mathematical level. In the last chapter, we present some numerical results and give a discussion of these results.

1.1 The Governing Equations for Flame Sheet Model

The flame sheet equations consist of the total mass and momentum conservation equations, constituting the flow field problem and using vorticity-velocity formulation of the steady-state, coupled with a conserved scalar equation. The governing equations for flame sheet model with axis symmetry can be stated in the following form.

$$\left\{ \begin{array}{l} \frac{\partial^2 V_r}{\partial r^2} + \frac{\partial^2 V_r}{\partial z^2} = \frac{\partial \omega}{\partial z} - \frac{1}{r} \frac{\partial V_r}{\partial r} + \frac{V_r}{r^2} - \frac{\partial}{\partial r} \left(\frac{V \cdot \nabla \rho}{\rho} \right), \\ \frac{\partial^2 V_z}{\partial r^2} + \frac{\partial^2 V_z}{\partial z^2} = -\frac{\partial \omega}{\partial r} - \frac{1}{r} \frac{\partial V_r}{\partial z} - \frac{\partial}{\partial z} \left(\frac{V \cdot \nabla \rho}{\rho} \right), \\ \frac{\partial^2 \mu \omega}{\partial r^2} + \frac{\partial^2 \mu \omega}{\partial z^2} + \frac{\partial}{\partial r} \left(\frac{\mu \omega}{r} \right) = \rho V_r \frac{\partial \omega}{\partial r} + \rho V_z \frac{\partial \omega}{\partial z} - \frac{\rho V_r}{r} \omega + \bar{\nabla} \rho \cdot \nabla \frac{V^2}{2} - \bar{\nabla} \rho \cdot g \\ \quad + 2 \left(\bar{\nabla} (\text{div}(V)) \cdot \nabla \mu - \nabla V_r \cdot \bar{\nabla} \frac{\partial \mu}{\partial r} - \nabla V_z \cdot \bar{\nabla} \frac{\partial \mu}{\partial z} \right), \\ \frac{1}{r} \frac{\partial}{\partial r} \left(r \rho D \frac{\partial S}{\partial r} \right) + \frac{\partial}{\partial z} \left(\rho D \frac{\partial S}{\partial z} \right) = \rho V_r \frac{\partial S}{\partial r} + \rho V_z \frac{\partial S}{\partial z}, \end{array} \right.$$

where $V = (V_r, V_z)$ is the velocity vector with radial V_r and axial V_z components, $\omega = \frac{\partial V_r}{\partial z} - \frac{\partial V_z}{\partial r}$ is vorticity, and S is the conserved scalar. In addition, the scalar ρ is the mass density of the mixture, measured as g/cm^3 , given by the equation of



$$\rho = \frac{PW}{RT}$$

where P is the pressure, R is the universal gas constant, T is the temperature, and W is the mean molecular weight of the mixture. The scalar μ is the viscosity coefficient of the mixture, measured as $g/(cm \cdot S)$, given by the equation of

$$\mu = \mu_0 \left(\frac{T}{T_0} \right)^r$$

where $r = 0.7$, $T_0 = 298K$, $\mu_0 = 1.85 \times 10^{-4}$. The scalar D is a diffusion coefficient, measured as cm^2/s , given by the equation of

$$\rho D = \frac{\mu}{P_r}$$

where $P_r = 0.75$. And the vector $\bar{\nabla} \beta = \left(\frac{\partial \beta}{\partial z}, -\frac{\partial \beta}{\partial r} \right)$ is curl of β .

1.2 Discretized Form for Flame Sheet Model

We use the finite difference approximations to discrete the governing equations for flame sheet model on the grid points in the computational domain, shown in **Figure 1**, which cover from $r = 0$ to 7.5cm in the radial direction and from $z = 0$ to 30cm in the axial direction. In the diffusion and source terms, we use standard centered differences. In the axial, we use a monotonicity preserving upwind scheme to discrete the convective terms. Next we write down the discretized form for each equation.

Radial Velocity :

$$\frac{\partial}{\partial r} \left(r \frac{\partial V_r}{\partial r} \right) + r \frac{\partial^2 V_r}{\partial z^2} - r \frac{\partial \omega}{\partial z} - \frac{V_r}{r} + r \frac{\partial}{\partial r} \left(\frac{V \cdot \nabla \rho}{\rho} \right) = 0.$$

(i) (ii) (iii) (iv) (v)

(i) For $i = 3 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\frac{\partial}{\partial r} \left(r \frac{\partial V_r}{\partial r} \right) \Big|_{i,j} \approx \frac{1}{r_{i+1/2} - r_{i-1/2}} \left(r \frac{\partial V_r}{\partial r} \Big|_{i+1/2,j} - r \frac{\partial V_r}{\partial r} \Big|_{i-1/2,j} \right)$$

$$\approx \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{r_{i+1} + r_i}{2} \frac{(V_r)_{i+1,j} - (V_r)_{i,j}}{r_{i+1} - r_i} - \frac{r_i + r_{i-1}}{2} \frac{(V_r)_{i,j} - (V_r)_{i-1,j}}{r_i - r_{i-1}} \right).$$

For $i = 2$ and $j = 2 \sim (m - 1)$

$$\frac{\partial}{\partial r} \left(r \frac{\partial V_r}{\partial r} \right) \Big|_{2,j} \approx \frac{1}{r_{2+1/2} - r_1} \left(r \frac{\partial V_r}{\partial r} \Big|_{2+1/2,j} - r \frac{\partial V_r}{\partial r} \Big|_{1,j} \right) \quad \text{where } r_1 = 0$$

$$\approx \frac{2}{r_2 + r_3} \left(\frac{r_2 + r_3}{2} \frac{(V_r)_{3,j} - (V_r)_{2,j}}{r_3 - r_2} \right) = \frac{(V_r)_{3,j} - (V_r)_{2,j}}{r_3 - r_2}.$$

(ii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$r \frac{\partial^2 V_r}{\partial z^2} \Big|_{i,j} \approx r_i \frac{1}{z_{j+1/2} - z_{j-1/2}} \left(\frac{\partial V_r}{\partial z} \Big|_{i,j+1/2} - \frac{\partial V_r}{\partial z} \Big|_{i,j-1/2} \right)$$

$$\approx r_i \frac{2}{z_{j+1} - z_{j-1}} \left(\frac{(V_r)_{i,j+1} - (V_r)_{i,j}}{z_{j+1} - z_j} - \frac{(V_r)_{i,j} - (V_r)_{i,j-1}}{z_j - z_{j-1}} \right).$$

(iii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$r \frac{\partial \omega}{\partial z} \Big|_{i,j} \approx r_i \frac{\omega_{i,j+1} - \omega_{i,j-1}}{z_{j+1} - z_{j-1}}.$$

(iv) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\left. \frac{V_r}{r} \right|_{i,j} \approx \frac{(V_r)_{i,j}}{r_i}.$$

(v) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} r \frac{\partial}{\partial r} \left(\frac{V \cdot \nabla \rho}{\rho} \right) \Big|_{i,j} &= r \frac{\partial}{\partial r} \left(\frac{V_r \partial \rho}{\rho \partial r} \right) \Big|_{i,j} + r \frac{\partial}{\partial r} \left(\frac{V_z \partial \rho}{\rho \partial z} \right) \Big|_{i,j} \\ &\approx \frac{r_i}{r_{i+1/2} - r_{i-1/2}} \left(\frac{V_r \partial \rho}{\rho \partial r} \Big|_{i+1/2,j} - \frac{V_r \partial \rho}{\rho \partial r} \Big|_{i-1/2,j} \right) \\ &\quad + \frac{r_i}{r_{i+1} - r_{i-1}} \left(\frac{V_z \partial \rho}{\rho \partial z} \Big|_{i+1,j} - \frac{V_z \partial \rho}{\rho \partial z} \Big|_{i-1,j} \right) \\ &\approx \frac{r_i}{r_{i+1} - r_{i-1}} \left[\left(\frac{(V_r)_{i+1,j}}{\rho_{i,j}} + \frac{(V_r)_{i,j}}{\rho_{i+1,j}} \right) \frac{\rho_{i+1,j} - \rho_{i,j}}{r_{i+1} - r_i} \right. \\ &\quad \left. - \left(\frac{(V_r)_{i-1,j}}{\rho_{i,j}} + \frac{(V_r)_{i,j}}{\rho_{i-1,j}} \right) \frac{\rho_{i,j} - \rho_{i-1,j}}{r_i - r_{i-1}} \right] \\ &\quad + \frac{r_i}{r_{i+1} - r_{i-1}} \left[\frac{(V_z)_{i+1,j} \rho_{i+1,j+1} - \rho_{i+1,j-1}}{\rho_{i+1,j} z_{j+1} - z_{j-1}} \right. \\ &\quad \left. - \frac{(V_z)_{i-1,j} \rho_{i-1,j+1} - \rho_{i-1,j-1}}{\rho_{i-1,j} z_{j+1} - z_{j-1}} \right]. \end{aligned}$$

Axial Velocity :

$$\frac{\partial^2 V_z}{\partial r^2} + \frac{\partial^2 V_z}{\partial z^2} + \frac{\partial \omega}{\partial r} + \frac{1}{r} \frac{\partial V_r}{\partial z} + \frac{\partial}{\partial z} \left(\frac{V \cdot \nabla \rho}{\rho} \right) = 0.$$

(i) (ii) (iii) (iv) (v)

(i) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} \frac{\partial^2 V_z}{\partial r^2} \Big|_{i,j} &\approx \frac{1}{r_{i+1/2} - r_{i-1/2}} \left(\frac{\partial V_z}{\partial r} \Big|_{i+1/2,j} - \frac{\partial V_z}{\partial r} \Big|_{i-1/2,j} \right) \\ &\approx \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{(V_z)_{i+1,j} - (V_z)_{i,j}}{r_{i+1} - r_i} - \frac{(V_z)_{i,j} - (V_z)_{i-1,j}}{r_i - r_{i-1}} \right). \end{aligned}$$

(ii) For $i = 2 \sim (n-1)$ and $j = 2 \sim (m-1)$

$$\begin{aligned} \frac{\partial^2 V_z}{\partial z^2} \Big|_{i,j} &\approx \frac{1}{z_{j+1/2} - z_{j-1/2}} \left(\frac{\partial V_z}{\partial z} \Big|_{i,j+1/2} - \frac{\partial V_z}{\partial z} \Big|_{i,j-1/2} \right) \\ &\approx \frac{2}{z_{j+1} - z_{j-1}} \left(\frac{(V_z)_{i,j+1} - (V_z)_{i,j}}{z_{j+1} - z_j} - \frac{(V_z)_{i,j} - (V_z)_{i,j-1}}{z_j - z_{j-1}} \right). \end{aligned}$$

(iii) For $i = 2 \sim (n-1)$ and $j = 2 \sim (m-1)$

$$\frac{\partial \omega}{\partial r} \Big|_{i,j} \approx \frac{\omega_{i+1,j} - \omega_{i-1,j}}{r_{i+1} - r_{i-1}}.$$

(iv) For $i = 2 \sim (n-1)$ and $j = 2 \sim (m-1)$

$$\frac{1}{r} \frac{\partial V_z}{\partial z} \Big|_{i,j} \approx \frac{(V_z)_{i,j+1} - (V_z)_{i,j-1}}{r_i(z_{j+1} - z_{j-1})}.$$

(v) For $i = 2 \sim (n-1)$ and $j = 2 \sim (m-1)$

$$\begin{aligned} \frac{\partial}{\partial z} \left(\frac{V \cdot \nabla \rho}{\rho} \right) \Big|_{i,j} &= \frac{\partial}{\partial z} \left(\frac{V_r \partial \rho}{\rho \partial r} \right) \Big|_{i,j} + \frac{\partial}{\partial z} \left(\frac{V_z \partial \rho}{\rho \partial z} \right) \Big|_{i,j} \\ &\approx \frac{1}{z_{j+1} - z_{j-1}} \left(\frac{V_r \partial \rho}{\rho \partial r} \Big|_{i,j+1} - \frac{V_r \partial \rho}{\rho \partial r} \Big|_{i,j-1} \right) \\ &\quad + \frac{1}{z_{j+1/2} - z_{j-1/2}} \left(\frac{V_z \partial \rho}{\rho \partial z} \Big|_{i,j+1/2} - \frac{V_z \partial \rho}{\rho \partial z} \Big|_{i,j-1/2} \right) \\ &\approx \frac{1}{z_{j+1} - z_{j-1}} \left[\frac{(V_r)_{i,j+1} \rho_{i+1,j+1} - \rho_{i-1,j+1}}{\rho_{i,j+1} r_{i+1} - r_{i-1}} - \frac{(V_r)_{i,j-1} \rho_{i+1,j-1} - \rho_{i-1,j-1}}{\rho_{i,j-1} r_{i+1} - r_{i-1}} \right] \\ &\quad + \frac{1}{z_{j+1} - z_{j-1}} \left[\left(\frac{(V_r)_{i,j+1}}{\rho_{i,j}} + \frac{(V_r)_{i,j}}{\rho_{i,j+1}} \right) \frac{\rho_{i,j+1} - \rho_{i,j}}{z_{j+1} - z_j} \right. \\ &\quad \left. - \left(\frac{(V_r)_{i,j-1}}{\rho_{i,j}} + \frac{(V_r)_{i,j}}{\rho_{i,j-1}} \right) \frac{\rho_{i,j} - \rho_{i,j-1}}{z_j - z_{j-1}} \right]. \end{aligned}$$

Vorticity :

$$\frac{\partial}{\partial r} \left(r \frac{\partial \mu \omega}{\partial r} \right) + r \frac{\partial^2 \mu \omega}{\partial z^2} - \frac{\mu \omega}{r} - \left(r \rho V_r \frac{\partial \omega}{\partial r} + r \rho V_z \frac{\partial \omega}{\partial z} - \rho V_r \omega \right) - r \bar{\nabla} \rho \cdot \nabla \frac{V^2}{2} +$$

(i) (ii) (iii) (iv) (v)

$$r \bar{\nabla} \rho \cdot g - 2r (\bar{\nabla}(\operatorname{div}(V)) \cdot \nabla \mu) + 2r \left(\nabla V_r \cdot \bar{\nabla} \frac{\partial \mu}{\partial r} + \nabla V_z \cdot \bar{\nabla} \frac{\partial \mu}{\partial z} \right) = 0.$$

(vi)

(vii)

(viii)

(i) For $i = 3 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} \left. \frac{\partial}{\partial r} \left(r \frac{\partial \mu \omega}{\partial r} \right) \right|_{i,j} &\approx \frac{1}{r_{i+1/2} - r_{i-1/2}} \left(\left. r \frac{\partial \mu \omega}{\partial r} \right|_{i+1/2,j} - \left. r \frac{\partial \mu \omega}{\partial r} \right|_{i-1/2,j} \right) \\ &\approx \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{r_{i+1} + r_i}{2} \frac{\mu_{i+1,j} \omega_{i+1,j} - \mu_{i,j} \omega_{i,j}}{r_{i+1} - r_i} \right. \\ &\quad \left. - \frac{r_i + r_{i-1}}{2} \frac{\mu_{i,j} \omega_{i,j} - \mu_{i-1,j} \omega_{i-1,j}}{r_i - r_{i-1}} \right). \end{aligned}$$

For $i = 2$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} \left. \frac{\partial}{\partial r} \left(r \frac{\partial \mu \omega}{\partial r} \right) \right|_{2,j} &\approx \frac{1}{r_{2+1/2} - r_1} \left(\left. r \frac{\partial \mu \omega}{\partial r} \right|_{2+1/2,j} - \left. r \frac{\partial \mu \omega}{\partial r} \right|_{1,j} \right) \quad \text{where } r_1 = 0 \\ &\approx \frac{2}{r_3 + r_2} \left(\frac{r_3 + r_2}{2} \frac{\mu_{3,j} \omega_{3,j} - \mu_{2,j} \omega_{2,j}}{r_3 - r_2} \right) = \frac{\mu_{3,j} \omega_{3,j} - \mu_{2,j} \omega_{2,j}}{r_3 - r_2}. \end{aligned}$$

(ii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} \left. r \frac{\partial^2 \mu \omega}{\partial z^2} \right|_{i,j} &\approx r_i \frac{1}{z_{j+1/2} - z_{j-1/2}} \left(\left. \frac{\partial \mu \omega}{\partial z} \right|_{i,j+1/2} - \left. \frac{\partial \mu \omega}{\partial z} \right|_{i,j-1/2} \right) \\ &\approx r_i \frac{2}{z_{j+1} - z_{j-1}} \left(\frac{\mu_{i,j+1} \omega_{i,j+1} - \mu_{i,j} \omega_{i,j}}{z_{j+1} - z_j} - \frac{\mu_{i,j} \omega_{i,j} - \mu_{i,j-1} \omega_{i,j-1}}{z_j - z_{j-1}} \right). \end{aligned}$$

(iii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\left. \frac{\mu \omega}{r} \right|_{i,j} \approx \frac{\mu_{i,j} \omega_{i,j}}{r_i}.$$

(iv) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} &\left(r \rho V_r \frac{\partial \omega}{\partial r} + r \rho V_z \frac{\partial \omega}{\partial z} - \rho V_r \omega \right) \Big|_{i,j} \\ &\approx r_i \rho_{i,j} \left[\max \left\{ \frac{(V_r)_{i,j} + (V_r)_{i-1,j}}{2}, 0 \right\} \frac{\omega_{i,j} - \omega_{i-1,j}}{r_i - r_{i-1}} \right. \\ &\quad \left. - \max \left\{ -\frac{(V_r)_{i+1,j} + (V_r)_{i,j}}{2}, 0 \right\} \frac{\omega_{i+1,j} - \omega_{i,j}}{r_{i+1} - r_i} \right] \\ &\quad + r_i \rho_{i,j} \left[\max \left\{ \frac{(V_z)_{i,j} + (V_z)_{i,j-1}}{2}, 0 \right\} \frac{\omega_{i,j} - \omega_{i,j-1}}{z_j - z_{j-1}} \right. \\ &\quad \left. - \max \left\{ -\frac{(V_z)_{i,j+1} + (V_z)_{i,j}}{2}, 0 \right\} \frac{\omega_{i,j+1} - \omega_{i,j}}{z_{j+1} - z_j} \right] - \rho_{i,j} (V_r)_{i,j} \omega_{i,j}. \end{aligned}$$

(v) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned} r \bar{\nabla} \rho \cdot \nabla \frac{V^2}{2} \Big|_{i,j} &= r \frac{\partial \rho}{\partial z} \left(V_r \frac{\partial V_r}{\partial r} + V_z \frac{\partial V_z}{\partial r} \right) \Big|_{i,j} - r \frac{\partial \rho}{\partial r} \left(V_r \frac{\partial V_r}{\partial z} + V_z \frac{\partial V_z}{\partial z} \right) \Big|_{i,j} \\ &\approx r_i \frac{\rho_{i,j+1} - \rho_{i,j-1}}{z_{j+1} - z_{j-1}} \left[(V_r)_{i,j} \frac{(V_r)_{i+1,j} - (V_r)_{i-1,j}}{r_{i+1} - r_{i-1}} + (V_z)_{i,j} \frac{(V_z)_{i+1,j} - (V_z)_{i-1,j}}{r_{i+1} - r_{i-1}} \right] \\ &\quad - r_i \frac{\rho_{i+1,j} - \rho_{i-1,j}}{r_{i+1} - r_{i-1}} \left[(V_r)_{i,j} \frac{(V_r)_{i,j+1} - (V_r)_{i,j-1}}{z_{j+1} - z_{j-1}} + (V_z)_{i,j} \frac{(V_z)_{i,j+1} - (V_z)_{i,j-1}}{z_{j+1} - z_{j-1}} \right]. \end{aligned}$$

(vi) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$r \bar{\nabla} \rho \cdot g \Big|_{i,j} = -r g_z \frac{\partial \rho}{\partial r} \Big|_{i,j} \approx (980.65) r_i \frac{\rho_{i+1,j} - \rho_{i-1,j}}{r_{i+1} - r_{i-1}}.$$

(vii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

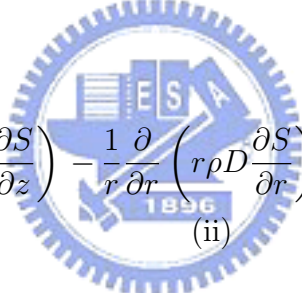
$$\begin{aligned} &2r (\bar{\nabla}(\text{div}(V)) \cdot \nabla \mu) \Big|_{i,j} \\ &= 2r \left\langle \frac{\partial}{\partial z} \left(\frac{1}{r} \frac{\partial}{\partial r} (r V_r) + \frac{\partial}{\partial z} V_z \right), -\frac{\partial}{\partial r} \left(\frac{1}{r} \frac{\partial}{\partial r} (r V_r) + \frac{\partial}{\partial z} V_z \right) \right\rangle \cdot \left\langle \frac{\partial \mu}{\partial r}, \frac{\partial \mu}{\partial z} \right\rangle \Big|_{i,j} \\ &= 2r \frac{\partial \mu}{\partial r} \left(\frac{\partial^2 V_r}{\partial z \partial r} + \frac{\partial^2 V_z}{\partial z^2} + \frac{1}{r} \frac{\partial V_r}{\partial z} \right) \Big|_{i,j} - 2r \frac{\partial \mu}{\partial z} \left(\frac{\partial^2 V_z}{\partial r \partial z} + \frac{\partial^2 V_r}{\partial r^2} + \frac{1}{r} \frac{\partial V_r}{\partial r} - \frac{V_r}{r^2} \right) \Big|_{i,j} \\ &\approx 2r_i \frac{\mu_{i+1,j} - \mu_{i-1,j}}{r_{i+1} - r_{i-1}} \left[\frac{(V_r)_{i+1,j+1} - (V_r)_{i+1,j-1} - (V_r)_{i-1,j+1} + (V_r)_{i-1,j-1}}{(r_{i+1} - r_{i-1})(z_{j+1} - z_{j-1})} \right. \\ &\quad \left. + \frac{2}{z_{j+1} - z_{j-1}} \left(\frac{(V_z)_{i,j+1} - (V_z)_{i,j}}{z_{j+1} - z_j} - \frac{(V_z)_{i,j} - (V_z)_{i,j-1}}{z_j - z_{j-1}} \right) + \frac{1}{r_i} \frac{(V_r)_{i,j+1} - (V_r)_{i,j-1}}{z_{j+1} - z_{j-1}} \right] \\ &\quad - 2r_i \frac{\mu_{i,j+1} - \mu_{i,j-1}}{z_{j+1} - z_{j-1}} \left[\frac{(V_z)_{i+1,j+1} - (V_z)_{i+1,j-1} - (V_z)_{i-1,j+1} + (V_z)_{i-1,j-1}}{(r_{i+1} - r_{i-1})(z_{j+1} - z_{j-1})} \right. \\ &\quad \left. + \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{(V_r)_{i+1,j} - (V_r)_{i,j}}{r_{i+1} - r_i} - \frac{(V_r)_{i,j} - (V_r)_{i-1,j}}{r_i - r_{i-1}} \right) + \frac{1}{r_i} \frac{(V_r)_{i+1,j} - (V_r)_{i-1,j}}{r_{i+1} - r_{i-1}} - \frac{(V_r)_{i,j}}{r_i^2} \right]. \end{aligned}$$

(viii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned}
& 2r \left(\nabla V_r \cdot \bar{\nabla} \frac{\partial \mu}{\partial r} + \nabla V_z \cdot \bar{\nabla} \frac{\partial \mu}{\partial z} \right) \Big|_{i,j} \\
&= 2r \left(\left\langle \frac{\partial V_r}{\partial r}, \frac{\partial V_r}{\partial z} \right\rangle \cdot \left\langle \frac{\partial^2 \mu}{\partial z \partial r}, -\frac{\partial^2 \mu}{\partial r^2} \right\rangle + \left\langle \frac{\partial V_z}{\partial r}, \frac{\partial V_z}{\partial z} \right\rangle \cdot \left\langle \frac{\partial^2 \mu}{\partial z^2}, -\frac{\partial^2 \mu}{\partial r \partial z} \right\rangle \right) \Big|_{i,j} \\
&= 2r \left[\frac{\partial^2 \mu}{\partial z^2} \frac{\partial V_z}{\partial r} - \frac{\partial^2 \mu}{\partial r^2} \frac{\partial V_r}{\partial z} + \frac{\partial^2 \mu}{\partial z \partial r} \left(\frac{\partial V_r}{\partial r} - \frac{\partial V_z}{\partial z} \right) \right] \Big|_{i,j} \\
&\approx 2r_i \left[\frac{2}{z_{j+1} - z_{j-1}} \left(\frac{\mu_{i,j+1} - \mu_{i,j}}{z_{j+1} - z_j} - \frac{\mu_{i,j} - \mu_{i,j-1}}{z_j - z_{j-1}} \right) \frac{(V_z)_{i+1,j} - (V_z)_{i-1,j}}{r_{i+1} - r_{i-1}} \right. \\
&\quad - \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{\mu_{i+1,j} - \mu_{i,j}}{r_{i+1} - r_i} - \frac{\mu_{i,j} - \mu_{i-1,j}}{r_i - r_{i-1}} \right) \frac{(V_r)_{i,j+1} - (V_r)_{i,j-1}}{z_{j+1} - z_{j-1}} \\
&\quad \left. + \frac{\mu_{i+1,j+1} - \mu_{i+1,j-1} - \mu_{i-1,j+1} + \mu_{i-1,j-1}}{(r_{i+1} - r_{i-1})(z_{j+1} - z_{j-1})} \left(\frac{(V_r)_{i+1,j} - (V_r)_{i-1,j}}{r_{i+1} - r_{i-1}} - \frac{(V_z)_{i,j+1} - (V_z)_{i,j-1}}{z_{j+1} - z_{j-1}} \right) \right].
\end{aligned}$$

Conserved Scalar :

$$\left(\rho V_r \frac{\partial S}{\partial r} + \rho V_z \frac{\partial S}{\partial z} \right) - \frac{1}{r} \frac{\partial}{\partial r} \left(r \rho D \frac{\partial S}{\partial r} \right) - \frac{\partial}{\partial z} \left(\rho D \frac{\partial S}{\partial z} \right) = 0.$$

(i)

(iii)

(i) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned}
& \left(\rho V_r \frac{\partial S}{\partial r} + \rho V_z \frac{\partial S}{\partial z} \right) \Big|_{i,j} \\
&\approx \rho_{i,j} \left[\max \left\{ \frac{(V_r)_{i,j} + (V_r)_{i-1,j}}{2}, 0 \right\} \frac{S_{i,j} - S_{i-1,j}}{r_i - r_{i-1}} \right. \\
&\quad - \max \left\{ -\frac{(V_r)_{i+1,j} + (V_r)_{i,j}}{2}, 0 \right\} \frac{S_{i+1,j} - S_{i,j}}{r_{i+1} - r_i} \Big] \\
&\quad + \rho_{i,j} \left[\max \left\{ \frac{(V_z)_{i,j} + (V_z)_{i,j-1}}{2}, 0 \right\} \frac{S_{i,j} - S_{i,j-1}}{z_j - z_{j-1}} \right. \\
&\quad \left. - \max \left\{ -\frac{(V_z)_{i,j+1} + (V_z)_{i,j}}{2}, 0 \right\} \frac{S_{i,j+1} - S_{i,j}}{z_{j+1} - z_j} \right].
\end{aligned}$$

(ii) For $i = 3 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned}
& \left. \frac{1}{r} \frac{\partial}{\partial r} \left(r \rho D \frac{\partial S}{\partial r} \right) \right|_{i,j} = \left. \frac{1}{P_r} \frac{1}{r} \frac{\partial}{\partial r} \left(r \mu \frac{\partial S}{\partial r} \right) \right|_{i,j} \\
& \approx \frac{1}{P_r} \frac{1}{r_i} \frac{1}{r_{i+1/2} - r_{i-1/2}} \left(r \mu \frac{\partial S}{\partial r} \Big|_{i+1/2,j} - r \mu \frac{\partial S}{\partial r} \Big|_{i-1/2,j} \right) \\
& \approx \frac{1}{P_r} \frac{1}{r_i} \frac{2}{r_{i+1} - r_{i-1}} \left(\frac{r_{i+1} \mu_{i+1,j} + r_i \mu_{i,j}}{2} \frac{S_{i+1,j} - S_{i,j}}{r_{i+1} - r_i} - \frac{r_i \mu_{i,j} + r_{i-1} \mu_{i-1,j}}{2} \frac{S_{i,j} - S_{i-1,j}}{r_i - r_{i-1}} \right).
\end{aligned}$$

For $i = 2$ and $j = 2 \sim (m - 1)$

$$\begin{aligned}
& \left. \frac{1}{r} \frac{\partial}{\partial r} \left(r \rho D \frac{\partial S}{\partial r} \right) \right|_{2,j} = \left. \frac{1}{P_r} \frac{1}{r} \frac{\partial}{\partial r} \left(r \mu \frac{\partial S}{\partial r} \right) \right|_{2,j} \\
& \approx \frac{1}{P_r} \frac{1}{r_2} \frac{1}{r_{2+1/2} - r_1} \left(r \mu \frac{\partial S}{\partial r} \Big|_{2+1/2,j} - r \mu \frac{\partial S}{\partial r} \Big|_{1,j} \right) \quad \text{where } \left. \frac{\partial S}{\partial r} \right|_{1,j} = 0 \\
& \approx \frac{1}{P_r} \frac{1}{r_2} \frac{2}{r_3 + r_2} \left(\frac{r_3 + r_2}{2} \frac{\mu_{3,j} + \mu_{2,j}}{2} \frac{S_{3,j} - S_{2,j}}{r_3 - r_2} \right) = \frac{1}{P_r} \frac{1}{r_2} \left(\frac{\mu_{3,j} + \mu_{2,j}}{2} \frac{S_{3,j} - S_{2,j}}{r_3 - r_2} \right).
\end{aligned}$$

(iii) For $i = 2 \sim (n - 1)$ and $j = 2 \sim (m - 1)$

$$\begin{aligned}
& \left. \frac{\partial}{\partial z} \left(\rho D \frac{\partial S}{\partial z} \right) \right|_{i,j} = \left. \frac{1}{P_r} \frac{\partial}{\partial z} \left(\mu \frac{\partial S}{\partial z} \right) \right|_{i,j} \\
& \approx \frac{1}{P_r} \frac{1}{z_{j+1/2} - z_{j-1/2}} \left(\mu \frac{\partial S}{\partial z} \Big|_{i,j+1/2} - \mu \frac{\partial S}{\partial z} \Big|_{i,j+1/2} \right) \\
& \approx \frac{1}{P_r} \frac{2}{z_{j+1} - z_{j-1}} \left(\frac{\mu_{i,j+1} + \mu_{i,j}}{2} \frac{S_{i,j+1} - S_{i,j}}{z_{j+1} - z_j} - \frac{\mu_{i,j} + \mu_{i,j-1}}{2} \frac{S_{i,j} - S_{i,j-1}}{z_j - z_{j-1}} \right).
\end{aligned}$$

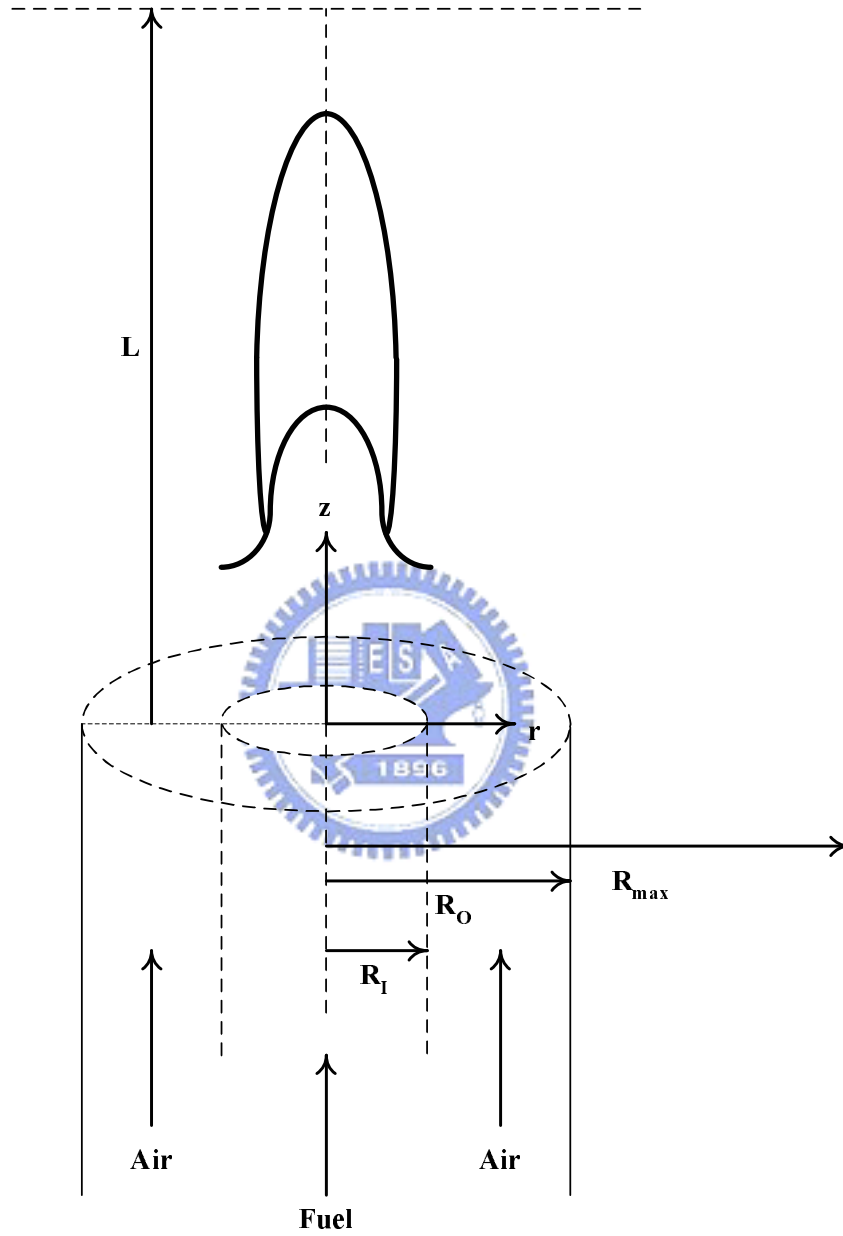


Figure 1: Physical configuration for diffusion flame model (not in scale)

Outer Boundary ($r = R_{\max}$) :

$$\begin{aligned} \frac{\partial V_r}{\partial r} = 0, & \quad \frac{\partial V_z}{\partial r} = 0, & \quad \omega = \frac{\partial V_r}{\partial z}, & \quad S = 0. \\ \text{(i)} & \quad \text{(ii)} & \quad \text{(iii)} & \quad \text{(iv)} \end{aligned}$$

(i) For $i = n$ and $j = 1 \sim m$

$$\frac{\partial V_r}{\partial r} \approx \frac{(V_r)_{n,j} - (V_r)_{n-1,j}}{r_n - r_{n-1}}.$$

(ii) For $i = n$ and $j = 1 \sim m$

$$\frac{\partial V_z}{\partial r} \approx \frac{(V_z)_{n,j} - (V_z)_{n-1,j}}{r_n - r_{n-1}}.$$

(iii) For $i = n$ and $j = 1$ or m

$$\frac{\partial \omega}{\partial r} \approx \frac{\omega_{n,j} - \omega_{n-1,j}}{r_n - r_{n-1}}.$$

For $i = n$ and $j = 2 \sim (m-1)$

$$\omega - \frac{\partial V_r}{\partial z} \approx \omega_{n,j} - \frac{(V_r)_{n,j+1} - (V_r)_{n,j-1}}{z_{j+1} - z_{j-1}}.$$

(iv) For $i = n$ and $j = 1 \sim m$

$$S \approx S_{n,j}.$$

Axis of Symmetry ($r = 0$) :

$$\begin{aligned} V_r = 0, & \quad \frac{\partial V_z}{\partial r} = 0, & \quad \omega = 0, & \quad \frac{\partial S}{\partial r} = 0. \\ \text{(i)} & \quad \text{(ii)} & \quad \text{(iii)} & \quad \text{(iv)} \end{aligned}$$

(i) For $i = 1$ and $j = 2 \sim (m-1)$

$$V_r \approx (V_r)_{1,j}.$$

(ii) For $i = 1$ and $j = 2 \sim (m - 1)$

We use the result (ii) to deal with axial velocity equation.

$$\begin{aligned} \frac{\partial^2 V_z}{\partial r^2} + \frac{\partial^2 V_z}{\partial z^2} + \frac{\partial \omega}{\partial r} + \frac{\partial}{\partial z} \left(\frac{V_z}{\rho} \frac{\partial \rho}{\partial z} \right) &\approx 2 \frac{(V_z)_{2,j} - (V_z)_{1,j}}{(r_2 - r_1)^2} \\ &+ \frac{2}{z_{j+1} - z_{j-1}} \left[\frac{(V_z)_{1,j+1} - (V_z)_{1,j}}{z_{j+1} - z_j} - \frac{(V_z)_{1,j} - (V_z)_{1,j-1}}{z_j - z_{j-1}} \right] + \frac{\omega_{2,j}}{r_2 - r_1} \\ &+ \frac{1}{z_{j+1} - z_{j-1}} \left[\left(\frac{(V_z)_{1,j+1}}{\rho_{1,j}} + \frac{(V_z)_{1,j}}{\rho_{1,j+1}} \right) \frac{\rho_{1,j+1} - \rho_{1,j}}{z_{j+1} - z_j} - \left(\frac{(V_z)_{1,j-1}}{\rho_{1,j}} + \frac{(V_z)_{1,j}}{\rho_{1,j-1}} \right) \frac{\rho_{1,j} - \rho_{1,j-1}}{z_j - z_{j-1}} \right]. \end{aligned}$$

(iii) For $i = 1$ and $j = 2 \sim (m - 1)$

$$\omega \approx \omega_{1,j}.$$

(iv) For $i = 1$ and $j = 2 \sim (m - 1)$

$$\frac{\partial S}{\partial r} \approx \frac{\omega_{2,j} - \omega_{1,j}}{r_2 - r_1}.$$

Inlet Boundary ($z = 0$) :

$$\begin{aligned} V_r = 0, \quad V_z = V_z^0(r), \quad \omega = \frac{\partial V_r}{\partial z} - \frac{\partial V_z}{\partial r}, \quad S = S^0(r). \end{aligned}$$

(i)
(ii)
(iii)
(iv)

(i) For $i = 1 \sim (n - 1)$ and $j = 1$

$$V_r \approx (V_r)_{i,1}.$$

(ii) For $i = 1 \sim (n - 1)$ and $j = 1$

$$V_z - V_z^0(r) \approx (V_z)_{i,1} - V_z^0(r_i).$$

(iii) For $i = 1$ and $j = 1$

$$\omega \approx \omega_{i,1}.$$

For $i = 2 \sim (n - 1)$ and $j = 1$

$$\omega - \frac{\partial V_r}{\partial z} + \frac{\partial V_z}{\partial r} \approx \frac{\omega_{i,1} + \omega_{i,2}}{2} - \frac{(V_r)_{i,2}}{z_2 - z_1} + \frac{(V_z)_{i+1,1} - (V_z)_{i-1,1}}{r_{i+1} - r_{i-1}}.$$

(iv) For $i = 1 \sim (n - 1)$ and $j = 1$

$$S - S^0(r) \approx S_{i,1} - S^0(r_i).$$

Outlet Boundary ($z = L$) :

$$\begin{array}{cccc} V_r = 0, & \frac{\partial V_z}{\partial z} = 0, & \frac{\partial \omega}{\partial z} = 0, & \frac{\partial S}{\partial z} = 0. \\ \text{(i)} & \text{(ii)} & \text{(iii)} & \text{(iv)} \end{array}$$

(i) For $i = 1 \sim (n - 1)$ and $j = m$

$$V_r \approx (V_r)_{i,m}.$$

(ii) For $i = 1 \sim (n - 1)$ and $j = m$

$$\frac{\partial V_z}{\partial z} \approx \frac{(V_z)_{i,m} - (V_z)_{i,m-1}}{z_m - z_{m-1}}.$$

(iii) For $i = 1$ and $j = m$

$$\omega \approx \omega_{i,m}.$$

For $i = 2 \sim (n - 1)$ and $j = m$

$$\frac{\partial \omega}{\partial z} \approx \frac{\omega_{i,m} - \omega_{i,m-1}}{z_m - z_{m-1}}.$$

(iv) For $i = 1 \sim (n - 1)$ and $j = m$

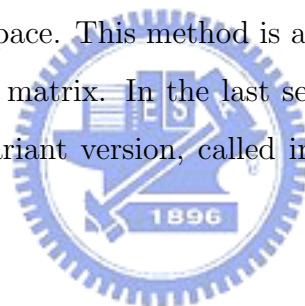
$$\frac{\partial S}{\partial z} \approx \frac{S_{i,m} - S_{i,m-1}}{z_m - z_{m-1}}.$$

2 Krylov Subspace Methods

A Krylov subspace method is a method for which the subspace $\mathcal{K}_m(A, r_0)$ of \mathbb{R}^n is the Krylov subspace with the form

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$ be initial residual in general and x_0 be initial guess. The Krylov subspace has well property that matrix-vector multiplication of basis is cheap to compute. In other words, if we given a basis for \mathcal{K}_m , then we can cheaply compute a basis for \mathcal{K}_{m+1} . In this chapter for a start, we introduce the general projection methods, namely orthogonal projection or oblique projection. Then we will show two optimal results. One is orthogonal projection just to minimize the A -norm of the error when A is symmetric positive definite. The other is oblique projection just to minimize the 2-norm of the residual when A be an arbitrary square matrix. In the second section, we introduce the Arnoldi's method to construct the Krylov subspace. This method is an orthogonal projection method onto \mathcal{K} for general non-Hermitian matrix. In the last section, we introduce the full orthogonalization method and its variant version, called incomplete orthogonalization method, to apply to linear system.



2.1 Projection Methods

Consider the linear system

$$Ax = b \tag{2.1}$$

where A is an $n \times n$ real matrix (or sparse matrix). We will use projection method for extracting an approximation to the solution of a linear system. This idea is to restrict the next step in an iterative method to a small subspace but pick “best” step in that subspace. In order to reach our goal as stated above, we find methods which can cheaply find the next iterate as far as possible but still minimize some measure of the error or residual at each step.

Let \mathcal{K} and \mathcal{L} are two m -dimensional subspaces of \mathbb{R}^n . A projection technique onto the subspace \mathcal{K} and orthogonal to \mathcal{L} is just to find an approximate solution \tilde{x} to (2.1) by

imposing two conditions. One is to find \tilde{x} belong to \mathcal{K} and the other is making residual vector must be orthogonal to \mathcal{L} . Now suppose we have current guess x_0 and let initial residual vector $r_0 = b - Ax_0$. The projection method is repressed as

$$\text{Find } \tilde{x} \in x_0 + \mathcal{K}, \quad \text{such that } \tilde{r} = b - A\tilde{x} \perp \mathcal{L}.$$

or equivalently

$$\text{Find } \delta \in \mathcal{K}, \quad \text{such that } \tilde{r} = b - A(x_0 + \delta) = r_0 - A\delta \perp \mathcal{L}. \quad (2.2)$$

Let $V = [v_1 \cdots v_m]$ and $W = [w_1 \cdots w_m]$ are $n \times m$ matrix whose column-vectors form a basis for \mathcal{K} and \mathcal{L} , respectively. Then $\tilde{x} = x_0 + \delta = x_0 + Vy$ for some $y \in \mathbb{R}^m$. Thus, we can transform (2.2) into matrix representation as following

$$W^T AVy = W^T r_0 \quad (2.3)$$

If we can find $W^T AV$ is nonsingular, then (2.3) has unique solution. For this purpose, the following proposition describes two ideal cases.

Proposition 1. *Let A , \mathcal{L} , and \mathcal{K} satisfy either one of the two following conditions,*

- i. A is positive definite and $\mathcal{L} = \mathcal{K}$, or*
- ii. A is nonsingular and $\mathcal{L} = A\mathcal{K}$.*

Then the matrix $B = W^T AV$ is nonsingular for any bases V and W of \mathcal{K} and \mathcal{L} , respectively.

Proof. To prove first case. Let V and W be any basis of \mathcal{K} and \mathcal{L} , respectively. Since \mathcal{L} and \mathcal{K} are the same, we let $W = VG$, where G is a $m \times m$ nonsingular matrix. Then

$$B = W^T AV = G^T V^T AV.$$

Because $V^T AV$ is positive definite, then we shows that B is nonsingular.

Consider the second case. Since $\mathcal{L} = A\mathcal{K}$, we let $W = AVG$, where G is a $m \times m$ nonsingular matrix. Then

$$B = W^T AV = G^T (AV)^T AV.$$

Since A is nonsingular, the $n \times m$ matrix AV is full rank. Then we have $(AV)^T AV$ is nonsingular. So we have a result. ■

Suppose **Proposition 1.** hold, then the approximate solution \tilde{x} can be repressed as

$$\tilde{x} = x_0 + Vy = x_0 + V(W^T AV)^{-1}W^T r_0.$$

A question is how much the quality of the approximate solution obtained from a general projection method? In order to answer this problem, two optimal results will be established.

Proposition 2. *Assume that A is symmetric positive definite and $\mathcal{L} = \mathcal{K}$. Then a vector \tilde{x} is the result of an (orthogonal) projection method onto \mathcal{K} with the starting vector x_0 if and only if it minimizes the A -norm of the error over $x_0 + \mathcal{K}$, that is,*

$$E(\tilde{x}) = \min_{x \in x_0 + \mathcal{K}} E(x),$$

where

$$E(x) \equiv (A(x_* - x), x_* - x)^{1/2} \equiv \|x_* - x\|_A.$$

Proof. Let $\tilde{x} = x_0 + \tilde{\delta}$, where $\tilde{\delta} \in \mathcal{K}$. Then

$$\begin{aligned} \min_{x \in x_0 + \mathcal{K}} \|x_* - x\|_A &= \min_{\delta \in \mathcal{K}} \|x_* - (x_0 + \delta)\|_A \\ &= \min_{\delta \in \mathcal{K}} \|d_0 - \delta\|_A \quad (\text{where } d_0 = x_* - x_0) \\ &= \|d_0 - \tilde{\delta}\|_A \end{aligned}$$

Therefore, we have

$$d_0 - \tilde{\delta} \perp_A \mathcal{K}.$$

It would be better to say that $\tilde{\delta}$ is the A -orthogonal projection of d_0 onto \mathcal{K} . ■

Proposition 3. *Let A be an arbitrary square matrix and assume that $\mathcal{L} = A\mathcal{K}$. Then a vector \tilde{x} is the result of an (oblique) projection method onto \mathcal{K} orthogonally to \mathcal{L} with the starting vector x_0 if and only if it minimizes the 2-norm of the residual vector $b - Ax$ over $x \in x_0 + \mathcal{K}$, that is,*

$$R(\tilde{x}) = \min_{x \in x_0 + \mathcal{K}} R(x),$$

where

$$R(x) \equiv \|b - Ax\|_2.$$

Proof. Let $\tilde{x} = x_0 + \tilde{\delta}$, where $\tilde{\delta} \in \mathcal{K}$. Then

$$\begin{aligned} \min_{x \in x_0 + \mathcal{K}} \|b - Ax\|_2 &= \min_{\delta \in \mathcal{K}} \|b - A(x_0 + \delta)\|_2 \\ &= \min_{A\delta \in A\mathcal{K}} \|r_0 - A\delta\|_2 \quad (\text{where } r_0 = b - Ax_0) \\ &= \|r_0 - A\tilde{\delta}\|_2 \end{aligned}$$

Therefore, we have

$$r_0 - A\tilde{\delta} \perp A\mathcal{K} = \mathcal{L}.$$

It would be better to say that $A\tilde{\delta}$ is the orthogonal projection of r_0 onto $A\mathcal{K}$. ■

By **Proposition 2.**, the result of the projection process can be interpreted as orthogonal projector acts on the initial error. The same is true of the **Proposition 3.** can be interpreted as oblique projector acts on the initial residual. The following properties will state conclusions from the above properties.

Proposition 4. *Let \tilde{x} be the approximate solution obtained from an orthogonal projection process onto \mathcal{K} , and let $\tilde{d} = x_* - \tilde{x}$ be the associated error vector. Then,*

$$\tilde{d} = (I - P_A)d_0,$$

where P_A denotes the projector onto the subspace \mathcal{K} , which is orthogonal with respect to the A -inner product.

Proof. By the result of **Proposition 2.** ■

Proposition 5. *Let \tilde{x} be the approximate solution obtained from a projection process onto \mathcal{K} orthogonally to $\mathcal{L} = A\mathcal{K}$, and let $\tilde{r} = b - A\tilde{x}$ be the associated residual. Then,*

$$\tilde{r} = (I - P)r_0,$$

where P denotes the orthogonal projector onto the subspace $A\mathcal{K}$.

Proof. By the result of **Proposition 3.** ■

2.2 Arnoldi's Method

We know that Schur factorization reduces a dense matrix A into upper triangular matrix U by applying unitary matrix V . In natural thought, the unitary matrix V is made up of Householder reflectors, and let $V = V_1V_2 \cdots V_m$. Then we have $V^*AV = U$. In the first step, V_1^* multiplied on the left of A and V_1 multiplied on the right of A . Note that V_1^* will change all rows of A and V_1 will change all columns of A . The entries are changed at each step and we write in boldface as following diagrams:

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \longrightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \end{pmatrix} \longrightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix}.$$

$A \qquad V_1^*A \qquad V_1^*AV_1$

By the same way to apply to other Householder reflectors, we can find this idea had to fail. Fortunately, Arnoldi's method suggest a good idea for us to reduce A to Hessenberg form. At the first step, we select Householder reflector V_2^* that leaves the first row unchanged. In other words, we let $\tilde{V} = V_2V_3 \cdots V_m$ and omit the reflector V_1 . We show the first step as following diagrams:

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} \longrightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \\ \mathbf{0} & \times & \times & \times & \times \end{pmatrix} \longrightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix}.$$

$A \qquad V_2^*A \qquad V_2^*AV_2$

Repeating this process by the same way on $V_2^*AV_2$ by V_3, \dots, V_m , we have Hessenberg form matrix

$$\tilde{V}^*A\tilde{V} = (V_m^* \cdots V_3^*V_2^*)A(V_2V_3 \cdots V_m) = H_m.$$

Arnoldi's method [1] is an orthogonal projection method onto \mathcal{K}_m for general non-Hermitian matrices. This basic idea is reducing a dense matrix into Hessenberg form by above way. For a start, we use standard Gram-Schmidt method to construct matrix V . The Arnoldi algorithm is stated as follow:

ALGORITHM 1. *Arnoldi*

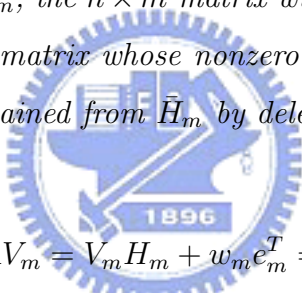
-
1. Choose a vector v_1 of norm 1
 2. For $j = 1, 2, \dots, m$ Do:
 3. $h_{ij} = (Av_j, v_i)$ for $i = 1, 2, \dots, j$
 4. $w_j = Av_j - \sum_{i=1}^j h_{ij}v_i$
 5. $h_{j+1,j} = \|w_j\|_2$
 6. If $h_{j+1,j} = 0$ then stop
 7. $v_{j+1} = w_j/h_{j+1,j}$
 8. EndDo
-

Proposition 6. Assume that Algorithm 1. does not stop before the m -th step. Then the vectors v_1, v_2, \dots, v_m form an orthonormal basis of the Krylov subspace

$$\mathcal{K}_m = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}.$$

Proof. Follow from the fact of steps 4, 5, and 7 of **Algorithm 1**. ■

Proposition 7. Denote by V_m , the $n \times m$ matrix with column vectors v_1, \dots, v_m , by \bar{H}_m , the $(m+1) \times m$ Hessenberg matrix whose nonzero entries h_{ij} are defined by Algorithm 1., and by H_m the matrix obtained from \bar{H}_m by deleting its last row. Then the following relations hold:



$$\begin{aligned} AV_m &= V_m H_m + w_m e_m^T = V_{m+1} \bar{H}_m, \\ V_m^T AV_m &= H_m, \end{aligned}$$

where $w_m = h_{m+1,m}v_{m+1}$.

Proof. From the fact of **Algorithm 1**, we have

$$Av_j = \sum_{i=1}^{j+1} h_{ij}v_i, \quad j = 1, 2, \dots, m.$$

Then we have all relations. ■

By **Algorithm 1**, we know that Arnoldi's method uses standard Gram-Schmidt orthonormalization to make V_m . In practice, the calculations of Gram-Schmidt formulas turn out to be numerically unstable. We can gain simple modification by using the modified Gram-Schmidt algorithm instead of the standard Gram-Schmidt algorithm. Let $A = [a_1, \dots, a_n]$

with columns $\{a_j\}$ and P_j be an $n \times n$ orthogonal projector, projects the vector orthogonally onto the space orthogonal to $\langle v_1, \dots, v_{j-1} \rangle$, of rank $n - (j - 1)$ such that

$$v_1 = \frac{P_1 a_1}{\|P_1 a_1\|}, \quad v_2 = \frac{P_2 a_2}{\|P_2 a_2\|}, \quad \dots, \quad v_j = \frac{P_j a_j}{\|P_j a_j\|},$$

with $P_1 = I_n$. It is not difficult to see that

$$P_j = P_{\perp v_{j-1}} \cdots P_{\perp v_2} P_{\perp v_1}.$$

Then the standard Gram-Schmidt algorithm just to do a single orthogonal projection,

$$v_j = P_j a_j.$$

However, the modified Gram-Schmidt algorithm uses successive orthogonal projections,

$$v_j = P_{\perp v_{j-1}} \cdots P_{\perp v_2} P_{\perp v_1} a_j$$

Therefore, we know that Gram-Schmidt and modified Gram-Schmidt are equivalent in mathematics. The modified Gram-Schmidt's version of Arnoldi is as follow:

ALGORITHM 2. *Arnoldi-Modified Gram-Schmidt*

1. Choose a vector v_1 of norm 1
2. For $j = 1, 2, \dots, m$ Do:
3. $w_j = Av_j$
4. For $i = 1, \dots, j$ Do:
5. $h_{ij} = (w_j, v_i)$
6. $w_j = w_j - h_{ij}v_i$
7. EndDo
8. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ then stop
9. $v_{j+1} = w_j/h_{j+1,j}$
10. EndDo

2.3 FOM, IOM, and DIOM

Let us return to our main subject, to solve the linear system $Ax = b$, we want to find approximate solution \tilde{x} cheaply. By the Arnoldi's method, we can apply this method to three types and obtain our goal in this section. The first type is directly applying algorithm of Arnoldi, called full orthogonalization method or FOM. It should be said with some emphasis that we use modified Gram-Schmidt process instead of Gram-Schmidt process from now on. The second type is applying incomplete orthogonalization process in FOM, called incomplete orthogonalization method or IOM. And the last type is improving IOM by using LU factorization, called direct incomplete orthogonalization or DIOM.

2.3.1 FOM

Now, given an initial guess x_0 to the linear system $Ax = b$, we consider an orthogonal projection method, $\mathcal{L} = \mathcal{K} = \mathcal{K}_m(A, r_0)$,

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2r_0, \dots, A^{m-1}r_0\},$$

where $r_0 = b - Ax_0$. Taking $v_1 = r_0/\beta$ and $\beta = \|r_0\|_2$ in Arnoldi's method, then we get

$$V_m^T AV_m = H_m$$

and by (2.2), we have

$$V_m^T(r_0 - AV_m y_m) = 0.$$

Then

$$y_m = H_m^{-1}(V_m^T r_0) = H_m^{-1}(\beta e_1). \quad (2.4)$$

The approximate solution is given by

$$x_m = x_0 + V_m y_m. \quad (2.5)$$

The process as above is called the full orthogonalization method (FOM). Let me stress again that we use modified Gram-Schmidt method to make V_m and the algorithm is stated as follow:

ALGORITHM 3. FOM

1. $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. Set $m \times m$ matrix $H_m = 0$
 3. For $j = 1, 2, \dots, m$ Do:
 4. $w_j = Av_j$
 5. For $i = 1, \dots, j$ Do:
 6. $h_{i,j} = (w_j, v_i)$
 7. $w_j = w_j - h_{i,j}v_i$
 8. EndDo
 9. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ then set $m = j$ and Goto 12
 10. $v_{j+1} = w_j/h_{j+1,j}$
 11. EndDo
 12. $y_m = H_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$
-

2.3.2 IOM

Sometimes we have a situation that our calculations may be dictated by computer's memory limitations. By FOM algorithm, the memory cost increases at least as $O(mn)$. As m increases, the largest value of m that can be used. We have two remedies to solve our problem. One remedy is to restart the FOM algorithm for reaching "small" m . And the other remedy is to truncate the Arnoldi algorithm, not using full orthogonalization. It is to say that we use incomplete orthogonalization process to make v_j and gain a banded Hessenberg matrix H_m with bandwidth $k + 1$, the number k maybe dictated by computer's memory limitations. The incomplete orthogonalization process with (2.4) and (2.5), called incomplete orthogonalization method (IOM), is performed as follow.

ALGORITHM 4. IOM

1. $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. Set $m \times m$ matrix $H_m = 0$
 3. For $j = 1, 2, \dots, m$ Do:
 4. $w = Av_j$
 5. For $i = \max\{1, j - k + 1\}, \dots, j$ Do:
 6. $h_{i,j} = (w, v_i)$
 7. $w = w - h_{i,j}v_i$
 8. EndDo
 9. $h_{j+1,j} = \|w\|_2$. If $h_{j+1,j} = 0$ then set $m = j$ and Goto 12
 10. $v_{j+1} = w/h_{j+1,j}$
 11. EndDo
 12. $y_m = H_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$
-

2.3.3 DIOM

We know that FOM and IOM algorithms use the same relation (2.4) to find approximate solution x_m . These methods must to compute the inverse of H_m , then we need to solve m -th linear system. To avoid computing the inverse problem, we can modify the algorithm of IOM. The direct incomplete orthogonalization method (DIOM) is derived from the structure of the LU factorization, $H_m = L_m U_m$, of the Hessenberg matrix H_m , which obtained from the IOM. We assume that no pivoting is used, then matrix L_m , entries $\{l_{ij}\}$, is unit lower bidiagonal and U_m , entries $\{u_{ij}\}$, is banded upper triangular, with k diagonals. Thus the approximate solution is given by

$$x_m = x_0 + V_m U_m^{-1} L_m^{-1} (\beta e_1). \quad (2.6)$$

Defining the matrix

$$P_m = [p_1, \dots, p_m] = V_m U_m^{-1} \quad (2.7)$$

and the vector

$$z_m = [z_1, \dots, z_m]^T = L_m^{-1} (\beta e_1). \quad (2.8)$$

By (2.7) and the structure of U_m , we have

$$\sum_{i=m-k+1}^m u_{im} p_i = v_m,$$

which show the vector p_m to be updated from the previous p_i 's and v_m , then

$$p_m = \frac{1}{u_{mm}} \left[v_m - \sum_{i=m-k+1}^{m-1} u_{im} p_i \right].$$

And by (2.8), because of the structure of L_m , we have

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix}$$

where

$$\zeta_m = -l_{m,m-1} \zeta_{m-1}$$

with $z_1 = [\beta]$ and $\zeta_1 = \beta$. Then (2.6) can be rewritten as

$$\begin{aligned} x_m &= x_0 + [P_{m-1}, p_m] \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix} \\ &= x_0 + P_{m-1} z_{m-1} + \zeta_m p_m \\ &= x_{m-1} + \zeta_m p_m. \end{aligned}$$

The algorithm of DIOM is stated as follow:

ALGORITHM 5. DIOM

1. Choose x_0 and $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. For $m = 1, 2, \dots$, until convergence Do:
 3. $w = Av_m$
 4. For $i = \max\{1, m - k + 1\}, \dots, m$ Do:
 5. $h_{i,m} = (w, v_i)$
 6. $w = w - h_{i,m}v_i$
 7. EndDo
 8. $h_{m+1,m} = \|w\|_2$ and $v_{m+1} = w/h_{m+1,m}$
 9. Update the LU factorization of H_m . If $u_{mm} = 0$ then Stop.
 10. $\zeta_m = \{ \text{if } m = 1 \text{ then } \beta, \text{ else } -l_{m,m-1}\zeta_{m-1} \}$
 11. $p_m = u_{mm}^{-1} (v_m - \sum_{i=m-k+1}^{m-1} u_{im}p_i)$ (for $i \leq 0$ set $u_{im}p_i = 0$)
 12. $x_m = x_{m-1} + \zeta_m p_m$
 13. EndDo
-



3 Biconjugate Gradient Stabilized Method

Conjugate gradient (CG) and biconjugate gradient (BCG) algorithm are derived in this chapter. These algorithms generate the optimal approximation from the Krylov subspace. When A is symmetric positive definite, CG can be derived from the symmetric Lanczos process. Contrary to symmetric form, non-symmetric matrix, we use two-sided Lanczos algorithm (Lanczos biorthogonalization) to construct biorthogonal bases for the Krylov subspaces corresponding to A and A^T . Then BCG can be derived from the Lanczos biorthogonalization procedure. To avoid using A^T in BCG, the conjugate gradient squared method (CGS) can be derived from BCG. The CGS can obtain faster convergent behavior than BCG for the same computational cost. Since the polynomials in CGS are square, then rounding errors tend to be more damaging than in the BCG. Finally, we improved the CGS algorithm such that can smoothen in the convergent behavior, called biconjugate gradient stabilized method (BICGSTAB).

3.1 Conjugate Gradient Method

The Conjugate Gradient method [8] is one of the best known iterative techniques for solving sparse symmetric positive definite linear systems. The Conjugate Gradient algorithm can be derived from the DIOM, for the case when A is symmetric positive definite. In the first, we derive the similar case of FOM when A is symmetric, called Lanczos method. Given an initial guess x_0 , we can find the approximate solution x_m by rewriting (2.4) and (2.5),

$$\begin{aligned} x_m &= x_0 + V_m y_m \\ y_m &= T_m^{-1}(\beta e_1), \end{aligned}$$

where T_m be tridiagonal matrix of the following form,

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \beta_{m-1} & \alpha_{m-1} & \beta_m & \\ & & & \beta_m & \alpha_m & \end{pmatrix}. \quad (3.1)$$

Then the Lanczos method for linear system can be stated as follow:

ALGORITHM 6. *Lanczos method for linear systems*

-
1. $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. For $j = 1, 2, \dots, m$ Do:
 3. $w_j = Av_j - \beta_j v_{j-1}$ (If $j = 1$ then set $\beta_1 v_0 = 0$)
 4. $\alpha_j = (w_j, v_j)$
 5. $w_j = w_j - \alpha_j v_j$
 6. $\beta_{j+1} = \|w_j\|_2$. If $\beta_{j+1} = 0$ then set $m = j$ and Goto 9
 7. $v_{j+1} = w_j/\beta_{j+1}$
 8. EndDo
 9. $T_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$, and $V_m = [v_1, \dots, v_m]$.
 10. $y_m = T_m^{-1}(\beta e_1)$ and $x_m = x_0 + V_m y_m$
-

We should notice that the steps 3 until 5 of algorithm of Lanczos method can correspond with the steps 4 until 8 of algorithm of FOM as A is symmetric positive definite. Now, we will follow the same steps as for DIOM to construct direct version of the Lanczos method. Let $T_m = L_m U_m$ be LU factorization of T_m , where L_m is unit lower bidiagonal and U_m is upper bidiagonal. To take a simple example,

$$T_m = \begin{pmatrix} 1 & & & & & \\ \lambda_2 & 1 & & & & \\ & & \ddots & & & \\ & & & \ddots & & \\ & & & & \lambda_{m-1} & 1 \\ & & & & & \lambda_m & 1 \end{pmatrix} \times \begin{pmatrix} \eta_1 & \beta_2 & & & & \\ & \eta_2 & \beta_3 & & & \\ & & \ddots & \ddots & & \\ & & & \eta_{m-1} & \beta_m & \\ & & & & & \eta_m \end{pmatrix}.$$

Then an approximate solution x_m is

$$x_m = x_0 + V_m T_m^{-1}(\beta e_1) = x_0 + V_m U_m^{-1} L_m^{-1}(\beta e_1).$$

Defining the matrix

$$P_m = [p_1, \dots, p_m] = V_m U_m^{-1}$$

and the vector

$$z_m = [z_1, \dots, z_m]^T = L_m^{-1} \beta e_1.$$

Because of the structure of U_m , we have

$$p_m = \eta_m^{-1} [v_m - \beta_m p_{m-1}].$$

Note that β_m is a scalar computed from the Lanczos algorithm, while η_m results from the m -th Gaussian elimination step on the tridiagonal matrix, that is,

$$\lambda_m = \frac{\beta_m}{\eta_{m-1}},$$

$$\eta_m = \alpha_m - \lambda_m \beta_m.$$

By the same way, because of the structure of L_m ,

$$z_m = \begin{bmatrix} z_{m-1} \\ \zeta_m \end{bmatrix},$$

in which $\zeta_m = -\lambda_m \zeta_{m-1}$. Therefore, we can update x_m as

$$x_m = x_0 + P_m z_m = x_{m-1} + \zeta_m p_m.$$

Then we have the direct version of the Lanczos algorithm, called D-Lanczos, as follow:

ALGORITHM 7. *D-Lanczos*

-
1. $r_0 = b - Ax_0$, $\zeta_1 = \beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. set $\lambda_1 = \beta_1 = 0$ and $p_0 = 0$
 3. For $m = 1, 2, \dots$, until convergence Do:
 4. $w = Av_m - \beta_m v_{m-1}$ and $\alpha_m = (w, v_m)$
 5. If $m > 1$ then $\lambda_m = \frac{\beta_m}{\eta_{m-1}}$ and $\zeta_m = -\lambda_m \zeta_{m-1}$
 6. $\eta_m = \alpha_m - \lambda_m \beta_m$
 7. $p_m = \eta_m^{-1}(v_m - \beta_m p_{m-1})$
 8. $x_m = x_{m-1} + \zeta_m p_m$
 9. If x_m has converged then stop
 10. $w = w - \alpha_m v_m$
 11. $\beta_{m+1} = \|w\|_2$ and $v_{m+1} = w/\beta_{m+1}$
 12. EndDo
-

Proposition 8. Let $r_m = b - Ax_m$, $m = 0, 1, \dots$, be the residual vectors produced by the Lanczos and the D-Lanczos algorithms and p_m , $m = 0, 1, \dots$, the auxiliary vectors produced by D-Lanczos. Then,

1. Each residual vector r_m is such that $r_m = \sigma_m v_{m+1}$ where σ_m is a certain scalar. As a result, the residual vectors are orthogonal to each other.
2. The auxiliary vectors p_i form an A -conjugate set, i.e., $(Ap_i, p_j) = 0$, for $i \neq j$.

Proof. We prove the first part, by **Proposition 7**.

$$\begin{aligned}
r_m &= b - Ax_m \\
&= b - A(x_0 + V_m y_m) \\
&= r_0 - (V_m T_m + \beta_{m+1} v_{m+1} e_m^T) y_m \\
&= \beta v_1 - V_m \beta e_1 - \beta_{m+1} e_m^T y_m v_{m+1} \\
&= -\beta_{m+1} e_m^T y_m v_{m+1}.
\end{aligned}$$

For the second part, we will claim that $P_m^T A P_m$ is a diagonal matrix.

$$\begin{aligned}
P_m^T A P_m &= U_m^{-T} V_m^T A V_m U_m^{-1} \\
&= U_m^{-T} T_m U_m^{-1} \\
&= U_m^{-T} L_m.
\end{aligned}$$

Note that $U_m^{-T} L_m$ is a lower triangular and A is symmetric, then $P_m^T A P_m$ must be a diagonal matrix. ■

Let us now return to CG algorithm. For the proposition given above, we can derive CG by imposing two conditions, one is r_j 's orthogonality and the other is p_j 's A -conjugacy. In D-Lanczos, we can find

$$x_{j+1} = x_j + \alpha_j p_j.$$

Then the residual r_{j+1} can be repressed as

$$r_{j+1} = r_j - \alpha_j A p_j. \tag{3.2}$$

Since the r_j 's are to be orthogonal each other, we have

$$(r_j - \alpha_j A p_j, r_j) = 0.$$

Therefore,

$$\alpha_j = \frac{(r_j, r_j)}{(A p_j, r_j)}. \tag{3.3}$$

Also by DIOM, we have that the next search direction p_{j+1} is a linear combination of r_{j+1} and p_j , it is

$$p_{j+1} = r_{j+1} + \beta_j p_j.$$

Thus,

$$(Ap_j, r_j) = (Ap_j, p_j - \beta_{j-1}p_{j-1}) = (Ap_j, p_j).$$

Then (3.3) becomes

$$\alpha_j = \frac{(r_j, r_j)}{(Ap_j, p_j)}.$$

Because Ap_j orthogonal to p_{j+1} , then

$$\beta_j = -\frac{(r_{j+1}, Ap_j)}{(p_j, Ap_j)}.$$

From (3.2) we have

$$Ap_j = -\frac{1}{\alpha_j}(r_{j+1} - r_j),$$

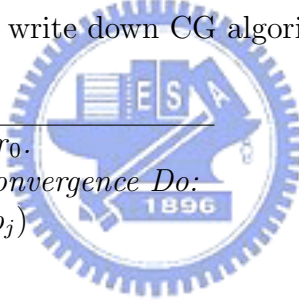
then

$$\beta_j = \frac{1}{\alpha_j} \frac{(r_{j+1}, (r_{j+1} - r_j))}{(Ap_j, p_j)} = \frac{(r_{j+1}, r_{j+1})}{(r_j, r_j)}.$$

By **Proposition 8.** and above manners, we can rewrite D-Lanczos algorithm to conjugate gradient (CG) algorithm. We write down CG algorithm as follow:

ALGORITHM 8. *CG*

1. $r_0 = b - Ax_0$, and $p_0 = r_0$.
2. For $j = 0, 1, \dots$, until convergence Do:
3. $\alpha_j = (r_j, r_j)/(Ap_j, p_j)$
4. $x_{j+1} = x_j + \alpha_j p_j$
5. $r_{j+1} = r_j - \alpha_j Ap_j$
6. $\beta_j = (r_{j+1}, r_{j+1})/(r_j, r_j)$
7. $p_{j+1} = r_{j+1} + \beta_j p_j$
8. EndDo



3.2 Lanczos Biorthogonalization

3.2.1 Two-Sided Lanczos

We know if the matrix A is symmetric matrix, then the Gram-Schmidt procedure for constructing an orthonormal basis for Krylov subspace of A reduces to three term recurrences. However, the matrix A is the case that A is non-symmetric in general. Fortunately, we can extend the symmetric Lanczos algorithm to the non-symmetric version, having three term recurrences. The Lanczos biorthogonalization use a pair of three term recurrences, one involving A and the other involving A^T , to construct biorthogonal bases for the Krylov spaces corresponding to A and A^T , respectively. Judging from the above, we have

$$\begin{aligned}\mathcal{K}_m(A, v_1) &= \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\} \\ \mathcal{K}_m(A^T, w_1) &= \text{span}\{w_1, A^T w_1, \dots, (A^T)^{m-1}w_1\},\end{aligned}$$

in which $(v_i, w_j) = 0$ for $i \neq j$. For the most part, we take $(v_i, w_i) = 1$. Then we show the algorithm as follow:

ALGORITHM 9. *The Lanczos Biorthogonalization Procedure*

-
1. Choose two vectors v_1, w_1 such that $(v_1, w_1) = 1$.
 2. $\beta_1 = \delta_1 = 0, w_0 = v_0 = 0$
 3. For $j = 1, 2, \dots, m$ Do:
 4. $\alpha_j = (Av_j, w_j)$
 5. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
 6. $\hat{w}_{j+1} = A^T w_j - \alpha_j w_j - \delta_j w_{j-1}$
 7. $\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}$. If $\delta_{j+1} = 0$ then stop
 8. $\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}$
 9. $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
 10. $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
 11. EndDo
-

Note that the **Algorithm 9** selects a manner to ensure that $(v_{j+1}, w_{j+1}) = 1$. In that case, it is a canonical choice to find two scalars $\beta_{j+1}, \delta_{j+1}$ such that satisfy the equality as following form

$$\delta_{j+1}\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1}).$$

If the algorithm does not break down, then we have T_m the tridiagonal matrix as below:

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix}.$$

The following proposition will interpret this result.

Proposition 9. *If the algorithm does not break down before step m , then the vector v_i , $i = 1, \dots, m$, and w_j , $j = 1, \dots, m$, form a biorthogonal system, i.e.,*

$$(v_j, w_i) = \delta_{i,j} \quad 1 \leq i, j \leq m.$$

Moreover, $\{v_i\}_{i=1,2,\dots,m}$ is a basis of $\mathcal{K}_m(A, v_1)$ and $\{w_i\}_{i=1,2,\dots,m}$ is a basis of $\mathcal{K}_m(A^T, w_1)$ and the following relations hold,

$$\begin{aligned} AV_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^T, \\ A^T W_m &= W_m T_m^T + \beta_{m+1} w_{m+1} e_m^T, \\ W_m^T AV_m &= T_m. \end{aligned}$$

Proof. We need prove here only the biorthogonality of the vectors v_i , w_i with induction. Since the proof of the above relations is similar to **Proposition 7**. Assume now that the vectors v_1, \dots, v_j and w_1, \dots, w_j are biorthogonal. Claim that $(v_{j+1}, w_i) = 0$ for $i \leq j$ as follow:

When $i = j$, by steps 5 and 9 of **Algorithm 9.**, then

$$\begin{aligned} (v_{j+1}, w_j) &= (\delta_{j+1}^{-1} [Av_j - \alpha_j v_j - \beta_j v_{j-1}], w_j) \\ &= \delta_{j+1}^{-1} [(Av_j, w_j) - \alpha_j (v_j, w_j) - \beta_j (v_{j-1}, w_j)] \\ &= \delta_{j+1}^{-1} [\alpha_j - \alpha_j \times 1 - 0] = 0. \end{aligned}$$

When $i < j$, by steps 5, 6, 9, and 10 of **Algorithm 9.**, then

$$\begin{aligned} (v_{j+1}, w_i) &= \delta_{j+1}^{-1} [(Av_j, w_i) - \alpha_j (v_j, w_i) - \beta_j (v_{j-1}, w_i)] \\ &= \delta_{j+1}^{-1} [(v_j, A^T w_i) - 0 - \beta_j (v_{j-1}, w_i)] \\ &= \delta_{j+1}^{-1} [(v_j, \beta_{i+1} w_{i+1} + \alpha_i w_i + \delta_i w_{i-1}) - \beta_j (v_{j-1}, w_i)]. \end{aligned}$$

For $i < j - 1$, $(v_{j+1}, w_i) = 0$ by inductive hypothesis.

For $i = j - 1$, then

$$\begin{aligned} (v_{j+1}, w_{j-1}) &= \delta_{j+1}^{-1}[(v_j, \beta_j w_j + \alpha_{j-1} w_{j-1} + \delta_{j-1} w_{j-2}) - \beta_j (v_{j-1}, w_{j-1})] \\ &= \delta_{j+1}^{-1}[\beta_j (v_j, w_j) - \beta_j (v_{j-1}, w_{j-1})] = 0. \end{aligned}$$

■

3.2.2 BCG

The Biconjugate Gradient method (BCG) [6] is a oblique projection process onto

$$\mathcal{K}_m = \text{span}\{v_1, Av_1, \dots, A^{m-1}v_1\}$$

orthogonally to

$$\mathcal{L}_m = \text{span}\{w_1, A^T w_1, \dots, (A^T)^{m-1} w_1\},$$

where $v_1 = r_0/\|r_0\|_2$ and takes w_1 to satisfy $(v_1, w_1) = 1$. Proceeding in the same way as for the CG algorithm from the symmetric Lanczos algorithm, we can derive BCG algorithm from **Algorithm 9.**, the two-sided Lanczos algorithm. First, we use the LU factorization of $T_m = L_m U_m$ and define two matrices

$$\begin{aligned} P_m &= [p_1, \dots, p_m] = V_m U_m^{-1}, \\ P_m^* &= [p_1^*, \dots, p_m^*] = W_m L_m^{-T}. \end{aligned}$$

By **Proposition 9.**, we have

$$(P_m^*)^T A P_m = L_m^{-1} W_m^T A V_m U_m^{-1} = L_m^{-1} T_m U_m^{-1} = I.$$

Thus, the column vectors p_i^* of P_m^* and those p_i of P_m are A -conjugate. Also, it is known that p_{j+1} and p_{j+1}^* can be expressed as

$$p_{j+1} = r_{j+1} + \beta_j p_j. \tag{3.4}$$

$$p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*. \tag{3.5}$$

From (3.5) and A -conjugacy,

$$(A p_j, r_{j+1}^* + \beta_j p_j^*) = 0.$$

Then by (3.2), we have

$$\beta_j = -\frac{(Ap_j, r_{j+1}^*)}{(Ap_j, p_j^*)} = \frac{(r_{j+1}, r_{j+1}^*)}{(r_j, r_j^*)}. \quad (3.6)$$

And like the Conjugate Gradient algorithm, we find the residual r_j and r_j^* are in the same direction as for v_{j+1} and w_{j+1} , respectively. For these information and by (3.5), we have

$$(r_j - \alpha_j Ap_j, r_j^*) = 0,$$

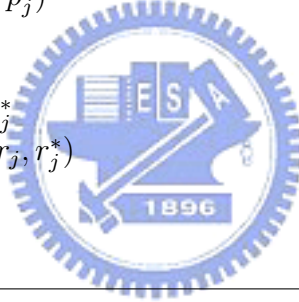
it imply

$$\alpha_j = \frac{(r_j, r_j^*)}{(Ap_j, p_j^*)}. \quad (3.7)$$

Putting these relations together by above, we have the BCG algorithm as follow:

ALGORITHM 10. *BCG*

1. $r_0 = b - Ax_0$, and choose r_0^* such that $(r_0, r_0^*) = 1$.
 2. $p_0 = r_0$, $p_0^* = r_0^*$
 3. For $j = 0, 1, \dots$, until convergence Do:
 4. $\alpha_j = (r_j, r_j^*) / (Ap_j, p_j^*)$
 5. $x_{j+1} = x_j + \alpha_j p_j$
 6. $r_{j+1} = r_j - \alpha_j Ap_j$
 7. $r_{j+1}^* = r_j^* - \alpha_j A^T p_j^*$
 8. $\beta_j = (r_{j+1}, r_{j+1}^*) / (r_j, r_j^*)$
 9. $p_{j+1} = r_{j+1} + \beta_j p_j$
 10. $p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$
 11. EndDo
-



3.3 Transpose-Free Variants

3.3.1 CGS

The BCG algorithm require multiplication by both A and A^T at each step. One thing, however, is certain: the vector p_{j+1}^* or w_j generated with A^T do not contribute to the solution directly, this mean extra work. For the reason given above, the Conjugate Gradient Squared (CGS) [10] was mainly to avoid using A^T in the BCG and to obtain faster convergence for the same computational cost. This idea is the residual vector r_j and the conjugate direction ϕ_j , in the BCG algorithm, can be expressed as

$$r_j = \phi_j(A)r_0, \quad p_j = \pi_j(A)r_0,$$

where ϕ_j and π_j are certain polynomials of degree j with $\phi_0(A) = 1$ and $\pi_0(A) = 1$. The same observation applies to the vectors r_j^* and p_j^* , defined as

$$r_j^* = \phi_j(A^T)r_0^*, \quad p_j^* = \pi_j(A^T)r_0^*.$$

By (3.6) and (3.7), the scalar α_j and β_j are given by

$$\alpha_j = \frac{(\phi_j(A)r_0, \phi_j(A^T)r_0^*)}{(A\pi_j(A)r_0, \pi_j(A^T)r_0^*)} = \frac{(\phi_j^2(A)r_0, r_0^*)}{(A\pi_j^2(A)r_0, r_0^*)}$$

$$\beta_j = \frac{(\phi_{j+1}(A)r_0, \phi_{j+1}(A^T)r_0^*)}{(\phi_j(A)r_0, \phi_j(A^T)r_0^*)} = \frac{(\phi_{j+1}^2(A)r_0, r_0^*)}{(\phi_j^2(A)r_0, r_0^*)}.$$

This indicates that the coefficients can be computed if we know r_0^* and $\phi_j^2(A)r_0$ and $\pi_j^2(A)r_0$.

From (3.4) and (3.5), it can be seen that the polynomials $\phi_{j+1}(t)$ and $\pi_{j+1}(t)$ satisfy the recurrences

$$\phi_{j+1}(t) = \phi_j(t) - \alpha_j t \pi_j(t) \tag{3.8}$$

$$\pi_{j+1}(t) = \phi_{j+1}(t) + \beta_j \pi_j(t), \tag{3.9}$$

and squaring both sides gives

$$\phi_{j+1}^2(t) = \phi_j^2(t) - 2\alpha_j t \phi_j(t) \pi_j(t) + \alpha_j^2 t^2 \pi_j^2(t) \tag{3.10}$$

$$\pi_{j+1}^2(t) = \phi_{j+1}^2(t) + 2\beta_j \phi_{j+1}(t) \pi_j(t) + \beta_j^2 \pi_j^2(t). \tag{3.11}$$

Multiplying $\phi_j(t)$ by the recurrence for $\pi_j(t)$ gives

$$\phi_j(t)\pi_j(t) = \phi_j^2(t) + \beta_{j-1}\phi_j(t)\pi_{j-1}(t), \quad (3.12)$$

and multiplying the recurrence for $\phi_{j+1}(t)$ by $\pi_j(t)$ gives

$$\begin{aligned} \phi_{j+1}(t)\pi_j(t) &= \phi_j(t)\pi_j(t) - \alpha_j t \pi_j^2(t) \\ &= \phi_j^2(t) + \beta_{j-1}\phi_j(t)\pi_{j-1}(t) - \alpha_j t \pi_j^2(t). \end{aligned} \quad (3.13)$$

Defining three new vectors as

$$\begin{aligned} r_j &= \phi_j^2(A)r_0, \\ p_j &= \pi_j^2(A)r_0, \\ q_j &= \phi_{j+1}(A), \end{aligned}$$

then (3.10) and (3.11) and (3.13) translate into

$$r_{j+1} = r_j - \alpha_j A(2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j Ap_j), \quad (\text{use (3.11)}) \quad (3.14)$$

$$p_{j+1} = r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j, \quad (3.15)$$

$$q_j = r_j + \beta_{j-1}q_{j-1} - \alpha_j Ap_j. \quad (3.16)$$

It is convenient to define two auxiliary vectors

$$u_j = r_j + \beta_{j-1}q_{j-1},$$

$$d_j = 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j Ap_j = u_j + q_j. \quad (\text{use (3.16)})$$

Utilizing these auxiliary vectors, we rewrite (3.14) and (3.15) and (3.16) as

$$r_{j+1} = r_j - \alpha_j A(u_j + q_j)$$

$$p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j)$$

$$q_j = u_j - \alpha_j Ap_j.$$

The first point to notice is the residual of CGS is different from the residual of BCG. In general, one should expect the result of CGS to converge twice as fast as BCG. Therefore, the CGS algorithm is given as below.

ALGORITHM 11. CGS

1. $r_0 = b - Ax_0$, and choose r_0^* such that $(r_0, r_0^*) = 1$.
 2. $p_0 = u_0 = r_0$
 3. For $j = 0, 1, \dots$, until convergence Do:
 4. $\alpha_j = (r_j, r_0^*) / (Ap_j, r_0^*)$
 5. $q_j = u_j - \alpha_j Ap_j$
 6. $x_{j+1} = x_j + \alpha_j(u_j + q_j)$
 7. $r_{j+1} = r_j - \alpha_j A(u_j + q_j)$
 8. $\beta_j = (r_{j+1}, r_0^*) / (r_j, r_0^*)$
 9. $u_{j+1} = r_{j+1} + \beta_j q_j$
 10. $p_{j+1} = u_{j+1} + \beta_j(q_j + \beta_j p_j)$
 11. EndDo
-



3.3.2 BICGSTAB

Note that the polynomials in the CGS are square, then the CGS residual usually increases by approximately the square of the increase of the BCG. But one difficulty in CGS is that rounding errors maybe lost or overflow and convergence curve maybe oscillate. To avoid the large oscillations in the CGS, one might try to produce iterates whose residual vectors are of the form

$$r'_j = \psi_j(A)\phi_j(A)r_0,$$

where $\phi_j(t)$ is again the BCG polynomial and $\psi_j(t)$ is a new polynomial which at each step is chosen to try and keep the smoothing convergence behavior. Specifically, we define $\psi_j(t)$ by the form

$$\psi_{j+1}(t) = (1 - \omega_j t)\psi_j(t) \quad (3.17)$$

in which the scalar ω_j can be chosen at each step to minimize $\|r_{j+1}\|_2$. This manner leads to the Biconjugate Gradient Stabilized (BICGSTAB) [12], the derivation is similar to CGS. First, leaving the discussion of the scalar ω_j aside for a moment, by using (3.8) and the residual polynomial

$$\begin{aligned} \psi_{j+1}(t)\phi_{j+1}(t) &= (1 - \omega_j t)\psi_j(t)\phi_{j+1}(t) \\ &= (1 - \omega_j t)(\psi_j(t)\phi_j(t) - \alpha_j t\psi_j(t)\pi_j(t)), \end{aligned} \quad (3.18)$$

which show that we can compute if we know the products $\psi_j(t)\pi_j(t)$. For this, by using (3.9) and (3.17), we have

$$\begin{aligned} \psi_j(t)\pi_j(t) &= \psi_j(t)(\phi_j(t) + \beta_{j-1}\pi_{j-1}(t)) \\ &= \psi_j(t)\phi_j(t) + \beta_{j-1}(1 - \omega_{j-1}t)\psi_{j-1}(t)\pi_{j-1}(t). \end{aligned} \quad (3.19)$$

In the BICGSTAB scheme, we require two recurrences

$$\begin{aligned} r_j &= \phi_j(A)\psi_j(A)r_0, \\ p_j &= \psi_j(A)\pi_j(A)r_0. \end{aligned}$$

According to the above formulas, it follow from (3.18) and (3.19) that

$$\begin{aligned} r_{j+1} &= (I - \omega_j A)(r_j - \alpha_j A p_j) \\ p_{j+1} &= r_{j+1} + \beta_j (I - \omega_j A) p_j. \end{aligned} \quad (3.20)$$

Finally, we need to express the coefficients α_j , β_j , and ω_j in terms of the new vectors. Let

$$\begin{aligned}\rho_j &= (\phi_j(A)r_0, \phi_j(A^T)r_0^*), \\ \tilde{\rho}_j &= (\phi_j(A)r_0, \psi_j(A^T)r_0^*).\end{aligned}$$

From BCG, we have $\phi_j(A)r_0$ orthogonal to all vectors $(A^T)^k r_0^*$, with $k < j$. In addition, $\phi_j(A)$ and $\psi_j(A)$ are polynomials of degree j . In particular, let $\eta_1^{(j)}$ and $\gamma_1^{(j)}$ be the leading coefficients for the polynomials $\psi_j(A)$ and $\phi_j(A)$, respectively. Then

$$\tilde{\rho}_j = \left(\phi_j(A)r_0, \frac{\eta_1^{(j)}}{\gamma_1^{(j)}} \phi_j(A^T)r_0^* \right) = \frac{\eta_1^{(j)}}{\gamma_1^{(j)}} \rho_j.$$

According to (3.8) and (3.17), we have

$$\gamma_1^{(j+1)} = -\alpha_j \gamma_1^{(j)}, \quad \eta_1^{(j+1)} = -\omega_j \eta_1^{(j)}.$$

As a result, we now compute β_j :

$$\begin{aligned}\beta_j &= \frac{(\phi_{j+1}(A)r_0, \psi_{j+1}(A^T)r_0^*)}{(\phi_j(A)r_0, \psi_j(A^T)r_0^*)} \times \frac{\alpha_j}{\omega_j} \\ &= \frac{(\psi_{j+1}(A)\phi_{j+1}(A)r_0, r_0^*)}{(\psi_j(A)\phi_j(A)r_0, r_0^*)} \times \frac{\alpha_j}{\omega_j} \\ &= \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}.\end{aligned}$$

To compute α_j by the same way, the polynomials in the right sides of the inner products in both the numerator and denominator can be replaced by their leading terms. And we also know that the leading coefficients for $\phi_j(A^T)r_0^*$ and $\pi_j(A^T)r_0^*$ are identical. Therefore,

$$\begin{aligned}\alpha_j &= \frac{(\phi_j(A)r_0, \phi_j(A^T)r_0^*)}{(A\pi_j(A)r_0, \pi_j(A^T)r_0^*)} \\ &= \frac{(\phi_j(A)r_0, \psi_j(A^T)r_0^*)}{(A\pi_j(A)r_0, \psi_j(A^T)r_0^*)} \\ &= \frac{(\psi_j(A)\phi_j(A)r_0, r_0^*)}{(A\psi_j(A)\pi_j(A)r_0, r_0^*)} \\ &= \frac{(r_j, r_0^*)}{(Ap_j, r_0^*)}.\end{aligned}$$

Lastly, let us now return to find the scalar ω_j to be minimize the residual r_{j+1} and by (3.20), we have

$$\begin{aligned}\min_{\omega_j \in \mathbb{R}} \|r_{j+1}\|_2 &= \min_{\omega_j \in \mathbb{R}} \|(I - \omega_j A)(r_j - \alpha_j Ap_j)\|_2 \\ &= \min_{\omega_j \in \mathbb{R}} \|(I - \omega_j A)s_j\|_2\end{aligned}$$

in which $s_j = r_j - \alpha_j Ap_j$. By Minimal Residual iteration, the optimal value for ω_j is given by

$$\omega_j = \frac{(As_j, s_j)}{(As_j, As_j)}.$$

Finally, we rewrite the residual r_{j+1} in the following form

$$r_{j+1} = r_j - \alpha_j Ap_j - \omega_j As_j = r_j - A(\alpha_j p_j + \omega_j s_j).$$

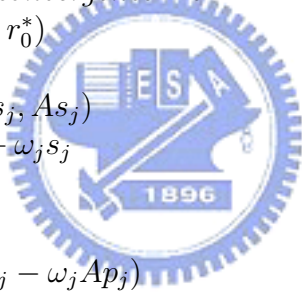
Then an approximate solution x_{j+1} can be repressed as

$$x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j.$$

Thus, we have the BICGSTAB algorithm as follow :

ALGORITHM 12. *BCGSTAB*

1. $r_0 = b - Ax_0$, and choose r_0^* such that $(r_0, r_0^*) = 1$.
 2. $p_0 = r_0$
 3. For $j = 0, 1, \dots$, until convergence Do:
 4. $\alpha_j = (r_j, r_0^*) / (Ap_j, r_0^*)$
 5. $s_j = r_j - \alpha_j Ap_j$
 6. $w_j = (As_j, s_j) / (As_j, As_j)$
 7. $x_{j+1} = x_j + \alpha_j p_j + \omega_j s_j$
 8. $r_{j+1} = s_j - \omega_j As_j$
 9. $\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \times \frac{\alpha_j}{\omega_j}$
 10. $p_{j+1} = r_{j+1} + \beta_j(p_j - \omega_j Ap_j)$
 11. EndDo
-



4 Transpose-Free QMR Method

In this chapter generalized minimal residual (GMRES), quasi-minimal residual (QMR), and transpose-free QMR (TFQMR) algorithms are derived. In line 12 of FOM, we find vector y_m can be obtained by dealing with the problem of inverse matrix. We will discuss it in different way from the least square method. In the first section, we derive GMRES and this variations by relying on application of Arnoldi's method. From the GMRES algorithm, we took advantage of the same techniques of IOM and DIOM to construct Quasi-GMRES and direct version of QGMRES, called DQGMRES. In the second section, we introduce QMR method. In algorithm of QMR, we will find an approximate solution x_m just as well as algorithm of GMRES, except for constructing the matrix T_m or H_m . In the last section, we introduce TFQMR method. This method is derived from the CGS algorithm. We shall have more to say about TFQMR later on.

4.1 GMRES, QGMRES, and DQGMRES

In this section, we will develop GMRES, QGMRES, and DQGMRES as well as the **section 2.3**. Only taking notice of one thing is to derive DQGMRES by using QR factorization, not LU factorization.

4.1.1 GMRES

The GMRES [11] is a oblique projection method based on taking $\mathcal{L}_m = A\mathcal{K}_m$, in which \mathcal{K}_m is the m -th Krylov subspace with $v_1 = r_0/\|r_0\|_2$. Now, we construct a basis for \mathcal{K}_m by using Arnoldi's method. Then resulting projection, oblique projection, should satisfy **Proposition 3**. If the approximate solution $x_m = x_0 + V_m y_m$, then

$$\begin{aligned} b - Ax_m &= b - A(x_0 + V_m y_m) \\ &= r_0 - AV_m y_m \\ &= \beta v_1 - V_{m+1} \bar{H}_m y_m \\ &= V_{m+1}(\beta e_1 - \bar{H}_m y_m). \end{aligned}$$

So the next guess x_m should satisfy

$$\begin{aligned} \min_{x \in x_0 + \mathcal{K}_m} \|b - Ax\|_2 &= \|b - Ax_m\|_2 \\ &= \|V_{m+1}(\beta e_1 - \bar{H}_m y_m)\|_2 \\ &= \|\beta e_1 - \bar{H}_m y_m\|_2, \end{aligned}$$

and the claim is that solving the least squares problem is inexpensive to compute. By the above relations, this gives the following algorithm.

ALGORITHM 13. GMRES

1. $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. Set $(m+1) \times m$ matrix $\bar{H}_m = 0$
 3. For $j = 1, 2, \dots, m$ Do:
 4. $w_j = Av_j$
 5. For $i = 1, \dots, j$ Do:
 6. $h_{ij} = (w_j, v_i)$
 7. $w_j = w_j - h_{ij}v_i$
 8. EndDo
 9. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ then set $m = j$ and Goto 12
 10. $v_{j+1} = w_j/h_{j+1,j}$
 11. EndDo
 12. Compute y_m the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$.
-



4.1.2 QGMRES

The same observation applies to QGMRES, we can use the same technique to derive an incomplete version of the GMRES algorithm, Quasi-GMRES (QGMRES). The algorithm of QGMRES is follow:

ALGORITHM 14. QGMRES

1. $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$, and $v_1 = r_0/\beta$
 2. Set $(m+1) \times m$ matrix $\bar{H}_m = 0$
 3. For $j = 1, 2, \dots, m$ Do:
 4. $w = Av_j$
 5. For $i = \max\{1, j - k + 1\}, \dots, j$ Do:
 6. $h_{ij} = (w, v_i)$
 7. $w = w - h_{ij}v_i$
 8. EndDo
 9. $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
 10. EndDo
 11. Compute y_m the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$.
-

Be continued until the 4-th rotation is applied by the same way, which transform the \bar{H}_4 into one involving the matrix and right-hand side,

$$\bar{H}_4^{(4)} = \begin{pmatrix} h_{11}^{(4)} & h_{12}^{(4)} & h_{13}^{(4)} & h_{14}^{(4)} \\ & h_{22}^{(4)} & h_{23}^{(4)} & h_{24}^{(4)} \\ & & h_{33}^{(4)} & h_{34}^{(4)} \\ & & & h_{44}^{(4)} \\ & & & & 0 \end{pmatrix}, \quad \bar{g}_4 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{pmatrix}.$$

By the above example, in general the scalars c_i and s_i of the i^{th} rotation Ω_i are defined as

$$s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}, \quad c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}. \quad (4.1)$$

Define Q_m the product of matrices Ω_i ,

$$Q_m = \Omega_m \Omega_{m-1} \dots \Omega_1$$

and

$$\begin{aligned} \bar{R}_m &= \bar{H}_m^{(m)} = Q_m \bar{H}_m, \\ \bar{g}_m &= Q_m(\beta e_1) = (\gamma_1, \dots, \gamma_{m+1})^T. \end{aligned}$$

Then

$$\min_y \|\beta e_1 - \bar{H}_m y\|_2 = \min_y \|\bar{g}_m - \bar{R}_m y\|_2.$$

Proposition 10. *Let Ω_i , $i = 1, \dots, m$ be the rotation matrices used to transform \bar{H}_m into an upper triangular form and \bar{R}_m , $\bar{g}_m = (\gamma_1, \dots, \gamma_{m+1})^T$ the resulting matrix and right-hand side. Denote by R_m the $m \times m$ upper triangular matrix obtained from \bar{R}_m by deleting its last row and by g_m the m -dimensional vector obtained from \bar{g}_m by deleting its last component. Then,*

1. *The rank of AV_m is equal to the rank of R_m . In particular, if $r_{mm} = 0$ then A must be singular.*
2. *The vector y_m which minimizes $\|\beta e_1 - \bar{H}_m y\|_2$ is given by*

$$y_m = R_m^{-1} g_m.$$

3. The residual vector at step m satisfies

$$b - Ax_m = V_{m+1}(\beta e_1 - \bar{H}_m y_m) = V_{m+1} Q_m^T (\gamma_{m+1} e_{m+1})$$

and, as a result,

$$\|b - Ax_m\|_2 = |\gamma_{m+1}|.$$

We need not elaborate or proof on above proposition, it is treated much more adequately in [9, p.169]. We can find y_m in **Proposition 10.**, then the direct version of QGMRES, called DQGMRES, is quite similar to DIOM. For example, if 5×4 Hessenberg matrix with bandwidth equal to 3, this is

$$\bar{H}_4 = \begin{pmatrix} h_{11} & h_{12} & & & \\ h_{21} & h_{22} & h_{23} & & \\ & h_{32} & h_{33} & h_{34} & \\ & & h_{43} & h_{44} & \\ & & & & h_{54} \end{pmatrix}, \quad \bar{g}_0 = \begin{pmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Multiply the Hessenberg matrix \bar{H}_4 and \bar{g}_0 by Ω_i , then the resulting is $\bar{R}_4 y = \bar{g}_4$, with

$$\bar{R}_4 = \begin{pmatrix} r_{11} & r_{12} & r_{13} & & & \\ & r_{22} & r_{23} & r_{24} & & \\ & & r_{33} & r_{34} & & \\ & & & & r_{44} & \\ & & & & & 0 \end{pmatrix}, \quad \bar{g}_4 = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \\ \gamma_5 \end{pmatrix}.$$

Therefore, the general approximate solution is given by

$$x_m = x_0 + V_m R_m^{-1} g_m$$

where R_m and g_m are obtained by removing the last row of \bar{R}_m and \bar{g}_m , respectively.

Defining the matrix

$$P_m = [p_1, \dots, p_m] = V_m R_m^{-1}$$

and the vector

$$g_m = \begin{bmatrix} g_{m-1} \\ \gamma_m \end{bmatrix},$$

in which

$$\gamma_m = c_m \gamma_m^{(m-1)},$$

where $\gamma_m^{(m-1)}$ is the last component of the vector \bar{g}_{m-1} and c_m defined by (4.1). Thus, x_m can be written as

$$\begin{aligned} x_m &= x_0 + P_m g_m \\ &= x_0 + P_{m-1} g_{m-1} + \gamma_m p_m \\ &= x_{m-1} + \gamma_m p_m. \end{aligned}$$

ALGORITHM 15. DQGMRES

1. $r_0 = b - Ax_0$, $\gamma_1 = \|r_0\|_2$, and $v_1 = r_0/\gamma_1$
 2. For $m = 1, 2, \dots$, until convergence Do:
 3. $w = Av_m$
 4. For $i = \max\{1, m - k + 1\}, \dots, m$ Do:
 5. $h_{im} = (w, v_i)$
 6. $w = w - h_{im}v_i$
 7. EndDo
 8. $h_{m+1,m} = \|w\|_2$ and $v_{m+1} = w/h_{m+1,m}$
 9. Update the QR factorization of \bar{H}_m , i.e.,
 10. Apply Ω_i , $i=m-k, \dots, m-1$ to the m -th column of \bar{H}_m
 11. Compute the rotation coefficients c_m, s_m by (4.1)
 12. $\gamma_{m+1} = -s_m \gamma_m$
 13. $\gamma_m = c_m \gamma_m$
 14. $h_{mm} = c_m h_{mm} + s_m h_{m+1,m}$ ($= \sqrt{h_{m+1,m}^2 + h_{mm}^2}$)
 15. $p_m = (v_m - \sum_{i=m-k}^{m-1} h_{im} p_i)/h_{mm}$
 16. $x_m = x_{m-1} + \gamma_m p_m$
 17. If $|\gamma_{m+1}|$ is small enough then stop.
 18. EndDo
-

4.2 Quasi-Minimal Residual

In BCG algorithm or **Proposition 9.**, we know those used the Lanczos biorthogonalization procedure to make

$$AV_m = V_{m+1}\bar{T}_m$$

in which \bar{T}_m is $(m+1) \times m$ tridagonal matrix of the form

$$\bar{T}_m = \begin{bmatrix} T_m \\ \delta_{m+1}e_m^T \end{bmatrix}.$$

Consider the DQGMRES algorithm make

$$AV_m = V_{m+1}\bar{H}_m$$

in which \bar{H}_m is $(m+1) \times m$ banded matrix of the form

$$\bar{H}_m = \begin{bmatrix} H_m \\ h_{m+1,m}e_m^T \end{bmatrix}.$$

If a bandwidth of \bar{H}_m equal to three. Then we can find \bar{T}_m and \bar{H}_m are similar. Then we can derive QMR by the same way as for GMRES or DQGMRES. In the first instance, if we have an approximate solution $x_m = x_0 + V_my_m$, then

$$\begin{aligned} b - Ax_m &= b - A(x_0 + V_my_m) \\ &= r_0 - AV_my_m \\ &= \beta v_1 - V_{m+1}\bar{T}_my_m \\ &= V_{m+1}(\beta e_1 - \bar{T}_my_m). \end{aligned}$$

So the next guess x_m should satisfy

$$\begin{aligned} \min_{x \in x_0 + \mathcal{K}_m} \|b - Ax\|_2 &= \|b - Ax_m\|_2 \\ &= \|V_{m+1}(\beta e_1 - \bar{T}_my_m)\|_2 \\ &\leq \|V_{m+1}\|_2 \cdot \|\beta e_1 - \bar{T}_my_m\|_2. \end{aligned}$$

Since the columns of V_{m+1} are not orthogonal in the Lanczos algorithm, it would be difficult to choose y_m to minimize the residual. However, we can easily choose to minimize the second factor, $\|\beta e_1 - \bar{T}_my_m\|_2$. Then we write down the algorithm as follow:

ALGORITHM 16. QMR

1. $r_0 = b - Ax_0$, $\gamma_0 = \|r_0\|_2$, $w_1 = v_1 = r_0/\gamma_1$ and $\beta_1 = \delta_1 = 0$
 2. For $m = 1, 2, \dots$, until convergence Do:
 3. $\alpha_m = (Av_m, w_m)$
 4. $\hat{v}_{m+1} = Av_m - \alpha_m v_m - \beta_m v_{m-1}$ ($v_0 = 0$)
 5. $\hat{w}_{m+1} = A^T w_m - \alpha_m w_m - \delta_m w_{m-1}$ ($w_0 = 0$)
 6. $\delta_{m+1} = |(\hat{v}_{m+1}, \hat{w}_{m+1})|^{1/2}$. If $\delta_{m+1} = 0$ then stop
 7. $\beta_{m+1} = (\hat{v}_{m+1}, \hat{w}_{m+1})/\delta_{m+1}$
 8. $w_{m+1} = \hat{w}_{m+1}/\beta_{m+1}$
 9. $v_{m+1} = \hat{v}_{m+1}/\delta_{m+1}$
 10. Update the QR factorization of \bar{T}_m , i.e.,
 11. Apply Ω_i , $i = m-2, m-1$ to the m -th column of \bar{T}_m
 12. Compute the rotation coefficients c_m, s_m by (4.1)
 13. $\gamma_{m+1} = -s_m \gamma_m$
 14. $\gamma_m = c_m \gamma_m$
 15. $\alpha_m = c_m \alpha_m + s_m \delta_{m+1}$
 16. $p_m = (v_m - \sum_{i=m-2}^{m-1} t_{im} p_i)/t_{mm}$
 17. $x_m = x_{m-1} + \gamma_m p_m$
 18. If $|\gamma_{m+1}|$ is small enough then stop.
 19. EndDo
-



4.3 TFQMR

The Transpose-Free QMR (TFQMR) [7] for solving general non-Hermitian linear systems is derived from CGS algorithm. By incorporating the QMR approach, residual norm of TFQMR produces smoother convergence behavior than CGS. In the first, we derive TFQMR from the CGS algorithm. We double all subscripts in the CGS algorithm, that is to say

$$\alpha_{2j} = (r_{2j}, r_0^*) / (Ap_{2j}, r_0^*) \quad (4.2)$$

$$q_{2j} = u_{2j} - \alpha_{2j}Ap_{2j} \quad (4.3)$$

$$x_{2j+2} = x_{2j} + \alpha_{2j}(u_{2j} + q_{2j}) \quad (4.4)$$

$$r_{2j+2} = r_{2j} - \alpha_{2j}A(u_{2j} + q_{2j}) \quad (4.5)$$

$$\beta_{2j} = (r_{2j+2}, r_0^*) / (r_{2j}, r_0^*) \quad (4.6)$$

$$u_{2j+2} = r_{2j+2} + \beta_{2j}q_{2j} \quad (4.7)$$

$$p_{2j+2} = u_{2j+2} + \beta_{2j}(q_{2j} + \beta_{2j}p_{2j}) \quad (4.8)$$

Observe that approximate solution x_{2j+2} in (4.4) can be split into the following two half-steps:

$$x_{2j+1} = x_{2j} + \alpha_{2j}u_{2j} \quad (4.9)$$

$$x_{2j+2} = x_{2j+1} + \alpha_{2j}q_{2j}. \quad (4.10)$$

When m is odd, set

$$\begin{cases} u_m = q_{m-1} \\ \alpha_m = \alpha_{m-1} \end{cases}. \quad (4.11)$$

Then (4.9) and (4.10) can be simplified by (4.11). Whether m is even or odd, the single equation is

$$x_m = x_{m-1} + \alpha_{m-1}u_{m-1}.$$

It must be noted that the intermediate iterates x_m does not exist in the original CGS algorithm with m is odd. And we define the $n \times m$ matrix,

$$U_m = [u_0, \dots, u_{m-1}]$$

and the vector

$$z_m = (\alpha_0, \dots, \alpha_{m-1})^T.$$

General iterate x_m and residual vector r_m are

$$x_m = x_{m-1} + \alpha_{m-1}u_{m-1} \quad (4.12)$$

$$= x_0 + U_m z_m, \quad (4.13)$$

$$r_m = r_{m-1} - \alpha_{m-1}Au_{m-1} \quad (4.14)$$

$$= r_0 - AU_m z_m. \quad (4.15)$$

From a result of (4.14), we have

$$Au_{m-1} = \frac{1}{\alpha_{m-1}}(r_{m-1} - r_m).$$

Judging from the above relation, we can translate into matrix form, this relation becomes

$$AU_m = R_{m+1}\bar{B}_m \quad (4.16)$$

where R_{m+1} is the $n \times (m+1)$ matrix,

$$R_{m+1} = [r_0, r_1, \dots, r_m]$$

and where \bar{B}_m is the $(m+1) \times m$ matrix with the following form,

$$\bar{B}_m = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \\ 0 & 0 & \cdots & 0 & -1 \end{pmatrix} \times \text{diag} \left\{ \frac{1}{\alpha_0}, \frac{1}{\alpha_1}, \dots, \frac{1}{\alpha_{m-1}} \right\}.$$

Let the $(m+1) \times (m+1)$ scaling matrix is

$$\Delta_{m+1} = \text{diag}\{\delta_0, \delta_1, \dots, \delta_m\},$$

in addition, each inverse of diagonal element can rescale the corresponding column of R_{m+1} equal to one. Then, the relation of (4.16) becomes

$$\begin{aligned} AU_m &= R_{m+1}\Delta_{m+1}^{-1}\Delta_{m+1}\bar{B}_m \\ &= \tilde{V}_{m+1}\bar{H}_m, \end{aligned} \quad (4.17)$$

if we define the $n \times (m + 1)$ matrix $\tilde{V}_{m+1} = R_{m+1}\Delta_{m+1}^{-1}$ and $(m + 1) \times m$ matrix $\bar{H}_m = \Delta_{m+1}\bar{B}_m$. With this result, equation (4.15) becomes

$$\begin{aligned} r_m &= r_0 - AU_m z_m = R_{m+1} [e_1 - \bar{B}_m z_m] \\ &= R_{m+1} \Delta_{m+1}^{-1} [\delta_0 e_1 - \Delta_{m+1} \bar{B}_m z_m]. \end{aligned} \quad (4.18)$$

This will lead us further into a consideration of whether we can exploit above relations by QMR. But before we come on to that, let us pause here to look at the following lemma.

Lemma 1. *Let \tilde{R}_m be the $m \times m$ upper part of the matrix $Q_{m-1}\bar{H}_m$ in FOM and let R_m be the $m \times m$ upper part of the matrix $Q_m\bar{H}_m$ in GMRES. Similarly, let \tilde{g}_m be the vector of the first m components of $Q_{m-1}(\beta e_1)$ and let g_m be the vector of the first m components of $Q_m(\beta e_1)$. Define*

$$\tilde{y}_m = \tilde{R}_m^{-1} \tilde{g}_m, \quad y_m = R_m^{-1} g_m$$

the y vectors obtained for an m -dimensional FOM and GMRES methods, respectively.

Then

$$y_m - \begin{pmatrix} y_{m-1} \\ 0 \end{pmatrix} = c_m^2 \left(\tilde{y}_m - \begin{pmatrix} y_{m-1} \\ 0 \end{pmatrix} \right)$$

in which c_m is the cosine used in the m -th rotation Ω_m , as defined in DQGMRES.

We need mention here only the result of lemma, this proof of lemma can be seen in [9, p.180]. Let us return to our main subject to derive TFQMR. By above lemma, we know it is also valid for the CGS/TFQMR pair. This relation provides a starting point to derive the TFQMR algorithm. Thus, the TFQMR iterates satisfy the following relation

$$x_m - x_{m-1} = c_m^2 (x_m^{CGS} - x_{m-1}). \quad (4.19)$$

Setting the vector

$$d_m = \frac{1}{\alpha_{m-1}} (x_m^{CGS} - x_{m-1}) \quad (4.20)$$

and the scalar

$$\eta_m = c_m^2 \alpha_{m-1}. \quad (4.21)$$

Then (4.19) can be expressed as

$$x_m = x_{m-1} + \eta_m d_m. \quad (4.22)$$

Moreover, by (4.12), (4.19), (4.20), and (4.21), we have

$$\begin{aligned}
d_m &= \frac{1}{\alpha_{m-1}} [(x_m^{CGS} - x_{m-1}^{CGS}) + (x_{m-1}^{CGS} - x_{m-2}) - (x_{m-1} - x_{m-2})] \\
&= u_{m-1} + \frac{1}{\alpha_{m-1}} (x_{m-1}^{CGS} - x_{m-2}) - \frac{1}{\alpha_{m-1}} (x_{m-1} - x_{m-2}) \\
&= u_{m-1} + \frac{1 - c_{m-1}^2}{\alpha_{m-1}} (x_{m-1}^{CGS} - x_{m-2}) \\
&= u_{m-1} + \frac{1 - c_{m-1}^2}{c_{m-1}^2} \frac{\eta_{m-1}}{\alpha_{m-1}} d_{m-1}.
\end{aligned}$$

Define the m -st tangent by $\theta_m = s_m/c_m$, then we have the new relation as below

$$d_{m+1} = u_m + \frac{\theta_m^2 \eta_m}{\alpha_m} d_m. \quad (4.23)$$

By the structure of \bar{H}_m in (4.17), the angle used in the $(j+1)$ -th rotation can obtain by (4.1), that is

$$s_{j+1} = \frac{-\delta_{j+1}}{\sqrt{\tau_j^2 + \delta_{j+1}^2}}, \quad c_{j+1} = \frac{\tau_j}{\sqrt{\tau_j^2 + \delta_{j+1}^2}}, \quad \theta_{j+1} = \frac{-\delta_{j+1}}{\tau_j}, \quad (4.24)$$

where τ_j be $(j+1)$ -th diagonal element of $\Omega_j \cdots \Omega_1 \bar{H}_m$. Moreover, after $j+1$ rotations, next diagonal element δ_{j+1} becomes τ_{j+1} , which is

$$\begin{aligned}
\tau_{j+1} &= \delta_{j+1} \times c_{j+1} = \frac{\tau_j \delta_{j+1}}{\sqrt{\tau_j^2 + \delta_{j+1}^2}} \\
&= -\tau_j \times s_{j+1} = -\tau_j \theta_{j+1} c_{j+1}.
\end{aligned} \quad (4.25)$$

Since only the square of scalar θ_{j+1} is invoked in the update of the direction d_{m+1} . We can ignore the negative symbol in (4.24) and (4.25). So far, we have seen the following relations:

- $d_{m+1} = u_m + (\theta_m^2/\alpha_m)\eta_m d_m$
- $\theta_{m+1} = \delta_{m+1}/\tau_m$
- $c_{m+1} = (1 + \theta_{m+1}^2)^{-\frac{1}{2}}$
- $\tau_{m+1} = \tau_m \theta_{m+1} c_{m+1}$
- $\eta_{m+1} = c_{m+1}^2 \alpha_m$.

The question which we must consider next is to derive the remain terms. We note a little earlier that the vectors r_m in (4.14) are no longer the actual residuals, we rewrite the new notation by w_m . Then we have

$$w_m = w_{m-1} - \alpha_{m-1}Au_{m-1}.$$

We may note, in passing, that the scalar δ_m equal to $\|w_m\|_2$. Setting the new vector $v_{2j} = Ap_{2j}$ and multiplying (4.8) by matrix A ,

$$\begin{aligned} v_{2j} &= Au_{2j} + \beta_{2j-2}(Aq_{2j-2} + \beta_{2j-2}Ap_{2j-2}) \\ &= Au_{2j} + \beta_{2j-2}(Au_{2j-1} + \beta_{2j-2}v_{2j-2}). \end{aligned} \quad (\text{by (4.16)})$$

The same observation applies to (4.3) and (4.7), we have

$$\begin{aligned} u_{2j+1} &= u_{2j} - \alpha_{2j}v_{2j} \\ u_{2j+2} &= w_{2j+2} + \beta_{2j}u_{2j+1}. \end{aligned}$$

To sum up all relations above, we have TFQMR algorithm as follow:

ALGORITHM 17. TFQMR

-
1. $w_0 = u_0 = r_0 = b - Ax_0, v_0 = Au_0, d_0 = 0$
 2. $\tau_0 = \|r_0\|_2, \theta_0 = \eta_0 = 0.$
 3. Choose r_0^* such that $\rho_0 = (r_0^*, r_0) \neq 0.$
 4. For $m = 0, 1, \dots$, until convergence Do:
 5. If m is even then
 6. $\alpha_{m+1} = \alpha_m = \rho_m / (v_m, r_0^*)$
 7. $u_{m+1} = u_m - \alpha_m v_m$
 8. EndIf
 9. $w_{m+1} = w_m - \alpha_m Au_m$
 10. $d_{m+1} = u_m + (\theta_m^2 / \alpha_m) \eta_m d_m$
 11. $\theta_{m+1} = \|w_{m+1}\|_2 / \tau_m; c_{m+1} = (1 + \theta_{m+1}^2)^{-\frac{1}{2}}$
 12. $\tau_{m+1} = \tau_m \theta_{m+1} c_{m+1}; \eta_{m+1} = c_{m+1}^2 \alpha_m$
 13. $x_{m+1} = x_m + \eta_{m+1} d_{m+1}$
 14. If m is odd then
 15. $\rho_{m+1} = (w_{m+1}, r_0^*); \beta_{m-1} = \rho_{m+1} / \rho_{m-1}$
 16. $u_{m+1} = w_{m+1} + \beta_{m-1} u_m$
 17. $v_{m+1} = Au_{m+1} + \beta_{m-1} (Au_m + \beta_{m-1} v_{m-1})$
 18. EndIf
 19. EndDo
-

5 Algorithm and Numerical Results

5.1 General Solution Algorithm

In the first, we refer to the finest grid (with 41×129 nodes) as level 1. Two additional coarser grids are constructed by successively discarding every other mesh line from one grid to the coarser one. These grids are numbered with increasing level number, we have

- level 1: 41×129 grid,
- level 2: 21×65 grid,
- level 3: 11×33 grid.

According to the characteristic of the results are given in **section 1.2**, we have a nonlinear system of the discretized equations in residual form,

$$F(\Phi) = 0. \quad (5.1)$$

Assume that nonlinear system start from an initial guess Φ_0 and have an accurate solution Φ_* . The vector Φ has $N_c N_x N_y$ three components for two dimensional problems, where N_c is the number of the solution components; N_x and N_y are the number of grid points in the r and z direction, respectively. In order to stabilize convergence, we use a damped Newton method [3] instead of Newton method. Suppose an initial solution Φ_0 is close to the solution Φ_* enough, the equations (5.1) can be solved by using a damped Newton method, that is

$$J(\Phi_n)(\Phi_{n+1} - \Phi_n) = -\lambda_n F(\Phi_n), \quad n = 0, 1, \dots \quad (5.2)$$

where $J(\Phi_n)$ is the Jacobian matrix at Φ_n with the form

$$J(\Phi_n) = \frac{\partial F}{\partial \Phi}(\Phi_n),$$

and the damping parameter λ_n is in the range of ($0 < \lambda_n \leq 1$). We denote the update vector as $\Delta \Phi_n = \Phi_{n+1} - \Phi_n$ and (5.2) with convergence tolerance

$$\|\Delta \Phi_n\|_2 < 10^{-5}.$$

In our numerical applications, we use likely modified Newton method in (5.2). This means that the Jacobian matrix is re-evaluated only twice Newton iterative step. If the rate of convergence is too slow, we form a new damping parameter.

We know that the governing equations of a combustion process are difficult to solve and a good initial solution estimate for an iteration process is hard to determine. For eliminating these difficulties, a time relaxation process is used. Due to the nonlinearity of the governing equations of a combustion process, we can use a pseudo transient process to produce a parabolic in time problem and bring the starting estimate into the convergence domain of the steady Newton method. Thus, the unsteady form of the governing equations can be obtained by adding the unsteady term to the steady-state equations. We have

$$\mathcal{F}(\Phi_{n+1}) = F(\Phi_{n+1}) + D \frac{\Phi_{n+1} - \Phi_n}{\Delta t^{n+1}} = 0, \quad (5.3)$$

where D is a diagonal scaling matrix with nonnegative entries and $\Delta t^{n+1} = t^{n+1} - t^n$ is n -st time step. In our calculations of program, in time dependent part, we use modified Newton method to solve the nonlinear system (5.3) which is similar to the system of equations in (5.2). In steady-state part, we use one way multigrid method and damped Newton method to solve the nonlinear system (5.1). One way multigrid method means that the coarser meshes are used only to initialize the next finer ones. Our goal is to obtain a converged numerical solution on level 1. For that purpose, we solve the nonlinear problem (5.1) and (5.3) starting at the coarsest level and ending at the finest. To summarize above informations, we write down these phases as following sample processes:

1. Time stepping on level 3 (in coarsest level).
2. Steady Newton iterations on level 3 and interpolation of the numerical solution from level 3 onto level 2.
3. Steady Newton iterations on level 2 and interpolation of the numerical solution from level 2 onto level 1.
4. Steady Newton iterations on level 1.

5.2 Numerical Results and Discussion

In this section, we present three Krylov subspace methods of numerical results, these methods are BICGSTAB and GMRES and TFQMR. First, a contour plot of the computed temperature for the flame sheet model is shown in **Figure 2** by using MATLAB. For computing efficiently in programs, we combined with a Gauss-Seidel left-preconditioner.

Table 1:
 Numerical results for one way multigrid during
 the time stepping phase (level 3).

Linear solver	Iterations	Residual	CPU time (seconds)
BICGSTAB	91	0.268043	4.0
GMRES	113	0.282073	4.0
TFQMR	384	0.24029	10.0

We left the problem, how to choose preconditioner matrix is, untouched. In our numerical calculations, the numerical result during time stepping are presented in **Table 1** and **Figure 3** is the residual norm during the time stepping phase. The **Table 2** contains the results for the one way multigrid and damped Newton methods in steady-state. In addition, a speed-up is with respect to the unilevel solution time (412.0). **Table 3** and **Table 4** and **Table 5** indicate that total numbers of iteration performed on each level during the steady-state. We present residual norm on each level during the steady-state Newton iterative steps in **Table 6**. Finally, we compare GMRES with others in maximum temperature of flame by using *C* or *FORTRAN* language in **Table 7**. These diagrams helps us to interpret the some facts. First, GMRES gives the best execution time in our program. Secondly, BICGSTAB gives the less iterations than the others, and gains the smoother residual norm during the time stepping phase. However, if we take the numbers of iteration on finest level as a criterion, then GMRES is just slightly better than BICGSTAB. Furthermore, we can find TFQMR is more expensive than all cases in our experience. Thus, we see that BICGSTAB and GMRES are better solvers to solve the flame sheet model.

Table 2:
Numerical results for damped Newton with one way multigrid during the steady-state phase.

Linear solver	Operation	level 1	level 2	level 3
BICGSTAB	Total CPU seconds	293.0	29.0	4.0
	Speed-up in time	1.4	14.2	103.0
GMRES	Total CPU seconds	190.0	25.0	5.0
	Speed-up in time	2.2	16.5	82.4
TFQMR	Total CPU seconds	412.0	65.0	5.0
	Speed-up in time	1.0	6.3	82.4

Table 3:
Numerical results for one way multigrid.
BICGSTAB(i) represents the total number of BICGSTAB iterations performed on level i during the steady-state Newton iterations.

Total levels	Level 3	Level 2	Level 1
BICGSTAB(1)	-	-	672
BICGSTAB(2)	-	350	676
BICGSTAB(3)	192	148	269
Total iterations	192	498	1617

Table 4:
Numerical results for one way multigrid.
GMRES(i) represents the total number of GMRES iterations performed on level i during the steady-state Newton iterations.

Total levels	Level 3	Level 2	Level 1
GMRES(1)	-	-	609
GMRES(2)	-	410	551
GMRES(3)	361	336	280
Total iterations	361	746	1440

Table 5:
Numerical results for one way multigrid.
TFQMR(i) represents the total number of TFQMR iterations performed on level i during the steady-state Newton iterations.

Total levels	Level 3	Level 2	Level 1
TFQMR(1)	-	-	1042
TFQMR(2)	-	746	1418
TFQMR(3)	324	1324	1798
Total iterations	324	2070	4258

Table 6: Residual norm on level i during the steady-state Newton iterative steps by using GMRES as linear solver.

*: bracket represent the new residual norm taking damped parameter $\lambda_n = 0.5$

Newton iteration	Level 3	Level 2	Level 1
original	1.979×10^{-1}	3.998×10^1	2.507×10^2
1	1.249×10^{-2}	1.259×10^2	$5.736 \times 10^2(1.665 \times 10^2)^*$
2	8.311×10^{-4}	3.571×10^1	$3.299 \times 10^2(1.274 \times 10^2)^*$
3	1.004×10^{-4}	2.863×10^0	1.478×10^2
4	5.234×10^{-5}	7.470×10^{-3}	1.185×10^1
5	-	2.144×10^{-4}	4.134×10^{-2}
6	-	-	2.961×10^{-4}

Table 7: Compare GMRES with others in maximum temperature of flame by using C or $FORTTRAN$.

Solver	C	$FORTTRAN$	$C - FORTTRAN$
GMRES - GMRES	0.0000	0.0000	0.0015
GMRES - BICGSTAB	0.2593	0.1846	0.1848
GMRES - TFQMR	0.2801	-	-



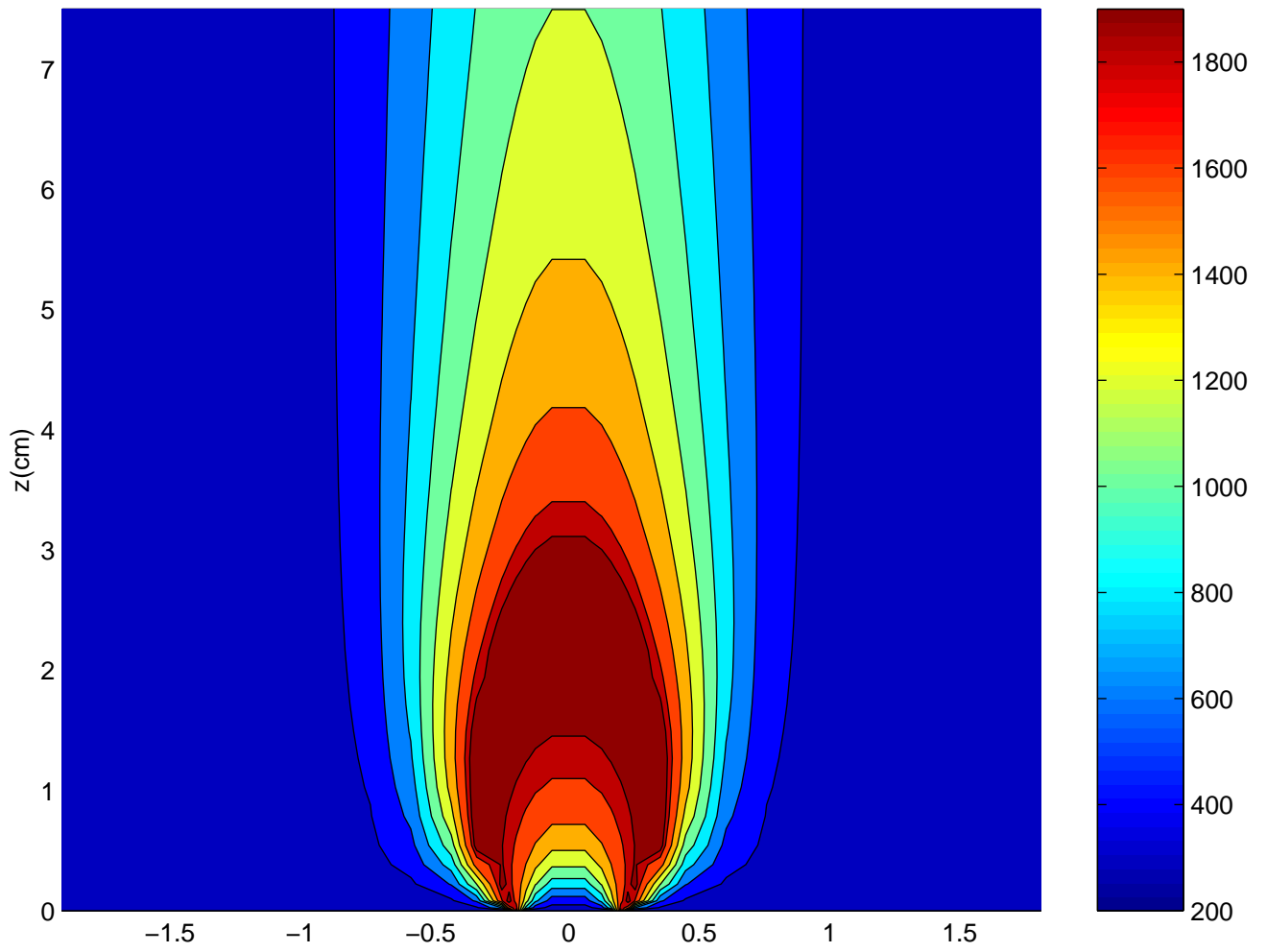


Figure 2: Contour plot of the temperature

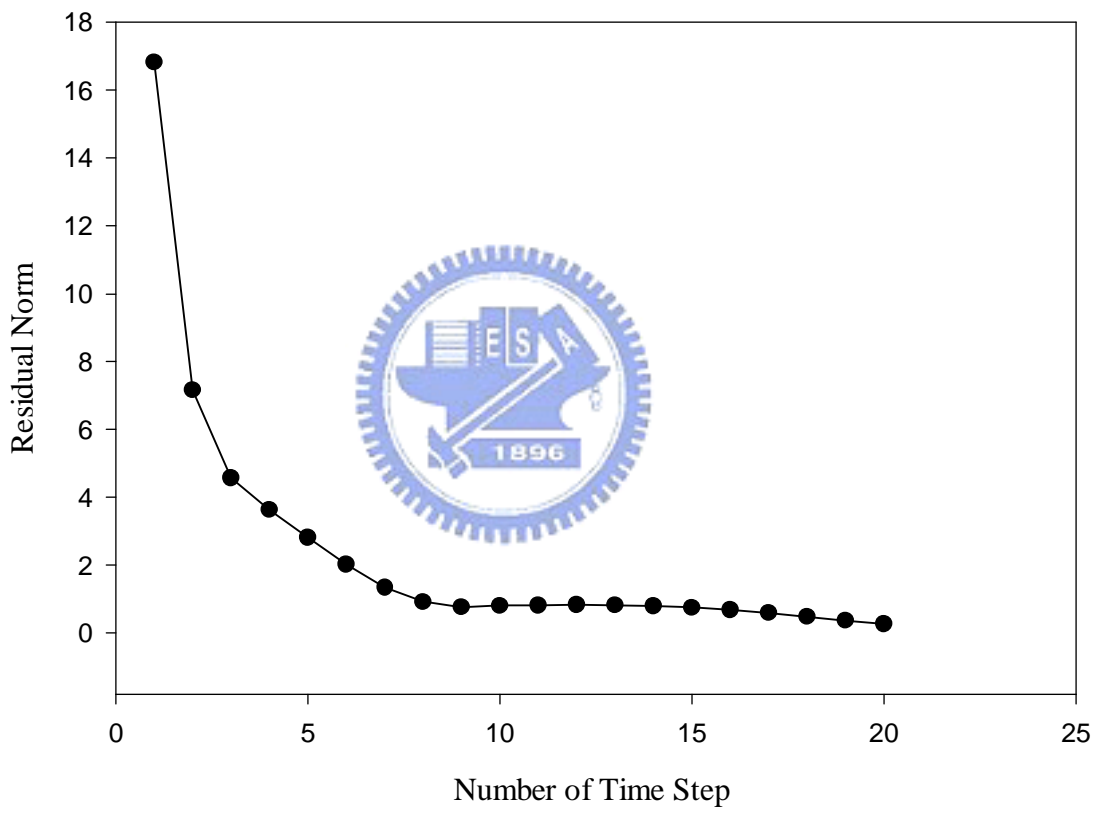


Figure 3: Residual norm of Newton step during the time stepping phase

References

- [1] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.
- [2] S. P. Burke and T. E. W. Schumann, Diffusion Flames, *Industrial and Engineering Chemistry*, 20, 998-1004, 1928.
- [3] Deuffhard, P., A Modified Newton Method for the Solution of Ill-conditioned Systems of Nonlinear Equations with Application to Multiple Shooting, *Numer. Math.*, Vol. 22, pp. 289-315, 1974.
- [4] C. C. Douglas and A. Ern, Numerical solution of flame sheet problems With and Without Multigrid Methods, *Sixth Copper Mountain Conference on Multigrid Methods*, N. D. Melson, T. A. Manteuffel, and S. F. McCormick, eds., NASA, Hampton, VA, 143-157, 1993.
- [5] A. Ern, *Vorticity-velocity modeling of chemically reacting flows*, PhD thesis, Yale University, February 1994. Mechanical Engineering Department.
- [6] R. Fletcher. Conjugate gradient methods for indefinite systems. In G. A. Watson, editor, *Proceedings of the Dundee Biennial Conference on Numerical Analysis 1974*, pages 73-89. Springer Verlag, New York, 1975.
- [7] R. W. Freund. A Transpose-Free Quasi-Minimal Residual algorithm for non-Hermitian linear systems. *SIAM Journal on Scientific Computing*, 14(2):470-482, 1993.
- [8] M. R. Hestenes and E. L. Stiefel. Methods of conjugate gradients for solving linear system. *Journal of Research of the National Bureau of Standards*, 49:409-436, 1952.
- [9] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, 2003.
- [10] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10(1):36-52, 1989.

- [11] Y. Saad and M. H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing*, 7:856-869, 1986.
- [12] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 12:631-644, 1992.

