FINDING RIGHTMOST EIGENVALUES OF LARGE SPARSE NONSYMMETRIC PARAMETERIZED EIGENVALUE PROBLEMS

Minghao Wu
Department of Mathematics
University of Maryland, College Park

Advisor: Dr. Howard Elman
Department of Computer Science
University of Maryland, College Park

Abstract

The goal of this report is to explore the numerical algorithm of finding rightmost eigenvalues of large sparse nonsymmetric parameterized eigenvalue problems, and how the rightmost eigenvalues vary as certain parameter changes. We implemented Arnoldi algorithm (both exact and inexact) and Implicitly Restarted Arnoldi algorithm with shift-invert transformation to reproduce computational results of several test problems in the literature. We also applied the algorithm to the study of dynamical equilibrium.

Introduction

Consider the eigenvalue problem

$$Ax = \lambda Bx \tag{1}$$

where A and B are large sparse non-symmetric real $N \times N$ matrices and they usually depend on one or several parameters in real applications. We are interested in computing its rightmost eigenvalues (namely, eigenvalues with the largest real parts). The motivation lies in the determination of the stability of steady state solutions of nonlinear systems of the form

$$B\frac{du}{dt} = f(u) \quad f : R^N \to R^N, u \in R^N \tag{2}$$

with large N and where u represents a state variable (velocity, pressure, temperature, etc). B is often called the mass matrix. Define the Jacobian matrix for the steady state u^* by $A = df/dx(u^*)$, then u^* is stable if all the eigenvalues of (1) have negative real parts. Typically, f arises from the spatial discretization of a PDE (partial differential equation). When finite differences are used to discretize a PDE, then often $B = I$ and (1) is called a standard eigenproblem. If the equations are discretized by finite elements, then the mass matrix $B \neq I$ and (1) is called a generalized eigenvalue problem. For problems arising from fluid mechanics, B is often singular[1].

Beside the numerical algorithm of computing the rightmost eigenvalues of (1), we are also interested in exploring the effect of parameters on the stability of steady state solutions. In particular, we are interested in finding the critical parameter value under which the rightmost eigenvalues cross the imaginary axis thus indicates that the steady state solution becomes unstable. This is an important part of the study of bifurcation of dynamical equilibrium.

The plan of this report is as follows: in the first part, we will describe and explain the algorithms we implemented; in the second part, computational results together with some technical details for several test problems will be presented.

Methodology

Eigenvalue Solvers

Since both A and B are large and sparse, direct methods such as the QZ algorithm for the generalized problem and the QR algorithm for the standard problem are not feasible. A more efficient approach is to solve the standard eigenvalue problem $Tx = \theta x$, which is a transformation of (1), by iterative methods like Arnoldi algorithm, subspace iteration and Lanczos' method[1]. Another reason why iterative methods are more suitable for our

problem is that we are only interested in computing the rightmost part of the spectrum instead of the complete spectrum. The eigenvalue solvers we implemented are Arnoldi algorithm and its variants, such as the Implicitly Restarted Arnoldi (IRA) algorithm.

1. Basic Arnoldi Algorithm

Arnoldi algorithm is an iterative eigensolver based on Krylov subspaces. Given a matrix A and a random normal vector u_1, a k-dimensional Krylov subspace is the subspace spanned by the columns of the following matrix:

$$K_k(A,u_1) = [u_1\ Au_1\ A^2u_1\ \dots\ A^{k-1}u_1]$$

provided that they are linearly independent. k+1 steps of Arnoldi algorithm give us the following decomposition:

$$AU_k = U_kH_k + \beta_k u_{k+1}e_k^T \tag{3}$$

where U_k is an orthonormal basis of the k-dimensional Krylov subspace, u_{k+1} is normal and orthogonal to U_k, H_k is a $k\times k$ upper Hessenberg matrix, β_k is a scalar and e_k is the $k\times 1$ vector $[0\ 0\ \dots\ 0\ 1]$. We usually choose k such that $k<<N$ (recall that the dimension of the matrices A and B is $N\times N$).

Leftmultiply (3) by U_k^T:

$$U_k^TAU_k = U_k^TU_kH_k + \beta_kU_k^Tu_{k+1}e_k^T = H_k. \tag{4}$$

Therefore, H_k is the projection of A onto the k-dimensional Krylov subspace. We hope that the eigenvalues of H_k can be good approximation of those of A so that we can solve a much smaller eigenvalue problem instead. Suppose (λ,z) is an eigenpair of H_k. Rightmultiply (4) by z:

$$U_k^TA(U_kz) = H_kz = \lambda z = \lambda(U_k^TU_k)z = U_k^T\lambda(U_kz). \tag{5}$$

We can view (λ,U_kz) as an approximated eigenpair of A with residual $A(U_kz) - \lambda(U_kz)$ and obtain the following from (5):

$$U_k^T(A(U_kz) - \lambda(U_kz)) = 0.$$

This means the residual is always orthogonal to the k-dimensional Krylov subspace. Fig. 1 illustrates their relation.

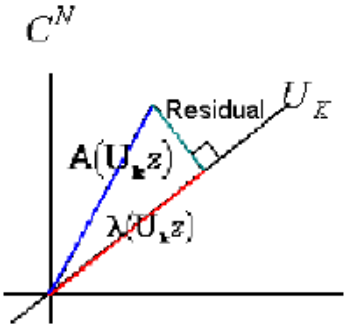


Fig. 1 The relation among $A(U_k z)$, $\lambda(U_k z)$ and the residual between them

As k increases, the residual will decrease and in the limit case, when $k = N$, $A(U_k z) = \lambda(U_k z)$, which means $(\lambda, U_k z)$ is an exact eigenpair of A.

However, we should point out that it is not a good idea to apply it naively to our problem. There are mainly two reasons: (1) since we are very often dealing with large matrices, increasing k to improve the performance of Arnoldi algorithm is not practical; for example, suppose A is of size 10,000×10,000 and $k = 100$, we need 10 megabytes to store the Krylov basis to double precision[2]; (2) in problems arise from fluid mechanics, mass matrix B is singular and this will give rise to spurious eigenvalues if we do not do anything clever. A lot of variants of Arnoldi algorithm follow the line of restarting the basic Arnoldi algorithm with a more carefully chosen vector u_1. We implemented one of them, the Implicitly Restarted Arnoldi algorithm (IRA).

2. Implicitly Restarted Arnoldi (IRA) Algorithm

The basic idea of IRA is to filter out unwanted eigendirections from the original starting vector u_1 by using the most recent spectrum information and also a clever filtering technique[2].

Suppose we want to find k rightmost eigenvalues and we first use the basic Arnoldi algorithm to find m $(m>k, m<<N)$ approximated eigenpairs:

$$(\mu_1,x_1),\ (\mu_2,x_2),\ \ldots,\ (\mu_m,x_m)$$

$(Re(\mu_i)\geq Re(\mu_j),\ i<j)$ and obtain the following Arnoldi decomposition:

$$AU_m = U_m H_m + \beta_m u_{m+1} e_m^T. \quad (6)$$

We want to filter out the eigendirections $x_{k+1}, \ldots, x_m$ from the starting vector u_1 so that the new starting vector can be richer in the k eigendirection we are interested in. Suppose now we want to filter out eigendirection x_{k+1}. We first subtract $\mu_{k+1}U_m$ from both sides of (6):

$$(A - \mu_{k+1}I)U_m = U_m(H_m - \mu_{k+1}I) + \beta_m u_{m+1} e_m^T$$

then compute the QR decomposition of $H_m - \mu_{k+1}I$ $(H_m - \mu_{k+1}I = Q_1R_1)$ and get:

$$(A - \mu_{k+1}I)U_m = U_m Q_1 R_1 + \beta_m u_{m+1} e_m^T. \quad (7)$$

Next, rightmultiply both sides of (7) by Q_1:

$$(A - \mu_{k+1}I)(U_m Q_1) = (U_m Q_1)(R_1 Q_1) + \beta_m u_{m+1} e_m^T Q_1 \quad (8)$$

and add $\mu_{k+1}U_m Q_1$ to both sides of (8):

$$A(U_m Q_1) = (U_m Q_1)(R_1 Q_1 + \mu_{k+1}I) + \beta_m u_{m+1} e_m^T Q_1.$$

Therefore we end up with a new Arnoldi decomposition:

$$AU_m^{(1)} = U_m^{(1)} H_m^{(1)} + \beta_m u_{m+1} e_m^T Q_1$$

where $U_m^{(1)} = U_m Q_1$ and $H_m^{(1)} = R_1 Q_1 + \mu_{k+1}I$. If we take a closer look at the matrix $U_m^{(1)}$, we will see that its first column is proportional to $(A - \mu_{k+1}I)u_1$, thus the unwanted eigendirection x_{k+1} has been filtered out from the starting vector. Repeat the exact same process with $\mu_{k+2}, \ldots, \mu_m$ and we will end up with the Krylov basis $U_m^{(m-k)}$ whose first column is proportional to $(A - \mu_{k+1}I)\ldots(A - \mu_m I)u_1$. Therefore, all the unwanted eigendirections have been filtered out from the original starting vector and we obtain an Arnoldi decomposition that is equivalent to the one starting with the new starting vector. The filtering is basically done by a sequence of *m-k QR* decomposition on a small Hessenberg matrix.

Matrix Transformation

Matrix transformation is crucial in solving problem like (1) for two reasons: a practical reason is that iterative methods like Arnoldi algorithm cannot solve generalized $(B \neq I)$ eigenvalue problems, which makes the transformation from generalized to standard eigenvalue problem necessary; a second reason is of a numerical nature; it is well known that iterative eigenvalue solvers applied to A quickly converge to the well-separated

extreme eigenvalues of A; however, when A arises from the spatial discretization of a PDE, the rightmost eigenvalues are in general not well separated and this implies slow convergence[1]. Therefore, we also need matrix transformation to map rightmost eigenvalues to well-separated eigenvalues. We explored two well-known transformations: shift-invert transformation and Cayley transformation.

1. Shift-Invert Transformation

The definition of shift-invert transformation of (1) is the following:

$$T_{SI}(A,B;\sigma) = (A - \sigma B)^{-1}B$$

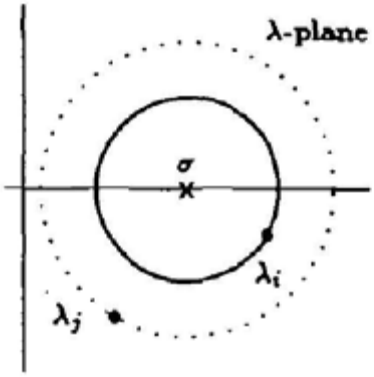
where σ is called the shift. The corresponding standard eigenvalue problem of (1) therefore is:

$$T_{SI}x = \theta x.$$

The eigenvalues of the original problem can easily be recovered by

$$\lambda = \sigma + 1/\theta.$$

Fig. 2 and Fig. 3 demonstrate this property of shift-invert transformation.



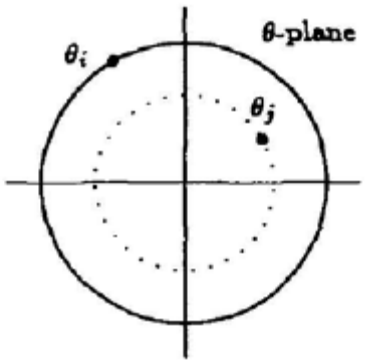


Fig. 2 *Fig. 3*

Fig. 2 Spectrum of the original problem $Ax = \lambda Bx$

Fig. 3 Spectrum of the new problem $T_{SI}x = \theta x$

To guarantee fast convergence, a good choice of σ should be an approximation of our targeted eigenvalue.

2. Cayley Transformation

Cayley transformation is defined by

$$T_C(A,B;\sigma,\tau) = 1 + (\sigma - \tau)T_{SI} = (A - \sigma B)^{-1}(A - \tau B)$$

where σ is called the shift and τ the anti-shift. Then the corresponding standard eigenvalue problem of (1) is

$$T_C x = \theta x.$$

To recover the eigenvalue of (1), we compute

$$\lambda = \sigma + (\sigma - \tau)/(\theta - 1).$$

Like shift-invert transformation, Cayley transformation maps λ close to σ to θ away from the unit circle; meanwhile, it also maps λ close to τ to θ with small modulus (that is where the name 'anti-shift' comes from). One interesting property of Cayley transformation is that the line $L(\sigma,\tau) = (\sigma + \tau)/2$ is mapped to the unit circle; moreover, λ to the left of this line are mapped inside the unit circle and λ to the right of this line are mapped outside of the unit circle. Fig. 4 and Fig.5 illustrate this property.

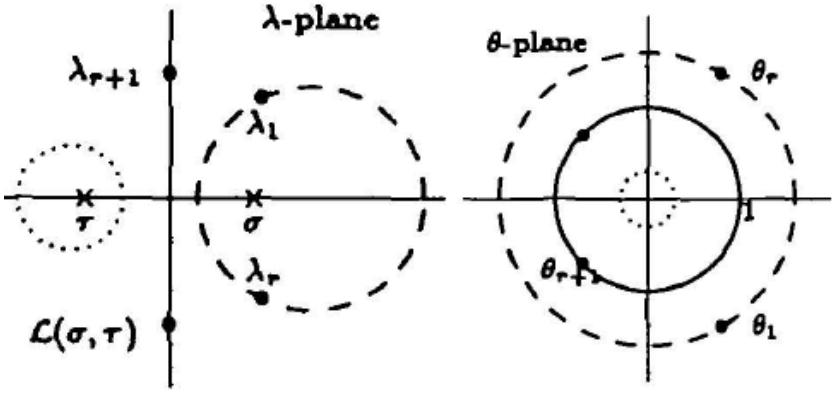


Fig. 4 *Fig. 5*

Fig. 4 Spectrum of the original problem $Ax = \lambda Bx$

Fig. 5 Spectrum of the new problem $T_C x = \theta x$

This property makes Cayley transformation specially suitable for the computation of rightmost eigenvalues, while the shift-invert transformation is suitable for computing eigenvalues lying close to the shift only. In practice, if we are interested in computing r

rightmost eigenvalues, we should choose σ and τ such that (approximated) λ_{r+1} lines on the line $L(\sigma,\tau)$.

3. Relation Between Shift-invert and Cayley transformation

Notice from its definition that Cayley transformation is just scaled and translated shift-invert transformation. Since Arnoldi's algorithm is translation invariant[3], the two transformation with the same starting vector will generate the same result in exact arithmetic. Therefore, if our eigensolver is exact Arnoldi algorithm, Cayley transformation will have no advantage over shift-invert transformation. However, when stationary iterative linear system solvers such as Jacobi and Gauss-Seidel are used, there are some advantages using Cayley transformation[4]. Nonetheless, we want to point out here that the most frequently used iterative linear system solver in this business is GMRES (Generalized Minimal Residual method) with preconditioning, which is much more robust and also insensitive to which matrix transformation one uses, so from now on we always consider shift-invert transformation, in either exact or inexact arithmetic.

Computation of Solution Paths

We consider the general problem

$$\boldsymbol{F}(\boldsymbol{x}, \rho) = \boldsymbol{0}$$

where $\boldsymbol{F}$: $R^{n+1} \to R^n$, $\boldsymbol{x} \in R^n$ is the state variable, and $\rho \in R$ is a parameter. We want to study the behavior of $\boldsymbol{x}$ as ρ varies. In particular, we are interested in the exchange of stability of $\boldsymbol{x}$ and the detection of Hopf bifurcation points.

1. Definitions

Definition: *solution path*:

The set

$$S = \{(\boldsymbol{x}, \rho) \in R^{n+1} : \boldsymbol{F}(\boldsymbol{x}, \rho) = \boldsymbol{0}\}$$

is called the *solution path*. Often in applications one is interested in computing the whole set of S or a continuous portion of it. For example in fluid mechanics $\boldsymbol{x}$ may represent the velocity and pressure of a flow, whereas ρ is some physical parameter such as the Reynolds number.

We introduce the following notation:

$$\boldsymbol{F}^0 = \boldsymbol{F}(\boldsymbol{x}_0, \rho_0),\ \boldsymbol{F}_{\boldsymbol{x}}{}^0 = \boldsymbol{F}_{\boldsymbol{x}}(\boldsymbol{x}_0, \rho_0),\ \boldsymbol{F}_{\rho}{}^0 = \boldsymbol{F}_{\rho}(\boldsymbol{x}_0, \rho_0).$$

Definition: *regular point*, *singular point* and *fold point (turning point)*

$(\boldsymbol{x}_0, \rho_0) \in S$ is called a *regular point* of S if $\boldsymbol{F}_{\boldsymbol{x}}{}^0$ is nonsingular. If a point $(\boldsymbol{x}_0, \rho_0) \in S$ is not regular, it is called a *singular point*. If $(\boldsymbol{x}_0, \rho_0) \in S$ is a singular point and if $\text{rank}(\boldsymbol{F}_{\boldsymbol{x}}{}^0) = n-1$ then $(\boldsymbol{x}_0, \rho_0)$ is called a *fold point* (or *turning point*) if $\boldsymbol{F}_{\rho}{}^0 \notin \text{image}(\boldsymbol{F}_{\boldsymbol{x}}{}^0)$. In this case, the $n \times (n+1)$ augmented Jacobian $[\boldsymbol{F}_{\boldsymbol{x}}{}^0 | \boldsymbol{F}_{\rho}{}^0]$ must have rank n and hence has a subset of n linearly independent columns.

Definition: *Hopf bifurcation point*

$(\boldsymbol{x}_0, \rho_0) \in S$ is called a *Hopf bifurcation point* if $\boldsymbol{F}_{\boldsymbol{x}}{}^0$ has a pair of pure imaginary eigenvalues and no other eigenvalues on the imaginary axis. At a Hopf bifurcation point, steady state solution loses stability and periodic solutions can be observed.

2. Keller's Pseudo-Arclength Continuation Method

When $\boldsymbol{F}_{\boldsymbol{x}}{}^0$ is nonsingular, Implicit Function Theorem implies that we can compute a nearby point $(\boldsymbol{x}_1, \rho_0 + \Delta\rho)$ of $(\boldsymbol{x}_0, \rho_0)$ on S by Newton's method. However, this method will fail when a singular point is approached. This is the motivation of Keller's pseudo-arclength continuation method[5].

We assume the following: there is an arc of S such that at all points on the arc

$$\text{rank}[\boldsymbol{F}_{\boldsymbol{x}} | \boldsymbol{F}_{\rho}] = n.$$

Consequently, any point on the arc is either a regular point or fold point of S. The Implicit Function Theorem thus implies that the arc is a smooth curve in R^{n+1}, and so there is a unique tangent direction at each point of the arc. Keller's idea is to reparameterize the system by a new parameter t, the projection of the arclength on to the tangent direction.

Suppose now we start from a point $(\boldsymbol{x}_0, \rho_0)$ at which the unit tangent vector is $\boldsymbol{\tau}_0 = [\boldsymbol{s}_0^T, \sigma_0]^T$, and we want to compute a nearby point $(\boldsymbol{x}_1, \rho_1)$ on the arc. We introduce an extended system

$$\boldsymbol{H}(\boldsymbol{y}, t) = \boldsymbol{0}$$

where $\boldsymbol{y} = (\boldsymbol{x}, \rho) \in R^{n+1}$ and $\boldsymbol{H}: R^{n+2} \rightarrow R^{n+1}$ is given by

$$\boldsymbol{F}(\boldsymbol{x}, \rho)$$
$$\boldsymbol{s}_0^T(\boldsymbol{x} - \boldsymbol{x}_0) + \sigma_0(\rho - \rho_0) - (t - t_0)$$

The last equation in the new system is the equation of the plane perpendicular to to $\boldsymbol{\tau}_0$ a distance $\Delta t = t - t_0$ from t_0 (see Fig. 6).

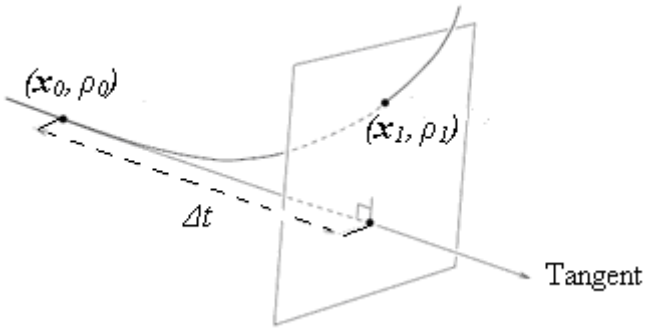


Fig. 6 Pseudo arclength continuation

Compute the Jacobian matrix of the new system:

$$H_y = \begin{bmatrix} F_x & F_\rho \\ s_0^T & \sigma_0 \end{bmatrix}.$$

The advantage of the extended system is that Jacobian matrix H_y is always nonsingular[5], so Newton's method always works.

3. Detection of Exchange of Stability and Hopf Bifurcation Points

Recall that both exchange of stability and Hopf bifurcation phenomena can be characterized by rightmost eigenvalues of the Jacobian matrix $\boldsymbol{F}_x$. Therefore, we can

detect them by applying the Arnoldi algorithm described earlier to $\boldsymbol{F_x}$ along the continuation process.

Computational Results

To test our code, we selected several test problems from the available literature and tried to reproduce the same results. In this section, we will present our computational results to various computation tasks we have completed for these test problems, and compare them with results in the literature.

Test Problem 1: Olmstead Model

1. Problem statement

The problem is from a paper by Olmstead *et al* in 1986[6]. The model is governed by a coupled system of PDEs:

$$\begin{cases} u_t = S_{xx} + cu_{xx} + Ru - u^3 \\ bS_t = (1-c)u - S \end{cases}$$

with boundary condition:

$$u = S = 0 \text{ at } x = 0, \pi.$$

This model represents the flow of a layer of viscoelastic fluid heated from below. u is the speed of the fluid, S is a quantity related to viscoelastic forces, and b, c, R are all parameters. In particular, R is called the Rayleigh number and is the parameter we are interested in.

2. Discretization of The PDE

We use second centered difference method to discretize the system of PDEs (parameter values: $b = 2$, $c = 0.1$ and $R = 0.6$). Grid size is $h = 1/(N/2)$ and the unknowns are ordered by grid points, i.e.:

$$\boldsymbol{y} = [u_1\ S_1\ u_2\ S_2\ \ldots\ u_{N/2}\ S_{N/2}]^T.$$

Then the system can be written as $d\mathbf{y}/dt = \mathbf{f}(\mathbf{y})$. We evaluate the Jacobian matrix A at the trivial steady state solution $\mathbf{y}^* = \mathbf{0}$. For this problem, the mass matrix $B = I$, it is a standard eigenvalue problem. Let $N = 1000$, so the resulting matrix is a 1000×1000 sparse nonsymmetric matrix with bandwidth 6 (see Fig. 7).

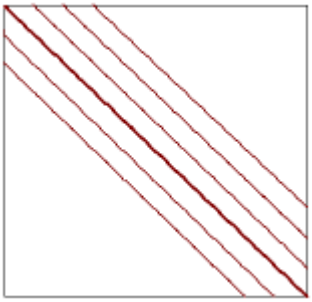


Fig. 7 Matrix structure of Olmstead model

3. Algorithm 1: Arnoldi method with shift-invert transformation

(1) Normalize random starting vector v_1

(2) for $i = 1$ to k do

(2.1) Solve for w_i: $(A - \sigma B)w_i = Bv_i$.

(2.2) Form $h_{ji} = v_j^H w_i, j = 1, \ldots, i$.

(2.3) Compute $w_i \leftarrow w_i - \sum_{j=1}^{i} h_{ji} v_j$. (QR - Gram - Schmidt)

(2.4) Form $h_{i+1,i} = ||w_i||_2$.

(2.5) From $v_{i+1} = w_i/h_{i+1,i}$. (Normalization)

(3) Let $H_k = \left[h_{ij}\right]_{i,j=1}^{k}$ where $h_{ij} = 0$ for $j>i+1$ and $V_k = [v_1,\ldots,v_k]$.

(4) Compute the eigenpairs (λ_i, z_i), $i = 1, \ldots, k$ of H_k by the QR-method.

(5) Compute the approximate eigenpairs $(\sigma + 1/\theta_i, V_k z_i)$, $i = 1, \ldots, k$.

4. Computational Results

When $b = 2$, $c = 0.1$, $R = 0.3$, the rightmost eigenvalues are $\lambda_{1,2} = -0.549994 \pm 2.01185i$[1]. We can use either of them as the shift of Arnoldi algorithm for our problem ($b = 2$, $c = 0.1$, $R = 0.6$). The result we obtained using exact Arnoldi algorithm (meaning in (2.1) of Algorithm 1, we use LU decomposition to solve the linear system) is:

rightmost eigenvalues: $\lambda_{1,2} = 0 \pm 0.4472i$

residual: $||Ax_i - \lambda_i x_i|| = 8.4504 \times 10^{-12}$, $i = 1,2$

and it agrees with the literature[1].

We also implemented inexact Arnoldi algorithm, which means in line (2.1) of Algorithm 1, linear systems of the form $(A - \sigma B)w = Bv$ are solved by GMRES with ILU(0) preconditioning. Using relaxation strategy to choose an appropriate error tolerance for GMRES can save us a lot of computational cost without decreasing the accuracy of the eigencomputation[7]. The relaxation strategy we implemented is

$$\tau_k = \frac{10^{-10}}{\|r_{k-1}\|_2}$$

where τ_k is the tolerance of GMRES at the k^{th} Arnoldi step and r_{k-1} is the relative error of the rightmost eigenpair computed at the $k\text{-}1^{th}$ Arnoldi step.

Let $N = 5000$, $b = 2$, $c = 0.1$, $R = 4.7$, and size of Krylov subspace is 20. Fig. 8 shows the different performance of GMRES with relaxation and without relaxation ($\tau_k = 10^{-10}$).

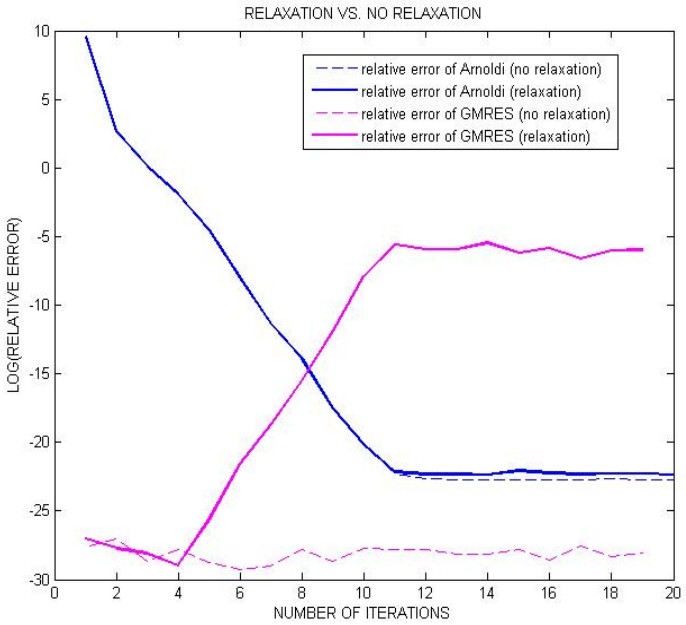


Fig. 8 Comparison of GMRES with relaxation and without relaxation

Observe that although our linear systems are solved less and less accurate, in the end we are able to gain the same order of accuracy in the rightmost eigenpair we computed. Again our computational result agrees with the literature[4].

The second computational task we conducted is to compute critical Rayleigh number R_C under different values of b and c:

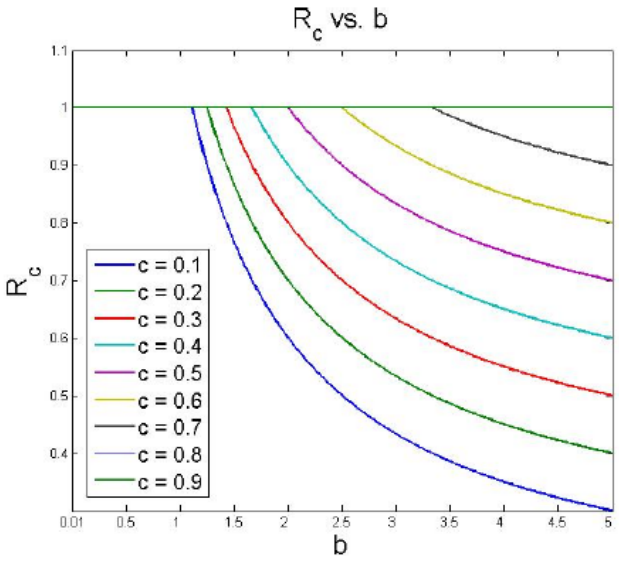


Fig. 9 Critical Rayleigh number

The theoretical result in the literature[6] is:

$$R_C = \begin{cases} 1, & b \le \dfrac{1}{1-c} \\ \dfrac{1}{b} + c, & b > \dfrac{1}{1-c} \end{cases}$$

to which our computational result also agrees (see Fig. 8).

Test Problem 2: Tubular Reactor Model

1. Problem Statement

The equations describing the conservation of reactant A and energy for the nonadiabatic tubular reactor with axial mixing appear below in dimensionless form[8]:

$$\begin{cases} \dfrac{\partial y}{\partial t} = \dfrac{1}{Pe_m}\dfrac{\partial^2 y}{\partial s^2} - \dfrac{\partial y}{\partial s} - Dye^{\gamma-\gamma/\theta} \\ \dfrac{\partial y}{\partial t} = \dfrac{1}{Pe_h}\dfrac{\partial^2 \theta}{\partial s^2} - \dfrac{\partial \theta}{\partial s} - \beta(\theta-\theta_0) + BDye^{\gamma-\gamma/\theta} \end{cases}$$

with boundary condition:

$$\frac{\partial y}{\partial s} = Pe_m\left(y-1\right) \text{ at } s = 0$$

$$\frac{\partial \theta}{\partial s} = Pe_h\left(\theta-1\right) \text{ at } s = 0$$

$$\frac{\partial y}{\partial s} = \frac{\partial \theta}{\partial s} = 0 \text{ at } s = 1.$$

y is the velocity, θ is the temperature, and Pe_m, Pe_h, B, D, β, γ are all parameters. D is called the Damkohler number and is the parameter we are interested in.

2. Discretization of The PDE

The discretization of this problem is basically the same with the last problem, except that one needs to compute the steady state solution by Newton's method first. We will skip the details here.

3. Algorithm 2: Keller's pseudo-arclength continuation

(1) Compute $\boldsymbol{x}_0$, $F_{\boldsymbol{x}}^{\,0}$ and $F_\rho^{\,0}$ at initial parameter value ρ_0.

(2) While $\rho_0 \le \rho_{end}$, do

(2.1) Compute the steady state solution $\boldsymbol{x}_0$ using Newton's method

(2.2) Compute $F_{\boldsymbol{x}}^{\,0}$ and $F_\rho^{\,0}$.

(2.3) Compute the rightmost eigenvalues of $F_{\boldsymbol{x}}^{\,0}$ using Arnoldi method.

(2.4) Solve for z_0: $F_{\boldsymbol{x}}^{\,0} z_0 = -F_\rho^{\,0}$,

and then compute unit tangent vector: $\begin{bmatrix} s_0 \\ \sigma_0 \end{bmatrix} = \frac{1}{\left(\|z_0\|^2 + 1\right)^{1/2}} \begin{bmatrix} z_0 \\ 1 \end{bmatrix}$.

(2.5) (Euler predictor) Choose a step length Δt and set

$$\begin{bmatrix} x^1 \\ \rho^1 \end{bmatrix} = \begin{bmatrix} x^0 \\ \rho^0 \end{bmatrix} + \Delta t \begin{bmatrix} s_0 \\ \sigma_0 \end{bmatrix}.$$

(2.6) (Newton's method) For $k = 1, \ldots, k_{max}$, iterate

$$\begin{bmatrix} x^{k+1} \\ \rho^{k+1} \end{bmatrix} = \begin{bmatrix} x^k \\ \rho^k \end{bmatrix} + \begin{bmatrix} d^k \\ \delta^k \end{bmatrix}$$

with

$$\begin{bmatrix} F_x^k & F_\rho^k \\ s_0^T & \sigma_0 \end{bmatrix} \begin{bmatrix} d^k \\ \delta^k \end{bmatrix} = -\begin{bmatrix} F^k \\ s_0^T\left(x^k - x_0\right) + \sigma_0\left(\rho^k - \rho_0\right) - \Delta t \end{bmatrix}.$$

(2.7) Update $\boldsymbol{x}_0 = \boldsymbol{x}^*$, $\rho_0 = \rho^*$, $F_{\boldsymbol{x}}{}^0 = F_{\boldsymbol{x}}{}^*$, $F_\rho{}^0 = F_\rho{}^*$ where $\boldsymbol{x}^*$, ρ^*, $F_{\boldsymbol{x}}{}^*$ and $F_\rho{}^*$ are computed in (2.3).

(2.8) Go back to (2).

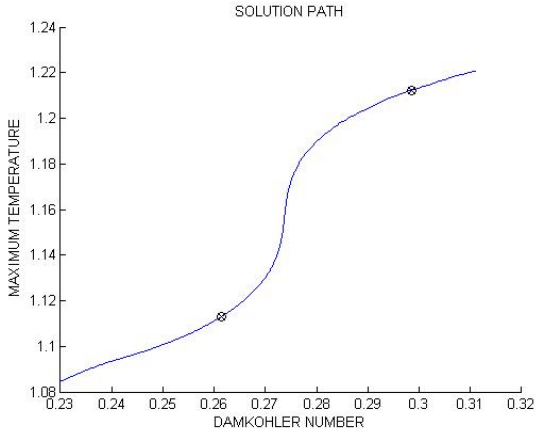
(3) Plot the solution path and bifurcation points computed.

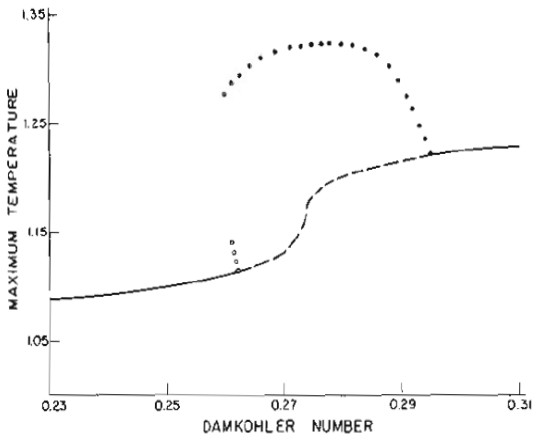
Note that the Jacobian matrix in (2.3) is H_y where $y = [\boldsymbol{x}^T, \rho]^T$. H_y is always nonsingular under the assumption that all the points on the solution path are either regular or fold points. In line (2.3) we compute the rightmost eigenvalues of the Jacobian matrix to detect changes of stability and Hopf bifurcation phenomena.

4. Computational Results

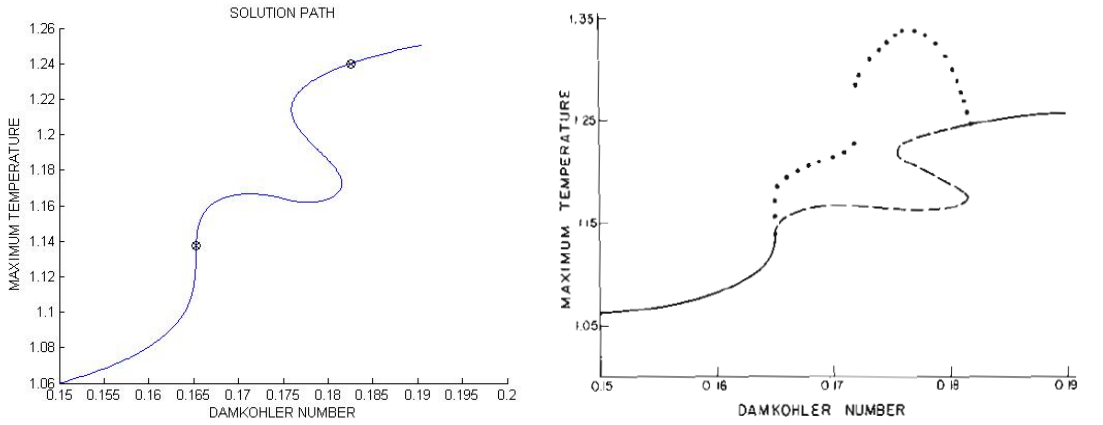
We computed the solution paths and detected fold points and Hopf bifurcation points of this problem under several different sets of parameters (Pe_m, Pe_h, B, β, γ). All our results agree with the results in the literature[8]. In Fig. 10, we compare the two.

$Pe_m = Pe_h = 5, B = 0.5, \beta = 3.5, \gamma = 25$:

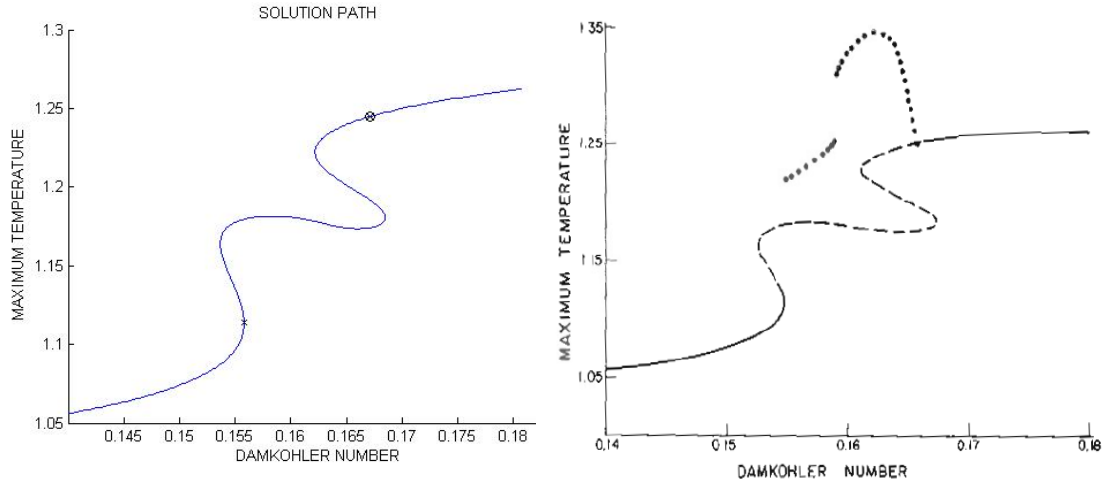




$Pe_m = Pe_h = 5, B = 0.5, \beta = 2.5, \gamma = 25$:



$Pe_m = Pe_h = 5, B = 0.5, \beta = 2.35, \gamma = 25$:



$Pe_m = Pe_h = 1, B = 0.5, \beta = 4, \gamma = 75$:

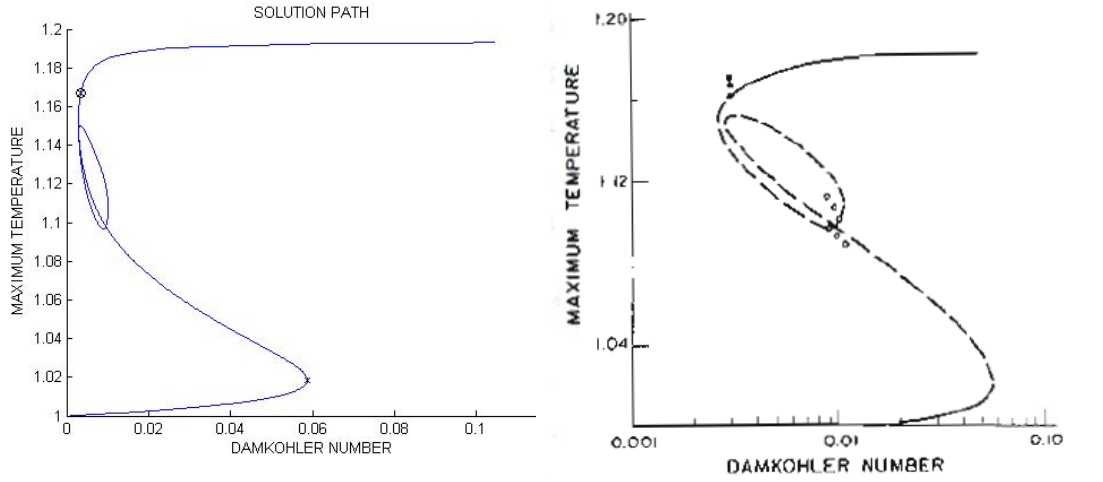


Fig. 10 Comparison of computational result and result in the literature

(note the different scaling of the two)

(left column: computational result ×: exchange of stabiltiy, o: Hopf bifurcation points;

right column: result in the literature ---: unstable, —: stable, •••: periodic solution, which appears at Hopf bifurcation points)

Test Problem 3

1. Problem Statement

This test problem comes from a paper by Meerbergen and Roose[9]. Consider the following eigenvalue problem:

$$\begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix} x = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} x$$

where K is a 200×200 matrix, C is a 200×100 matrix, and M is a 200×200 matrix. They are all of full rank. K, C and M are generated artificially and all the eigenvalues of the problem lie between -3 and 50. Eigenvalue problem with this kind of block structure appears in the stability analysis of steady state solution of Navier-Stokes equations for incompressible flow.

2. Algorithm 3: implicitly restarted Arnoldi method

Suppose we are interested in finding the k rightmost eigenpairs of $Ax = \lambda Bx$.

(1) Compute m ($m>k$) rightmost eigenpairs (λ_i, x_i) $i = 1,\ldots,m$ and $Re(x_i) \geq Re(x_j)$ for $i<j$ using Algorithm 1.

(2) Filter out the unwanted eigendirections by applying shifted QR algorithm with shifts $\lambda_{k+1}, \ldots, \lambda_m$.

(3) Contract the m-dimensional Arnoldi decomposition to a k-dimensional one.

(4) While $||Ax_i - \lambda_i Bx_i||_2 \geq tolerance$, $i = 1,\ldots,k$:

(4.1) Expand the k-dimensional Arnoldi decomposition back to an m-dimensional one.

(4.2) – (4.4) Repeat (1) to (3).

(4.5) Go back to (4).

(5) Output (λ_i, x_i) $i = 1,\ldots,k$ and their residual $||Ax_i - \lambda_i Bx_i||_2$.

3. Computational Result

We solved this problem with both the basic Arnoldi algorithm and Implicit Restarted Arnoldi (IRA) algorithm and compared their performance in the table below.

Table 1. Comparison of Arnoldi and IRA ($k = 10$, $\sigma = 60$)

Exact Eigenvalues	**Computed Eigenvalues (IRA)**	**Computed Eigenvalues (basic Arnoldi)**
49.9129 + 0i	49.9129 + 0i	**193.8412 + 7113830.9524i**
2.9112 + 1.1256i	2.9112 + 1.1256i	**193.8412 - 7113830.9524i**
2.9112 – 1.1256i	2.9112 – 1.1256i	49.9129 + 0i
2.5036 + 0.0624i	2.5036 + 0.0624i	3.0891 + 0i
2.5036 – 0.0624i	2.5036 – 0.0624i	2.6112 + 0i
2.3792 + 0i	2.3792 + 0i	-0.5752 + 2.7079i
2.1318 + 0.9356i	2.1318 + 0.9356i	-0.5752 - 2.7079i
2.1318 – 0.9356i	2.1318 – 0.9356i	-1.0901 + 0.7532i
2.1081 + 1.3539i	2.1081 + 1.3539i	-1.0901 - 0.7532i
2.1081 – 1.3539i	2.1081 – 1.3539i	**-47.3106 + 0i**

Notice that the basic Arnoldi algorithm produces spurious eigenvalues (in bold print), while the IRA gives the correct result. The reason for this is that matrix B is singular[1].

Test Problem 4: 2D Driven-Cavity Problem

1. Problem Statement

This is a classic test problem used in fluid dynamics, known as driven-cavity flow. It is a model of the flow in a square cavity ([-1,1]×[0,1]) with the lid moving from left to right. Different choices of the nonzero horizontal velocity on the lid give rise to different computational models[10]. We studied the regularized cavity model, which is governed by the following Navier-Stokes equation:

$$u_t - \upsilon\nabla^2\vec{u} + \vec{u}\cdot\nabla\vec{u} + \nabla p = 0$$
$$\nabla\cdot\vec{u} = 0$$

with boundary condition

$$u_x = 1 - x^4 \text{ at } y = 1.$$

$\vec{u} = (u_x, u_y)$ is the velocity, p is the pressure and $\upsilon > 0$ is a given constant called the kinematic viscosity. The parameter we are interested in is the so called Reynolds number, which will be defined soon.

As one can see from the above Navier-Stokes equation, there is a diffusion term $\upsilon\nabla^2\vec{u}$ and also a convection term $\vec{u}\cdot\nabla\vec{u}$. Having a quantitative measure of the relative contributions of viscous diffusion and convection is very useful. It turns out that the Reynolds number

$$R = \frac{UL}{\upsilon}$$

is such a measure. Here, U is the maximum magnitude of velocity on the inflow, and L denotes a characteristic length scale for the domain, for example, the constant in Poincare inequality. In the driven-cavity problem, $R = 2/\upsilon$.

2. Discretization of The PDE

We used the Mixed Finite Element Method to discretize the Navier-Stokes equation. Q_2 elements are used on velocity space, and Q_1 elements are used on pressure space. Q_2-Q_1 discretization is proved to be stable (meaning solution is unique)[10]. The eigenvalue problem arises from the discretization has the form

$$\begin{bmatrix} K & C^T \\ C & 0 \end{bmatrix} x = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} x$$

which has the same structure as test problem 3.

3. Computational Result

It is well-known that steady state solution of this problem is not stable when Reynolds number is very large (greater than 10^4) and the flow pattern develops into a time-periodic state[10]. Our goal is to find out the critical Reynolds number at which the steady state solution loses stability. There are two major computational difficulties.

First of all, when Reynolds number becomes larger, the computation of steady state solution becomes harder. It requires finer mesh (which implies larger linear system) therefore much heavier work in order to solve the linear systems to the same level of accuracy. Every time we refine the mesh, the matrices become almost 16 times bigger! Also, it has been observed that the radius of convergence of Newton's iteration is inversely proportional to the Reynolds number[10]. Thus, Newton's iteration will not converge for large Reynolds number. An alternative is Picard iteration which has a huge ball of convergence but the convergence rate is only linear. Therefore, the solution is to use a 'hybrid' method to solve the linear system: apply a number of Picard iterations first to get a good enough initial guess and then use Newton's iteration starting from this good initial guess. Fig. 11 shows the performance of the hybrid method ($R = 8000$, mesh size $h = 2^{-5}$, $tolerance = 10^{-10}$):

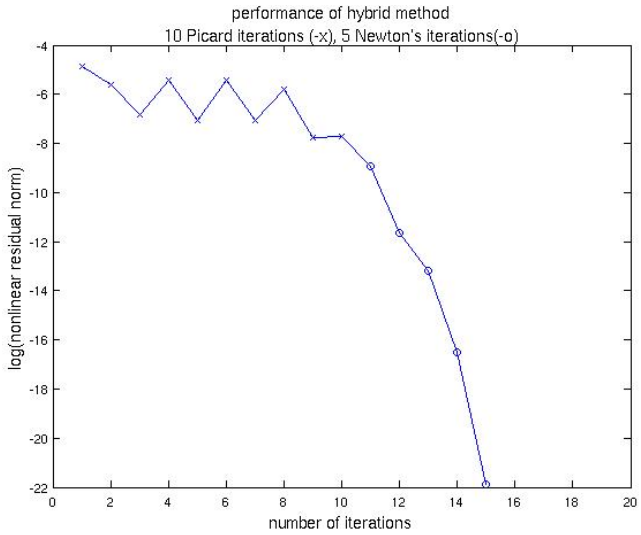


Fig. 11 Performance of hybrid method

We can see that Picard iteration converges slowly and the solution oscillates. Newton's iteration following it converges rapidly.

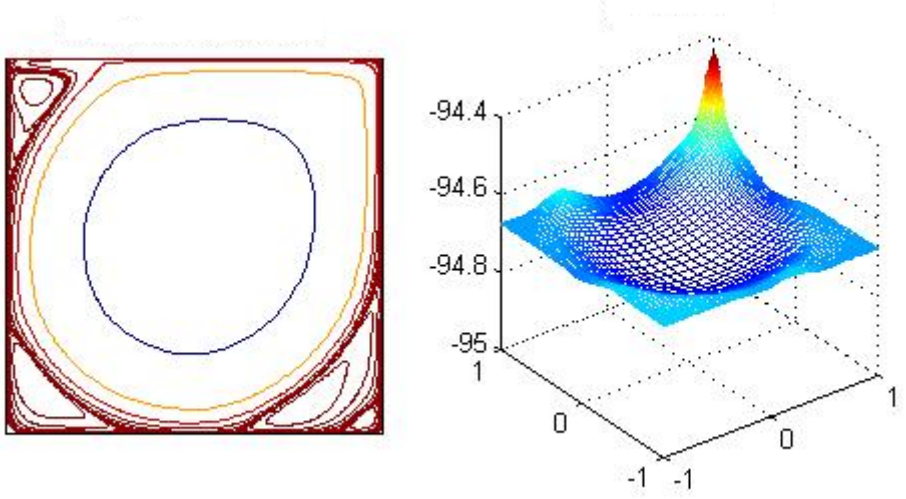


Fig. 12 Exponentially spaced streamline plot (left) and pressure plot (right) of a Q2-Q1 approximation with R = 8000

Secondly, it is not clear how to choose a good shift for IRA when we are approaching the critical Reynolds number. If the rightmost eigenvalue is real, then 0 is usually a good choice. However, when Reynolds number is large, the rightmost eigenvalue change from a real number to a complex conjugate pair whose imaginary part is much larger than the real part (in modulus). In this case, 0 is no longer a good choice. Complex shift becomes necessary. However, so far, we do not know how to choose such a shift.

In Table 2, we showed the rightmost eigenvalues we computed for Reynolds number from 7900 to 8500.

Table 2. Rightmost eigenvalues for R = 7900 to 8500 (meshsize 2^{-4})

Reynolds Number	Rightmost Eigenvalues
7500	-0.0044
7600	-0.0043
7700	-0.0043
7800	-0.0042
7900	-0.0042
8000	-0.0041

8100	-0.0038 ± 1.2926i
8200	-0.0020 ± 1.2932i
8300	-0.0002 ± 1.2937i
8400	0.0017 ± 1.2942i
8500	0.0037 ± 1.2947i

It can be observed that at a Reynolds number between 8000 and 8100, the rightmost eigenvalue change from real to a complex conjugate pair; also there is a Hopf bifurcation point when Reynolds number is around 8300, where rightmost eigenvalues cross the imaginary axis. This agrees with the literature[11]. When Reynolds number is 8300, the imaginary part of the rightmost eigenvalues is more than 6000 times greater than the real part (in modulus). Next, we will show a heuristic of detecting this kind of eigenvalues.

We first use 0 as our shift and compute 50 eigenvalues closest to this shift using IRA. The rightmost eigenvalues computed this way are shown in table 3:

Table 3. Rightmost eigenvalues computed by IRA with $\sigma = 0$, $k = 50$

Reynolds Number	**Rightmost Eigenvalues**
7500	-0.0044
7600	-0.0043
7700	-0.0043
7800	-0.0042
7900	-0.0042
8000	-0.0041
8100	-0.0041
8200	-0.0040
8300	-0.0040
8400	-0.0040
8500	-0.0039

We then draw the largest circle that exclude eigenvalue(s) with the largest modulus but include all the other eigenvalues (see Fig. 13).

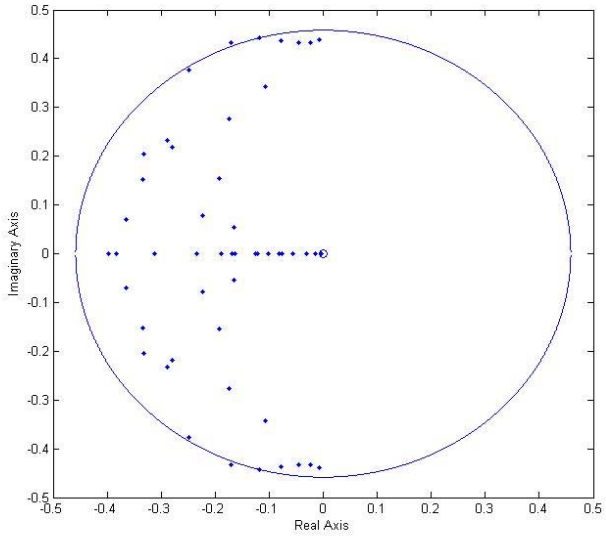


Fig. 13

Therefore every eigenvalue that lies in that circle must have been computed by IRA (otherwise, the eigenvalue(s) with the largest modulus won't be computed at all). Assume that the rightmost eigenvalue is not computed and has large imaginary part. Then it must sit very close to the imaginary axis and outside of the circle. Therefore, we use the point where the circle intersects with imaginary axis ($\pm ri$, r is the radius of the circle) as the new shift. We compute another 20 eigenvalues using the new shift (ri) and draw the circle again (see Fig. 14). Repeat this process unless we reach a certain number of times or a new rightmost eigenvalue with larger real part than the previous one is found. The results in table 2 are obtained after this validation process.

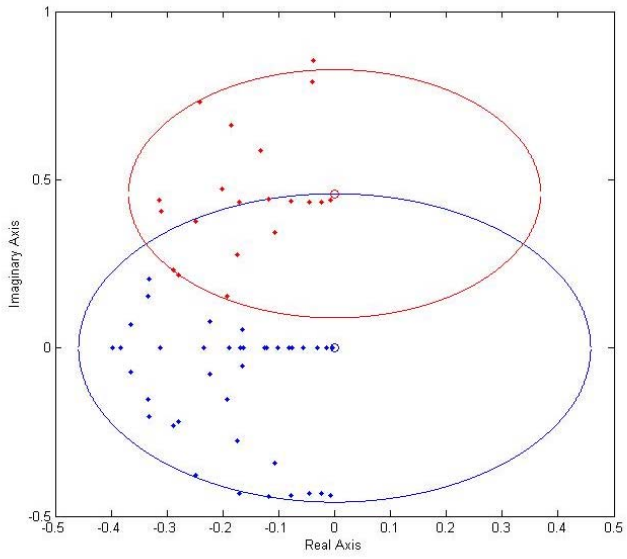


Fig. 14 (due to scaling, circles become ellipses)

Validation Test

Codes for *algorithm 1: Arnoldi method with shift-invert matrix transformation* and *algorithm 3: implicitly restarted Arnoldi method* are both tested on several benchmark problems in the existing literature[1,3,4,8,9,11] and they produced the same results as the literature. Another kind of validation test we have done is to generate random matrices using Matlab function 'rand', solve the eigenvalue problem and compare the results with those produced by Matlab function 'eig' or 'eigs'. A large amount of the second kind tests have been done and our codes always give the expected results.

Code for algorithm *2: Keller's pseudo-arclength continuation method* is tested by *test problem 2: tubular reactor model*[8]. We compute the solution paths and bifurcation points under 4 different sets of parameters and our results also agree with the literature. Comparing to algorithm 1 and 3, however, we have not done a large number of validation tests for algorithm 2. The main reason is that it is much harder to find or construct appropriate test problems. We will look into that in the future.

Future Work

There are still a lot of open questions that we need to answer. One of them is how to precondition IRA and implement iterative linear system solver inside IRA. Another one is how to detect rightmost eigenvalues whose imaginary part is much larger than their real part in modulus.

Acknowledgement

The IRA code we used is written by Fei Xue, a PhD student of Department of Mathematics, University of Maryland.

The software we used to discretize Navier-Stokes equations and find its steady state solution is IFISS, Incompressible Flow Iterative Solution Software, developed by Howard Elman (Department of Computer Science, University of Maryland), David Silvester (Department of Mathematics, University of Manchester) and Andy Wathern (Oxford University, Computing Laboratory).

Reference

[1] Meerbergen, K & Roose, D 1996 Matrix transformation for computing rightmost eigenvalues of large sparse non-symmetric eigenvalue problems. *SIAM J. Numer. Anal.* **16**, 297-346.

[2] Stewart, G. W. 2001 Matrix algorithms, volume II: eigensystems. *SIAM*.

[3] Meerbergen, K & Spence, A & Roose, D 1994 Shift-invert and Cayley transformations for detection of rightmost eigenvalues of nonsymmetric matrices. *BIT* **34**, 409-423.

[4] Meerbergen, K & Roose, D 1997 The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues. *SIAM J. Matrix Anal. Appl.* **18**, 1-20.

[5] Spence, A & Graham, Ivan G., Numerical Methods for Bifurcation Problems.

[6] Olmstead, W. E., Davis, W. E., Rosenblat, S. H., & Kath, W. L. 1986 Bifurcation with memory. *SIAM J. Appl. Math.* **40**, 171-188.

[7] Simoncini, V 2005 Variable accuracy of matrix-vector products in projection methods for eigencomputation. *SIAM J. Numer. Anal.* **43**, 1155-1174.

[8] Heinemann, R & Poore, A 1981 Multiplicity, stability, and oscilatory dynamics of the tubular reactor. Chemical Engineering Science, **36** 1411-1419.

[9] Meergergen, K & Spence, A 1997 Implicitly restarted Arnoldi with purification for the shift-invert transformation. *Math. Comput.* **66**, 667-698.

[10] Elman, H & Silvester, D & Wathen, A 2005 Finite elements and fast iterative solvers with applications in incompressible fluid dynamics. Oxford University Press

[11] Bruneau, C-H & Saad, M 2006 The 2D lid-driven cavity problem revisited. Computers & fluids **35**, 326-348.