Open access • Book Chapter • DOI:10.1007/3-540-55895-0_427

Parallel Homotopy Algorithm for Large Sparse Generalized Eigenvalue Problems: Application to Hydrodynamic Stability Analysis — Source link

G. Chen, H. B. Keller, S. H. Lui, Bernard Roux

Institutions: California Institute of Technology, Hong Kong University of Science and Technology

Published on: 01 Sep 1992 - Joint International Conference on Vector and Parallel Processing: Parallel Processing

Topics: n-connected, Homotopy analysis method, Intel iPSC, Square matrix and Diagonal matrix

Parallel Homotopy Algorithm For Large Sparse Generalized Eigenvalue Problems: Application to Hydrodynamic Stability Analysis

G. Chen, H.B. Keller, S.H. Lui, B. Roux

Parallel Homotopy Algorithm For Large Sparse Generalized Eigenvalue Problems: Application to Hydrodynamic Stability Analysis

G. Chen[1], H. B. Keller[2], S. H. Lui[3] and B. Roux[1]

[1] Institut de Mécanique des Fluides
1, rue Honnorat, 13003 Marseille, France

[2] Applied Mathematics 217-50, California Institute of Technology
Pasadena, CA 91125, U.S.A

[3] Dept. of Mathematics, Hong Kong University of Science and Technology
Hong Kong

Abstract. A parallel homotopy algorithm is presented for finding a few selected eigenvalues (for example those with the largest real part) of $Az = \lambda Bz$ with real, large, sparse, and nonsymmetric square matrix A and real, singular, diagonal matrix B. The essence of the homotopy method is that from the eigenpairs of $Dz = \lambda Bz$, we use Euler-Newton continuation to follow the eigenpairs of $A(t)z = \lambda Bz$ with $A(t) \equiv (1-t)D + tA$. Here D is some initial matrix and "time" t is incremented from 0 to 1. This method is, to a large degree, parallel because each eigenpath can be computed independently of the others. The algorithm has been implemented on the Intel hypercube. Experimental results on a 64-node Intel iPSC/860 hypercube are presented. It is shown how the parallel homotopy method may be useful in applications like detecting Hopf bifurcations in hydrodynamic stability analysis.

1 Introduction

We consider the generalized eigenvalue problem

$$Az = \lambda Bz \,, \tag{1}$$

with real, large, sparse, and nonsymmetric square matrix A and real, singular, diagonal matrix B. By a large sparse matrix we mean one that is prohibitively expensive to factorise by either a QR or a LU decomposition needed for the standard eigenvalue problem or by the QZ algorithm [1] for the generalized eigenvalue problem. We must therefore rule out all these traditional algorithms. Such sparse generalized eigenvalue problems are often encountered in linear hydrodynamic stability analysis, in which the matrix A arises from the discretization of the Navier-Stokes equations (by finite difference or finite element methods), and B is usually called "mass matrix" which is singular due to the continuity equation (and possibly to the boundary conditions). Usually, only certain eigenvalues and corresponding eigenvectors are required. These are either the ones closest to some complex number or the ones in some part of the complex plane. Projection methods, like the Lanczos and Arnoldi methods, are popularly used because they avoid factorising the matrix [2, 3].

Lecture Notes in Computer Science **634**, pp. 331-341 (1992)

With advances in parallel processing technology, it is now feasible to achieve high performance in solving truly complex problems. However, in order to use those general purpose computers in a specific application, algorithms that are to a large extent parallel in nature need to be developed. Homotopy algorithm is one of the parallel algorithms for eigenvalue problems. Remarkable numerical results have been obtained by using the homotopy algorithm for the tridiagonal symmetric matrix eigenvalue problem [4, 5]. Very recently, Li, Zeng and Cong [6] developed a homotopy algorithm for real nonsymmetric eigenvalue problems. Its performance is very encouraging. However, if the matrix is large and sparse, this algorithm suffers a serious drawback. In the reduction of the given matrix to a similar matrix in upper Hessenberg form by the Householder transformation, the matrix usually loses its sparsity. Hence the algorithm requires the explicit storage of the entire matrix. This may pose a problem if the matrix is so large that not all its entries can be accommodated within the main memory of the computer.

In this paper, we present a homotopy continuation algorithm for finding a few selected eigenvalues (for example those with the largest real part) and corresponding eigenvectors of Eq.(1). The essence of the homotopy method is that from the eigenpairs of

$$Dz = \lambda Bz, \tag{2}$$

we use Euler-Newton continuation to follow the eigenpairs of

$$A(t)z = \lambda Bz\ , \tag{3}$$

where $A(t) \equiv (1-t)D + tA$. Here D is some real initial matrix and "time" t is incremented from 0 to 1. At $t = 1$, we have the eigenpairs of Eq. (1). A great advantage of the homotopy method is that it is, to a large degree, parallel because each eigenpath can be computed independently of the others. With a special choice of initial matrix D, the storage requirement is proportional to the number of nonzero elements of the given matrix.

In §2, we describe the homotopy algorithm for computing the eigenpaths including the choice of the initial matrix D, stepsize selection and the method of solution of the linear systems which arise in the Newton iteration. This algorithm was developed in [7]. We have implemented a parallel program on the Intel hypercube, which is a distributed-memory, message-passing, multiprocessor machine. The results of numerical tests obtained on a 64-node Intel iPSC/860 hypercube are presented in §4 and our conclusion follows in §5.

With advances in parallel processing technology, it is now feasible to achieve high performance in solving truly complex problems. However, in order to use those general purpose computers in a specific application, algorithms that are to a large extent parallel in nature need to be developed. Homotopy algorithm is one of the parallel algorithms for eigenvalue problems. Remarkable numerical results have been obtained by using the homotopy algorithm for the tridiagonal symmetric matrix eigenvalue problem [4, 5]. Very recently, Li, Zeng and Cong [6] developed a homotopy algorithm for real nonsymmetric eigenvalue problems. Its performance is very encouraging. However, if the matrix is large and sparse, this algorithm suffers a serious drawback. In the reduction of the given matrix to a similar matrix in upper Hessenberg form by the Householder transformation, the matrix usually loses its sparsity. Hence the algorithm requires the explicit storage of the entire matrix. This may pose a problem if the matrix is so large that not all its entries can be accommodated within the main memory of the computer.

In this paper, we present a homotopy continuation algorithm for finding a few selected eigenvalues (for example those with the largest real part) and corresponding eigenvectors of Eq.(1). The essence of the homotopy method is that from the eigenpairs of

$$Dz = \lambda Bz, \tag{2}$$

we use Euler-Newton continuation to follow the eigenpairs of

$$A(t)z = \lambda Bz\ , \tag{3}$$

where $A(t) \equiv (1-t)D + tA$. Here D is some real initial matrix and "time" t is incremented from 0 to 1. At $t = 1$, we have the eigenpairs of Eq. (1). A great advantage of the homotopy method is that it is, to a large degree, parallel because each eigenpath can be computed independently of the others. With a special choice of initial matrix D, the storage requirement is proportional to the number of nonzero elements of the given matrix.

In §2, we describe the homotopy algorithm for computing the eigenpaths including the choice of the initial matrix D, stepsize selection and the method of solution of the linear systems which arise in the Newton iteration. This algorithm was developed in [7]. We have implemented a parallel program on the Intel hypercube, which is a distributed-memory, message-passing, multiprocessor machine. The results of numerical tests obtained on a 64-node Intel iPSC/860 hypercube are presented in §4 and our conclusion follows in §5.

2 Numerical Algorithm

2.1. Following the eigenpaths

Let $H: C^n \times C \times [0, 1] \rightarrow C^{n+1}$ be defined by

$$H(z,\lambda,t) = (1 - t)\begin{pmatrix} Dz - \lambda Bz \\ n(z) \end{pmatrix} + t\begin{pmatrix} Az - \lambda Bz \\ n(z) \end{pmatrix} = \begin{pmatrix} A(t)z - \lambda Bz \\ n(z) \end{pmatrix} \tag{4}$$

where $A(t) \equiv (1-t)D + tA$, with $0 \le t \le 1$.

Here, D is the initial matrix and n(z) is a normalisation equation. In our implementation of the homotopy algorithm, we use

$$n(z) \equiv \begin{pmatrix} (z^*z - 1)/2 \\ \text{Im } z_j \end{pmatrix} = 0 \tag{5}$$

where the superscript * denotes the conjugate transpose of a complex vector, z_j is the J^{th} component of z and is required to be nonzero (its imaginary part is put to zero). The normalisation (5) is not analytical as complex equations but their real forms are infinitely differentiable. In practice we choose J so that $|\text{Re } z_j| \ge |\text{Re } z_i|$ for all $i = 1,..., n$.

Let $DH = (H_z, H_\lambda, H_t)$, where H_z, H_λ, H_t denote the partial derivatives of H with respect to z, λ, and t respectively. When DH has full rank at $(z_0, \lambda_0, t_0) \in H^{-1}(0)$, the local solution set of $H(z, \lambda, t) = 0$ consists of a smooth 1-manifold $(z(s), \lambda(s), t(s))$ passing through (z_0, λ_0, t_0). Such a curve $(z(s), \lambda(s), t(s))$ is called an eigenpath. To follow an eigenpath $\Gamma = (z(s), \lambda(s), t(s))$ through (z_0, λ_0, t_0), we apply the usual Euler-Newton continuation method.

Namely, we first calculate the tangent vector $(\dot{z}, \dot{\lambda}, \dot{t})$ at (z_0, λ_0, t_0) on Γ by solving

$$H_z\dot{z} + H_\lambda\dot{\lambda} + H_t\dot{t} = 0, \tag{6}$$

with the normalization

$$\|\dot{z}\|^2 + |\dot{\lambda}|^2 + |\dot{t}|^2 = 1, \tag{7}$$

We have used dot to denote derivative with respect to s.

Using the Euler prediction

$$(z_0^1, \lambda_0^1, t_0^1) = (z_0, \lambda_0, t_0) + ds\ (\dot{z}, \dot{\lambda}, \dot{t}) \tag{8}$$

with stepsize ds, we apply Newton's method to $\mathbf{H} = 0$ with initial guess $(z_0^1, \lambda_0^1, t_0^1)$. Newton iteration will converge quadratically to the solution at a later time t_1 provided $t_1 - t_0$ is small enough.

It should be noted that in the Newton's corrections, we employ the pseudo-arclength method due to Keller [8]. That is, we augment $\mathbf{H} = 0$ with the equation

$$\mathrm{Re}\,[\,\dot{z}^*\,(z - z_0) + \bar{\dot{\lambda}}\,(\lambda - \lambda_0)\,] + \dot{t}\,(t - t_0) - ds = 0, \tag{9}$$

which represents the plane perpendicular to the tangent vector $(\dot{z}, \dot{\lambda}, \dot{t})$ at a distance ds from (z_0, λ_0, t_0). This method is used to follow successfully the eigenpath in the case where (z_0, λ_0, t_0) may be close to a bifurcation point.

In [6, 7], a method is developed to identify the bifurcation point and continuously follow the bifurcation branches. In practice we have two cases to consider (see Fig. 1):

1. Γ is a real eigenpath (Fig. 1a). On the opposite side of this bifurcation point, a complex eigenpath and its complex conjugate emerge. Using the tangent vector $\phi = (\dot{z}, \dot{\lambda}, \dot{t})$ at the bifurcation point, we need only to follow one of the bifurcation branches with the initial tangent vector $i\phi$ or $-i\phi$.

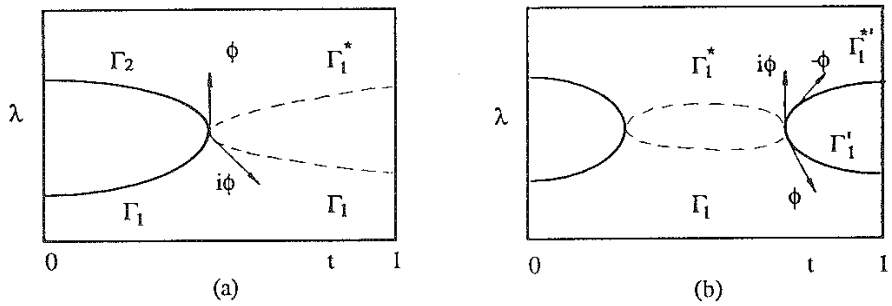

Fig. 1. Transition from (a) real eigenpath to complex eigenpath; (b) complex eigenpath to real eigenpath, at a bifurcation point. Dotted lines denote complex eigenpath.

2. Γ is a complex eigenpath (Fig. 1b). On the opposite side of this bifurcation point, two real eigenpaths emerge. We follow them in real space with the real tangent vectors ϕ and $-\phi$ respectively. Because the problem is being solved in real space, there is no chance of converging back to the complex solution. In our parallel implementation, the processor which became idle at a bifurcation point (in case 1) can be invoked to carry out the computation along one of those directions. This will be described in §3.

2.2 Choice of Initial Matrix D

We first rescale the given n x n matrices A and B so that the largest element of A (in absolute value) equals to one. Using Gerschgorin's theorem, we obtain a number r such that all the eigenvalues of A lie within a circle of radius r in the complex plane. The initial matrix D is chosen as a diagonal matrix whose elements are the diagonal elements of A, possibly perturbed so that

$$| a_{ii} - d_{jj} | > 2r/n^{3/2}, \quad i \neq j , \tag{10}$$

where a_{ij} denotes the (i, j) entry of A and d_{jj} denotes the (j, j) diagonal entry of D. More precisely, we define d_{11} to be a_{11}, and for $i > 1$, we take d_{ii} to be $a_{ii} + \delta$, where δ is the smallest number in magnitude which makes d_{ii} satisfy (10) for all $j < i$. With this choice of D, the initial tangents $(\dot{z}, \dot{\lambda}, \dot{t})$ at $t = 0$ are easily computed. There is no theoretical justification for the bound in (10). From numerical experiments on random matrices, the spectrum of D seems to be reasonably distributed with this bound.

2.3 Solution of Nonsymmetric Linear Systems

To compute the tangent vector and each Newton iteration, large, sparse, nonsymmetric linear systems must be solved. Due to memory, operation count, and parallelization considerations, iterative methods are used to solve these linear systems. In our implementation, we employ the conjugate gradients squared (CGS) algorithm [9] which is being widely used to solve large sparse linear systems. This method is simple to implement and has low storage requirements. In order to improve the convergerce rate, we use the preconditioning techniques based on the incomplete Gaussian elimination [10].

2.4 Stepsize Control

1. Initial stepsize

Let $(\dot{z}, \dot{\lambda}, \dot{t})$ be the unit tangent with $\dot{t} > 0$ at the initial point $(z_0, \lambda_0, 0) \in H^{-1}(0)$. If $\dot{t} \approx 1$, then the size of $(\dot{z}, \dot{\lambda})$ is relatively small and the eigenpath may be close to a straight line. A larger stepsize ds therefore can be used. In this case, we choose $ds = \dot{t}$. When $\dot{t} << 1$, we use $ds = \max\{0.01, |\dot{t}|^5\}$.

2. Increasing and cutting the stepsize

Let (z_0, λ_0, t_0) and (z_1, λ_1, t_1) be two points on the eigenpath with $t_0 < t_1 < 1$, we choose stepsize ds_2 for next point (z_2, λ_2, t_2) as follows:

$$ds_2 = ds_1 \{ \mathrm{Re}\,(\dot{z}_1^* \dot{z}_0 + \bar{\dot{\lambda}}_1 \dot{\lambda}_0) + \dot{t}_1 \dot{t}_0 + 0.5 \}, \qquad (11)$$

where ds_1 is the stepsize used to obtain (z_1, λ_1, t_1) from (z_0, λ_0, t_0). The idea is that when the two previous tangents are parallel, then we increase the stepsize by 50%. If the tangents are perpendicular, we decrease the stepsize by a half. We use the above scheme until the time t is close to one; then we solve the problem $H(z, \lambda, 1) = 0$.

Whenever the iteration in the Newton correction step fails to converge after, say, 8 iterations, we cut the stepsize by half and restart the correction step.

3 Parallel Implementation

We have implemented our algorithm on the Intel iPSC/860 which is a MIMD machine consisting of 64 processors in an hypercube connection. Each such processor, also called a node, executes its own program on data in its own memory. The nodes are controlled by another processor, called the host, which loads the programs into the nodes and starts them. Host and nodes communicate by message passing. Communication between nodes are also performed by sending and receiving messages, either synchronously, i.e., the processing stops until a message is sent or received; or asynchronously, where processing and communication overlap.

It is seen, therefore, that an algorithm is well suited on such a machine if it may be set as several independent processes, each working on its own data with a minimum of interprocess communication.

The homotopy method is, to a large degree, parallel in the sense that each eigenpath can be computed independently of the others. If the matrices A and B are stored in each node, then there is no communication overhead at all other than the trivial broadcast of the location of bifurcation points.

The problem considered, i.e., equation (1), is a special case of the generalized eigenvalue problem. When B is a singular diagonal matrix with m zero diagonal elements, the characteristic polynomial $\det(A-\lambda B) = 0$ is of degree equal to (n-m), where n is the dimension of A. Thus, there is not a complete set of eigenvalues for the problem. The missing eigenvalues may be regarded as "infinite". In the present paper, we intend to find the eigenvalue of (1) with the largest real part.

Computations of several test problems using random matrices A and B have been carried out. We observe that most of the eigenpaths move in a relatively simple fashion as t progresses. That is, there are no wild oscillations and the eigenpaths, in general, maintain their order where they are ordered according to $\mathrm{Re}(\lambda)$. However, we have not been able to derive a mechanism to guarantee that an eigenpath will end up (at $t = 1$) having an eigenvalue with the largest real part. This is the reason for which several eigenpaths have to be followed. This increases the likelihood that one of them will be the relevant.

In practice, to find an eigenvalue with the largest real part, we reorder the initial matrix D so that d_{ii}/b_{ii} is in descending order for $b_{ii} \neq 0$, where d_{ii} and b_{ii} denotes the (i, i) diagonal entry of D and B respectively. Then we compute k ($<<$n-m) eigenpairs by following the eigenpaths $(z(s), \lambda(s), t(s))$ which start from the eigenpairs of (2) corresponding to the k largest eigenvalues. Each node (processor) then follows k/p eigenpaths, where p is the number of nodes to be used (for simplicity, we assume that k/p is an integer). For a sample of ten random 640 by 640 matrices A and B, and for $k \geq 8$, the largest (real part) eigenvalue was found among the first k eigenpaths computed.

Using p nodes, the parallel homotopy algorithm consists of the following steps:

1. Initiating

The non-zero entries of the matrix A, and the diagonal elements of the matrix B are stored in two one-dimensional arrays, say RA and b respectively. To locate a special entry of A, two address arrays IR and IC are needed. IR(i) gives the address in RA of the first non-zero entry in row number i in A. IC(j) gives the column number (in A) of the entry

RA(j). IR has dimension n+1, while RA and IC have dimension nz, where nz is the number of non-zero elements of A.

Let us identify the nodes to be used by 0, 1,..., p-1, respectively. Node 0 reads input data from the file in which n, nz, b, RA, IR and IC are stored. Then we send these data to all other nodes from node 0. After this point, each node has the necessary data to follow the eigenpaths.

2. Following the eigenpaths

Divide the number of eigenpaths as equally as possible among the p nodes. Each node builds the initial matrix D and then follows its own eigenpaths concurrently, with the procedure described in §2.

3. Broadcasting the bifurcation points

The eigenpaths start off from D and advance using the Euler-Newton continuation, solving the problem in real space. When it detects that it is going backwards in time, i.e., $\dot{t} < 0$, then a bifurcation point has been passed. We identify this point first and store the values of (λ, t) at this bifurcation point in an array. We then broadcast these values to all other nodes. When a real eigenpath encounters a bifurcation point (Fig. 1a), it first checks whether the bifurcation point has been visited before. If it has been, then, the node stops following this path. By this way, only one path of a complex conjugate pair of paths is computed. This node is then free to compute one of the two real eigenpaths, say Γ_1' in Fig. 1b.

With the above considerations at the bifurcation points, computation time (and cost) would be reduced.

4 Numerical Experiments

In this section we present numerical results of our implementation of the parallel homotopy algorithm for two examples. All computations were done on the 64-node "Gamma" machine of CCSF at Caltech with double precision.

Our first example is just a test case, constructed as follows:

$$A = \begin{pmatrix} C & -I & & \\ -I & C & & \\ & & -I & \\ & & -I & C \end{pmatrix}, \text{ where } C = \begin{pmatrix} 4 & a & & \\ b & 4 & & \\ & & 4 & a \\ & & b & 4 \end{pmatrix}$$

with $a = -1 + \delta$, $b = -1 - \delta$, $\delta = 0.5$ (asymmetry parameter). We take **B** to be the unit matrix. The problem therefore becomes a standard eigenvalue problem $Az = \lambda z$. The dimensions of matrices **C** and **A** are 32 and 640 respectively. The number of non-zero elements of **A** is 3096. We computed the 64 eigenpairs corresponding to the 64 largest eigenvalues of the initial matrix **D**. The performance for this test example is shown in Table 1. The speedup S_p on p nodes is defined as

$$S_p = T_1 / T_p \ , \tag{12}$$

where T_1 is the execution time using one node, and T_p is the execution time using p nodes. The efficiency, E_p, is the ratio of the speedup and p. For this test example, we obtained an efficiency higher than 86%.

TABLE 1. Performance of the parallel homotopy algorithm, n = 640, nz = 3096.

Number of Nodes p	Execution Time (sec)	Residual $\max_i \| Az_i - \lambda z_i \|_2$	Speedup S_p	Efficiency E_p
1	4736	1.2657×10^{-11}		
4	1331	1.4567×10^{-11}	3.6	0.889
8	671	1.2789×10^{-11}	7.1	0.881
16	338	1.7524×10^{-11}	14.0	0.876
32	170	1.6435×10^{-11}	27.9	0.870
64	86	1.2534×10^{-11}	55.0	0.860

As expected, the largest eigenvalue ($\lambda = 7.7036491$) was found by following the first eigenpath, i.e., the eigenpath with the largest eigenvalue of the initial matrix.

Our second example is concerned with the prediction of the transition from steady to oscillatory surface-tension-driven flows in a liquid bridge [11]. The governing equations of the problem for velocity $U \equiv (u, v, w)$, temperature Θ, and pressure p in the liquid bridge of volume $V=\{ (r, \theta, z) \mid 0 \le r \le R, 0 \le \theta \le 2\pi, -L \le z \le L\}$ are the Navier-Stokes and energy equations:

$$U_t + Re\,(U\cdot\nabla)\cdot U = -\nabla p + \nabla^2 U, \tag{13}$$

$$\nabla\cdot U = 0, \tag{14}$$

$$\Theta_t + Re\,(U\cdot\nabla)\,\Theta = 1/Pr\;\nabla^2\Theta, \tag{15}$$

with appropriate boundary conditions. Re and Pr denote the Reynolds-Marangoni and the Prandtl numbers, respectively. For a given fluid, molten silicon, for example, we have a Prandtl number of 0.023. The flow behaviour is then characterized by the Reynolds-Marangoni number. At sufficiently low values of Re, the flow is laminar, steady and axisymmetric. When Re exceeds a critical value, the flow undergoes a transition to an oscillatory mode of convection which depends on the boundary conditions. Such transition in fact corresponds to a Hopf bifurcation where a generalized eigenvalue of the discretized Jacobian matrix of the linearized perturbation equations of (13-15) crosses the imaginary axis. We describe briefly here the method used to solve the stationary equations and to compute the eigenvalues of the perturbation equations.

The spatial finite difference discretization of (13-15) yields a system of ordinary differential equations of the form

$$M\frac{\partial x}{\partial t} = f(x, \lambda, \mu)\,, \tag{16}$$

where x is the solution vector (velocities, pressures and temperatures), λ is the bifurcation parameter (Re in the present case), and μ is the vector of the remaining fixed parameters in the problem. For fixed λ, the steady-state solution (which is assumed to be axisymmetric) x_0 of (16) will satisfy the equation

$$f(x_0, \lambda, \mu) = 0\,, \tag{17}$$

The stability of the steady-state $x_0(\lambda)$ can be analyzed by taking small perturbations of the form $\zeta(r, z)e^{\sigma t + in\theta}$ and linearizing (16) about x_0, obtaining

$$f_x\zeta = \sigma M\zeta\,, \tag{18}$$

where n is the azimuthal wave number and f_x is the Jacobian matrix.

As Re is varied, an axisymmetric steady-state solution may lose stability in one of two ways. One or more generalized eigenvalues of (18) may cross the imaginary axis with zero imaginary part. This case corresponds either to a limit point or to a bifurcation to another steady solution. Alternatively the eigenvalue may cross the imaginary axis at a nonzero imaginary value. This corresponds to a Hopf bifurcation to a periodic solution.

At Re = 71739 for n = 0, we solved the axisymmetric steady equations (17) using the Newton-Raphson method. The computation, using the above presented parallel homotopy algorithm, proceeds by following the first 8 eigenpaths on 8 nodes. The resulting eigenvalues with the largest real part among the 8 computed are the complex conjugate pair $\sigma = -41.36 \pm 404$ i. To locate a nearby Hopf bifurcation, we solved an extended steady-state system of equations proposed in [12] using the steady state solution, the imaginary part of eigenvalue ($\sigma_i = 404$) and the corresponding eigenvectors at that guessed Reynolds-Marangoni number. After 6 iterations, the system converged to a Hopf bifurcation point at a critical Reynolds-Marangoni number of 78000 and an angular frequency $\sigma_i = 446.2$. This result is obtained on a variable (r, z) mesh, 21x41. The dimension of the generalized eigenvalue problem is 2340x2340 with 18885 non-zero elements of the Jacobian matrix.

5 Conclusion

We have presented a homotopy method for solving a real, large, sparse generalized eigenvalue problem. The essence of the method is that all the eigenpairs can be obtained by following the eigenpaths starting from an initial known generalized eigenvalue problem. The algorithm presented here is well suited for parallel computer architectures because of the independence of the eigenpaths to be followed.

By a special choice of the initial matrix and following only several eigenpaths, we determined selected eigenvalues, for example those with the largest real part. Such an approach, which is of interest in linear stability analysis for hydrodynamic problems, has been applied to a concrete problem of finding the Hopf bifurcation point for surface-tension-driven flows in a liquid bridge.

It would be interesting to compare the present algorithm with the widely used Arnoldi algorithm described in [13].

Acknowledgments

Most of the work of the author (G. C) was done during a visit to the Center for Research on Parallel Computation at Caltech with support from "Conseil Régional Provence-Alpes-Côte d'Azur (France)". All the 64-node computations have been performed on the "Gamma" machine of the Caltech Concurrent Supercomputing Facilities.

References

1. C. B. Moler and G. W. Steward: An algorithm for generalized matrix eigenvalue problems, *SIAM J. Numer. Anal.* **10**, 241-256 (1973)

2. W. Kerner: Large-scale complex eigenvalue problems, *J. Compt. Phys.* **85**, 1-85 (1989)

3. Y. Saad: Numerical solution of large nonsymmetric eigenvalue problems, *Computer Physics Comm.* **53**, 71-90 (1989)

4. T. Y. Li and N. H. Rhee: Homotopy algorithm for symmetric eigenvalue problems, *Numer. Math.* **55**, 265-280 (1989)

5. T. Y. Li, H. Zhang and X. H. Sun: Parallel homotopy algorithm for the symmetric tridiagonal eigenvalue problem, *SIAM J. Sci. Stat. Comput.* **12**, 469-487 (1991)

6. T. Y. Li, Z. G. Zeng and L. Cong: Solving eigenvalue problems of real nonsymmetric matrices with real homotopies, *SIAM J. Numer. Anal.*, to appear

7. S. H. Lui: Ph.D Thesis part 2, Parallel homotopy method for the real nonsymmetric eigenvalue problem, California Institute of Technology, Pasadena, 1991

8. H. B. Keller: Numerical solutions of bifurcation and nonlinear eigenvalue problems. In: P.H. Rabinowitz (ed.): *Applications of bifurcation theory*, New York, Academic Press 1977

9. P. Sonneveld: CGS, A fast Lanczos-type solver for nonsymmetric systems, *SIAM J. Sci. Stat. Comput.* **10**, 36-52 (1989)

10. H.P. Langtangen: Conjugate gradient methodes and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns, *Int. J. Numer. Meth. Fluids* **9**, 213-233 (1989)

11. J-J. Xu and S. H. Davis: Convective thermocapillary instabilities in liquid bridges, *Phys. Fluids* **27**, 1102-1107 (1984)

12. A. D. Jepson: Ph.D Thesis part 2, Numerical Hopf bifurcation, California Institute of Technology, Pasadena, 1981

13. R. Natarajan: An Arnoldi-based iterative scheme for nonsymmetric matrix pencils arising in finite element stability problems, *J. Compt. Phys.* **100**, 128-142 (1992)