97

Applications of Quasi-Newton Methods for the Numerical Coupling of Some Nonlinear Problems

C.-H. Lai

1 Introduction

In a typical domain decomposition method, substructure solutions are used to construct preconditioners that are to be used in a conjugate gradient algorithm applied to the entire discretised problem. The approach used in this paper is to solve, instead of the entire discretised problem, the reduced interface problem which arisen from domain decomposition methods. This research has been motivated by a current project on viscous/inviscid coupling [LCP96]. In order to couple two models, an iterative scheme first developed by Schwarz for elliptic problems is usually employed [Sch90]. However the rate of convergence is not satisfactory, even for linear problems, without the use of preconditioners.

The interfacial problem along the subproblems interface is usually obtained as certain defects. Early experiences in this context can be found in [Lai93][Lai94]. The approach is based on a two level scheme. At the finer level, each subproblem is described by a nonlinear continuous model and solved independent of other subproblems using a local Newton's method. These subproblem solutions contribute to the evaluation of the defects along the interface of subproblems. At the coarse level, the defect equation is being solved by means of a quasi-Newton method. Two reasons of using quasi-Newton methods. First, the difficulty in computing the Jacobian matrix and second, the analytic form of the Jacobian is not known. A comparison of various quasi-Newton methods for a linear convection-diffusion problem can be found in [Lai94].

This paper is organised as follows. First, a simple interface problem is introduced followed by a description of some quasi-Newton methods. The performance of the nonlinear coupling on distributed computing environment is discussed with numerical examples.

Ninth International Conference on Domain Decomposition Methods
Editor Petter E. Bjørstad, Magne S. Espedal and David E. Keyes

2 A Simple Interface Problem

We consider the interface problem of the following two-point boundary value problem,

$$\frac{d^2\phi}{dx^2} = g(x, \phi, \frac{d\phi}{dx}), \qquad \in \qquad \Omega = \{x|a < x < b\} \tag{2.1}$$

subject to Dirichlet boundary conditions $\phi(a) = \phi_a$ and $\phi(b) = \phi_b$. The domain Ω is split into $s+1$ nonoverlapped subdomains, Ω_k, $k = 1, 2, \cdots, s+1$, such that $\Omega = \{\cup_{k=1}^{s+1}\Omega_k\} \cup \{\cup_{k=1}^{s}\Gamma_k\}$ where $\Omega_k = \{x|x_{k-1} < x < x_k\}$, $\partial\Omega_k = \{x_{k-1}, x_k\}$, $\Gamma_k = x_k$. Each of the subdomains has the following associated two-point boundary value problem,

$$\frac{d^2u_k}{dx^2} = g(x, u_k, \frac{du_k}{dx}), \qquad \in \qquad \Omega_k \tag{2.2}$$

subject to boundary conditions $u_k(x_{k-1}) = \lambda_{k-1}$ and $u_k(x_k) = \lambda_k$, and $u_1(x_0) = \phi_a$ and $u_{s+1}(x_{s+1}) = \phi_b$ where $x_0 = a$, $x_{s+1} = b$. Let $u_k = u_k(x; \boldsymbol{\lambda})$ denote the solution of (2.2) in Ω_k, where $\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2 \ \cdots \ \lambda_s] \in \Omega_D \subset R^s$. In order to obtain unique values of $\phi'(x_k)$, $k = 1, 2, \cdots, s$, we define the defect $\mathbf{D} : \Omega_D \subset R^s \to R^s$ as

$$\mathbf{D}(\boldsymbol{\lambda}) = [D_k(\boldsymbol{\lambda})] \equiv [\frac{\partial}{\partial x}u_k(x_k; \boldsymbol{\lambda}) - \frac{\partial}{\partial x}u_{k+1}(x_k; \boldsymbol{\lambda})] \tag{2.3}$$

and require to solve the defect equation $\mathbf{D}(\boldsymbol{\lambda}) = 0$. The continuity of the function ϕ across the interfaces is implicit in (2.2). The defect equation guarantees the continuity of ϕ' across the interfaces. It can be easily seen that $\mathbf{D} \in C^1(\Omega_D)$. In the case of two subdomains, the defect equation has one unknown. In the case of multi-subdomain, the Jacobian matrix $J(\boldsymbol{\lambda}) = \mathbf{D}'(\boldsymbol{\lambda})$ is a nonsymmetric tridiagonal matrix [Lai94]. If $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*$ is a root of $\mathbf{D}(\boldsymbol{\lambda}) = 0$, then the function

$$\phi(x) = \begin{cases} \lambda_{k-1}^* & x = x_{k-1} \\ u_k(x; \boldsymbol{\lambda}^*) & x_{k-1} < x < x_k, \\ \lambda_k^* & x = x_k \end{cases} \qquad k = 1, 2, \cdots, s+1 \tag{2.4}$$

where $\lambda_0^* = \phi_a$ and $\lambda_{s+1}^* = \phi_b$, is a solution of (2.1).

Note that (2.3) is the equilibrium state of the variables $\frac{\partial}{\partial x}u_k(x_k; \boldsymbol{\lambda})$ and $\frac{\partial}{\partial x}u_{k+1}(x_k; \boldsymbol{\lambda})$. There are other equilibrium states, e.g. integrate the difference of the derivatives along the interface, or perhaps other physically viable states. Once the mathematical interface coupling is defined, the defect equation can be easily set up and contributions to the defect equation from different subdomains can be separately computed which ensures distributed computing tasks.

3 Some Quasi-Newton Methods

In this section we consider a number of quasi-Newton methods for the solution of the defect equation $\mathbf{D}(\boldsymbol{\lambda}) = 0$. Let $\mathbf{D} : \Omega_D \subset R^s \to R^s$ where Ω_D is an open and convex set, $\mathbf{D} \in C^1(\Omega_D)$, $\mathbf{D}(\boldsymbol{\lambda}^*) = 0$, $J(\boldsymbol{\lambda}^*)$ nonsingular, and, for all $\boldsymbol{\lambda} \in \Omega_D$

$$||J(\boldsymbol{\lambda}) - J(\boldsymbol{\lambda}^*)|| \leq L||\boldsymbol{\lambda} - \boldsymbol{\lambda}^*||^p \tag{3.5}$$

for some norm $||.||$, and some L, $p > 0$. The general quasi-Newton method for the solution of $\mathbf{D}(\boldsymbol{\lambda}) = 0$ is given by

$$\boldsymbol{\lambda}^{(n+1)} = \boldsymbol{\lambda}^{(n)} - \alpha_n^{-1}\mathbf{D}(\boldsymbol{\lambda}^{(n)}) \tag{3.6}$$

where α_n is a nonsingular matrix approximating the Jacobian matrix J($\boldsymbol{\lambda}$). The convergence result of such an algorithm can be found in [DM74]. We choose six algorithms for the present comparison: a modified Newton method, Broyden's method, Schubert's method, Schubert-Kim method, Schubert-Powell method and an adaptive-α method. We are interested to apply these methods to the linear system $\mathbf{D}(\boldsymbol{\lambda}) \equiv J\boldsymbol{\lambda} - \mathbf{b} = 0$ where J is an $s \times s$ matrix as well as to the nonlinear system $\mathbf{D}(\boldsymbol{\lambda}) = 0$ in general. For some interface problems, the sparsity structure of the Jacobian matrices is known, therefore one should employ the so-called Schubert's update rather than Broyden's update. The difference between Broyden's method and Schubert's method is that the former does not take care of the sparsity of the Jacobian matrix while the latter preserves the sparseness structure of the Jacobian matrix. The naming convention for the algorithms being tested in this paper is that any name beginning with Schubert has its sparseness structure of the Jacobian matrix being preserved.

In order to avoid the update of the approximate Jacobian every iteration, a modified Newton method is applied here in such a way that the Jacobian is only calculated once and is used in all subsequent iteration.

Algorithm 3.1 Modified Newton's Method [Lai94]. Given $\boldsymbol{\lambda}^{(0)}$ and α_0, compute $\boldsymbol{\lambda}^{(1)}$ using (3.6). Then evaluate $J(\boldsymbol{\lambda}^{(1)})$ by means of a finite difference approximation. Finally use (3.6) to compute $\boldsymbol{\lambda}^{(n+1)}$ by choosing $\alpha_n = J(\boldsymbol{\lambda}^{(1)})$, $n = 1, 2, \ldots$.

α_0 is chosen as a diagonal matrix or in such a way that its sparseness is the same as that of the Jacobian matrix. Also α_n is calculated once and is kept for all subsequent iteration because it is expensive to evaluate the Jacobian matrix in every iterative step.

One classical technique of choosing α_n is called Broyden's update. Let $Q_{u,v} = \{\hat{\alpha} \in \mathbf{L}(R^s), \mathbf{u}, \mathbf{v} \in R^s : \hat{\alpha}\mathbf{u} = \mathbf{v}\}$. Then Broyden's update is obtained as the solution to the minimisation problem [DS79] $\min\{||\hat{\alpha} - \alpha_n||_F : \hat{\alpha} \in Q_{\mathbf{s}_n,\mathbf{y}_n}\}$ where $\mathbf{s}_n = \boldsymbol{\lambda}^{(n+1)} - \boldsymbol{\lambda}^{(n)}$, $\mathbf{y}_n = \mathbf{D}(\boldsymbol{\lambda}^{(n+1)}) - \mathbf{D}(\boldsymbol{\lambda}^{(n)})$, $||.||_F$ is the Frobenius norm. The solution of the minimisation problem is given by

$$\alpha_{n+1} = \alpha_n + \frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{s}_n^T}{< \mathbf{s}_n, \mathbf{s}_n >} \tag{3.7}$$

Suppose $\hat{W}$ and W are nonsingular matrices in $\mathbf{L}(R^s)$, then a weighted update can also be obtained as the solution to the minimisation problem [DS79] $\min\{||\hat{W}(\hat{\alpha}-\alpha_n)W||_F : \hat{\alpha} \in Q_{\mathbf{s}_n,\mathbf{y}_n}\}$ and is given by

$$\alpha_{n+1} = \alpha_n + \frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{v}_n^T}{< \mathbf{v}_n, \mathbf{s}_n >} \tag{3.8}$$

where $\mathbf{v}_n = W^{-T}W^{-1}\mathbf{s}_n$.

Algorithm 3.2 Broyden's Method [DS79]. Given $\boldsymbol{\lambda}^{(0)}$ and α_0, compute $\boldsymbol{\lambda}^{(n+1)}$ using (3.6) and α_{n+1} using (3.7), for $n = 0, 1, \ldots$.

For simplicity α_0 is chosen as a diagonal matrix. However the subsequent generated matrices, α_n, are full matrices. The inverses of these matrices are expensive and there is no guarantee that these matrices are nonsingular.

Algorithm 3.3 Schubert's Method [Mar79]. Perform the same steps as that given in Algorithm 3.2 but the successive updates α_{n+1} are made in such a way that the sparseness structure is preserved.

For simplicity α_0 is chosen as a diagonal matrix. As in the previous case, the sequence of matrices generated by (3.6) is not necessarily nonsingular.

Kim and Tewarson [KT92] proposed a weighted mean of (3.7) and (3.8) with $\mathbf{v}_n = -\alpha_n^T \mathbf{D}(\boldsymbol{\lambda}^{(n)})$, i.e.

$$\alpha_{n+1} = \alpha_n + (1-\mu)\frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{s}_n^T}{<\mathbf{s}_n, \mathbf{s}_n>} + \mu\frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{v}_n^T}{<\mathbf{v}_n, \mathbf{s}_n>} \tag{3.9}$$

where μ is chosen to satisfy $\|\frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{s}_n^T}{<\mathbf{s}_n,\mathbf{s}_n>}\|_F = \|\mu\frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{v}_n^T}{<\mathbf{v}_n,\mathbf{s}_n>}\|_F$ from which μ is obtained as

$$\mu = \frac{< -\alpha_n^T \mathbf{D}(\boldsymbol{\lambda}^{(n+1)}), \mathbf{s}_n >^2}{<\mathbf{s}_n, \mathbf{s}_n>< -\alpha_n^T \mathbf{D}(\boldsymbol{\lambda}^{(n+1)}), -\alpha_n^T \mathbf{D}(\boldsymbol{\lambda}^{(n+1)}) >} \tag{3.10}$$

We implemented the weighted update of Kim and Tewarson by means of Schubert's approach, i.e. the sparsity of the Jacobian matrix is preserved.

Algorithm 3.4 Schubert - Kim Method. Given $\boldsymbol{\lambda}^{(0)}$ and α_0, compute $\boldsymbol{\lambda}^{(n+1)}$ using (3.6) and α_{n+1} using (3.9), for $n = 0, 1, \ldots$.

It is worth to note that when $\mu = 1$, the method actually fails to converge for some interface problems, in particular the nonlinear boundary value problem described later.

None of the methods described so far has equipped with a technique to advoid singular matrix α_{n+1}. Using the well known determinant property $\det(I + \mathbf{u}\mathbf{v}^T) = 1 + <\mathbf{u}, \mathbf{v}>$ for any $\mathbf{u}$ and $\mathbf{v}$ in R^s, one can deduce that if α_n is nonsingular then, α_{n+1} is nonsingular if and only if $<\mathbf{s}_n, \alpha_n^{-1}\mathbf{y}_n> \neq 0$. Powell defined a modification to Broyden's method given by [MT76]

$$\alpha_{n+1} = \alpha_n + \theta_n \frac{\mathbf{D}(\boldsymbol{\lambda}^{(n+1)})\mathbf{s}_n^T}{<\mathbf{s}_n, \mathbf{s}_n>} \tag{3.11}$$

where θ_n is chosen so that α_{n+1} is nonsingular. In other words, we require

$$|\det \alpha_{n+1}| \geq \eta |\det \alpha_n|, \qquad |\theta_n - 1| \leq \eta \tag{3.12}$$

for any given $\eta \in (0,1)$. Using the above determinant property, one can easily deduce the following relation

$$\theta_n = \begin{cases} 1, & |\gamma_n| \geq \eta \\ \frac{1 - \text{sign}(\gamma_n)\eta}{1-\gamma_n}, & |\gamma_n| < \eta \end{cases} \tag{3.13}$$

where $\gamma_n = <\mathbf{s}_n, \alpha_n^{-1}\mathbf{y}_n> / <\mathbf{s}_n, \mathbf{s}_n>$, and $\text{sign}(0) = 1$. We implement, in Algorithm 3.5, the above modification by means of Schubert's approach.

Algorithm 3.5 Schubert - Powell Method [Mar79]. Given $\boldsymbol{\lambda}^{(0)}$ and α_0, compute $\boldsymbol{\lambda}^{(n+1)}$ using (3.6) and α_{n+1} using (3.11) and (3.13) for $n = 0, 1, \ldots$.

It should be noted that while Powell's modification to Broyden's method leads to global and superlinear convergence in the case of linear systems, it does not hold for general nonlinear functions [MT76].

Finally, we describe an algorithm based on a sequence of adaptive parameters [Lai94]. Here the technique for a scalar defect equation is essentially an optimal one-point iteration method where α_n is obtained by setting $\mathbf{G}' = 0$, where $\mathbf{G} = \boldsymbol{\lambda}^{(n)} - \alpha_n^{-1}\mathbf{D}(\boldsymbol{\lambda}^{(n)})$. This adaptive parameter α_n is equivalent to the scalar ϵ-algorithm [Lai94]. An adaptive α for the extension [Lai94] to s-dimensional problems is

$$\alpha_{n+1} = \alpha_n \frac{\|\mathbf{D}(\boldsymbol{\lambda}^{(n+1)}) - \mathbf{D}(\boldsymbol{\lambda}^{(n)})\|}{\|\mathbf{D}(\boldsymbol{\lambda}^{(n)})\|} \tag{3.14}$$

Algorithm 3.6 Adaptive α [Lai94]. Given $\boldsymbol{\lambda}^{(0)}$ and α_0, compute $\boldsymbol{\lambda}^{(n+1)}$ using (3.6) and α_{n+1} using (3.14), for $n = 0, 1, \ldots$.

Since an arbitrary initial approximation is chosen, there is no guarantee that $\boldsymbol{\lambda}^{(0)}$ is sufficiently close to $\boldsymbol{\lambda}^*$, in particular for nonlinear problems, which is an essential requirement for global convergence. One way to generate better initial approximation for nonlinear problems is to use Algorithm 3.6 which provides a small step during the initial few updates of the Jacobian. In the numerical tests shown later, Algorithm 3.6 is employed 3 or 4 times before the other algorithms are employed. The number of iterative updates, n_{it}, as presented in the Tables as shown later includes the above number of initial iterates. The reason for including these initial iterates in the iteration count becomes clear when the feasibility for parallel implementation is discussed. A stopping criterion for the above algorithms is $\|\boldsymbol{\lambda}^{(n)} - \boldsymbol{\lambda}^*\| < \epsilon$ where $\boldsymbol{\lambda}^*$ is given and ϵ is a small tolerance.

4 Performance Analysis

Now the approach involved in the present study is to divide the problem into two levels. At the fine level, the problem is divided into a number of subproblems to be computed in parallel. At the coarse level, the problem is small enough not to warranty for parallel implementation and the computational work is taken as the sequential overhead.

Let M denote the total number of nodes in the entire computational domain. One work unit is defined as the computational work required to solve the discretised problem with M nodes. For steady linear problems, solving the entire computational problem requires one work unit. For nonlinear problems, solving the entire computational problem requires n_s work unit, where n_s is the number of linearisations. For time dependent problems, solving the entire computational problem requires $n_t n_s$ work units where n_t is the number of time steps. Let the entire computational domain be divided into $s+1$ subdomains, M_k be the number of nodes in the k-th subdomain, $k = 1, 2, \cdots, s+1$ and n_{it} be the number of updates in order to obtain a converged solution $\boldsymbol{\lambda}^{(n_{it})}$ along the interfaces by using a quasi-Newton scheme. Suppose there is a set of $s+1$ concurrent processors and that the connectivity is the same as the layout of the subdomains. Since most of the computational work is devoted to the solutions of subproblems, it is possible to estimate the parallel computing time by taking the sum of the maximum work unit in any iteration involved in the subproblem solutions, i.e.

$$\tau = \sum_{n=1}^{n_{it}} \max_{1 \le k \le s+1} \{\frac{M_k}{M}\} \tag{4.15}$$

Hence a quasi-Newton algorithm described previously is considered as a feasible parallel algorithm provided $\tau < n_s$. Note that the present performance monitor does not include the overheads required to solve the linear systems and that such overhead becomes negligible for Algorithm 3.6.

5 Numerical Examples

We consider the following convection dominant flow problem,

$$\frac{d^2\phi}{dx^2} - \gamma\frac{d\phi}{dx} = 0, \qquad \phi(0) = 0, \phi(1) = 1 \tag{5.16}$$

where $\gamma \gg 1$. The domain is subdivided into $s+1$ subdomains with interfaces, Γ_k, $k = 1, 2, \ldots, s$, distributed evenly across the domain. We use exact solutions in each of the subdomains and are interested to compare the number of iterations, n_{it}, required to update the function values along the interfaces. The value ϵ of the stopping criterion is chosen to be 0.5×10^{-5}. For the present studies, the defect equation system for the convection diffusion problem is a linear system and Algorithm 3.5 is included in the test set. The number of iterations are presented in the middle column of Table 1 and values obtained by using (4.15) are presented in the third column of Table 1. Note that an "x" represents divergence of the test. More results can be found in ([Lai93]).

Table 1 Linear problem: $\gamma = 50$.

			$s+1$					$s+1$		
	4	8	16	32	64	4	8	16	32	64
Alg.			n_{it}					τ		
3.1	2	2	2	2	2	0.500	0.250	0.125	0.063	0.031
3.2	6	14	25	48	88	1.500	1.750	1.563	1.500	1.375
3.3	7	13	19	21	36	1.750	1.625	1.188	0.656	0.563
3.4	7	13	18	24	x	1.750	1.625	1.125	0.750	x
3.5	9	14	16	22	36	2.250	1.750	1.000	0.688	0.563
3.6	10	19	31	51	102	2.500	2.375	1.938	1.594	1.594

Obviously, Algorithm 3.1 is very attractive but the construction of $J(\boldsymbol{\lambda}^{(1)})$ requires $2s$ subproblem solvers. Algorithms 3.3, 3.4, and 3.5 are more efficient compare with 3.2 and 3.6.

Table 2 Nonlinear problem: $a = 1$ and $b = 0.5$, 161 mesh points.

	$s+1$				$s+1$			
$(n_s = 4)$	4	8	16	32	4	8	16	32
Alg.	n_{it}				τ			
3.1	12	11	12	11	3.000	1.375	0.750	0.343
3.2	11	17	34	57	2.750	2.125	2.125	1.781
3.3	10	18	19	17	2.500	2.250	1.188	0.531
3.4	11	17	x	x	2.750	2.125	x	x
3.6	12	20	72	114	3.000	2.500	4.500	3.563

Next, we consider the nonlinear elliptic boundary value problem,

$$\frac{d}{dx}(K(\phi)\frac{d\phi}{dx}) = f(x), \qquad \phi(0) = 0, \phi(1) = 1, \tag{5.17}$$

where $K(\phi) = a + b\phi$ is the thermal conductivity and $f(x) = 2a + 6bx^2$ for real numbers of a and b. The analytic solution of the problem is $\phi = x^2$. Since the problem is nonlinear, therefore the defect equation is also nonlinear. In order to solve the nonlinear subproblem, a linearisation based on Newton's method is applied at the subproblem level and a finite difference method is then applied to the linearised subproblem, i.e. $F'(u_k)(u_k^{new} - u_k) = -F(u_k)$ where $F(u_k) = f(x) - \frac{d}{dx}(K(u_k)\frac{du_k}{dx})$. The resulting set of linear equation is solved by a Gaussian elimination. It is not intended in this paper to study the efficiency of linear solvers at the subproblem level, and there are plenty of fast linear solvers available. The important issue here is the convergence rate of the quasi-Newton method, i.e. the number of updates, n_{it}, of the defect equation along the interface. Results for $a = 1.0$ and $b = 0.5$ are presented for the cases of 161 and 321 mesh points. Since the defect equation is a nonlinear system, Algorithm 3.5 does not guarantee that the approximate Jacobian matrices to be nonsingular and therefore it is not included in the test set. Tables 2 and 3 show the relationship between n_{it} and $s + 1$ for the nonlinear boundary value problem.

Table 3 Nonlinear problem: $a = 1$ and $b = 0.5$, 321 mesh points.

	$s+1$					$s+1$				
$(n_s = 4)$	4	8	16	32	64	4	8	16	32	64
Alg.	n_{it}					τ				
3.1	13	11	13	14	30	3.250	1.375	0.813	0.438	0.469
3.2	11	17	36	57	121	2.750	2.125	2.250	1.781	1.891
3.3	10	19	19	20	21	2.500	2.375	1.188	0.625	0.328
3.4	11	18	28	x	x	2.750	2.250	1.750	x	x
3.6	13	23	61	126	206	3.250	2.875	3.813	3.938	3.219

For test problems with 161 mesh points, one cannot have 64 subdomains. The weighted mean method described in Algorithm 3.4 seems to be less attractive for nonlinear problems because it diverges for a fairly small number of interfacial unknowns. In general, quasi-Newton methods performs better for nonlinear boundary value problems compare with linear boundary value problems.

6 Conclusions

A novel application of quasi-Newton methods for the solution of interface problems arisen from domain decomposition methods is examined. Performance analysis shows that Algorithms 3.2, 3.3, and 3.4 have been identified to be suitable for nonlinear problems. Algorithm 3.6 is included as a feasible parallel algorithm for nonlinear problems but is not suitable for linear problems. Algorithm 3.1 is extremely effective for linear problems only if enough parallel processors are available to construct the Jacobian. Algorithm 3.6 is also important in the sense that it provides stable early iterates in Algorithm 3.1 at a cheap overhead in order to produce an approximate solution in the neighbour of the exact solution. Extension to 2-D problems is currently being studied.

REFERENCES

[DM74] Dennis J. J. and Moré J. (1974) A characterization of superlinear convergence and its application to quasi-newton methods. *Math Comp* 28: 549–560.

[DS79] Dennis J. J. and Schnabel R. (1979) Least change secant updates for quasi-newton methods. *SIAM Review* 21: 443–459.

[KT92] Kim S. and Tewarson R. (1992) A quasi-newton method for solving nonlinear algebraic equations. *Computers Math. Applic.* 24: 93 – 97.

[Lai93] Lai C.-H. (1993) Comparing quasi-newton method for solving sparse interface problem. CWI Technical Report NM-R9303, CWI, Amsterdam, the Netherlands.

[Lai94] Lai C.-H. (1994) An iteration scheme for non-symmetric interface operator. In et. al. R. G. (ed) *Contemporary Mathematics*, number 157, pages 279–285. American Mathematical Society.

[LCP96] Lai C.-H., Cuffe A., and Pericleous K. (1996) A domain decomposition algorithm for viscous/inviscid coupling. *Advances in Engineering Software* 26: 151–159.

[Mar79] Marwil E. (1979) Convergence results for schubert's method for solving sparse nonlinear equations. *SIAM J. Num. Anal.* 16: 588–604.

[MT76] Moré J. and Trangenstein J. (1976) On the global convergence of broyden's method. *Math. Comp.* 30: 523–540.

[Sch90] Schwarz H. (1890) über einen grenzübergang durch alternierendes verfahren. *Gesammelte Mathematische Abhandlungen* 41: 133–143.