*Fourteenth International Conference on Domain Decomposition Methods*
Editors: Ismael Herrera , David E. Keyes, Olof B. Widlund, Robert Yates 

# 35. An Aitken-Schwarz method for efficient metacomputing of elliptic equations

N. Barberou[1], M. Garbey[1,2], M. Hess[3], M. Resch[2,3], T. Rossi[4], J. Toivanen[4], D. Tromeur-Dervout[1]

**1. Introduction.** Metacomputing as defined by Larry Smarr [13] has been implemented in many projects among which GLOBUS is the most widely developed one [2]. Experiments with large configurations have however shown that the latency of wide area networks is prohibitively high and that substantial bandwidth can hardly be achieved [10]. From that some people have concluded that metacomputing does not make sense.

However, there are two strong arguments for metacomputing. First, with the introduction of clusters of fat nodes, varying latencies, and bandwidths are a characteristic feature of any modern hardware. Algorithms developed in metacomputing environments are therefore well suited also for such systems. Second, some large problems require a level of computing power not available on a single system. Especially in cases of industrial or natural disasters reliable predictions based on extremely large models may only be achievable on clustered supercomputers in a metacomputing environment. Such simulations of emergency scenarios will again need clever algorithms that can tolerate the bad performance of wide area communication networks.

The development of such algorithms is difficult. For Poisson or Helmholtz operators the speed of propagation of information in the spatial domain is infinite. However, two factors help to design latency-aware algorithms; firstly information propagating at infinite speed can be damped in space relatively fast, secondly, more than 90 percent of the information carried in a practical computation is noise.

In this paper, we address the significant challenge to build a fast solver for the Helmholtz operator. It combines the Aitken-Schwarz domain decomposition method [4, 5] associated with the Partial Solution variant of Cyclic Reduction (PSCR) method [11, 12] on large scale parallel computers.

## 2. Numerical methods.

### 2.1. The PDC3D inner solver.

The parallel solver PDC3D developed by T. Rossi and J. Toivanen [11] [12] following the ideas of Y. Kuznetsov [9] and P. Vassilevski [15] is a parallel fast direct solution method for linear systems with separable block tridiagonal matrices. Such systems appear, for example, when discretizing the Poisson equation in a rectangular domain using the seven-point finite difference scheme or piecewise linear finite elements on a triangulated, possibly nonuniform rectangular mesh. The method under consideration has the arithmetical complexity $\mathcal{O}(N \log^2 N)$ and is closely related to the cyclic reduction method. But instead of using the matrix polynomial factorization the so-called partial solution technique is employed. Based on the analysis of [12], the radix-4 variant is chosen for the parallel implementation using the MPI standard. However, the method works for blocks of arbitrary dimension. [11] [12] show that the sequential efficiency and numerical stability of the PSCR method compares favorably to the well-known BLKTRI implementation of the generalized cyclic reduction method. The current PDC3D code is using a two-dimensional domain decomposition. It requires a high performance communication network mainly because of global reduction operations to gather the partial solution. Its very good scalability has been shown on a CrayT3E (table 4.2).

[1]MCS/CDCSP/ISTIL - University Lyon 1, 69622 Villeurbanne, France
[2]Department of Computer Science, University of Houston, USA
[3]HLRS, University of Stuttgart, Germany
[4]Department of Mathematical Information Technology, University of Jyväskylä, Finland

**2.2. The Aitken-Schwarz outer solver.** Following a different approach than the PDC3D solver, M. Garbey and D. Tromeur-Dervout [4] [5] [6] have developed an Aitken-Schwarz algorithm (respectively Steffensen-Schwarz) for the Helmholtz operator, (respectively for general linear and non linear elliptic problems) that is highly tolerant to low bandwidth and high latency. In the specific case of separable elliptic operators, the Aitken-Schwarz algorithm might be less efficient in terms of arithmetic complexity than PDC3D as the number of processors increases [5], but it is very competitive when using $O(10)$ sub-domains.

Let us recall briefly the salient feature of the method in 1D to solve

$$L[u] = f, \quad on \ \ \Omega = [0,1] \tag{2.1}$$

$$B[u] = 0 \tag{2.2}$$

where operator $B$ denotes the linear operator on boundary conditions, and $L$ is some linear operator. Let $\Omega_i = (x_i^l, x_i^r),\ i = 1..q$ be a partition of $\Omega$ with $x_2^l < x_1^r < x_3^l < x_2^r, ..., x_q^l < x_{q-1}^r$. We consider the additive Schwarz algorithm:

$for\ i = 1..q, do \qquad L[u_i^{n+1}] = f\ in\ \Omega_i,\ u_i^{n+1}(x_i^l) = u_{i-1}^n(x_i^l),\ u_i^{n+1}(x_i^r) = u_{i+1}^n(x_i^r),$

*enddo*

Let $u_i^{l,n+1} = u_i^{n+1}(x_i^l),\ u_i^{r,n+1} = u_i^{n+1}(x_i^r)$ and $\tilde{u}^n$ (respectively $\tilde{u}$) be the $n$ iterated (respectively exact) solution restricted at the interface, i.e

$$\tilde{u}^n = (u_2^{l,n}, u_1^{r,n}, u_3^{l,n}, u_2^{r,n}, ..., u_q^{l,n}, u_{q-1}^{r,n})$$

The operator $\tilde{u}^n - \tilde{u} \to \tilde{u}^{n+1} - \tilde{u}$ is linear. Let us denote its matrix by $P$. $P$ has the following pentadiagonal structure:

$$\left|\begin{array}{ccccccccc}
0 & \delta_1^r & 0 & 0 & .... & & & & \\
\delta_2^{l,l} & 0 & 0 & \delta_2^{l,r} & ... & & & & \\
\delta_2^{r,l} & 0 & 0 & \delta_2^{r,r} & ... & & & & \\
 & & & & & & & & \\
 & & & ... & \delta_{q-1}^{l,l} & 0 & 0 & \delta_{q-1}^{l,r} & \\
 & & & ... & \delta_{q-1}^{r,l} & 0 & 0 & \delta_{q-1}^{r,r} & \\
 & & & ... & 0 & 0 & \delta_q^r & 0 &
\end{array}\right|$$

The sub-blocks $P_i = \left|\begin{array}{cc} \delta_i^{l,l} & \delta_i^{l,r} \\ \delta_i^{r,l} & \delta_i^{r,r} \end{array}\right|$ $i = 2..q-1$ can be computed with 3 Schwarz iterates as follows.

We have $(u_{i-1}^{r,n+1} - \tilde{u}_{i-1}^r, u_{i+1}^{l,n+1} - \tilde{u}_{i+1}^l)^t = P_i(u_i^{l,n} - \tilde{u}_i^l, u_i^{r,n} - \tilde{u}_i^r)^t$. Therefore

$$\begin{pmatrix} u_{i-1}^{r,n+3} - u_{i-1}^{r,n+2} & u_{i-1}^{r,n+2} - u_{i-1}^{r,n+1} \\ u_{i+1}^{l,n+3} - u_{i+1}^{l,n+2} & u_{i+1}^{l,n+2} - u_{i+1}^{l,n+1} \end{pmatrix} = P_i \begin{pmatrix} u_i^{l,n+2} - u_i^{l,n+1} & u_i^{l,n} - u_i^{l,n} \\ u_i^{r,n+2} - u_i^{r,n+1} & u_i^{r,n} - u_i^{r,n} \end{pmatrix}$$

In practice the last matrix on the right hand side of the previous equation is non singular and $P_i$ can be computed, but it cannot be guaranteed. However, one can always compute beforehand the coefficients of $P_i$ -see [4]. For the Helmholtz operator $L[u] = u'' - \lambda u$, or generally speaking elliptic problems with constant coefficients, the matrix $P$ is known analytically.

From the equality $\tilde{u}^{n+1} - \tilde{u} = P(\tilde{u}^n - \tilde{u})$, one obtains the generalized Aitken acceleration as follows:

$$\tilde{u}^\infty = (Id - P)^{-1}(\tilde{u}^{n+1} - P\tilde{u}^n). \tag{2.3}$$

If the additive Schwarz method converges, then $||P|| < 1$ and $Id - P$ is non singular.

The aim of our paper is therefore to combine the two methods in order to have a highly efficient solver for the Helmholtz operator for metacomputing environments. This solver can be used to solve the elliptic equations satisfied by the velocity components of a incompressible Navier Stokes (NS) code written in velocity-vorticity formulation. The elliptic part of such an NS solver is usually the most time consuming part as these equations must be solved very accurately to satisfy the velocity divergence free constraint. Similarly this solver can be used for the pressure solve in NS written in velocity pressure formulation using the projection method.

Our parallel implementation is then as follows: first one decomposes the domain of computation into a one-dimensional domain decomposition (DD) of $O(10)$ macro sub-domains. This first level of DD uses the Aitken Schwarz algorithm. The macro sub-domains are distributed among clusters or distinct parallel computers. Secondly, each macro sub-domain is decomposed into a two-dimensional DD: this level of DD uses the PDC3D solver. Globally we have a three-dimensional DD and a two-level algorithm that matches the hierarchy of the network and access to memory.

**3. Hardware and software components for metacomputing.** Metacomputing in heterogeneous environments introduces problems that are partly similar to those well known from clusters and partly very new and specific. Among the most critical ones is the concurrent scheduling of resources. Another one is the mapping of processes to processors. For these problems we refer to projects like GLOBUS [2], Legion [7] or TME [14]. In this section we focus on communication.

The communication can be done using several MPI-implementations [1] [3][8]. All these implementations provide a simple way to start and run MPI-applications across a metacomputer. They differ, however, in completeness of implementing the MPI-standard and in the degree of optimization. For the experiments described in this paper we have chosen PACX-MPI [3] from the High Performance Computing Center Stuttgart (HLRS) which allows metacomputing for MPI-codes without any changes [10]. Based on the experience of several projects the library relies on four main concepts:

- For the programmer the metacomputer looks like any other parallel system.
- Usage of communication daemons to clearly split external from internal communication and ease support of different communication protocols (e.g. TCP/IP, ATM).
- Use of native MPI for internal communication and standard protocols for external communication. MPI-implementations based on native protocols typically are superior in performance to any other approach.
- Optimized global communication by minimizing traffic between systems.

**4. Results.** We are going to present some numerical experiences in metacomputing environments. For simplicity, we restrict ourselves to a network of two or three parallel computers. For large scale metacomputing experiments, we are using the hardware described in Table 4.1. Once and for all we denote **CrayS** the Cray of HLRS in Stuttgart University, **CrayN** of the von Neumann Institute in Jülich (NIC), **CrayP** of the Pittsburgh center of high performance computing in USA and **CrayH** the Cray T3E of the National Scientific Computing Center of Finland at CSC. The goal is to demonstrate on classical problems that make intense use of Poisson solves, that efficient numerical results and high performance are attainable in a metacomputing environment with standard network connections.

**4.1. Fast Poisson solver experiment.** We make three hypotheses:

- First, we restrict ourselves to the Poisson problem, i.e the Helmholtz operator with $\lambda = 0$. As a matter of fact, it is the worst situation for metacomputing because any perturbation at an artificial interface decreases linearly in space, instead of exponentially as for the Helmholtz operator.
- Second, we do a priori load balancing on the heterogeneous network of Cray supercom-

| Machine | #proc | MHz | internal latency | internal bandwidth | Localization |
|---|---|---|---|---|---|
| CrayS | 512 | 450 | 12 $\mu s$ | 320 MB/s | HLRS, Stuttgart |
| CrayP | 512 | 450 | 12 $\mu s$ | 320 MB/s | PSC, Pittsburgh |
| CrayH | 512 | 375 | 12 $\mu s$ | 320 MB/s | CSC, Helsinki |
| CrayN | 512 | 375 | 12 $\mu s$ | 320 MB/s | NIC , Jülich |

Table 4.1: System configuration at Stuttgart, Pittsburgh, Helsinki, Jülich.

puters. We verified that PDC3D solver is roughly 30% slower on CrayH than on CrayS for our test cases. The number of grid points in the Aitken domain decomposition is balanced in such way that PDC3D in each parallel computer uses approximatively the same CPU time.

• Third, and this is a key point, we are running our metacomputing experiment on two or three supercomputers with the existing ordinary area network. During all our experiments, the bandwidth fluctuated in the range of $(1.6Mb/s - 5.Mb/s)$ and the latency was about $30ms$.

Let us show that a fast elliptic solver that is quasi optimal on a single parallel computer gives poor performance in a metacomputing environment.

The PDC3D solver is an almost optimal solver with good parallel scalability in parallel computers having a good balance of network and processor. By analyzing the PDC3D algorithm, we can deduce that the number of communications per processor is of the order of $\log(p_x) + (N_x/p_x)\log(N_x)\log(p_y)$ and the total length of all messages for one processor is of the order of $(N_y/p_y)N_z\log(p_x) + (N_x/p_x)\log(N_x)N_z\log(p_y)$ floating point numbers.
Each processor stores $(N_x/n_x)(N_y/n_y)N_z$ floating point numbers which is considerably more than the amount of communication. Thus, the computational work per processor greatly exceeds the amount of data to be transferred. This leads to a rather efficient code as can be seen in Table 4.2, where the results of experiments made on CrayS are presented. Also, it can be seen from the communication estimates and the numerical results that it is favorable to choose $p_y$ to be larger than $p_x$.

The PDC3D solver obviously cannot be used efficiently in a metacomputing environment. Based on the performances -see Table 4.2- of the PDC3D on CrayS, we select the most efficient data distribution and run the same problem on the metacomputing architecture, i.e on CrayS and CrayH that share equally the total number of processors used. Table 4.2 gives a representative set of the performance of PDC3D on the metacomputing architecture (CrayS-CrayH). We conclude that no matter what the number of processors, most of the elapsed time is spent in communication between the two computer sites. This conclusion holds for a problem of smaller size, that is $256^3$: the elapsed time grows continuously from 0.76 s, up to 18.73 s with 512 processors. Obviously the PDC3D performance degrades drastically when using a slow network. In the following we show how Aitken-Schwarz can overcome this problem.

**4.2. Aitken-Schwarz experiment.** We proceed with a performance evaluation of our two level domain decomposition method combining Aitken-Schwarz and PDC3D (**AS**). We define the barrier between low and medium size frequencies in each space variable to be 1/4 of the number of waves; We do not accelerate the highest half of the frequencies. We checked that the impact on the numerical error against an exact polynomial solution is in the interval $[10^{-7}, 10^{-6}]$ for our test cases with minimum overlap between macro sub-domains. Let us give first the performance of our method on a single Cray.

Figure 4.1 gives the elapse time for the following growing size of Poisson problems $158 \times 192 \times 384$, $316 \times 192 \times 384, 633 \times 192 \times 384$. When increasing the number of do-

| **No metacomputing**: localization of processors : 100% $\in$ **CrayS** | | |
|---|---|---|
| 128 procs ($p_x \times p_y$) | 256 procs ($p_x \times p_y$) | 512 procs ($p_x \times p_y$) |
| 25.9 s (4 x 32) | 17.6s (4 x 64) | |
| 22.0 s (16 x 8) | 11.5s (16 x 16) | 7.2 (16 x 32) |
| 21.8 s (64 x 2) | 11.2s (64 x 4) | **5.77 (64 x 8)** |
| **Metacomputing**: localization of processors :50% $\in$ **CrayS** and 50% $\in$ **CrayH** | | |
| 72.0s (64 x 2) | 77.2s (64 x 4) | **75.1 (64 x 8)** |

Table 4.2: **Elapsed time in (s) for the PDC3D solver on CrayS and on metacomputing architecture (CrayS, CrayH) to solve a problem of global size** $511 \times 511 \times 512$

mains in the same proportion as the number of processors the elapsed time remains constant. Our solver has therefore good scalability properties on the Cray T3E. Further, our method requires no more than 6 seconds to solve the problem with 46 $10^6$ unknowns on a Cray T3E with 256 processors running at 450 $MHz$. Figure 5.2 shows also that the speedup of our solver is fairly good. Now let us proceed with the metacomputing experiment. We make the two following hypothesis:

• First we fix the size of our problem in such a way that it cannot be solved on one single computer at our disposal. As a matter of fact, we use almost all memory available on our network of supercomputers.

• Second, we focus our study in this context on the extensibility properties of our direct linear solver. To benefit of the Gustafson law for scalability, we believe that a direct measurement of the speedup is not appropriate. We have also not estimated the speedup from a model analysis, because our two level domain decomposition method is too complex to give any realistic estimate in a metacomputing environment. Table 4.3 summarizes our results. Let us notice that each case has been run several times and our measurements give elapsed time with a variation of few seconds, depending on the quality of the network during the experiment. We provide here an average value that corresponds to two or three consecutive runs excluding the cases where the network died during the runs.

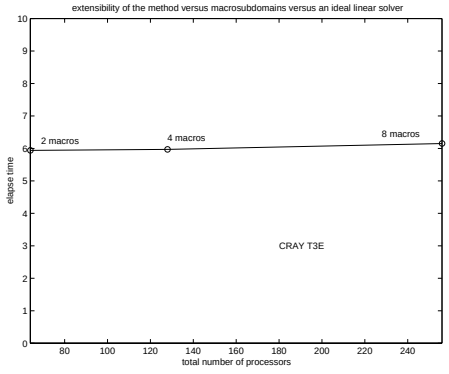


Figure 4.1: Extensibility of the Aitken Schwarz algorithm for the 3D Poisson problem

Our two main observations are as follows:

• We have in our experiments an irreducible overhead that varies from $17s$ to $24s$ and

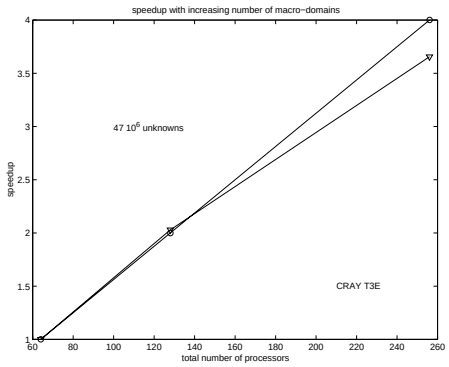


Figure 4.2: Speedup of the Aitken Schwarz algorithm for the 3D Poisson problem

that depends mostly on the speed of the network that interlinks our supercomputers. We recall that the bandwidth of the network was in the range of 2 to 5 Mb/s. This overhead is quasi independent of the size of the problem considered here.

• Beside the overhead due to the network between distant sites, we observe an excellent scalability of our Poisson solver. This result is a combination of two factors: first the arithmetic complexity of our solver grows almost linearly with the number of macrosubdomains. Second the ratio of computation time per macrodomains to communication time is large even with a slow network and fast supercomputers.

Finally, we would like to underline that the conditions of our experiments were by definition difficult. It is not realistic to stop simultaneously the production of several national computing centers for long. We had therefore few windows for experiments with few hours each time. We are extremely grateful to all centers participating in these sets of experiments for their cooperations. Further, to our knowledge, there are no other known results of efficient large scale metacomputing simulations of PDE problems that assume tidily coupled computation as it is the case in a Poisson solver. This work is currently extended to 3D Navier Stokes equation using our Poisson solver as a preconditioner.

**5. Conclusions.** In this paper, we demonstrate the feasibility of numerical efficient metacomputing between distantly located parallel computing resources for tidily coupled problems as Helmholtz solvers. In order to achieve this result, we use the best components we can get at each stage that is to say:

• PDC3D : one of the best efficient solvers for separable operators that scales well on homogeneous computers with fast communication network.

• PACX-MPI to achieve excellent performance of MPI communication on both the internal and external communication network.

• Aitken-Schwarz that is numerically efficient, tolerant to slow communication networks and high latencies and scales well up to $O(10)$ domains.

High latencies and slow communication networks with fluctuating bandwidth shared by thousands of users are typical difficulties encountered in grid computing. The Aitken-Schwarz DDM seems to be an example of a numerical tools that adresses to such difficulties.

**Acknowledgment**: The authors would like to thank the John von Neumann Institute and the Pittsburgh Supercomputing Center for providing access to their systems for experiments. All authors would like to thank their home organizations Cines, CSC and HLRS for support in this work. The work of T. Rossi and J. Toivanen was supported by the Academy of Finland, grants #43066, #53588 and #66407. The work of N. Barberou was supported

| # of pts in $x$ per MD | | | # of MD per machine | | | # of pts in $y, z$ per proc. | | # of proc. per MD | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $N_{xJ}$ | $N_{xS}$ | $N_{xP}$ | $MJ$ | $MS$ | $MP$ | $n_y$ | $n_z$ | $p_y$ | $p_z$ | Time (s) |
| 225 | 321 | 0 | 1 | 1 | 0 | 49 | 49 | 16 | 16 | 60 |
| 225 | 321 | 0 | 1 | 2 | 0 | 49 | 49 | 16 | 16 | 58 |
| | | | | | | | | | | |
| 0 | 321 | 0 | 2 | 0 | 0 | 49 | 49 | 8 | 16 | 30 .4 |
| 225 | 321 | 0 | 1 | 1 | 0 | 49 | 49 | 8 | 16 | 47 |
| 225 | 321 | 0 | 1 | 2 | 0 | 49 | 49 | 8 | 16 | 47 |
| | | | | | | | | | | |
| 0 | 321 | 0 | 0 | 2 | 0 | 49 | 49 | 8 | 8 | 27.3 |
| 0 | 321 | 0 | 0 | 3 | 0 | 49 | 49 | 8 | 8 | 27.3 |
| 0 | 321 | 0 | 0 | 4 | 0 | 49 | 49 | 8 | 8 | 27.2 |
| 225 | 321 | 0 | 4 | 4 | 0 | 49 | 49 | 8 | 8 | 51 |
| | | | | | | | | | | |
| 0 | 321 | 0 | 0 | 2 | 0 | 43 | 43 | 16 | 16 | 25,4 |
| 225 | 321 | 0 | 2 | 2 | 0 | 43 | 43 | 16 | 16 | 50 |
| 225 | 321 | 321 | 2 | 2 | 1 | 43 | 43 | 16 | 16 | 59 |
| 0 | 401 | 0 | 0 | 2 | 0 | 43 | 43 | 16 | 16 | 30.5 |
| 281 | 401 | 401 | 2 | 2 | 1 | 43 | 43 | 16 | 16 | 62 |

Table 4.3: Extensibility of the Aitken-Schwarz on the Poisson problem in a metacomputing framework

by the "abondement de l'ANVAR". The work of D. Tromeur-Dervout was supported by the ACI-Grid project of the French Ministry.

REFERENCES

[1] I. Foster and N. Karonis. A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems.

[2] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Intl J. Supercomputer Applications*, 11(2):115–128, 1997.

[3] E. Gabriel, M. Resch, T. Beisel, and R. Keller. Distributed computing in a heterogenous computing environment. In V. Alexandrov and J. Dongarra, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface, Lecture Notes in Computer Science*, pages 180–188. Springer, 1998.

[4] M. Garbey and D. Tromeur-Dervout. Two Level Domain Decomposition for Multi-cluster. In H. K. T. Chan, T. Kako and O. Pironneau, editors, *Proc. Int. Conf. on Domain Decomposition Methods DD12*, pages 325–340. DDM org, 2001.

[5] M. Garbey and D. Tromeur-Dervout. Aitken-Schwarz Method on Cartesian grids. In N. Debit, M. Garbey, R. Hoppe, J. Périaux, and D. Keyes, editors, *Proc. Int. Conf. on Domain Decomposition Methods DD13*, pages 53–65. CIMNE, 2002.

[6] M. Garbey and D. Tromeur-Dervout. On some Aitken like acceleration of the Schwarz method. *Int. J. of Numerical Methods in Fluids*, 2002. to appear.

[7] A. Grimshaw, A. Ferrari, F. Knabe, and M. Humphrey. Legion: An Operating System for Wide-Area Computing. Technical Report CS-99-12, University of Virginia, 1999.

[8] H. Koide, T. Imamura, and H. Takemiya. MPI based communication library for a heterogeneous parallel computer cluster, stampi. Technical report, Japan Atomic Energy Research Institute, 1997. http://ssp.koma.jaeri.go.jp/en/stampi.html.

[9] Y. Kuznetsov. Numerical methods in subspaces. *Vychislitel'nye Processy i Sistemy II*, 1985. G. I. Marchuk editor, Nauka, Moscow.

[10] S. Pickles, J. Brooke, F. Costen, E. Gabriel, M. Muller, M. Resch, and S. Ord. Metacomputing across intercontinental networks. *Future Generation Computer Systems*, 17(8):911–918, 2001.

[11] T. Rossi and J. Toivanen. A Nonstandard Cyclic Reduction Method, its Variants and Stability. *SIAM J. Matrix Anal. Appl.*, 3:628–645, 1999.

[12] T. Rossi and J. Toivanen. A Parallel Fast Direct Solver for Block Tridiagonal Systems with Separable Matrices of Arbitrary Dimension. *SIAM J. Sci. Comput.*, 5:1778–1796, 1999.

[13] L. Smarr and C. Catlett. Metacomputing. *Communications of the ACM*, 35(6):45–52, 1992.

[14] H. Takemiya, T.Imamura, and H. Koid. TME a visual programming and execution environment for a meta-application. Technical report, Jaeri Internal Report, 2000.

[15] P. S. Vassilevski. Fast Algorithm for Solving a Linear Algebraic Problem with Separable Variables. *C.R. Acad. Bulgare Sci.*, 37:305–308, 1984.