A fast domain decomposition high order Poisson solver

by

Bertil Gustafsson and Lina Hemmingsson

Report No. 206/1998

UPPSALA UNIVERSITET
Institutionen för
teknisk databehandling

UPPSALA UNIVERSITY
Department of
Scientific Computing

A fast domain decomposition high order Poisson solver *

Bertil Gustafsson and Lina Hemmingsson†

March, 1998

Abstract

In this paper we present a fast high order Poisson solver for implementation on parallel computers. The method uses deferred correction, such that high order accuracy is obtained by solving a sequence of systems with a narrow stencil on the left hand side. These systems are solved by a domain decomposition method. The method is direct in the sense that for any given order of accuracy, the number of arithmetic operations is fixed. Numerical experiments show that these high order solvers easily outperform standard second order ones. The very fast algorithm in combination with the coarser grid allowed for by the high order method, also makes it quite possible to compete with adaptive methods and irregular grids for problems with solutions containing widely different scales.

*The second author was supported by the Swedish Research Council for Engineering Sciences, under contract No. 282-95-1000

†Department of Scientific Computing, Box 120, S-751 04 Uppsala, Sweden

Contents

1 Introduction

We consider numerical methods for Poisson's equation in D space-dimensions

$$\begin{aligned} \Delta u = F(\mathbf{x}) \quad &, \quad \mathbf{x} \in \Omega = [0,1]^D, \\ u = g(\mathbf{x}) \quad &, \quad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{1.1}$$

Fast Poisson solvers are characterized by the operation count $\sim N^D \log N$, where N is the number of grid-points in each direction. In the sixties and early seventies many such solvers were developed, most of them based on second order finite differences. Surveys of such solvers are found in [2], [5], and [17], [9]. With the introduction of parallel computers, it became necessary to find efficient parallel implementations. One way is to use domain decomposition, where the so called Schur complement system for the unknowns at the internal boundaries is solved by itself, and the subdomain problems are solved in parallel. Such an algorithm was developed by Chan and Resasco [4], again for the standard second order scheme.

For a large class of problems, in particular when very accurate solutions are required, higher order methods can be expected to be more effective. This is demonstrated for a model problem in Section 2 of this paper. However, the parallelization is complicated by the fact that the stencils are wider. In particular, if the internal boundaries separating the subdomains are defined in the standard way, the subdomains become coupled to each other through the wider stencils, and this destroys the parallel solution procedure. If, on the other hand, we define the internal boundaries to contain several grid-lines, the solution of the corresponding Schur complement system becomes more difficult. Therefore, if we can achive high order accuracy, without widening the stencil on the left hand side, we can keep the fast solution procedure. Such a method is obtained by using the principle of deferred correction, where high order accuracy is obtained by solving a sequence of systems, each one with the standard second order accurate formula on the left hand side. On the right hand side it is necessary to have wider stencils, since approximations of the truncation error must be included. However, transferring known values (from the previous computational step) on one gridline from one processor to another, is easy compared to the complication that arises when there is a coupling between the unknowns in different processors.

The method of deferred correction has been developed by Pereyra in a long series of papers, see for example [11], [12], [13], [14], and by Pereyra et.al. [15]. There is one inherent difficulty with deferred correction, namely the problem of constructing the difference operators close to the boundary in such a way that the truncation error approximation used in each step is obtained to the right order of accuracy. In this paper we use polynomial extrapolation to obtain extra points outside the boundary, such that the usual centered operators can be used at all internal points and provide the necessary accuracy.

High order approximations can also be constructed by using the differential equation to substitute higher order derivatives, occurring in the truncation error, by lower order ones. For example, a fourth order approximation in two space dimensions can be obtained by using a nine-point compact formula. However, for higher order than four, the stencils necessarily becomes wider, and the domain decomposition concept cannot be applied in any simple manner. The method presented in this paper allows for any even order of accuracy, still with one and the same narrow stencil on the left-hand side.

In Section 2 we analyze a simple 1D model problem and derive estimates of work and memory requirements for different orders of accuracy. In Section 3 we present the algorithm, while we present error estimates in Section 4. The advantages with our technique in a domain decomposition setting of the problem is demonstrated in Section 5. Finally, in Section 6 we present numerical results, and make some concluding remarks in Section 7.

2 A model problem

Consider the model problem

$$\begin{aligned} &u_{xx} = \omega^2 \sin(\omega x) \quad , \ 0 \le x \le \pi \, , \\ &u(0) = u(\pi) = 0 \, , \end{aligned}$$

which has the solution

$$u(x) = -\sin(\omega x) \, .$$

We shall analyze this problem and show that higher order methods are more efficient than lower order ones.

Let Q be the standard second order accurate difference operator

$$Qu_j = \frac{1}{h^2}(u_{j+1} - 2u_j + u_{j-1})$$

be defined on the grid $\{x_j = jh\}$, $Nh = \pi$. Then difference approximations of order 2,4,6 are defined by

$$Q^{\langle p \rangle} u_j^{\langle p \rangle} = \omega^2 \sin(\omega x_j) \ \ , \ j = 1, 2, \dots , N-1 \, ,$$

where

$$\begin{aligned} Q^{\langle 2 \rangle} &= Q \, , \\ Q^{\langle 4 \rangle} &= Q - \frac{h^2}{12} Q^2 \, , \\ Q^{\langle 6 \rangle} &= Q - \frac{h^2}{12} Q^2 - \frac{h^4}{90} Q^4 \, . \end{aligned}$$

The solution is uniquely defined with the boundary conditions

$$\left. \begin{aligned} u_{-\nu}^{\langle p \rangle} &= -u_{\nu}^{\langle p \rangle} \\ u_{N+\nu}^{\langle p \rangle} &= -u_{N-\nu}^{\langle p \rangle} \end{aligned} \right\} \ \nu = 0, 1, 2 \, .$$

With

$$\hat{Q} = \frac{2}{h^2}(\cos\xi - 1) \quad , \; \xi = \omega h \, ,$$

the solution is obtained as

$$u_j^{\langle p \rangle} = \frac{\omega^2}{\hat{Q}_p} \sin(\omega x_j) \quad , \; p = 2, 4, 6 \, ,$$

where

$$\begin{aligned} \hat{Q}^{\langle 2 \rangle} &= \hat{Q} \, , \\ \hat{Q}^{\langle 4 \rangle} &= \hat{Q} - \frac{h^2}{12}\hat{Q}^2 \, , \\ \hat{Q}^{\langle 6 \rangle} &= \hat{Q} - \frac{h^2}{12}\hat{Q}^2 - \frac{h^4}{90}\hat{Q}^4 \, . \end{aligned}$$

A Taylor expansion gives

$$\hat{Q}^{\langle p \rangle} = -\omega^2 + \omega^2 \delta^{\langle p \rangle}(\xi) \, ,$$

where

$$\begin{aligned} \delta^{\langle 2 \rangle}(\xi) &= \frac{\xi^2}{12} + \mathcal{O}(\xi^4) \, , \\ \delta^{\langle 4 \rangle}(\xi) &= \frac{\xi^4}{90} + \mathcal{O}(\xi^6) \, , \\ \delta^{\langle 6 \rangle}(\xi) &= \frac{\xi^6}{432} + \mathcal{O}(\xi^8) \, . \end{aligned}$$

The error in the solution is

$$\epsilon = \max_{x_j} |u(x_j) - u_j^{\langle p \rangle)}| = |-1 - \frac{\omega^2}{\hat{Q}^{\langle p \rangle}}| = |1 - \frac{1}{1 + \delta^{\langle p \rangle}(\xi)}| \approx \delta^{\langle p \rangle}(\xi) \, .$$

By prescribing the error tolerance ϵ, the approximate number of gridpoints $N^{\langle p \rangle} = \pi / h$ is obtained as

$$\begin{aligned} N^{\langle 2 \rangle} &= \frac{\pi\omega}{(12\epsilon)^{1/2}} \, , \\ N^{\langle 4 \rangle} &= \frac{\pi\omega}{(90\epsilon)^{1/4}} \, , \\ N^{\langle 6 \rangle} &= \frac{\pi\omega}{(432\epsilon)^{1/6}} \, . \end{aligned}$$

In this paper we suggest a different way of constructing the higher order method, but the necessary number of grid-points is essentially the same as above. However, the required work $W^{\langle p \rangle}$ for the method of order p is only

$$W^{\langle p \rangle}(N) = \frac{p}{2} W^{\langle 2 \rangle}(N)$$

on the same grid. Therefore the relevant ratios for comparison are

$$\begin{aligned} \frac{W^{\langle 2\rangle}(N^{\langle 2\rangle})}{W^{\langle 4\rangle}(N^{\langle 4\rangle})} &= \frac{(90\epsilon)^{1/4}}{(12\epsilon)^{1/2}} = \frac{0.44}{\epsilon^{1/4}}, \\ \frac{W^{\langle 2\rangle}(N^{\langle 2\rangle})}{W^{\langle 6\rangle}(N^{\langle 6\rangle})} &= \frac{(432\epsilon)^{1/6}}{(12\epsilon)^{1/2}} = \frac{0.26}{\epsilon^{1/3}}. \end{aligned}$$

The following table shows these ratios for different values of ϵ:

	ϵ				
	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
W_2/W_4	1	4	14	44	139
W_2/W_6	1	6	26	121	560

Table 1: Work-ratios as a function of required precision

It is only for the lowest precision that the second order method can compete, and not even in that case it is any faster than the higher order ones. Furthermore, in the real multi-dimensional case the memory requirement becomes so large for a second order method, that the problem might become impossible to solve in practice. As an example, $\epsilon = 10^{-6}$, $\omega = 10$ implies $N^{\langle 2\rangle} \approx 10^4$, which corresponds to 10^{12} points in three space-dimensions. For the $6th$ order method, the much more reasonable number $N^{\langle 6\rangle} \approx 10^2$ is sufficient. In Section 6 we will demonstrate these effects for the two-dimensional problem.

3 High order methods and deferred correction

Introduce a uniform grid defined by

$$x_{d,j} = jh_d \quad , \quad h_d = \tfrac{1}{N_d} \quad , \quad j = 0,\ldots,N_d \quad , \quad d = 1,\ldots,D. \tag{3.1}$$

Also define

$$\begin{aligned} \bar{\Omega}_h &= \{0, h_1, 2h_1, \ldots, 1\} \times \{0, h_2, 2h_2, \ldots, 1\} \times \ldots \times \{0, h_D, 2h_D, \ldots, 1\}, \\ \partial\Omega_h &= \bar{\Omega}_h \bigcap \partial\Omega, \\ \Omega_h &= \bar{\Omega}_h \setminus \partial\Omega_h. \end{aligned}$$

We will assume $h_d = \mathcal{O}(h)$, $\forall d$.

We use centered finite differences and obtain high order accuracy using deferred correction. The strategy is to operate in several stages, gaining 2 orders of accuracy in each stage. We demonstrate with an example how to achieve fourth order accuracy for a model problem in 2D, where we assume that $h = h_1 = h_2$, $N = N_1 = N_2$. Then we proceed with a general description of the algorithm.

Define the operators Q_{x_d}, $d = 1, 2$, by

$$Q_{x_d} = \frac{1}{h^2}\Delta_{+x_d}\Delta_{-x_d}, \tag{3.2}$$

where

$$\Delta_{+x_1} u_{ij} = u_{i+1,j} - u_{ij} \quad , \quad \Delta_{-x_1} u_{ij} = u_{ij} - u_{i-1,j} \tag{3.3}$$

and similarly for $\Delta_{\pm x_2}$. Also define the operator Δ_h by

$$\Delta_h = Q_{x_1} + Q_{x_2}. \tag{3.4}$$

A centered second-order discretization of (1.1) yields

$$\Delta_h u^{\langle 2 \rangle} = F, \tag{3.5}$$

i.e.,

$$u^{\langle 2 \rangle} = u + \mathcal{O}(h^2). \tag{3.6}$$

Now

$$\frac{\partial^2 u}{\partial x_d^2} = \left(Q_{x_d} - \tfrac{h^2}{12} Q_{x_d}^2 \right) u + \mathcal{O}(h^4) \quad , \quad d = 1, 2, \tag{3.7}$$

i.e., we can obtain fourth order accuracy in space by solving

$$\left(\Delta_h - \frac{h^2}{12} Q_4 \right) u^{\langle 4 \rangle} = F, \tag{3.8}$$

where

$$Q_4 = Q_{x_1}^2 + Q_{x_2}^2. \tag{3.9}$$

To solve (3.8) directly, we would have to use an iterative solver or a banded direct solver. Moreover, in a domain decomposition setting of the problem, we would get difficulties near the internal boundaries as mentioned above.

Assuming that $u^{\langle 4 \rangle} = u^{\langle 2 \rangle} + h^2 q(\mathbf{x})$, where $q(\mathbf{x})$ is a sufficiently smooth function and noting that Q_4 approximates $\partial^4/\partial x_1^4 + \partial^4/\partial x_2^4$ to second order accuracy, we get

$$\left(\Delta_h - \frac{h^2}{12} Q_4 \right) u^{\langle 4 \rangle} = \Delta_h u^{\langle 4 \rangle} - \frac{h^2}{12} Q_4 u^{\langle 2 \rangle} - \frac{h^4}{12} \left(\frac{\partial^4 q}{\partial x_1^4} + \frac{\partial^4 q}{\partial x_2^4} \right) + \mathcal{O}(h^6). \tag{3.10}$$

By considering the last two terms as a fourth order error, we get

$$\Delta_h \tilde{u}^{\langle 4 \rangle} = F + \frac{h^2}{12} Q_4 u^{\langle 2 \rangle}, \tag{3.11}$$

i.e., we obtain a fourth order approximate solution by solving a second system of equations which is exactly of the same type as the first one defined in (3.5). In Section 5 we will describe how we can make use of this in a domain decomposition setting of the problem.

The general algorithm to obtain a p:th order approximation to u is as follows. Let the coefficients c_j be defined by

$$\begin{cases} c_1 = 1 \\ \Delta u = \sum_{k=1}^{j-1} c_k \sum_{d=1}^{D} h_d^{2(k-1)} Q_{x_d}^k u + c_j \sum_{d=1}^{D} h_d^{2(j-1)} \frac{\partial^{2j} u}{\partial x_d^{2j}} + \mathcal{O}(h^{2j}), \end{cases}$$

$$j = 2, \dots, p/2.$$

Then $u^{\langle p \rangle}$ is obtained by solving the following sequence of problems

$$\sum_{d=1}^{D} Q_{x_d} u^{\langle 2j \rangle} = F - \sum_{k=2}^{j} c_k \sum_{d=1}^{D} h_d^{2(k-1)} Q_{x_d}^k u^{\langle 2(j-1) \rangle} \quad , \quad j = 1, \dots, p/2.$$

This algorithm requires $u^{\langle 2(k-1) \rangle}$ to be defined at $p/2 - 1$ gridlines outside the domain. For this we use extrapolation with a polynomial of order $p + 2$. In Section 4 we will show that this yields a solution that is accurate of order p.

By using this algorithm a high order accurate solution of order p is obtained through $p/2$ applications of a fast domain decomposition Poisson solver with different right hand sides.

4 Error estimates

Until now it has been assumed that the truncation error at all levels is smooth. This requires first of all that $u(\mathbf{x})$ is a smooth solution. This is of course not always the case in applications, but the problem of determining the accuracy of high order approximations for non-smooth solutions is beyond the scope of this paper. However, even under the assumption of a smooth solution, we must make sure that the intermediate approximate solutions are smooth enough, such that the final solution has the right accuracy. This requires that the approximation near the boundaries is constructed carefully.

For the operator on the left hand side, no special procedure is required, since we are assuming a rectangular domain, and the Dirichlet boundary condition is sufficient to close the system. On the right hand side, we apply the centered difference operators at all points inside $\partial\Omega_h$ to approximate the higher order derivatives, and use polynomial extrapolation to define the necessary points outside. This procedure was suggested by Peryera et.al. in [15].

In order to illustrate the technique and the analysis, we shall derive the error estimate for the simplest 2D-case with the fourth order accurate approximation. The generalization to higher order approximations is straightforward. The problem is

$$\begin{aligned} \Delta u = F(\mathbf{x}) \quad &, \quad \mathbf{x} \in \Omega = [0,1] \times [0,1], \\ u = 0 \quad &, \quad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{4.1}$$

In the following of this section we will assume $h = h_1 = h_2$ and $N = N_1 = N_2$.

In order to achieve sufficient smoothness of the solution, we assume that $F \in C^\infty(\mathbf{x})$. Furthermore we need certain compatibility conditions in the corners, and for convenience we assume

$$\frac{\partial^{l+k}F}{\partial x_1^l \partial x_2^k}(\mathbf{x}) = 0 \ , \ \mathbf{x} \in \partial\Omega \ , \quad 0 \le l, k \, . \tag{4.2}$$

In particular, this assumption guarantees that

$$\frac{\partial^{2k}u}{\partial x_1^{2k}} = \frac{\partial^{2k}u}{\partial x_2^{2k}} = 0 \ , \ \mathbf{x} \in \partial\Omega \ , \quad 0 \le k \, . \tag{4.3}$$

The fourth order approximation is defined by the two-stage procedure

$$\begin{array}{rcll} \Delta_h u_{ij}^{\langle 2\rangle} & = & F_{ij} & , \ \mathbf{x}_{ij} \in \Omega_h \, , \\ u_{ij}^{\langle 2\rangle} & = & 0 & , \ \mathbf{x}_{ij} \in \partial\Omega_h \, , \end{array} \tag{4.4}$$

$$\begin{array}{rcll} \Delta_h u_{ij}^{\langle 4\rangle} & = & F_{ij} + \frac{h^2}{12} Q_4 u_{ij}^{\langle 2\rangle} & , \ \mathbf{x}_{ij} \in \Omega_h \, , \\ u_{ij}^{\langle 4\rangle} & = & 0 & , \ \mathbf{x}_{ij} \in \partial\Omega_h \, . \end{array} \tag{4.5}$$

The operator Q_4 definied in (3.9) approximates $\partial^4/\partial x_1^4 + \partial^4/\partial x_2^4$ to second order accuracy. The corresponding stencil is 5 points wide in each direction, and when applied at the gridlines next to the boundary, it requires $u^{\langle 2\rangle}$ to be defined on one extra gridline outside the domain in each direction. This is accomplished through extrapolation as desribed in Section 3.

In order to get the correct order of accuracy, it is essential that $Q_4 u_j^{\langle 2\rangle}$ is accurate enough. Even if $u^{\langle 2\rangle}$ has second order accuracy, we need it to be smooth enough such that $Q_4 u_j^{\langle 2\rangle}$ approximates $\left(\partial^4/\partial x_1^4 + \partial^4/\partial x_2^4\right) u$ with a second order error.

For the Poisson equation there is a maximum principle, also for the discrete second order approximation. Since we intend to use our method even for systems of elliptic equations with variable coefficients, where this principle does not necessarily apply, we shall use l_2-theory that can be generalized in a straightforward way. The scalar product and norm are defined by

$$(u,v)_{\substack{\mu,\nu \\ m,n}} = \sum_{i=\mu}^{\nu} \sum_{j=m}^{n} u_{ij} v_{ij} h^2 \ , \quad ||u||^2_{\substack{\mu,\nu \\ m,n}} = (u,u)_{\substack{\mu,\nu \\ m,n}} \ .$$

We shall also use the notation

$$(u,v) = (u,v)_{\substack{1,N-1 \\ 1,N-1}} \quad , \quad ||u||^2 = (u,u)$$

We begin by the well known Sobolev inequality, (see for example [18]):

Lemma 4.1 *Gridfunctions f with $f_{ij} = 0 \, , \ \mathbf{x} \in \partial\Omega_h$ satisfy the inequality*

$$||f||^2 \le \mathrm{const}(||D_{+x_1} f||^2_{\substack{0,N-1 \\ 1,N-1}} + ||D_{+x_2} f||^2_{\substack{1,N-1 \\ 0,N-1}}) \, .$$

Consider next the problem

$$\begin{array}{rcll} \Delta_h u_{ij} & = & F_{ij} & , \ \mathbf{x}_{ij} \in \Omega_h \, , \\ u_{ij} & = & 0 & , \ \mathbf{x}_{ij} \in \partial\Omega_h \, . \end{array} \tag{4.6}$$

Proposition 4.2 *The solutions of* (4.6) *satisfy the estimate*

$$||u|| \leq \text{const}||F|| \, .$$

Proof We take the scalar product of the difference equation with u, and use summation by parts to obtain

$$-||D_{+x_1}u||^2_{\substack{0,N-1 \\ 1,N-1}} - ||D_{+x_2}u||^2_{\substack{1,N-1 \\ 0,N-1}} = (u, F) \, ,$$

which gives

$$||D_{+x_1}u||^2_{\substack{0,N-1 \\ 1,N-1}} + ||D_{+x_2}u||^2_{\substack{1,N-1 \\ 0,N-1}} \leq \delta ||u||^2 + \frac{1}{4\delta}||F||^2 \, ,$$

where δ is any positive constant. By Lemma 4.1 it follows that

$$||u||^2 \leq \text{const}(\delta ||u||^2 + \frac{1}{4\delta}||F||^2) \, .$$

By choosing δ sufficiently small, we obtain the desired estimate. ■

To derive the error estimate, we use the technique that is the basis for Richardson extrapolation. Since $u(\mathbf{x})$ is a smooth solution, we have

$$\Delta_h u(\mathbf{x}_{ij}) = \Delta u(\mathbf{x}_{ij}) + h^2 \psi^{(2)}(\mathbf{x}_{ij}) + h^4 \psi^{(4)}(\mathbf{x}_{ij}) + h^6 \psi^{(6)}_{ij} \, . \tag{4.7}$$

(Throughout this paper we use the notation $\psi(\mathbf{x}_{ij})$ for functions defined everywhere, while ψ_{ij} denotes gridfunctions). Here $\psi^{(2)}, \psi^{(4)}$ are even derivatives of u multiplied by a constant. They are smooth functions, and by (4.3) they satisfy

$$\frac{\partial^{2k}\psi^{(\nu)}}{\partial x_1^{2k}}(\mathbf{x}) = \frac{\partial^{2k}\psi^{(\nu)}}{\partial x_2^{2k}}(\mathbf{x}) = 0 \ , \ \mathbf{x} \in \partial\Omega, \ \ \nu = 2, 4 \ , \ \ k = 0, 1, \dots, . \tag{4.8}$$

We make the ansatz

$$u^{\langle 2 \rangle}_{ij} = u(\mathbf{x}_{ij}) + h^2 \phi^{(2)}(\mathbf{x}_{ij}) + h^4 \phi^{(4)}(\mathbf{x}_{ij}) + h^6 \phi^{(6)}_{ij} \, , \tag{4.9}$$

and we want to show that $\phi^{(2)}, \phi^{(4)}$ are smooth functions, and that $\phi^{(6)}$ is a bounded gridfunction. By combining the equations (4.7) and (4.9) and using (4.1), we obtain

$$\begin{array}{l} h^2(\psi^{(2)}(\mathbf{x}_{ij}) + \Delta_h \phi^{(2)}(\mathbf{x}_{ij})) + h^4(\psi^{(4)}(\mathbf{x}_{ij}) + \Delta_h \phi^{(4)}(\mathbf{x}_{ij})) + \\ h^6(\psi^{(6)}_{ij} + \Delta_h \phi^{(2)}_{ij}) = 0 \ , \ \ \mathbf{x}_{ij} \in \Omega_h \, . \end{array} \tag{4.10}$$

Let $\phi^{(2)}(\mathbf{x})$ be the solution of

$$\begin{array}{rcll} \Delta\phi^{(2)}(\mathbf{x}) & = & -\psi^{(2)}(\mathbf{x}) & , \ \mathbf{x} \in \Omega \, , \\ \phi^{(2)}(\mathbf{x}) & = & 0 & , \ \mathbf{x} \in \partial\Omega \, , \end{array} \tag{4.11}$$

By our smoothness assumption and (4.8) we conclude that $\phi^{(2)}(\mathbf{x})$ is smooth. Furthermore we have

$$\Delta_h \phi^{(2)}(\mathbf{x}_{ij}) = \Delta \phi^{(2)}(\mathbf{x}_{ij}) + h^2 \tilde{\psi}^{(2)}(\mathbf{x}_{ij}) + h^4 \tilde{\psi}^{(4)}_{ij} ,$$

where $\tilde{\psi}^{(2)}(\mathbf{x})$ consists of even derivatives of $\phi^{(2)}$, i.e. even derivatives of u satisfying the condition (4.3) at the boundary. Next we let $\phi^{(4)}(\mathbf{x})$ be the solution of

$$\begin{aligned} \Delta \phi^{(4)}(\mathbf{x}) &= -\psi^{(4)}(\mathbf{x}) - \tilde{\psi}^{(2)}(\mathbf{x}) \quad , \ \mathbf{x} \in \Omega , \\ \phi^{(4)}(\mathbf{x}) &= 0 \quad , \ \mathbf{x} \in \partial\Omega . \end{aligned}$$

Since the difference between $\Delta\phi^{(4)}$ and $\Delta_h \phi^{(4)}$ is $\mathcal{O}(h^2)$, we can repeat the arguments above, and collect all remaining terms in (4.10) to obtain

$$\begin{aligned} \Delta_h \phi^{(6)}_{ij} &= -\tilde{\tilde{\psi}}^{(6)}_{ij} \quad , \ \mathbf{x}_{ij} \in \Omega_h , \\ \phi^{(6)}_{ij} &= 0 \quad , \ \mathbf{x}_{ij} \in \partial\Omega_h . \end{aligned}$$

The solution $\phi^{(6)}$ of this problem is bounded, and therefore the expansion (4.9) has the desired properties.

The grid-function $u^{\langle 2 \rangle}$ is well defined for all $\mathbf{x} \in \bar{\Omega}_h$. The extra gridline outside the domain required by $Q_4 u^{\langle 2 \rangle}$ is defined by extrapolation, in our case of order 6 (or higher). As an example, consider the extra line to the right of the domain, and the extrapolation

$$[\Delta^p_{-x_1} u^{\langle 2 \rangle}_{ij}]_{i=N+1} = 0 \ , \quad p \geq 6 ,$$

i.e. we represent $u^{\langle 2 \rangle}$ locally by a polynomial of degree $p - 1 \geq 5$.

Let $f(\mathbf{x})$ be a general smooth function in $\Omega \cup \partial\Omega$, and define f on the external gridline by

$$[\Delta^p_{-x_1} f(\mathbf{x}_{ij})]_{i=N+1} = 0 \ , \quad p \geq 6 .$$

Then there exists a smooth function $\tilde{f}$ defined in the extended domain, that is equal to f in $\Omega_h \cup \partial\Omega_h$ and

$$f(\mathbf{x}_{N+1,j}) = \tilde{f}(\mathbf{x}_{N+1,j}) + \mathcal{O}(h^6) .$$

Since $Q_4 \sim h^{-4}$, we have

$$Q_4 f(\mathbf{x}_{ij}) = (\frac{\partial^4}{\partial x_1^4} + \frac{\partial^4}{\partial x_2^4}) f(\mathbf{x}_{ij}) + \mathcal{O}(h^2) \ , \quad \mathbf{x}_{ij} \in \Omega_h .$$

Therefore, when applying Q_4 on $u^{\langle 2 \rangle}$, we use the expansion (4.9) and the fact that u, $\phi^{(2)}$, $\phi^{(4)}$ are all smooth and $\phi^{(6)}$ bounded. We get

$$\begin{aligned} Q_4 u^{\langle 2 \rangle}_{ij} &= (\tfrac{\partial^4}{\partial x_1^4} + \tfrac{\partial^4}{\partial x_2^4})(u(\mathbf{x}_{ij}) + h^2 \phi^{(2)}(\mathbf{x}_{ij}) + h^4 \phi^{(4)}(\mathbf{x}_{ij})) + \mathcal{O}(h^2) \\ &= (\tfrac{\partial^4}{\partial x_1^4} + \tfrac{\partial^4}{\partial x_2^4}) u(\mathbf{x}_{ij}) + \mathcal{O}(h^2) . \end{aligned} \tag{4.12}$$

In order to derive the final error estimate, we note that

$$\Delta_h u(\mathbf{x}_{ij}) = F(\mathbf{x}_{ij}) + \frac{h^2}{12}(\frac{\partial^4}{\partial x_1^4} + \frac{\partial^4}{\partial x_2^4})u(\mathbf{x}_{ij}) + \mathcal{O}(h^4) \ , \quad \mathbf{x}_{ij} \in \Omega_h \, . \tag{4.13}$$

By subtracting (4.13) from (4.5) and using (4.12), we get for the error $w_{ij} = u_{ij}^{\langle 4 \rangle} - u(\mathbf{x}_{ij})$

$$\begin{array}{rcll} \Delta_h w_{ij} & = & \mathcal{O}(h^4) & , \ \mathbf{x}_{ij} \in \Omega_h \, , \\ w_{ij} & = & 0 & , \ \mathbf{x}_{ij} \in \partial\Omega_h \, . \end{array} \tag{4.14}$$

By using Proposition 4.2, we obtain

Proposition 4.3 *The error in the solution of* (4.4), (4.5) *satisfies the estimate*

$$||u_{ij}^{\langle 4 \rangle} - u(\mathbf{x}_{ij})|| \leq \text{const. } h^4 \, . \tag{4.15}$$

5 Domain Decomposition

Next we will introduce a domain decomposition setting to the problem. Domain decomposition methods for elliptic problems have been considered by many authors in e.g. [1], [3], [4], [6], [7] and [10]. Here we will consider the method described in [4].

We divide the original domain in q nonoverlapping subdomains Ω_m. The internal boundaries that separate two subdomains are denoted by Γ_m, $m = 1, 2, \ldots, q-1$. We will restrict our formulation to the unit-square in 2D and internal boundaries oriented in the x_1-direction, see Figure 1.

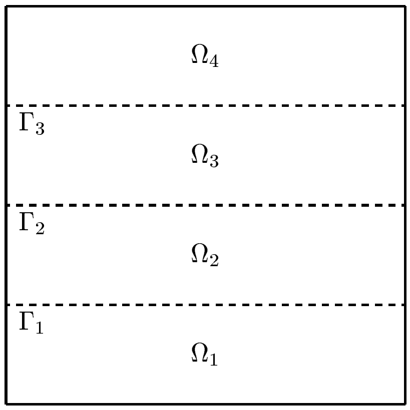

Figure 1: The original domain decomposed in four subdomains.

We consider an implementation on a parallel computer with q processors, where processor number m stores the unknowns in Ω_m and Γ_m.

In the following we will omit the superindices and discuss the solution of

$$\begin{aligned} \Delta_h u = F \quad &, \quad \mathbf{x}_{ij} \in \Omega_h, \\ u = g \quad &, \quad \mathbf{x}_{ij} \in \partial\Omega_h. \end{aligned} \tag{5.1}$$

By denoting the unknowns in $\bigcup_{m=1}^{q-1} \Gamma_m$ by u_0 and those in $\bigcup_{m=1}^{q} \Omega_m$ by u_1 we obtain the following partitioned system to solve:

$$\begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}. \tag{5.2}$$

Note that by having forced the operator on the left-hand side of (3.11) to involve only three grid-points in each direction, we only have one gridline in the internal boundary. For a high order discretization and hence a wider stencil this is not the case.

Block-Gaussian elimination on (5.2) yields the so called Schur-complement system

$$C u_0 = F_0 - A_{01} A_{11}^{-1} F_1, \tag{5.3}$$

where

$$C = A_{00} - A_{01} A_{11}^{-1} A_{10}. \tag{5.4}$$

Once (5.3) is solved, u_0 is inserted into (5.2) and all subdomain systems can be solved in parallel, independent of each other. We summarize this in the following algorithm:

Algorithm 5.1
Solve $A_{11} w = F_1$,
Solve $C u_0 = F_0 - A_{01} w$,
Solve $A_{11} v = A_{10} u_0$,
$u_1 = w - v$.

In [4] it is shown that the capacitance matrix C has a decomposition in sine-matrices

$$C = (I_{q-1} \otimes S_{N_1-1})\, T\, (I_{q-1} \otimes S_{N_1-1}),$$

where

$$S_{N_1-1}(j,k) = \sqrt{\tfrac{2}{N_1}} \sin\left(\tfrac{jk\pi}{N_1}\right) \quad , \quad k, j = 1, \ldots, N_1 - 1 \ .$$

The matrix T has the form

$$T = \begin{pmatrix} \Lambda_1 & \Theta_1 & & \\ \Theta_2 & \ddots & \ddots & \\ & \ddots & \ddots & \Theta_{q-2} \\ & & \Theta_{q-1} & \Lambda_{q-1} \end{pmatrix},$$

where Λ_j and Θ_j are diagonal. Hence the solution of (5.3) can be accomplished through

1. $q-1$ fast sine transforms of length N_1-1 (see e.g. [9]),
2. the solution of N_1-1 tridiagonal systems of order $q-1$,
3. $q-1$ fast sine transforms of length N_1-1.

The $q-1$ fast sine transforms in stage 1. and 3. can take place in parallel, indepent of each other. The tridiagonal systems however, are spread over the processors. We solve these using cyclic reduction, [8].

Similarly, the coefficient-matrices of the subdomain systems have a decomposition in sine-matrices (see e.g. [9]). Denote by $A_{11}^{(m)}$ the coefficient-matrix to the subdomain system in Ω_m and by $N_2^{(m)}-1$ the number of unknowns in the x_2-direction in Ω_m. Then

$$A_{11}^{(m)} = \left(I_{N_2^{(m)}-1} \otimes S_{N_1-1}\right) M \left(I_{N_2^{(m)}-1} \otimes S_{N_1-1}\right),$$

where M is a matrix on the form

$$M = \begin{pmatrix} D_1 & G_1 & & \\ G_2 & \ddots & \ddots & \\ & \ddots & \ddots & G_{N_2^{(m)}-2} \\ & & G_{N_2^{(m)}-1} & D_{N_2^{(m)}-1} \end{pmatrix},$$

where D_j and G_j are diagonal. Hence, we can solve each subdomain system through

1. $N_2^{(m)}-1$ fast sine transforms of length N_1-1,
2. the solution of N_1-1 tridiagonal systems of order $N_2^{(m)}-1$,
3. $N_2^{(m)}-1$ fast sine transforms of length N_1-1.

Note, that all subdomain systems can be solved completely in parallel. Assuming $N_1 = N_2 = N$ and $q << N$, the number of arithmetic operations for the complete algorithm is $\approx 5pN^2\log_2 N$. Hence, with q parallel processors, the effective computing time is proportional to $5\frac{p}{q}N^2\log_2 N$.

6 Numerical experiments

In this section we will present the results from the numerical experiments performed using the method described in this paper. We consider Poisson's equation in 2D. In order to be able to compare the error in the solution for different methods, we prescribe the solution $u(\mathbf{x})$. In Figures 2-6 we display contour-plots of the different solutions.

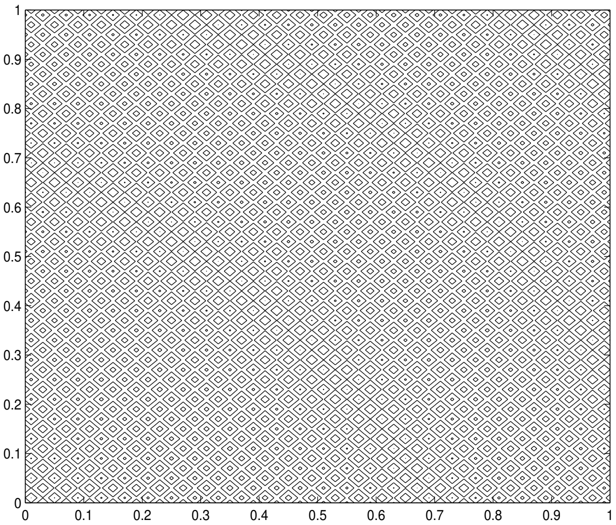

Figure 2: $u = \sin(50\pi x_1) + \sin(50\pi x_2) + e^{x_1} + e^{x_2}$.

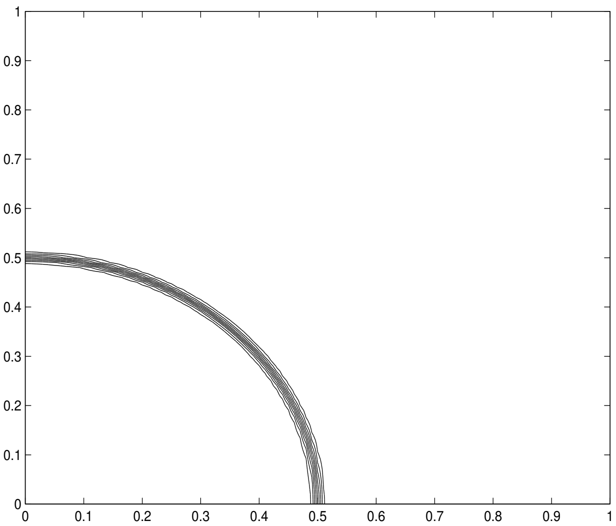

Figure 3: $u = \tanh\left(\frac{r-0.5}{0.01}\right)$, $r = \sqrt{x_1^2 + x_2^2}$.

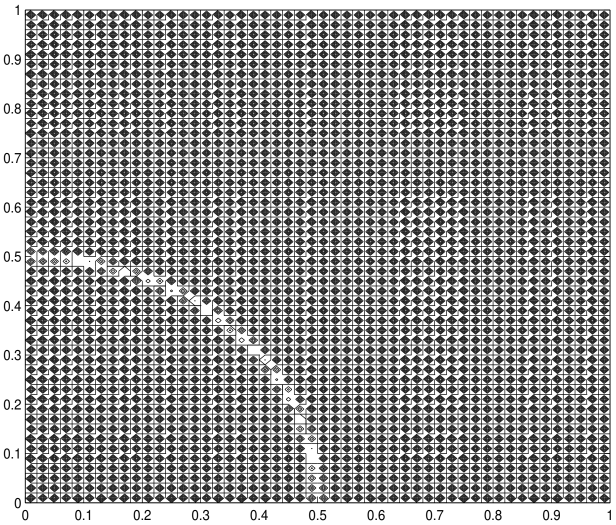

Figure 4: $u = \tanh\left(\frac{r-0.5}{0.01}\right)\sin(50\pi x_1)\sin(50\pi x_2)$, $r = \sqrt{x_1^2 + x_2^2}$.

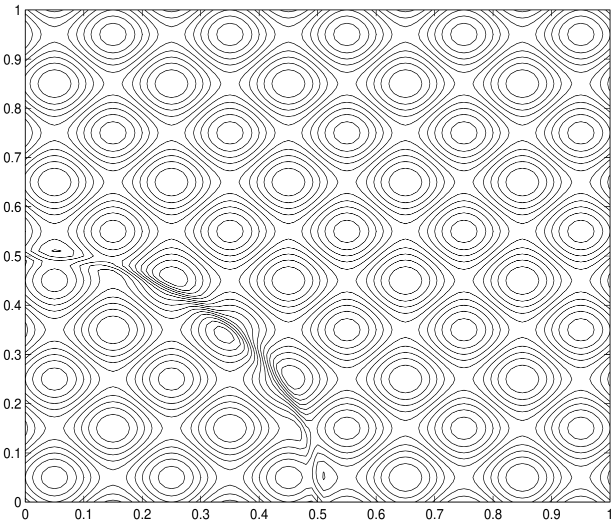

Figure 5: $u = \tanh\left(\frac{r-0.5}{0.01}\right) + \sin(10\pi x_1) + \sin(10\pi x_2)$, $r = \sqrt{x_1^2 + x_2^2}$.

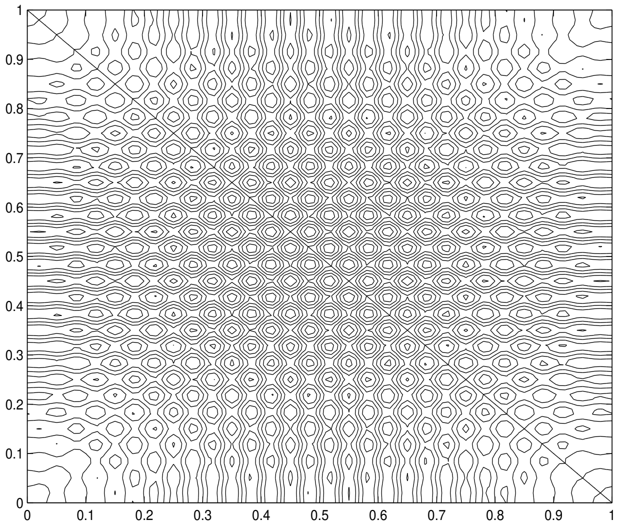

Figure 6: $u = \sin(\pi x_1)\sin(10\pi x_1) + \sin(\pi x_2)\sin(10\pi x_2)$.

The computer used was "Yggdrasil" at the Department of Scientific Computing in Uppsala. This computer consists of three Digital AlphaServer 5/300, each housing:

- Four Alpha 21164 ("EV5") processors running at 300 MHz. Each processor has an on-chip cache and a 4MByte secondary level cache.

- 1 GByte shared memory.

In all computations presented we used 8 processors. We started to take time *after* a factorization of the coefficient-matrices involved was performed. However, the time to compute the right-hand side and to compute the solution for that actual right-hand side is included in the times presented.

$N-1$	$\Vert(u_{ij}^{\langle p\rangle}-u(\mathbf{x}_{ij}))\Vert_2$		wall clock-time(s)	
	p=2	p=6	p=2	p=6
$u=\sin(50\pi x_1)\sin(50\pi x_2)+e^{x_1}+e^{x_2}$				
255	$3.2\cdot10^{-2}$	$1.0\cdot10^{-4}$	0.065	0.15
511	$7.8\cdot10^{-3}$	$7.9\cdot10^{-6}$	0.28	0.56
1023	$2.0\cdot10^{-3}$	$8.5\cdot10^{-8}$	0.97	2.5
2047	$4.9\cdot10^{-4}$	$7.6\cdot10^{-10}$	4.0	12.8
$u=\tanh\left(\frac{r-0.5}{0.01}\right),\ r=\sqrt{x_1^2+x_2^2}$				
255	$8.9\cdot10^{-4}$	$9.8\cdot10^{-6}$	0.065	0.16
511	$2.2\cdot10^{-4}$	$1.9\cdot10^{-7}$	0.25	0.62
1023	$5.5\cdot10^{-5}$	$3.1\cdot10^{-9}$	1.1	2.9
2047	$1.4\cdot10^{-5}$	$4.9\cdot10^{-11}$	4.2	14.3
$u=\tanh\left(\frac{r-0.5}{0.01}\right)\sin(50\pi x_1)\sin(50\pi x_2),\ r=\sqrt{x_1^2+x_2^2}$				
255	$1.6\cdot10^{-2}$	$5.7\cdot10^{-5}$	0.076	0.15
511	$3.9\cdot10^{-3}$	$1.4\cdot10^{-6}$	0.29	0.60
1023	$9.8\cdot10^{-4}$	$2.0\cdot10^{-8}$	1.2	2.7
2047	$2.5\cdot10^{-4}$	$2.8\cdot10^{-10}$	5.0	14.0
$u=\tanh\left(\frac{r-0.5}{0.01}\right)+\sin(10\pi x_1)+\sin(10\pi x_2),\ r=\sqrt{x_1^2+x_2^2}$				
255	$1.5\cdot10^{-3}$	$9.8\cdot10^{-6}$	0.097	0.16
511	$3.7\cdot10^{-4}$	$1.9\cdot10^{-7}$	0.28	0.59
1023	$9.2\cdot10^{-5}$	$3.1\cdot10^{-9}$	1.2	2.6
2047	$2.3\cdot10^{-5}$	$4.9\cdot10^{-11}$	4.8	14.3
$u=\sin(\pi x_1)\sin(30\pi x_1)+\sin(\pi x_2)\sin(30\pi x_2),\ r=\sqrt{x_1^2+x_2^2}$				
255	$8.0\cdot10^{-3}$	$9.4\cdot10^{-6}$	0.064	0.16
511	$2.0\cdot10^{-3}$	$6.2\cdot10^{-8}$	0.24	0.57
1023	$4.9\cdot10^{-4}$	$6.9\cdot10^{-10}$	1.0	2.4
2047	$1.2\cdot10^{-4}$	$1.0\cdot10^{-11}$	4.0	12.5

Table 2: Error in the solution and elapsed wall clock-time.

The sixth order method on the coarsest grid gives a smaller error than the second order method on the finest grid, and it does so with a computing time that is more than 25 times shorter in all cases. The largest problem possible to fit into the memory was 2047×2047. Hence, it was not even possible in any of these cases to obtain the same small error in the solution with the standard second order method as with the sixth order method with 255×255 points.

Note that the required number of grid-points and the computing time well agree with the analysis in Section 2 (generalized to the two-dimensional case). Note also that the small constant multiplying $N^2 \log_2 N$ in the operation count is reflected in the very short computing time. For example, the highly oscillatory solution in the first problem in Table 2 is computed with an accuracy of 10^{-4} in 0.15 seconds on the medium sized computer used for these experiments.

7 Conclusions

By analyzing a simple model problem, we have given a theoretical foundation for the use of higher order approximations instead of lower order ones for elliptic problems. Domain decomposition methods are ideally suited for parallel implementation. One class of such methods is based on the solution of the Schur complement system together with the solution of the subdomain systems. In order to keep the subdomain systems mutually independent for high order accurate approximations, we use the principle of deferred corrections. In that way, the order of accuracy p is achieved by a $p/2$-step procedure, where in each step the left hand side corresponds to the standard second order approximation. Each such step is solved by a fast direct domain decomposition method.

The fast algorithm requires a structured rectangular grid. Numerical experiments show that a sixth order method by far outperforms the standard second order one, both in CPU-time and memory requirement. It was not even possible to obtain the same accuracy in the solution with the second order method as with the sixth order method due to memory restrictions. The very fast algorithm makes it possible to resolve features in the solution that normally would require adaptive methods and irregular grids with much higher complexity.

The type of method used here can be generalized to general ellipic systems. However, the direct fast second order solver used in each step must be substituted by an iterative solver.

References

[1] P. Bjorstad, and O. Widlund, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1097-1120.

[2] B.L. Buzbee, G.H. Golub, and C.W. Nielson, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627-656.

[3] T.F. Chan, *Analysis of preconditioners for domain decomposition*, SIAM J. Numer. Anal., 24 (1987), pp. 382-390.

[4] T.F. Chan, and D.C. Resasco, *A domain-decomposed fast Poisson solver on a rectangle*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 14-26.

[5] F. Dorr, *The direct solution of the discrete Poisson equation on a rectangle*, SIAM Rev., 12 (1970), pp. 248-263.

[6] M. Dryja, *A capacitance matrix method for Dirichlet problem on polygonal region*, Numer. Math., 39 (1982), pp. 51-64.

[7] G.H. Golub, and D. Mayers, *The use of pre-conditioning over irregular regions*, 1983. Lecture at Sixth International Conference on Computing Methods in Applied Sciences and Engineering, Versailles, France, December 1983.

[8] R.W. Hockney, and C.R. Jesshope, *Parallel computers 2*, IOP Publishing Ltd, 1988.

[9] C.F. van Loan, *Computational frameworks for the fast Fourier transform*, SIAM, Philadelphia 1992.

[10] D.E. Keyes, and W.D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 166-202.

[11] V. Pereyra, *Accelerating the convergence of discretizations algorithms*, SIAM J. Numer. Anal., 4 (1967), pp. 508-532.

[12] V. Pereyra, *Iterated deferred corrections for nonlinear operator equations*, Numer. Math., 10 (1967), pp. 316-323.

[13] V. Pereyra, *Iterated deferred corrections for nonlinear boundary value problems*, Numer. Math., 11 (1968), pp. 111-125.

[14] V. Pereyra, *Highly accurate numerical solution of casilinear elliptic boundary-value problems in n dimensions*, Math. Comp., 24 (1970), pp. 771-783.

[15] V. Pereyra, W. Proskurowski, and O. Widlund, *High order fast Laplace solvers for the Dirichlet problem on general regions*, Math. Comp., 31 (1977), pp. 1-16.

[16] G. Starius, *Asymptotic expansions for a class of elliptic difference schemes*, Math. Comp., 37 (1981), pp. 321-326.

[17] P.N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle*, SIAM Rev., 19 (1977), pp. 490-501.

[18] V. Thomée, *Elliptic difference operators and Dirichlet's problem*, Contrib. to Diff. Equations, III no. 3 (1964), pp. 301-324.