Reuse of preconditioners in implicit variable time-step methods

Jawad Al-Temimi

RuG

Mathematics

Master's Thesis

Reuse of preconditioners in implicit variable time-step methods

Jawad Al-Temimi

University of Groningen
Informatica
Postbus 800
9700 AV Groningen

April 19, 2005

Contents

1 Introduction 2

2 Problem Description 2
- 2.1 Model Problem . . . 2
- 2.2 Space Discretization of the Problem . . . 3
- 2.3 Time Discretization . . . 5
- 2.4 The variable time-step algorithm . . . 6

3 Various rewritings of the linear system and results 7
- 3.1 Introduction . . . 7
- 3.2 Method 1 . . . 8
- 3.3 Method 2 . . . 11
- 3.4 Method 3 . . . 13
- 3.5 Method 4 . . . 14
- 3.6 Condition numbers and iteration counts . . . 16

4 Discussion and Conclusion 18

1 Introduction

In this report we study the reuse of a LU-decomposition in an *implicit method* for time-integration with a variable time step. The model problem is a *parabolic equation* and the θ-method is used for the time-integration. The discretization of this problem leads in every time step to a system of the form $Ax = f$.

In general, the work required to compute the LU-decomposition is $O(n^3)$ and the work to backsolve with computed LU factors is $O(n^2)$. Though these amounts are for full matrices, also for sparse matrices the factorization and solve is expensive. In any case, it is efficient to reuse the LU-decomposition as much as possible.

If Δt is constant, the system will not change and we can solve $Ax = f$ easily by using the preconditioner or LU factorization of A. If Δt will take various values, the diagonal of the matrix will change. We will study several rewritings of the system in order to be able to reuse the preconditioner of previous time steps.

2 Problem Description

In this section we will describe the continuous model problem and its discretization. For the discretization we follow the method of lines approach, hence we first descretize in space and next apply a suitable method to the resulting system of ODE's. In one case we apply the θ-method

2.1 Model Problem

We want to solve the following *parabolic equation* in two dimensions

$$u_t = u_{xx} + u_{yy} + g(x, y, t), \quad 0 < x, y < 1, \quad t > 0$$

subject to the initial and boundary conditions

$$u(x, y, 0) = f(x, y), \quad \text{for} \quad 0 < x, y < 1$$

$$u(0, y, t) = f_L(y, t) = 0, \quad u(1, y, t) = f_R(y, t) = 0$$

$$u(x, 0, t) = f_B(x, t) = \sin((f_1 \exp(-0.1t) + f_2)t) \sin(\pi x), \quad \text{for} \quad x = 0, 1$$

$$u(x, 1, t) = f_T(x, t) = 0, \quad \text{and} \quad g(x, y, t) = 0, \quad \text{for} \quad t > 0$$

2.2 Space Discretization of the Problem

For the space discretization we use a cartesian grid with meshsize

$$h = \frac{1}{N+1}, \quad x_i = ih, \quad y_j = jh, \quad 1 \leq i, j \leq N$$

On this grid we use the following approximations

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j, t) \cong \frac{U_{i+1,j}(t) - 2U_{i,j}(t) + U_{i-1,j}(t)}{h^2},$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j, t) \cong \frac{U_{i,j+1}(t) - 2U_{i,j}(t) + U_{i,j-1}(t)}{h^2}$$

where $U_{i,j}(t) = U(x_i, y_j, t)$

This results use a system of ODE's

$$\frac{dU_{i,j}}{dt} = F_{i,j}(U, t) = HU_{i,j} + g_{i,j}(t),$$

where $g_{i,j}(t) = g(x_i, y_j, t)$, and the difference operator H is implicitly defined by

$$HU_{i,j}(t) = \frac{U_{i+1,j}(t) - 2U_{i,j}(t) + U_{i-1,j}(t)}{h^2} + \frac{U_{i,j+1}(t) - 2U_{i,j}(t) + U_{i,j-1}(t)}{h^2}$$

Hence we have the following system

$$\frac{dU_{i,j}}{dt} = \frac{U_{i+1,j}(t) - 2U_{i,j}(t) + U_{i-1,j}(t)}{h^2} + \frac{U_{i,j+1}(t) - 2U_{i,j}(t) + U_{i,j-1}(t)}{h^2} + g_{i,j}(t)$$

Along the boundaries we have

- $i = 1, j = 1, 2, \ldots, N$:

$$\frac{dU_{1,j}}{dt} = \frac{U_{2,j}(t) - 2U_{1,j}(t)}{h^2} + \frac{U_{1,j+1}(t) - 2U_{1,j}(t) + U_{1,j-1}(t)}{h^2} + \left(\frac{U_{0,j}(t)}{h^2} + g_{1,j}(t)\right)$$

- $i = N, j = 1, 2, \ldots, N$:

$$\frac{dU_{N,j}}{dt} = \frac{-2U_{N,j}(t) + U_{N-1,j}(t)}{h^2} + \frac{U_{N,j+1}(t) - 2U_{N,j}(t) + U_{N,j-1}(t)}{h^2} + \left(\frac{U_{N+1,j}(t)}{h^2} + g_{N,j}(t)\right)$$

- $j = 1, i = 1, 2, \ldots, N$:

$$\frac{dU_{i,1}}{dt} = \frac{U_{i+1,1}(t) - 2U_{i,1}(t) + U_{i-1,1}(t)}{h^2} + \frac{U_{i,2}(t) - 2U_{i,1}(t)}{h^2} + \left(\frac{U_{i,0}(t)}{h^2} + g_{i,1}(t)\right)$$

- $j = N, i = 1, 2, \ldots, N$:

$$\frac{dU_{i,N}}{dt} = \frac{U_{i+1,N}(t) - 2U_{i,N}(t) + U_{i-1,N}(t)}{h^2} + \frac{-2U_{i,N}(t) + U_{i,N-1}(t)}{h^2} + \left(\frac{U_{i,N+1}(t)}{h^2} + g_{i,N}(t)\right)$$

Where

$$U_{0,j}(t) = U(x_0, y_j, t) = f_L(y_j, t), \quad U_{N+1,j}(t) = U(x_{N+1}, y_j, t) = f_R(y_j, t),$$

$$U_{i,0}(t) = U(x_i, y_0, t) = f_B(x_i, t), \quad \text{and} \quad U_{i,N+1}(t) = U(x_i, y_{N+1}, t) = f_T(x_i, t),$$

where $i = 1, 2, \ldots, N, \quad j = 1, 2, \ldots, N, \quad \text{and} \quad t > 0$

For the implementation it is convenient to define

$$G_{i,j} = g_{i,j}, \quad \text{for} \quad 2 \leq i, j \leq N-1$$

$$G_{1,j} = g_{1,j}(t) + \frac{U_{0,j}(t)}{h^2} = g_{1,j}(t) + \frac{f_L(y_j,t)}{h^2}, \quad \text{for} \quad j = 2, 3, \ldots, N-1$$

$$G_{N,j} = g_{N,j}(t) + \frac{U_{N+1,j}(t)}{h^2} = g_{N,j}(t) + \frac{f_R(y_j,t)}{h^2}, \quad \text{for} \quad j = 2, 3, \ldots, N-1$$

$$G_{i,1} = g_{i,1}(t) + \frac{U_{i,0}(t)}{h^2} = g_{i,1}(t) + \frac{f_B(x_i,t)}{h^2}, \quad \text{for} \quad i = 2, 3, \ldots, N-1$$

$$G_{i,N} = g_{i,N}(t) + \frac{U_{i,N+1}(t)}{h^2} = g_{i,N}(t) + \frac{f_T(x_i,t)}{h^2}, \quad \text{for} \quad i = 2, 3, \ldots, N-1$$

and

$$G_{1,1} = g_{1,1}(t) + \tfrac{1}{h^2}[f_B(x_1,t) + f_L(y_1,t)]$$

$$G_{1,N} = g_{1,N}(t) + \tfrac{1}{h^2}[f_T(x_1,t) + f_L(y_N,t)]$$

$$G_{N,1} = g_{N,1}(t) + \tfrac{1}{h^2}[f_B(x_N,t) + f_R(y_1,t)]$$

$$G_{N,N} = g_{N,N}(t) + \tfrac{1}{h^2}[f_T(x_N,t) + f_R(y_N,t)]$$

By using this G instead of g we can solve the same problem using homogeneous Dirichlet boundary conditions. In the following we write the system in matrix vector notation.
First define the following matrices

$$L = \begin{pmatrix} -4 & 1 & 0 & \ldots & 0 \\ 1 & -4 & 1 & \ldots & 0 \\ 0 & 1 & -4 & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 0 & 0 & 0 & 1 & -4 \end{pmatrix}, \hat{H} = \frac{1}{h^2} \begin{pmatrix} L & I & 0 & \ldots & 0 \\ I & L & I & \ldots & 0 \\ 0 & I & L & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & I \\ 0 & 0 & 0 & I & L \end{pmatrix}$$

where I is the identity matrix of order N. Furthermore, we define the vectors

$$U = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_{N-1} \\ U_N \end{pmatrix}, U_j = \begin{pmatrix} U_{1,j} \\ U_{2,j} \\ U_{3,j} \\ \vdots \\ U_{N-1,j} \\ U_{N,j} \end{pmatrix}$$

and likewise

$$G(t) = \begin{pmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ \vdots \\ G_{N-1}(t) \\ G_N(t) \end{pmatrix}, G_j(t) = \begin{pmatrix} G_{1,j}(t) \\ G_{2,j}(t) \\ G_{3,j}(t) \\ \vdots \\ G_{N-1,j}(t) \\ G_{N,j}(t) \end{pmatrix}$$

Herewith the system of ordinary differential equation can be written as

$$\frac{dU}{dt} = \hat{H}U + G \tag{2.1}$$

$$U(0) = F_I \tag{2.2}$$

2.3 Time Discretization

We can apply any suitable time-step method to (2.1). We choose the θ-method which yields

$$\frac{U^{n+1}-U^n}{\Delta t_n} = \theta(\hat{H}U^{n+1} + G(t^{n+1})) + (1-\theta)(\hat{H}U^n + G(t^n)),$$

where Δt_n denotes the current time step.
This is in fact a reuse of the form

$$\hat{A}_{n+1}U^{n+1} = P_nU^n + V \tag{2.3}$$

where

$$\hat{A}_{n+1} = \frac{1}{\Delta t_n}I - \theta\hat{H},$$

$$P_n = \frac{1}{\Delta t_n}I + (1-\theta)\hat{H}$$

$$V = \theta G(t^{n+1}) + (1-\theta)G(t^n)$$

For $\theta = \frac{1}{2}$ we have a second order account discretization. For other values of $\theta \in [0,1]$ it is first order only. For the analysis it is convenient to rewrite the system to be solved.
Define

$$a_n = \frac{h^2}{\theta\Delta t_n},\ F = -h^2\hat{H},\ f_{n+1} = \frac{h^2}{\theta}(P_nU^n + V),\ A_{n+1} = a_nI + F \tag{2.4}$$

then (2.3) is equal to

$$A_{n+1}U^{n+1} = f_{n+1} \tag{2.5}$$

2.4 The variable time-step algorithm

In this section we will sketch an algorithm to control the step size Δt in our time integration problem.

Algorithm 1.

Choose $\Delta t = \Delta t_0$,

(a) *We will take two steps (algorithm 2.) to solve* (2.3) *from* t *till* $t+2\Delta t$, *First we begin from* t *to* $t+\Delta t$, *Second from* $t+\Delta t$ *to* $t+2\Delta t$ *with step size* Δt, *we have the initial state* u_{n-1}, *then we will compute* u_n, u_{n+1}.

(b) *We compute* f_{n-1}, f_n, f_{n+1} *and* D, *where* $D = -\frac{f_{n+1}-2f_n+f_{n-1}}{12\Delta t^2}$, *where*

$f_n = \hat{H}u_n + g(t^n)$, [$D\Delta t^3$ *approximates local truncation error*].

(c) *If not* $\frac{\varepsilon\Delta t}{4} < |D\Delta t^3| < \varepsilon\Delta t$, *where* ε *is the global error, then we will compute*

new Δt, *by* $\Delta t = \sqrt{\frac{\varepsilon}{2|D|}}$. *The new value of* Δt *has to be not more than two time the old value if we want to avoid rapid changes in* Δt.

(d) *If* Δt *not decreased, then increase* t *by two times the old value of* Δt.

(e) *go to* (a).

Algorithm 2.

Step: (Perform one time step).

(a) *Compute* f_{n+1}, A_{n+1} *from* (2.4).

(b) *Solve* $A_{n+1}U^{n+1} = f_{n+1}$ *from* (2.5).

We assume that we have a good preconditioner P_A for A_n, so the equation $A_nU^n = f_n$ can be solved easily. The goal is to find a way to solve the equation $A_{n+1}U^{n+1} = f_{n+1}$ without constructing a new preconditioner for the matrix A_{n+1}, but rewrite the equation such that we can reuse P_A. $A_{n+1}U^{n+1} = f_{n+1}$ is solved in four ways to be described in the next section.

The result of this algorithm Figure 1 is the solution of *parabolic equation* (two dimension) using *Crank-Nicolson* method ($\theta = \frac{1}{2}$) with variable time-stepping, for two time values (top) and of a specific point in space (bottom left), time-step during integration (bottom right).

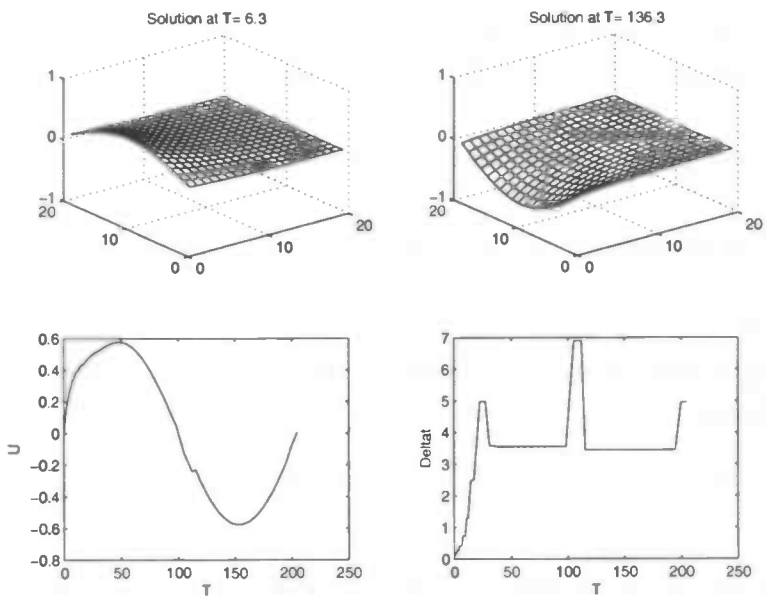

Figure 1: solution of the model problem using *Crank-Nicolson* ($\theta = \frac{1}{2}$) with variable time-stepping, for two time values (top) and of a specific point in space (bottom left), time-step during integration (bottom right)

3 Various rewritings of the linear system and results

For convenience we define the systems $By = g$ and $Ax = f$ which are equivalent to $A_{n+1}U^{n+1} = f_{n+1}$ and $A_nU^n = f_n$, respectively. We will discuss four ways to solve the linear system $By = g$ without constructing a preconditioner for B, but by rewriting B to an expression such that we can reuse the preconditioner of A. In this report we will take as preconditioner of A the inverse of A. The application of this preconditioner means that we have to solve a system with A. For that we use the LL^T factorization of A.

3.1 Introduction

We need to know how we can find the eigenvalues for our matrices and also the condition numbers because the convergence depends on the condition number. First we state an important theorem (*Gerschgorin theorem*) to locate the eigenvalues of the matrix A:

Theorem 1. *(Gerschgorin's circle theorem) Let A be a square complex matrix. Around every element a_{ii} on the diagonal of the matrix, we can draw a circle with radius the sum of the absolute value of the other elements on the same*

row $\sum_{j \neq i} |a_{ij}|$. *The interior of such circles are called* Gerschgorin discs. *Every eigenvalue of* A *lies in one of these Geschgorin discs.*

Theorem 2. *The eigenvalues of a real symmetric matrix are real.*

Corollary 1. *The eigenvalues of the matrix* F *defined in (2.4) are real and on the interval* $0 \leq \lambda \leq 8$.

Definition 1. *If* V *is a vector space over* $\mathbb{C}$, *the spectrum of a linear mapping* $T : V \to V$ *is the set*

$$\sigma(T) = \{\lambda \in \mathbb{C} : \mid T - \lambda I \mid \text{ is not invertible}\},$$

where I *denotes the identity mapping. If* V *is finite dimensional, the spectrum of* T *is precisely the set of its eigenvalues. For infinite dimensional space this is not generally true, although it is true that each eigenvalue of* T *belong to* $\sigma(T)$. *The spectral radius of* T *is*

$$\rho(T) = \sup\{\lambda : \lambda \in \sigma(T)\}.$$

In the following we rewrite the linear system $By = g$ such that the preconditioner of A can be employed and analyse it. The first is simply preconditioning by A. The second is a rewriting of the system. The third is the second with A as preconditioner. In the fourth method we try to minimize the condition number using a free parameter.

Lemma 1. *Let* $h(\lambda)$ *be a rational function. Suppose* λ *is an eigenvalue of* F *then* $h(\lambda)$ *is an eigenvalue of* $h(F)$.

Proof:
Suppose $U^*FU = \Lambda$, where U is the matrix of all eigenvectors of F, and Λ is a diagonal matrix with the eigenvalues of F on the diagonal. Then $U^*(h(F))U = h(U^*FU) = h(\Lambda)$. Hence $h(\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)) = \text{diag}(h(\lambda_1), h(\lambda_2), \ldots, h(\lambda_n))$, giving the eigenvalues of $h(F)$ on the diagonal.

3.2 Method 1

We will use the preconditioner of A to solve the equation $By = \tilde{g}_1$, so we have to solve

$$(aI + F)^{-1}[bI + F]y = \tilde{g}_1, \quad \text{where} \quad \tilde{g}_1 = (aI + F)^{-1}g$$

We can rewrite this system in the following way:

$$h_1(F)y = \tilde{g}_1$$

where

$$h_1(F) = (aI + F)^{-1}[bI + F]$$

Using Lemma 1 the eigenvalues of $h_1(F)$ are:

$$h_1(\lambda) = (a + \lambda)^{-1}[b + \lambda] \tag{3.1}$$

where λ is an eigenvalue of F.
In Figure 3 we have drawn $h_1(\lambda)$ on the interval $[0, 8]$ where the eigenvalues according to Corollary 1 are located. It seems that $h_1(\lambda)$ is decreasing function for $a < b$, and increasing function for $a > b$.

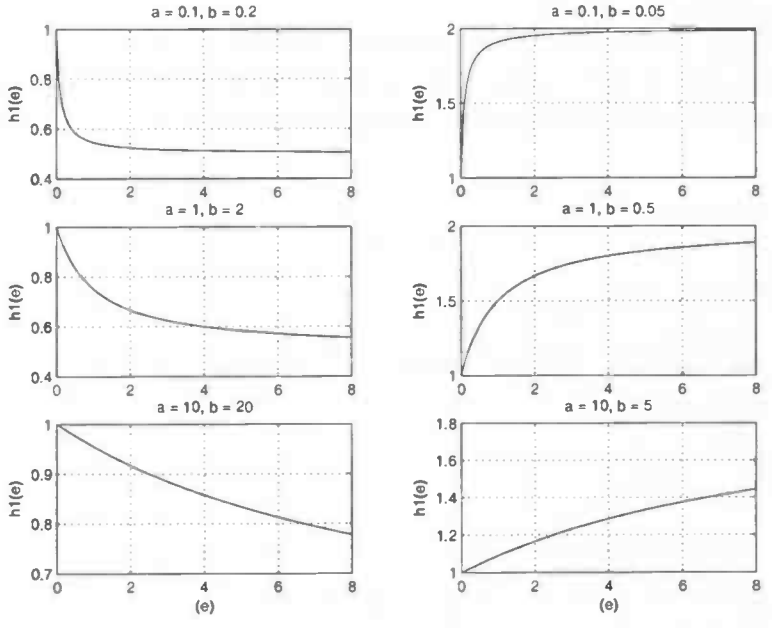

Figure 2: Plots of $h_1(\lambda)$ for various values of a and b

This property is convenient to get an impression of the condition number $\kappa(h_1(F))$ which is bounded by:

$$\kappa(h_1(F)) \leq \hat{\kappa}_1(a, b), \text{ where } \hat{\kappa}_1(a, b) = \frac{\max_{0 \leq \lambda \leq 8} |h_1(\lambda)|}{\min_{0 \leq \lambda \leq 8} |h_1(\lambda)|} \tag{3.2}$$

In order to show that indeed $h_1(\lambda)$ is monotonous, we consider the first derivative of $h_1(\lambda)$:

$$h_1'(\lambda) = \frac{(a+\lambda)-(b+\lambda)}{(a+\lambda)^2} = \frac{(a-b)}{(a+\lambda)^2}$$

Since we take positive time-steps, a and b are positive. Then we have two cases:

- if $a < b$, then $h_1'(\lambda) \leq 0$, we conclude that $h_1(\lambda)$ is a decreasing function, and condition number is bounded by:

$$\kappa(h_1(F)) \leq \frac{h_1(0)}{h_1(8)} = \frac{b(a+8)}{a(b+8)} \tag{3.3}$$

- if $a > b$, then $h_1'(\lambda) \geq 0$, and the condition number will be bounded by:

$$\kappa(h_1(F)) \leq \frac{h_1(8)}{h_1(0)} = \frac{a(b+8)}{b(a+8)} \tag{3.4}$$

The upperbound $\hat{\kappa}_1(a,b)$ is also a good approximation since, for the standard discretization employed, we know that the extremal eigenvalues of F tend to 0 and 8 when the mesh is refined.
In Figure 4, $\hat{\kappa}_1$ the estimate of the condition number of $h_1(F)$, is depicted for some values of a and $\frac{a}{2} \leq b \leq 2a$. We bound b in a since usually in adaptive time-step algorithms one allows only a change by a factor 2.

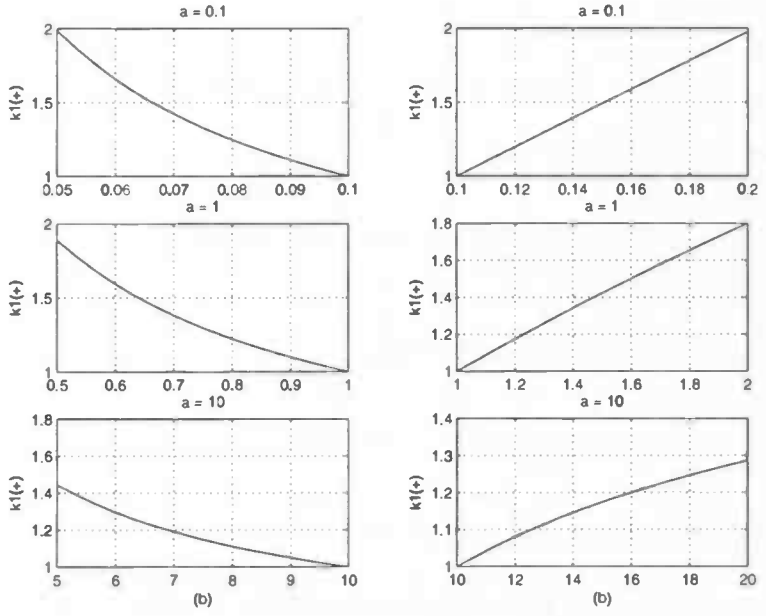

Figure 3: The estimated condition number $\hat{\kappa}_1(a,b)$ for some values of a, $\frac{a}{2} \leq b \leq 2a$

The estimated condition numbers for $b = 2a$ and $b = \frac{a}{2}$ (the end points in the graphs of Figure 4) are:

$$\hat{\kappa}_1(a, \frac{a}{2}) = \frac{a+16}{a+8} \in [1,2] \quad \text{and} \quad \hat{\kappa}_1(a,2a) = \frac{a+8}{a+4} \in [1,2] \tag{3.5}$$

3.3 Method 2

In this section we will try to rewrite the equation $By = g$ such that preconditioning with A^{-1} has more effect. We will consider the rewriting in this section and we will study the preconditioned version in the next section. We will rewrite B in an expression of A as follows:

$$B = (\tfrac{b}{a})[A - F] + F = (\tfrac{b}{a})A + F - (\tfrac{b}{a})F = \tfrac{b}{a}(A + \tfrac{a}{b}F - F)$$

Define $C = (\frac{a}{b} - 1)F$, then

$$By = (\tfrac{b}{a})(A + C)y = g$$

Multiplying the left- and right-hand side by $(I - CA^{-1})$ yields the second system we will study:

$$(\tfrac{b}{a})(A - CA^{-1}C)y = \tilde{g}_2$$

where

$$\tilde{g}_2 = (I - CA^{-1})g$$

Again the system matrix can be expressed in F:

$$(\tfrac{b}{a})(A - CA^{-1}C) = (\tfrac{b}{a})[(aI + F) - (\tfrac{a}{b} - 1)F(aI + F)^{-1}(\tfrac{a}{b} - 1)F]$$

So we can rewrite the above system as follows:

$$h_2(F)y = \tilde{g}_2$$

where we can rewrite $h_2(F)$ to

$$h_2(F) = (\tfrac{b}{a})[(aI + F) - (\tfrac{a}{b} - 1)F(aI + F)^{-1}(\tfrac{a}{b} - 1)F]$$

$$= (\tfrac{b}{a})(aI + F)^{-1}[(aI + F)^2 - F^2(\tfrac{a}{b} - 1)^2]$$

$$= (\tfrac{b}{a})(aI + F)^{-1}[a^2 + 2aF + F^2 - \tfrac{a^2}{b^2}F^2 + \tfrac{2a}{b}F^2 - F^2]$$

$$= (\tfrac{b}{a})(aI + F)^{-1}(\tfrac{a}{b})[\tfrac{ab^2 + 2b^2F - aF^2 + 2bF^2}{b}]$$

$$= (aI + F)^{-1}[\tfrac{(2b - a)F^2 + 2b^2F + ab^2}{b}]$$

So using Lemma 1, the eigenvalues of the matrix $h_2(F)$ are given by:

$$h_2(\lambda) = (a + \lambda)^{-1}[\frac{(2b - a)\lambda^2 + 2b^2\lambda + ab^2}{b}] \tag{3.6}$$

where λ is an eigenvalue of F.
Figure 5 shows $h_2(\lambda)$ for some values of $a = 0.1, 1, 10$ and $b = 2a$ and $b = \frac{a}{2}$, and we see that for these cases $h_2(\lambda)$ is an increasing function.
We can also show that $h_2(\lambda)$ is in general a monotonic increasing function by considering the derivative:

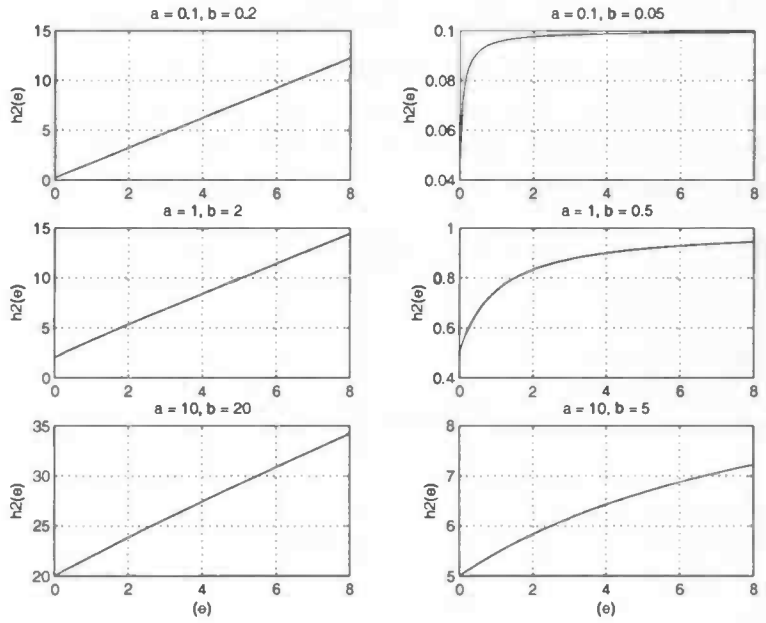

Figure 4: Plots of $h_2(\lambda)$ for various values of a and b

$$h_2'(\lambda) = \frac{b(a+\lambda)[2\lambda(2b-a)+2b^2]-b[(2b-a)\lambda^2+2b^2\lambda+ab^2]}{b^2(a+\lambda)^2} = \frac{\lambda^2(2-\frac{a}{b})+2a\lambda(2-\frac{a}{b})+ab}{(a+\lambda)^2}$$

We consider in our experiments only cases where $(2-\frac{a}{b}) \geq 0$, which makes the counter positive for all $\lambda \geq 0$. Hence, since the denominator is positive, we have $h_2'(\lambda) > 0$.
Hence $h_2(\lambda)$ is increasing function. So $h_2(8)$ is the maximum of $h_2(F)$, and $h_2(0)$ is the minimum:

$$h_2(8) = (a+8)^{-1}[\frac{(2b-a)(8)^2+2b^2(8)+ab^2}{b}] = \frac{64(2-\frac{a}{b})+16b+ab}{(a+8)} = \frac{64(2-\frac{a}{b})+b(a+16)}{(a+8)},$$

$$h_2(0) = (a)^{-1}[\frac{ab^2}{b}] = b$$

An estimate of the condition number is now given by:

$$\hat{\kappa}_2(a,b) = \frac{h_2(8)}{h_2(0)} = \frac{64(2-\frac{a}{b})+b(a+16)}{b(a+8)} = \frac{64(2-\frac{a}{b})}{b(a+8)} + \frac{a+16}{a+8} \qquad (3.7)$$

In Figure 6, $\hat{\kappa}_2$ the estimated condition number depicted for some values of a and b with $\frac{a}{2} \leq b \leq 2a$.
For $b = \frac{a}{2}, 2a$ we have:

$$\hat{\kappa}_2(a,\frac{a}{2}) = \frac{a+16}{a+8} \in [1,2] \quad \text{and} \quad \hat{\kappa}_2(a,2a) = \frac{48}{a(a+8)} + \frac{a+16}{a+8} \in [1,\infty) \quad (3.8)$$

Note that $\hat{\kappa}_2(a,\frac{a}{2}) = \hat{\kappa}_1(a,\frac{a}{2})$ and that $\hat{\kappa}_2(a,2a)\hat{\kappa}_1(a,2a)$. So no advantage with this approach is expected, it is only a starting part for method 3.

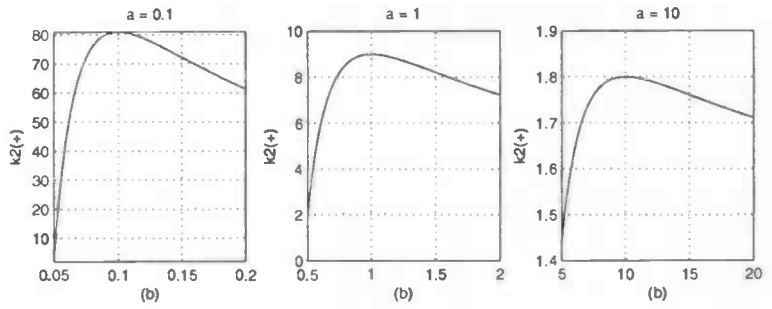

Figure 5: The estimated condition number $\hat{\kappa}_2(a,b)$ for some values of a, $\frac{a}{2} \leq b \leq 2a$

3.4 Method 3

We will study in this section the system of the previous section with preconditioner A:

$$A^{-1}(\tfrac{b}{a})(A - CA^{-1}C)y = A^{-1}\tilde{g}_2$$

Hence, up to a factor, the system matrix is of the form $\frac{b}{a}(I - (A^{-1}C)^2)$. So for $(A^{-1}C)$ small we will have a condition number close to 1. Again we write the system matrix as a function of F.
Clearly:

$$h_3(F) = (aI + F)^{-1}h_2(F)$$

$$h_3(F) = (aI + F)^{-2}[\tfrac{(2b-a)F^2+2b^2F+ab^2}{b}]$$

So using Lemma 1, the eigenvalues of $h_3(F)$ are given by:

$$h_3(\lambda) = (a + \lambda)^{-2}[\frac{(2b - a)\lambda^2 + 2b^2\lambda + ab^2}{b}] \tag{3.9}$$

where λ is an eigenvalue of F.
In Figure 7, $h_3(\lambda)$ is drawn for some values of a, $b = 2a$, and $b = \frac{a}{2}$, and we see that this function is decreasing.
We can show that $h_3(\lambda)$ is in general a monotonic decreasing function by considering the derivative:

$$h_3'(\lambda) = \tfrac{b(a+\lambda)^2[2\lambda(2b-a)+2b^2]-[2b(a+\lambda)][(2b-a)\lambda^2+2b^2\lambda+ab^2]}{b^2(a+\lambda)^4}$$

$$= \tfrac{-\lambda(2a^2-4ab+2b^2)}{b(a+\lambda)^3} = \tfrac{-2\lambda(a^2-2ab+b^2)}{b(a+\lambda)^3} = \tfrac{-2\lambda(a-b)^2}{b(a+\lambda)^3}$$

Hence, for the considered values of a, b and λ (all being positive) $h_3'(\lambda) < 0$, and therefore $h_3(\lambda)$ is a decreasing function. So $h_3(0)$ is the maximum of $h_3(F)$ and $h_3(8)$ is the minimum:

$$h_3(8) = (a+8)^{-2}\left[\frac{(2b-a)(8)^2+2b^2(8)+ab^2}{b}\right] = \frac{64(2-\frac{a}{b})+16b+ab}{(a+8)^2},$$

$$h_3(0) = (a)^{-2}\left[\frac{ab^2}{b}\right] = \frac{b}{a}$$

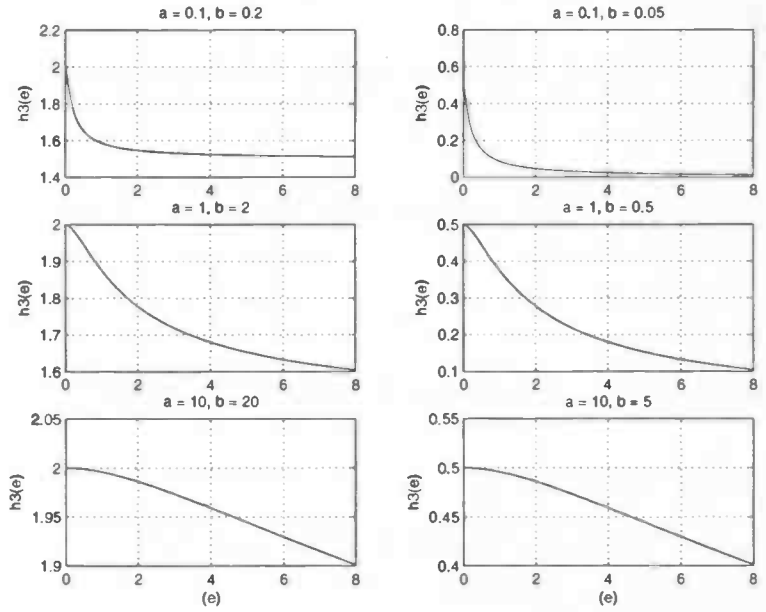

Figure 6: Plots of $h_3(\lambda)$ for various values of a and b

An estimate of the condition number is now given by:

$$\hat{\kappa}_3(a,b) = \frac{h_3(0)}{h_3(8)} = \frac{\frac{b}{a}}{\frac{64(2-\frac{a}{b})+16b+ab}{(a+8)^2}} = \frac{b(a+8)^2}{64a(2-\frac{a}{b})+16ab+a^2b} \tag{3.10}$$

We see in Figure 8 the estimated condition number depicted for some values of a and b with $\frac{a}{2} \leq b \leq 2a$.
For $b = \frac{a}{2}, 2a$ we have:

$$\hat{\kappa}_3(a,\frac{a}{2}) = \frac{(a+8)^2}{a(16+a)} \in [1,\infty) \quad \text{and} \quad \hat{\kappa}_3(a,2a) = \frac{(a+8)^2}{(a+4)(a+12)} \in [1,\frac{4}{3}] \tag{3.11}$$

Note that $\hat{\kappa}_3(a,2a) < \hat{\kappa}_1(a,2a)$.

3.5 Method 4

In this section we will study (using Mathematica) the following preconditioned system:

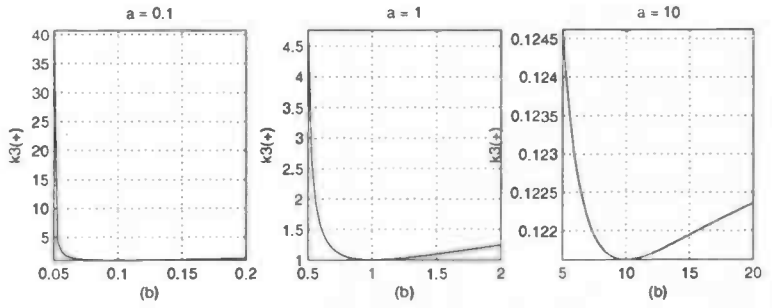

Figure 7: The estimated condition number $\hat{\kappa}_3(a,b)$ for some values of a, $\frac{a}{2} \leq b \leq 2a$

$$(A^{-1} - \gamma I)By = g_4$$

where

$$g_4 = (A^{-1} - \gamma I)g$$

Again we can express the system matrix in F:

$$(A^{-1} - \gamma I)B = h_4(F)$$

where

$$h_4(F) = ((aI + F)^{-1} - \gamma I)(bI + F)$$

The freedom γ is chosen such that $h_4(\lambda)$ is as constant as possible on the interval $[0, 8]$. This leads to the requirement that $h_4(0) = h_4(8)$ which results in $\gamma = \frac{a-b}{a(8+a)}$. So $h_4(\lambda)$ assumes the form:

$$h_4(\lambda) = \frac{b+\lambda}{a+\lambda} - \frac{(a-b)(b+\lambda)}{a(8+a)} \tag{3.12}$$

In Figure 8, $h_4(\lambda)$ is shown and we see that there is only one internal extreme which is a maximum for $a > b$ and a minimum for $a < b$. The extreme is found for $\lambda = -a + \sqrt{8a + a^2}$ with value:

$$\frac{(-a + b + \sqrt{8a + a^2})^2}{8a + a^2} \tag{3.13}$$

For $a > b$ we can see that $h_4(0)$ is the minimum eigenvalue of $h_4(F)$, then the condition number for $a > b$ is bounded by:

$$\kappa(h_4(F)) \leq \hat{\kappa}_4(a,b) = \frac{(-a + b + \sqrt{8a + a^2})^2}{b(8+b)} \tag{3.14}$$

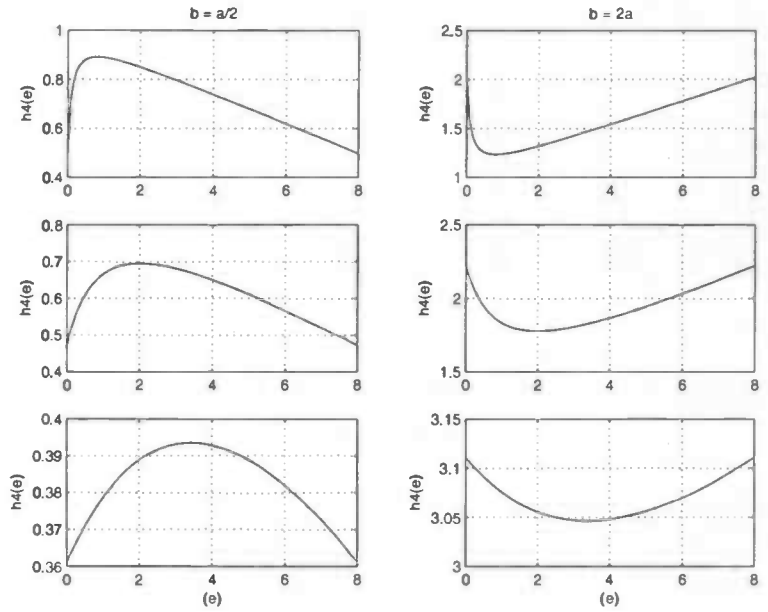

Figure 8: $h_4(\lambda)$ for various values of a, $b = 2a$ and $b = \frac{a}{2}$

For $a < b$ the estimated condition number is just the reciprocal of the one for $a > b$.
In figure 9, $\hat{\kappa}_4$ the estimated condition number is depicted for some values of a and b in $\frac{a}{2} \leq b \leq 2a$. Note that $\hat{\kappa}_4(a, \frac{a}{2}) < \hat{\kappa}_3(a, \frac{a}{2})$ and that $\hat{\kappa}_4(a, b) < \hat{\kappa}_1(a, b)$ for $a > 0$ and $\frac{a}{2} \leq b \leq 2a$.

3.6 Condition numbers and iteration counts

In this section, we compare condition numbers and iteration counts. We see in Table 1 the estimated condition number for all methods for some values of a and b. In Table 2 we find the iteration numbers of all methods using MATLAB's CG to solve the system $By = g$. The initial guess is zero, g is a random vector, and the iteration is stopped if the residual is less than 1.10^{-6}.

We see the estimated condition number of Method 2 is always larger than that of Method 1, as already derived in general in Section (3.3). This is expressed in the number of iterations shown in Table 2. Method 3 gives condition numbers smaller than those of Method 1 if $a < b$ as noticed in Section (3.4). However, for $a > b$ it is much worse than that of Method 1 for a, b small, so it is not appropriate to use it in that case.

The estimated condition number of Method 4 is smaller than that of Method 1,

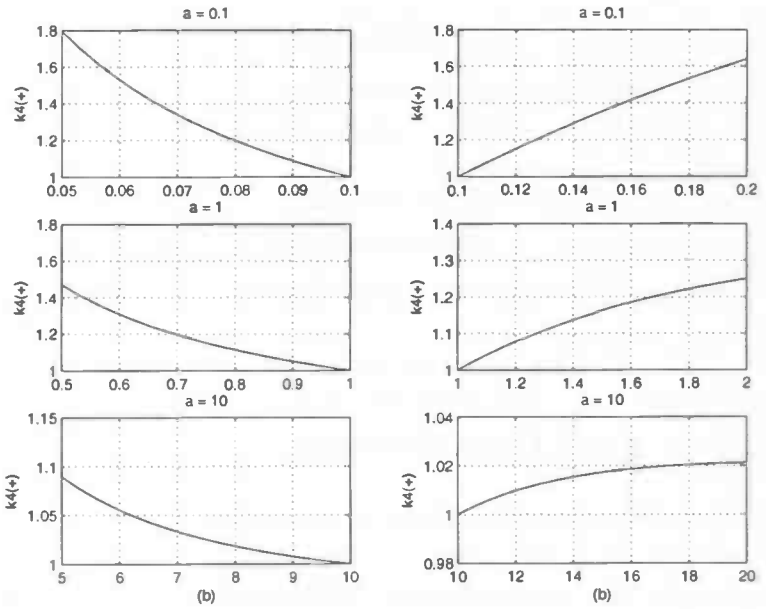

Figure 9: The estimated condition number $\hat{\kappa}_4(a,b)$ for some values of a, $\frac{a}{2} \leq b \leq 2a$

which is not surprising since Method 1 is a special case in the class of considered methods to derive Method 4. Method 4 will be the one with the smallest condition number in this class. We see that smaller condition numbers do not always result in a smaller number of iterations. This might be due to the initial solution and right-hand side.

a	b	$\kappa(h_1(F))$	$\kappa(h_2(F))$	$\kappa(h_3(F))$	$\kappa(h_4(F))$
0.1	0.2	1.9756	61.2469	1.3225	1.64
0.1	0.05	1.9877	1.9877	40.7516	1.7950
1	2	1.8000	7.2222	1.2462	1.25
1	0.5	1.8889	1.8889	4.7647	1.4705
10	20	1.2857	1.7111	1.0519	1.0212
10	5	1.4444	1.4444	1.2462	1.0897

Table 1: The condition number κ of $h_1(F)$, $h_2(F)$, $h_3(F)$ and $h_4(F)$ for $a = 0.1, 1, 10$ and $\frac{a}{2} \leq b \leq 2a$

a	b	$y_1(iter)$	$y_2(iter)$	$y_3(iter)$	$y_4(iter)$
0.1	0.2	5	25	5	7
0.1	0.05	4	4	16	7
1	2	6	15	5	5
1	0.5	6	6	12	6
10	20	5	7	4	3
10	5	6	6	5	4

Table 2: The iteration number for $n = 10$

4 Discussion and Conclusion

Method 4 shows that if we want to reduce the condition number then that can be obtained by using a free parameter. It would be of interest to generalize this idea. For example, the preconditioner of Method 3 is simply the product $A^{-1}(I - CA^{-1})$ and, since C is just a factor times F, it belongs to the class of preconditioners of the form $A^{-1} - \gamma A^{-1}FA^{-1}$. Method 1 is in this class for $\gamma = 0$, the method with the smallest condition number in this class will be better than both Methods 1 and 3.

It is doubtful whether the extra effort that is put in a preconditioner is payed back in a sufficient decrease of the number of iterations. We see in Method 3 an extra solve, which often dominates the cost, is necessary per iteration. We hope that the number of iterations is reduced by a factor 2. This is in the cases where it functions well, $a < b$ clearly not the case.

Maybe an ideal method in the above mentioned class will do better. Method 4 gives a reduction of the condition numbers with hardly no extra costs, so this will do better in general than Method 1. In practice we deal with an approximation of the inverse of A due to an incomplete factorization. Then the optimization process based on the inverse of A only leads to reasonable results if the incomplete factorization is accurate. In that case it does not make sense to consider sophisticated approximation with more free parameters. So only simple forms are relevant.

Our conclusion in this study is that computation time can be brought down by doing something just slightly less trivial than just preconditioning with the previous matrix. It remains however to be investigated whether this holds also if we consider incomplete factorization.

References

[1] John C. Strikwerda, *Finite Difference Schemes & Partial Differential Equations*, University of Wisconsin-Madison, 1989.

[2] Anne Greenbaum, *Iterative Methods for Solving Linear Systems*, University of Washington, Seattle, Washington, 1997.

[3] Kendall E. Atkinson, *An Introduction to Numerical Analysis*, University of Iowa, 1988.

[4] Joel N. Franklin, *Matrix Theory*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1968.