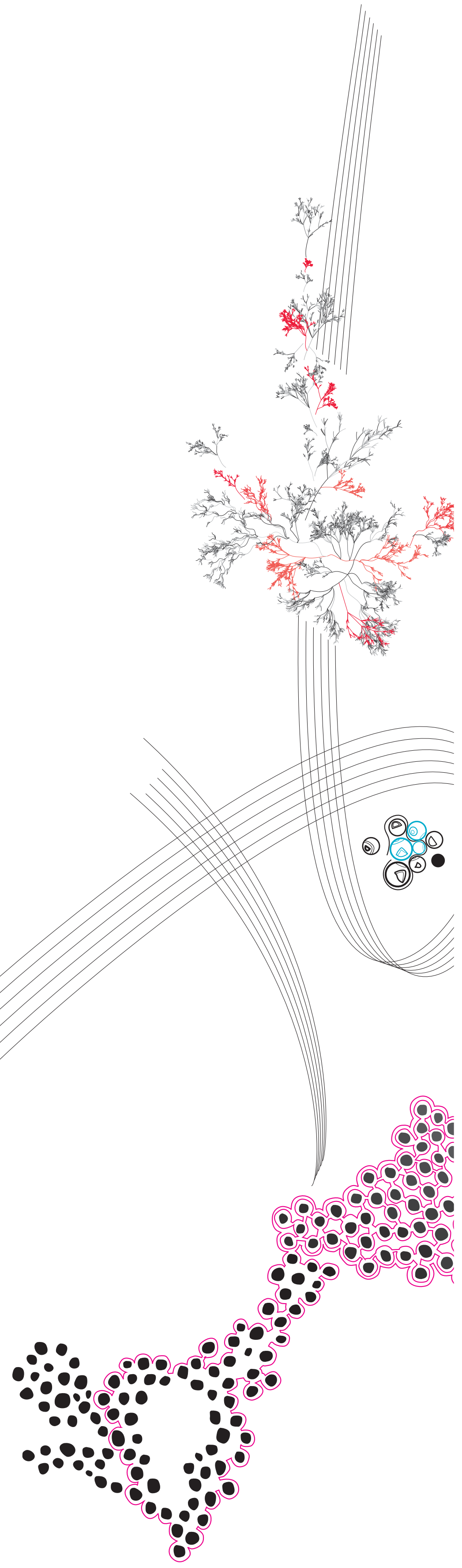

BSc Thesis Applied Mathematics

Accelerating iterative methods for the anisotropic radiative transfer equation using Anderson Acceleration

Daan Velthuis

Supervisor: M. Schlottbom

August, 2023

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I want to thank Matthias Schlottbom for proposing this topic and his guidance and feedback during my research.

Accelerating iterative methods for the anisotropic radiative transfer equation using Anderson Acceleration

Daan T. J. Velthuis[*]

August, 2023

Abstract

This paper concerns the iterative solution of the linear system which comes from the discretization of the anisotropic radiative transfer equation. The goal is to create a fast converging method using Anderson Acceleration. With Anderson Acceleration a method can converge within fewer iterations or it might turn a diverging method into a converging one. The results are shown for numerical examples, which showcase the effect of Anderson Acceleration. The mixed-system with subspace correction and Anderson Acceleration converges with a low computation time for all our examples, making it a great candidate if a proof can be given in the future that the method will converge for all values for the parameters.

Keywords: Anistotropic radiative transfer, Iterative methods, Anderson Acceleration

1 Introduction

In this paper we will discuss methods to solve the anisotropic radiative transfer equation. This equation describe the way radiation waves move through a turbid medium, it describes the streaming, absorption and scattering. It has many applications, for example heat transfer [4] and medical imaging [6]. The equation is as follows:

$$s \cdot \nabla_r u(s,r) + \sigma_t(r) u(s,r) = \sigma_s(r) \int_S k(s \cdot s') u(s', r) ds' + q(s,r), \text{ where} \tag{1}$$

$$k(s \cdot s') = \frac{1}{4\pi} \frac{1-g^2}{[1 - 2g(s\cdot s') + g^2]^{3/2}} \tag{2}$$

Here $u(s,r)$ is the intensity we are after. It depends on both $r \in R$, which is the position, and $s \in S$, which is a unit vector that indicates the direction of propagation. The scattering rate is given by σ_s. The loss of intensity is given by the total attenuation coefficient $\sigma_t = \sigma_a + \sigma_s$, where σ_a is the absorption rate. We define $c = \|\sigma_s/\sigma_t\|_\infty$. The function $k(s \cdot s')$ is the scattering function, which has a constant g. We consider only $0 \leq g < 1$. For $g = 0$, we have isotropic scattering and for g close to 1 the scattering is forward peaked. Lastly, $q(s,r)$ characterizes the internal sources of radiation.

We will work with the linear system obtained by the mixed finite element discretization of equation (1) [3]:

$$\begin{bmatrix} R + M^+ & -A^T \\ A & M^- \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \end{bmatrix} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \end{bmatrix} + \begin{bmatrix} q^+ \\ q^- \end{bmatrix} \tag{3}$$

[*]Email: d.t.j.velthuis@student.utwente.nl

Here $+$ indicate the even part and $-$ the odd part. The matrix R accounts for the boundary conditions. The matrices M^+ and M^- are mass matrices and K^+ and K^- represents scattering. Lastly A discretizes $s \cdot \nabla_r$.

This system is too large to solve with direct methods, so iterative methods are used. There are already multiple methods in place, each with their advantages and shortcomings. Firstly, the preconditioned even-parity system is provably robustly convergent, but is computationally expensive per iteration. Secondly, the mixed system also is provable convergent and has a low computation time per iteration. However, the number of iterations goes up quickly when c approaches 1. The last method which we will discuss is the mixed system with subspace correction. The correction improves the convergence behaviour when $c \approx 1$, but the system might diverge for certain parameters.

The goal of this research is to create an iterative method which incorporates all positives from the existing methods. We want to have a method which converges to a solution fast in terms of computation time for all values of the parameters. To achieve this we will use Anderson Acceleration.

2 Anderson acceleration

The fixed-point iteration is a method used to find a fixed point of a function. In other words, it approximates a solution for the equation $f(x) = x$ by starting with an initial guess x_0 and calculates the sequence $x_{k+1} = f(x_k), k = 0, 1, 2, \ldots$, which converges to the true fixed point $\tilde{x}$ if f is a contraction. The speed with which this sequence converges to the solution is the convergence rate. Anderson acceleration is a method used for increasing this convergence rate.

2.1 Method

We will now look at Anderson acceleration. We have a function $f : \mathbb{R}^n \to \mathbb{R}^n$ for which we want to calculate the fixed point, so we want to solve $f(x) = x$. We define $g(x) = f(x) - x$, which gives the residual and denote $f_k = f(x_k)$ and $g_k = g(x_k)$. The algorithm has a parameter m, which is the number of last estimates taken into consideration when calculating the next value of x. The method works as follows:

Algorithm 1 Anderson Acceleration

```
Start with initial guess $x_0$
$x_1 = f(x_0)$
Set $k = 1$
while $\|g(x_k)\|$ is not sufficiently small do
    $m_k = \min(m, k)$
    $G_k = [g_{k-m_k}, \ldots, g_k]$
    $\beta = \operatorname{argmin}_{\alpha \in A_k} \|G_k \alpha\|$, where $A_k = \{\alpha \in \mathbb{R}^{m_k} : \sum_{i=0}^{m_k} \alpha_i = 1\}$
    $x_{k+1} = \sum_{i=0}^{m_k} \beta_i f_{k-m_k+i}$
    k = k+1
end while
```

2.2 Derivation

Solving $f(x) = x$ is the same problem as solving $g(x) = 0$, therefore we could see this problem as a minimization problem; minimize $\|g(x)\|$.

As stated above, the fixed-point iteration chooses the next value of x by evaluating the function at the previous value of x, so $x_{k+1} = f(x_k)$. However, when using Anderson acceleration x_{k+1} will be a linear combination of the $m_k + 1$ last values of the sequence x_k, where the coefficients sum to one, so $\hat{x}_{k+1} = X_k\beta$, where $X_k = [x_{k-m_k}, \ldots, x_k]$. We want to choose β such that it minimizes $\|g(\hat{x}_{k+1})\|$. If we have β, we can calculate our next iterate. We will rewrite the problem.

$$g(\hat{x}_{k+1}) = g(X_k\beta) = g\left(\sum_{i=0}^{m_k} \beta_i x_{k-m_k+i}\right)$$

We will now use the first order Taylor approximation of $g(x) \approx ax + b$.

$$\begin{aligned} g\left(\sum_{i=0}^{m_k} \beta_i x_{k-m_k+i}\right) &\approx a\sum_{i=0}^{m_k} \beta_i x_{k-m_k+i} + b \\ &= \sum_{i=0}^{m_k} \beta_i a x_{k-m_k+i} + \sum_{i=0}^{m_k} \beta_i b \\ &= \sum_{i=0}^{m_k} \beta_i (a x_{k-m_k+i} + b) \\ &\approx \sum_{i=0}^{m_k} \beta_i g(x_{k-m_k+i}) \\ &= G_k\beta \end{aligned}$$

We will apply Anderson Acceleration on a linear system, so our $g(x)$ is affine linear, meaning in our case we do not need the approximation, but it is an equality.

We have now translated our problem of choosing β to choosing it in such a way that it minimizes over $\|G_k\beta\|$. Our final step is to include the fixed-point iteration step as well, this means we set the true next value of x to $f(\hat{x}_{k+1})$. We can calculate it in the following way, where we use the similar linear approximation for f as we did for g.

$$x_{k+1} := f(\hat{x}_{k+1}) = f(X_k\beta) \approx F_k\beta, \text{ where } F_k = [f_{k-m_k}, \ldots, f_k]$$

3 Applying Anderson acceleration

3.1 Splitting methods

The condition number of a problem indicates how much sensitive the problem is to small errors. A low condition number means the problem is well-conditioned, meaning not much precision is lost during the calculations and this makes the problem suitable for iterative solvers. A preconditioner can be applied to the problem to reduce the condition number. A wide class of iterative solution techniques is based on splitting matrix A into $M - N$. Using this we see $Ax = b$ is equivalent to $x = x + M^{-1}(b - Ax)$. We see that this is already a fixed-point problem with $f(x) = x + M^{-1}(b - Ax)$.

Matrix M should now be chosen in a smart way, it should resemble A, but it has to be computationally cheap to invert. Choosing M to be the diagonal of A is called the Jacobi preconditioner and choosing M to be the lower triangular matrix, obtained by setting all entries above the diagonal to zero in A, is the Gauss-Seidel preconditioner.[1]

can rewrite the equation by introducing our iteration matrix $T = I - M^{-1}A$.

$$x = M^{-1}b + (I - M^{-1}A)x = M^{-1}b + Tx \tag{4}$$

Transforming this into an iterative scheme we obtain:

$$x_{k+1} = M^{-1}b + Tx_k. \tag{5}$$

3.2 Convergence

To show the method actually converges, we look at the error at each iteration: $e_k = x - x_k$. We want to have $e_k \to 0$ as $k \to \infty$. We can subtract our iterative scheme (5) from equation (4) to get the following

$$\begin{aligned} x - x_{k+1} &= M^{-1}b + Tx - (M^{-1}b + Tx_k), \text{ i.e.} \\ e_{k+1} &= Te_k \end{aligned}$$

This means $e_k = T^k e_0$. Taking the norm on both sides gives us an inequality which shows the upper-bound of the error.

$$\|e_k\| = \|T^k e_0\| \leq \|T\|^k \|e_0\|$$

This shows the error will converge to 0 if $\|T\| < 1$.[1]

3.3 Termination criterion

Before we put the different methods to the test, we first need to decide when the process should stop. We want to have the error to be smaller than a chosen absolute tolerance t. A common choice is to check the difference between consecutive iterations and stop when the difference becomes sufficiently small. This raises a problem when the rate of convergence is very small, i.e. $\|T\| \approx 1$, because the solution may not have been reached yet, but the update is smaller than the chosen absolute tolerance. This is why we need to include the norm of T when determining when the difference $\|x_k - x_{k-1}\|$ is actually sufficiently small.

$$\begin{aligned} \|e_{k+1}\| \leq \|T\|\|e_k\| = \|T\|\|x - x_k\| &= \|T\|\|x - x_{k+1} + x_{k+1} - x_k\| \\ &\leq \|T\| \left(\|x - x_{k+1}\| + \|x_{k+1} - x_k\|\right) \\ &= \|T\|\|e_{k+1}\| + \|T\|\|x_{k+1} - x_k\| \end{aligned}$$

This can be rewritten to

$$\|e_{k+1}\| = \frac{\|T\|}{1 - \|T\|}\|x_{k+1} - x_k\|$$

This means if we want to have $\|e_{k+1}\| \leq t$, than $\|x_{k+1} - x_k\| \leq \frac{1-\|T\|}{\|T\|}t$, which could become very small if $\|T\|$ is close to 1.[1]

3.4 Test case

To see the effect of applying Anderson acceleration, we will consider a small example. We will look at a second-order one-dimensional boundary value problem with Dirichlet boundary conditions:

$$\begin{aligned} -u'' + u &= f \text{ on the interval (0,1)} \\ u(0) &= 0 \\ u(1) &= 0, \end{aligned}$$

where f is given and u is unknown. We discretize this problem by splitting the interval $[0,1]$ in N subintervals, all of length $h = \frac{1}{N}$. We define points on the interval $x_i = ih, i = 0, \ldots, N$ and $u_i \approx u(x_i)$. We now take the central difference approximation of u, $u_i'' \approx \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}, i = 1, \ldots, N-1$. This way we can rewrite our equation to a linear system.

$$(A + I)U = F$$

Here A is a $N-1 \times N-1$ tridiagonal matrix with $\frac{2}{h^2}$ on the main diagonal and $-\frac{1}{h^2}$ on both minor diagonals. $U = (u_1, \ldots u_{N-1})^T$ and $F = (f(x_1), \ldots, f(x_2))^T$.

We have that $A+I$ is strictly diagonally dominant, which is a necessary condition to ensure convergence for the Jacobi and Gauss-Seidel iteration. [1] The iteration matrix for this problem is $T = M^{-1}(A+I)$.

3.5 Results

3.5.1 Terminating criterion

We have calculated $\frac{1-\|T\|}{\|T\|}$ for different values of N for both preconditioners to see the difference.

Table 1: $\frac{1-\|T\|}{\|T\|}$ for both preconditioners, where T is the iteration matrix in the boundary value problem.

N	Jacobi	Gauss-Seidel
10	5.672e-2	9.425e-2
20	1.373e-2	2.535e-2
40	3.406e-3	6.573e-3
80	8.497e-4	1.672e-3
160	2.123e-4	4.213e-4
320	5.307e-5	1.057e-4
640	1.327e-5	2.649e-5
1280	3.317e-6	6.628e-6
2560	8.293e-7	1.658e-6

In table 1 we can see that for both methods $\frac{1-\|T\|}{\|T\|}$ gets approximately 4 times as small as N gets twice as large. This means we expect to need 4 times as many iterations to reach the same precision if N doubles in size.

3.5.2 Boundary value problem

Moreover, we have also compared the number of iterations needed to solve the problem by the standard fixed point iteration and when using Anderson acceleration. We have chosen an absolute tolerance of 10^{-6} and $u(x) = x^5 - x^3 + x^2 - x$, which means $f(x) = -20x^3 + 6x - 2$. The iteration count can be seen in table 2, dashed indicate that the method did not converge within 300,000 iterations.

TABLE 2: Iteration count for solving boundary value problem with $u(x) = x^5 - x^3 + x^2 - x$ and tolerance is 10^{-6}.

N	Jacobi	Jacobi (AA)	Gauss-Seidel	Gauss-Seidel (AA)
10	248	50	127	18
20	1026	304	519	37
40	4217	1323	2118	85
80	17288	4502	8661	197
160	70795	16128	35431	350
320	289723	29552	144926	1030
640	-	54325	-	3777
1280	-	119140	-	6119

We have done the same for a different choice of function, namely $u(x) = e^x - (1 + (e-1)x)$ and $f(x) = -e^x$ and the results can be seen in table 3.

TABLE 3: Iteration count for solving boundary value problem with $u(x) = e^x - (1 + (e-1)x)$ and tolerance is 10^{-6}.

N	Jacobi	Jacobi (AA)	Gauss-Seidel	Gauss-Seidel (AA)
10	238	44	122	12
20	986	195	497	34
40	4056	926	2035	65
80	16643	3195	8334	203
160	68213	8441	34130	447
320	279394	17166	139741	1022
640	-	39484	-	4751
1280	-	96444	-	14071

Without Anderson Acceleration we see that the number of iterations roughly quadruples each time N doubles for both methods in both cases. When Anderson Acceleration is applied the number of iterations goes down by a significant factor. It only roughly doubles when N is doubled as well. For high values of N the iteration is more than 100 times smaller for the Gauss-Seidel method.

3.6 Anderson acceleration with an eigenvector

While running tests, a special case came up where the size of A did not influence the numbers of iterations needed, the solution was always found after one iteration. It turned out that here we were solving $Ax = b$ with b being an eigenvector of A with corresponding eigenvalue λ. We will show that indeed the solution will always be found after one iteration when Anderson acceleration is applied to the fixed-point problem $x = x + b - Ax$.

Theorem: When Anderson Acceleration is applied to the system $Ax = b$, x_2 will be a solution, if $x_0 = \mathbf{0}$ and b is an eigenvector of A with eigenvalue λ.

Proof: We start of with $x_0 = \mathbf{0}$ and thus $x_1 = b$. Using these values we can calculate that $G_1 = [b, (1-\lambda)b]$. Now we will choose β, such that

$$\begin{aligned}\beta &= \arg\min_{\alpha \in A_1} \|G_1\alpha\|, \text{ where } A_1 = \{\alpha \in \mathbb{R}^2 : \alpha_0 + \alpha_1 = 1\} \\ &= \arg\min_{\alpha_0+\alpha_1=1} \|(1-\alpha_1\lambda)b\|.\end{aligned}$$

We see for $\alpha_1 = \frac{1}{\lambda}$ the value becomes zero, which is the minimal value of a norm. This means we have the vector $\beta = \left(\frac{\lambda-1}{\lambda}, \frac{1}{\lambda}\right)$. This can be used to calculate the next iterate,

$$x_2 = \beta_0 f_0 + \beta_1 f_1 = \frac{1}{\lambda} b.$$

We see this is indeed a solution to our equation $x = x + b - Ax$, which proves our theorem. ■

4 Anisotropic radiative transfer system

Recall from the introduction that we want to apply Anderson acceleration to the more complex system

$$\begin{bmatrix} R + M^+ & -A^T \\ A & M^- \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \end{bmatrix} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} \begin{bmatrix} u^+ \\ u^- \end{bmatrix} + \begin{bmatrix} q^+ \\ q^- \end{bmatrix}. \tag{6}$$

We will consider the multiple methods named in the introduction and compare the computation time and iteration count for different parameters, both with and without Anderson Acceleration applied.

4.1 Approaches

First we explain the derivation of the methods and later on we will add preconditioners to try to lower the numbers of iterations needed.

4.1.1 Even-parity system

By taking the Schur complement of (6) we get

$$Eu^+ = K^+u^+ + q,$$

where $E = A^T(M^- - K^-)^{-1}A + M^+ + R$ and $q = q^+ + A^T(M^- - K^-)^{-1}q^-$. In iterative form this will be

$$u^+_{k+1} = u^+_k - (E - K^+)u^+_k - q. \tag{7}$$

We see this system uses the mentioned inverse of $(M^- - K^-)$, which is computationally expensive to calculate. Once we have calculated u^+, we can use it to calculate u^-.

4.1.2 Iteration on mixed system

The mixed system approach is a splitting method as in chapter 3.1. It starts with turning (6) into an iterative scheme:

$$\begin{bmatrix} R+M^+ & -A^T \\ A & M^- \end{bmatrix} \begin{bmatrix} u_{k+1}^+ \\ u_{k+1}^- \end{bmatrix} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} \begin{bmatrix} u_k^+ \\ u_k^- \end{bmatrix} + \begin{bmatrix} q^+ \\ q^- \end{bmatrix}. \tag{8}$$

We see this is of the form $Mx_{k+1} = Nx_k + b$.

This method will first calculate u_{k+1}^+ using the Schur complement. Secondly it will use this found value to calculate u_{k+1}^- as well. Lastly, it will use the entire vector $u_{k+1} = (u_{k+1}^+, u_{k+1}^-)$ for Anderson acceleration, opposed to the previous method, which only considered u_{k+1}^+ at each iteration.

The Schur complement of (8) gives us the equation for u_{k+1}^+ and u_{k+1}^-:

$$\begin{aligned} (R+M^+ + A^T(M^-)^{-1}A)u_{k+1}^+ &= f_k^+ + A^T(M^-)^{-1}f_k^- \\ u_{k+1}^- &= (M^-)^{-1}(f_k^- - Au_{k+1}^+), \end{aligned}$$

where $f_k^+ = K^+ + u_k^+ + q^+$ and $f_k^- = K^- + u_k^- + q^-$.

We can prove that this system has a convergence rate c. We start by converting (8) into error form by subtracting it of (6):

$$\begin{bmatrix} R+M^+ & -A^T \\ A & M^- \end{bmatrix} \begin{bmatrix} e_{k+1}^+ \\ e_{k+1}^- \end{bmatrix} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} \begin{bmatrix} e_k^+ \\ e_k^- \end{bmatrix}.$$

We rename the blockdiagonal matrix with K^+ and K^- to K. Next we multiply both sides by e_{k+1}^T:

$$\begin{bmatrix} (e_{k+1}^+)^T & (e_{k+1}^-)^T \end{bmatrix} \begin{bmatrix} R+M^+ & -A^T \\ A & M^- \end{bmatrix} \begin{bmatrix} e_{k+1}^+ \\ e_{k+1}^- \end{bmatrix} = e_{k+1}^T K e_k.$$

This can be simplified to

$$(e_{k+1}^+)^T(R+M^+)e_{k+1}^+ + (e_{k+1}^-)^T M^- e_{k+1}^- = e_{k+1}^T K e_k.$$

Matrix R is a symmetric positive semidefinite matrix, which means $(e_{k+1}^+)^T R e_{k+1}^+ \geq 0$, we also rename the blockdiagonal matrix with M^+ and M^- to M:

$$e_{k+1}^T M e_{k+1} \leq e_{k+1}^T K e_k.$$

We know from the properties of the anisotropic radiative transfer equation that $x^T K x \leq c x^T M x$. That gives us the following inequalities using the Cauchy-Schwarz inequality:

$$\begin{aligned} e_{k+1}^T M e_{k+1} &\leq c(e_{k+1}^T M e_k) \\ \|e_{k+1}\|_M^2 &\leq c(\|e_k\|_M \|e_{k+1}\|_M) \\ \|e_{k+1}\|_M &\leq c\|e_k\|_M \end{aligned}$$

This concludes the proof that the method has a contraction rate capped by c, meaning it will always converge as $c = \frac{\sigma_s}{\sigma_a + \sigma_s} < 1$.

4.2 Applying preconditioners

4.2.1 Even-parity system

Two preconditioners P_1 and P_2 will be applied to (7), resulting in

$$u^+_{k+1} = u^+_k - P_2 P_1((E - K^+)u^+_k + q).$$

This is done in the same way as in the paper on robustly convergent methods[2]. The first preconditioner P_1 is chosen to resemble E^{-1}. It can not be calculated directly, so another iterative approach is taken, which is computationally expensive. This preconditioner ensures a contraction rate of c.

The second preconditioner P_2 aims to improve the convergence behaviour for $c \approx 1$ using subspace correction [5]. The system is projected onto a smaller suitable subspaces using Galerkin projection.

This method is provably convergent. [5] Moreover, the method is robust, meaning that the number of iterations needed does not depend on the size of the system.

4.2.2 Mixed system

Before we get to our preconditioner, we first do some manipulations on our system. First we take (8), however we do not directly calculate u_{k+1}, but first an intermediate step $u_{k+1/2}$.

$$\begin{bmatrix} R + M^+ & -A^T \\ A & M^- \end{bmatrix} u_{k+1/2} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} u_k + \begin{bmatrix} q^+ \\ q^- \end{bmatrix}. \tag{9}$$

We introduce $e_k := u - u_k$ and we subtract (9) from our original system (6):

$$\begin{bmatrix} R + M^+ & -A^T \\ A & M^- \end{bmatrix} e_{k+1/2} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} e_{k+1/2} + \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} (e_k - e_{k+1/2}).$$

We know $e_k - e_{k+1/2} = u - u_k - u + u_{k+1/2}$, so we can rewrite our system to

$$\begin{bmatrix} R + M^+ & -A^T \\ A & M^- \end{bmatrix} e_{k+1/2} = \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} e_{k+1/2} + \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} (u_{k+1/2} - u_k).$$

If we know could calculate our error $e_{k+1/2}$, we can use our value for $u_{k+1/2}$ perfectly to get the exact value of u. Unfortunately, we see that calculating the error is equivalent to solving the entire system, which was our goal in the first place. What we can do, is approximate our error with a correction value u_c and use it to get a better value for our next iterate.

To get to this approximation we will apply subspace correction as above, in [5]. We construct a block-diagonal matrix:

$$P = \begin{bmatrix} P^+ & \\ & P^- \end{bmatrix}.$$

This matrix, when applied, will lower the dimensions significantly. We will project the entire system, which creates a way smaller and approachable linear system, which we can simply solve using traditional methods. Once the solution is found, we use it to get a solution in our original large space again, which approximates the real solution. P has to be chosen properly to ensure a unique solution.

We multiply our matrices from the left side with P^T and from the right with P to create a smaller system. This smaller system will have v_c as its variable:

$$P^T \begin{bmatrix} R+M^+ & -A^T \\ A & M^- \end{bmatrix} P \begin{bmatrix} v_c^+ \\ v_c^- \end{bmatrix} = P^T \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} P \begin{bmatrix} v_c^+ \\ v_c^- \end{bmatrix} + P^T \begin{bmatrix} K^+ & \\ & K^- \end{bmatrix} (u_{k+1/2} - u_k).$$

This can be rewritten to

$$\begin{bmatrix} (P^+)^T(R+M^+)P^+ & -(P^+)^T A^T P^- \\ (P^-)^T A P^+ & (P^-)^T M^- P^- \end{bmatrix} \begin{bmatrix} v_c^+ \\ v_c^- \end{bmatrix} =$$

$$\begin{bmatrix} (P^+)^T K^+ P^+ & \\ & (P^-)^T K^- P^- \end{bmatrix} \begin{bmatrix} v_c^+ \\ v_c^- \end{bmatrix} + \begin{bmatrix} (P^+)^T K^+ & \\ & (P^-)^T K^- \end{bmatrix} (u_{k+1/2} - u_k).$$

After this system has been solved we can get $u_c = Pv_c$. Lastly we can get our new iterate $u_{k+1} = u_{k+1/2} + u_c$.

5 Results

The preconditioned even-parity system (P-EPS), the mixed system (MS) and the mixed system with subspace correction (MS-SC) are implemented in MATLAB, both with and without Anderson Acceleration (AA). The iteration count and computation time for different values for the parameters g and c are shown in the tables below.
The value of c is set to a specific value by changing the values of σ_s and σ_a. This is done in the same way as in [5] using a checkerboard pattern.
For the MS-SC method, Anderson Acceleration does not take the half-steps $u_{k+1/2}$ into account, but only the corrected estimatation u_k.

Table 4: Iteration count and computation count (s) for $g = 0$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	8	1.20	6	0.89	16	0.28	9	0.35	10	0.34	6	0.33
0.7500	10	1.38	7	0.98	25	0.42	12	0.50	12	0.40	7	0.39
0.9000	11	1.53	8	1.10	46	0.84	16	0.69	13	0.53	8	0.53
0.9900	12	2.14	8	1.27	246	4.12	49	2.43	14	0.47	9	0.52
0.9990	11	1.52	8	1.08	-	-	202	9.96	13	0.44	9	0.52
0.9999	12	1.66	9	1.21	-	-	-	-	15	0.63	9	0.52

Table 5: Iteration count and computation count (s) for $g = 0.1$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	8	2.25	6	1.74	15	0.24	9	0.36	10	0.34	7	0.40
0.7500	10	2.78	7	1.99	24	0.42	12	0.52	12	0.41	8	0.48
0.9000	12	3.34	8	2.23	44	0.77	15	0.67	13	0.45	9	0.53
0.9900	12	3.47	9	2.46	233	3.90	42	1.98	14	0.47	9	0.53
0.9990	11	3.04	9	2.49	-	-	208	10.28	13	0.45	9	0.53
0.9999	12	3.36	9	2.48	-	-	-	-	14	0.60	9	0.54

Table 6: Iteration count and computation count (s) for $g = 0.2$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	8	2.73	6	2.18	15	0.25	9	0.38	10	0.34	7	0.39
0.7500	10	3.44	7	2.47	24	0.41	12	0.53	12	0.40	9	0.54
0.9000	12	4.42	8	2.90	43	0.78	16	0.72	13	0.47	10	0.62
0.9900	13	5.41	9	3.30	219	3.97	39	1.97	14	0.50	10	0.64
0.9990	12	4.51	9	3.37	-	-	133	7.05	13	0.47	9	0.57
0.9999	13	4.79	9	2.70	-	-	-	-	15	0.52	10	0.58

Table 7: Iteration count and computation count (s) for $g = 0.4$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	9	3.69	6	2.68	14	0.25	10	0.44	12	0.49	9	0.62
0.7500	11	4.47	8	3.18	22	0.36	13	0.55	16	0.55	11	0.65
0.9000	13	5.26	9	3.50	39	0.65	17	0.74	20	0.66	13	0.77
0.9900	15	6.09	10	3.88	191	3.15	37	1.70	17	0.57	12	0.73
0.9990	14	5.79	10	3.92	-	-	206	10.14	15	0.51	11	0.65
0.9999	15	6.11	10	3.92	-	-	-	-	16	0.57	12	0.72

Table 8: Iteration count and computation count (s) for $g = 0.5$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	9	4.32	7	3.29	15	0.25	11	0.46	14	0.48	10	0.58
0.7500	12	5.76	8	3.72	21	0.35	13	0.55	23	0.77	13	0.79
0.9000	14	6.76	9	4.15	37	0.60	18	0.79	33	1.16	15	0.92
0.9900	17	8.22	10	4.79	176	3.18	37	1.85	26	0.89	15	0.93
0.9990	16	7.69	10	4.57	-	-	113	5.52	17	0.58	13	0.78
0.9999	17	7.90	11	4.40	-	-	-	-	19	0.63	14	0.85

Table 9: Iteration count and computation count (s) for $g = 0.7$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	10	6.19	7	4.20	19	0.31	12	0.51	21	0.71	12	0.73
0.7500	14	8.70	9	5.27	30	0.50	17	0.74	102	3.44	18	1.10
0.9000	18	12.46	11	6.85	41	0.68	22	0.98	-	-	24	1.52
0.9900	25	16.82	13	7.81	145	2.41	45	2.10	-	-	26	1.64
0.9990	25	16.28	13	7.40	812	13.91	151	7.43	29	0.97	20	1.25
0.9999	23	15.51	13	7.43	-	-	313	15.28	122	4.16	22	1.36

Table 10: Iteration count and computation count (s) for $g = 0.9$

c	P-EPS		P-EPS (AA)		MS		MS (AA)		MS-SC		MS-SC (AA)	
0.5000	13	11.61	9	7.61	24	0.40	14	0.61	40	1.35	17	1.05
0.7500	18	19.66	11	10.52	47	0.81	22	0.98	-	-	31	1.95
0.9000	26	30.56	14	14.69	85	1.41	34	1.59	-	-	75	4.85
0.9900	50	60.80	21	22.01	151	2.56	55	2.64	-	-	167	10.92
0.9990	60	71.07	22	22.86	516	8.65	132	6.41	-	-	68	4.55
0.9999	61	71.77	21	20.86	-	-	-	-	-	-	69	4.50

6 Discussion

For the preconditioned even-parity system, we see a low iteration count, but a high computation time per iteration, as expected. We know the contraction rate depends on g and c [2], which is confirmed by the results. If g or c increases, the number of iterations needed goes up. When Anderson Acceleration is applied the iteration count goes down and with it the computation time. However, it is still significantly higher than for the other methods.

The mixed-system has a low computation time per iteration, but as c approaches 1, the convergence becomes very slow. The dashes indicate that the method did not converge within 1,000 iterations. This makes sense, because we have proved that the contraction rate is capped by c. Anderson Acceleration gets the iteration count down, but it remains slow for high values of c.

When looking at the result for the mixed-system with subspace correction we see the method is fast and converges for the cases when g is at most 0.5. For larger values of g, the dashes indicate that the method diverged. Anderson Acceleration turns the divergent sequence into a convergent one and does manage to find a solution for all the cases, while still being a very fast method.

7 Conclusions

In conclusion, we can see that Anderson Acceleration lowers the number of iterations needed for all the methods and converges even when the original method diverges for all chosen parameters. The goal was to have a method which converges fast, the mixed-system with subspace correction and Anderson Acceleration fits this description.

However, it remains to be proven that the method converges for all possible values for the parameters. As of now, we have only shown it does so for these specific values. It is still possible that it fails for other values. This makes the method unreliable. It would be a great addition if the proof is given, because it would mean our goal is achieved and we have a fast convergent method in the mixed-system with subspace correction and Anderson Acceleration.

References

[1] Uri M. Ascher and Chen Greif. *A First Course in Numerical Methods.* Society for Industrial and Applied Mathematics, February 2011.

[2] Jürgen Dölz, Olena Palii, and Matthias Schlottbom. On robustly convergent and efficient iterative methods for anisotropic radiative transfer. *Journal of Scientific Computing*, 90(3), February 2022.

[3] Herbert Egger and Matthias Schlottbom. A mixed variational framework for the raditive transfer Equation. *Mathematical Models and Methods in Applied Sciences*, 22(03):1150014, March 2012.

[4] Michael Modest. *Radiative Heat Transfer.* Academic Press, 2 edition, May 2003.

[5] Olena Palii and Matthias Schlottbom. On a convergent DSA preconditioned source iteration for a DGFEM method for radiative transfer. *Computers &amp Mathematics with Applications*, 79(12):3366–3377, June 2020.

[6] Tanja Tarvainen, Aki Pulkkinen, Ben T. Cox, and Simon R. Arridge. Utilising the radiative transfer equation in quantitative photoacoustic tomography. In Alexander A. Oraevsky and Lihong V. Wang, editors, *Photons Plus Ultrasound: Imaging and Sensing 2017.* SPIE, March 2017.