

Adaptive SOR methods based on the Wolfe conditions

Yuto Miyatake*, Tomohiro Sogabe[†] and Shao-Liang Zhang[‡]

June 4, 2019

Abstract

Because the expense of estimating the optimal value of the relaxation parameter in the successive over-relaxation (SOR) method is usually prohibitive, the parameter is often adaptively controlled. In this paper, new adaptive SOR methods are presented that are applicable to a variety of symmetric positive definite linear systems and do not require additional matrix-vector products when updating the parameter. To this end, we regard the SOR method as an algorithm for minimising a certain objective function, which yields an interpretation of the relaxation parameter as the step size following a certain change of variables. This interpretation enables us to adaptively control the step size based on some line search techniques, such as the Wolfe conditions. Numerical examples demonstrate the favourable behaviour of the proposed methods.

1 Introduction

The successive over-relaxation (SOR) method is one of the most well-known stationary iterative methods for solving linear systems

$$A\mathbf{x} = \mathbf{b},$$

where $A \in \mathbb{R}^{n \times n}$ and $\mathbf{b} \in \mathbb{R}^n$ are given nonsingular matrix and vector, respectively. The SOR method is formulated as

$$\mathbf{x}^{(k+1)} = G_\omega \mathbf{x}^{(k)} + \mathbf{c}_\omega, \quad k = 0, 1, 2, \dots,$$

where

$$\begin{aligned} G_\omega &= (D + \omega L)^{-1}[(1 - \omega)D - \omega U], \\ \mathbf{c}_\omega &= \omega(D + \omega L)^{-1}\mathbf{b}. \end{aligned}$$

Here, the matrix A is expressed as the matrix sum

$$A = D + L + U,$$

where $D = \text{diag}(a_{11}, a_{22}, \dots, a_{nn})$, and L and U are strictly lower and upper triangular $n \times n$ matrices.

The convergence performance of the SOR method depends on the relaxation parameter ω . A necessary condition for the SOR method to converge is that $\omega \in (0, 2)$, and this condition is also sufficient for a symmetric positive definite matrix A (for further details of SOR theory see, e.g., [4, 18]). The relaxation parameter

*Cybermedia Center, Osaka University, 1-32 Machikaneyama, Toyonaka, Osaka 560-0043, Japan, miyatake@cas.cmc.osaka-u.ac.jp

[†]Department of Applied Physics, Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, 464-8603 Nagoya, Japan, sogabe@na.nuap.nagoya-u.ac.jp

[‡]Department of Applied Physics, Graduate School of Engineering, Nagoya University, Furo-cho, Chikusa-ku, 464-8603 Nagoya, Japan, zhang@na.nuap.nagoya-u.ac.jp

ω is ideally chosen such that the spectral radius of the iteration matrix G_ω is as small as possible. However, the optimal value is rarely available except in some special cases, such as consistently ordered p -cyclic matrices [18]. Thus, in practice a specific value obtained either empirically or heuristically is often used. A typical example is $\omega = 2 - O(\tilde{h})$, where \tilde{h} is the mesh spacing of the discretisation of the underlying physical domain. An alternative approach is to adaptively control the relaxation parameter. This approach is referred to as an adaptive SOR method.

In this paper, we are concerned with adaptive SOR methods. To explain our motivation, we first review two standard approaches and their features. One is to approximate the spectral radius of G_ω using the information obtained during iterations [5]. This approach is based on the observation that for a consistently ordered coefficient matrix with some additional assumptions, the spectral radius of G_ω is related to that of the Jacobi iteration matrix. While a highly accurate parameter is often estimated by this method, and the proportion of the cost of the estimation is small compared with a whole iteration, the applicability depends on the matrix properties. The second approach is to approximately minimise the residual instead of the spectral radius of G_ω , in order to handle more general linear systems [1] (see [12, 16] for some related approaches). The key observation in [1] is that the residual can be expanded as a polynomial in terms of ω , and the authors' approach is to first approximate or truncate the polynomial by a lower order one, then determine the real root of it. For symmetric positive definite linear systems, the authors also considered a certain quadratic form instead of the residual. While the efficiency was reported, the estimated parameter does not always satisfy the convergence condition, and several additional matrix-vector products are required to update the parameter.

By taking this background into account, the aim of this paper is to develop a new type of adaptive SOR method that is applicable to a variety of symmetric positive definite linear systems, without additional assumptions or a requirement for additional matrix-vector products when updating the relaxation parameter. To achieve this goal, we develop new adaptive SOR methods based on our previous work [13], which shows that for any symmetric positive definite linear system the SOR method can be regarded as an algorithm for solving a certain minimisation problem. The consequence of this discussion is that the relaxation parameter can be interpreted as the step size following the change of variables $h = 2\omega/(2 - \omega)$. This enables us to apply some line search techniques, developed in the context of unconditioned optimisation problems. In [13], we already tested an approach based on the steepest descent method, but this approach requires an additional matrix-vector product to update the relaxation parameter, and the convergence could be slow for certain linear systems, as will be illustrated later. In this work, we shall propose two new adaptive SOR methods based on other line search techniques. One is based on the Armijo condition, which makes the computational cost for updating the parameter inexpensive. But the estimated parameter sometimes tends to 0, which delays the convergence. To avoid such a situation, we consider another adaptive SOR method based on the Wolfe conditions, i.e. the curvature condition is also used in addition to the Armijo condition. These methods restrict A to be symmetric positive definite, but do not require additional assumptions on the coefficient matrix. Furthermore, they do not require additional matrix-vector products when updating the parameter, and it is guaranteed that the estimated value always satisfies the convergence condition $\omega \in (0, 2)$.

The remainder of this paper is organised as follows. In Section 2, the connection between the SOR method and an optimisation problem is explained, based on [13]. New adaptive SOR methods are presented in Section 3, and numerically tested in Section 4. Finally, concluding remarks are provided in Section 5.

2 Connection between the SOR method and an optimisation problem

This section briefly reviews our previous work [13], which reveals the connection between the SOR method and an optimisation problem.

2.1 Unconditioned optimisation problem for a strictly convex objective function

Let a differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be strictly convex¹ and coercive². Let us consider the unconditioned optimisation problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}). \quad (1)$$

Then, there exists a unique global minimiser \mathbf{x}_* for the function f :

$$\mathbf{x}_* = \arg \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}).$$

Note that this minimiser coincides with the unique solution to $\nabla f(\mathbf{x}) = \mathbf{0}$, and can also be realised as the equilibrium of the following gradient system:

$$\frac{d}{dt} \mathbf{x}(t) = -P \nabla f(\mathbf{x}(t)), \quad (2)$$

where $P \in \mathbb{R}^{n \times n}$ is a symmetric positive definite matrix. More precisely, the solution to (2) converges to the unique minimiser of the function f for any initial vector $\mathbf{x}(0) = \mathbf{x}_0$, i.e.,

$$\lim_{t \rightarrow \infty} \mathbf{x}(t) = \arg \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}).$$

In the following, we discuss algorithms for the optimisation problem (1) with an emphasis on the relation to the gradient system (2). Note that most gradient descent methods can be interpreted as the explicit Euler method for the gradient system (2):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - h P \nabla f(\mathbf{x}^{(k)}), \quad (3)$$

where h denotes the step size. In particular, (3) coincides with the steepest descent method when P is the identity matrix.

Instead of the explicit Euler method, we consider an alternative discretisation method for the gradient system (2). The key is to approximate the gradient using the so-called discrete gradient [2, 7, 8, 9, 11, 10, 14, 15]. We consider a discrete approximation to ∇f , which will be denoted by $\nabla_d f$. The function $\nabla_d f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ is defined by

$$\nabla_d f(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \frac{f(x_1, y_2, \dots, y_n) - f(y_1, \dots, y_n)}{x_1 - y_1} \\ \frac{f(x_1, x_2, y_3, \dots, y_n) - f(x_1, y_2, \dots, y_n)}{x_2 - y_2} \\ \vdots \\ \frac{f(x_1, \dots, x_n) - f(x_1, \dots, x_{n-1}, y_n)}{x_n - y_n} \end{bmatrix} \quad (4)$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. This function is called a discrete gradient, because it is an approximation to the gradient in the sense that

$$\lim_{\mathbf{y} \rightarrow \mathbf{x}} \nabla_d f(\mathbf{x}, \mathbf{y}) = \nabla f(\mathbf{x}). \quad (5)$$

¹A function $f \in \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be strictly convex if and only if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ ($\mathbf{x} \neq \mathbf{y}$) and $\lambda \in (0, 1)$, it holds that $f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) < \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y})$.

²A function $f \in \mathbb{R}^n \rightarrow \mathbb{R}$ is said to be coercive if and only if $f(\mathbf{x}) \rightarrow \infty$ for $\|\mathbf{x}\| \rightarrow \infty$.

Furthermore, it follows that

$$f(\mathbf{x}) - f(\mathbf{y}) = \nabla_{\mathbf{d}} f(\mathbf{x}, \mathbf{y})^\top (\mathbf{x} - \mathbf{y}) \quad (6)$$

for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. By utilising the discrete gradient (4), we discretise the gradient system (2) as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - hP\nabla_{\mathbf{d}} f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}). \quad (7)$$

Here, we have just replaced the gradient in (3) with the discrete gradient. However, there is a significant difference between (3) and (7) from the viewpoint of optimisation problems. For the solution to (7), the following discrete dissipation property holds:

$$\begin{aligned} \frac{1}{h} \left(f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)}) \right) &= \nabla_{\mathbf{d}} f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top \frac{\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}}{h} \\ &= -\nabla_{\mathbf{d}} f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top P \nabla_{\mathbf{d}} f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}) \leq 0 \end{aligned} \quad (8)$$

as long as $h > 0$. Thanks to this property, the sequence obtained by (7) converges to the minimiser.

Proposition 1 (see, e.g., [3]). Let a differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be strictly convex and coercive. Then, the sequence $\{\mathbf{x}^{(k)}\}_{k=0}^\infty$ obtained by the iteration (7) converges to the unique minimiser of the function f for any initial vector \mathbf{x}_0 , i.e.,

$$\lim_{k \rightarrow \infty} \mathbf{x}^{(k)} = \arg \min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}).$$

A rigorous proof of this is given in [3].

This proposition indicates that the convergence is guaranteed even for a relatively large step size h , which is not the case for standard gradient descent methods (3).

Let us mention the case that the step size h is controlled adaptively. We denote the step size at k th iteration by $h^{(k)}$. In this case, we need to avoid the situation that the step size tends to 0 or $+\infty$. The convergence is guaranteed if $0 < \varepsilon < h^{(k)} < M < \infty$ for two positive constants ε and M .

Remark 1. In the context of numerical ordinary differential equations, a function satisfying both (5) and (6) is called a discrete gradient, and is used to simulate systems of ordinary differential equations such as gradient systems and Hamiltonian systems. Such a function is not unique in general, and (4) is just an example, which was introduced by Itoh–Abe [8]. See, e.g., [2, 7, 8, 9, 11, 10, 14, 15] for further details regarding the discrete gradient method.

2.2 SOR method as an optimisation solver

We now show that the SOR method can be regarded as an algorithm to solve a certain optimisation problem.

Let

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top A \mathbf{x} - \mathbf{x}^\top \mathbf{b}. \quad (9)$$

If A is symmetric positive definite, then the function f is strictly convex and coercive, and thus it yields the unique minimiser $\mathbf{x}_* = A^{-1} \mathbf{b}$ (note that $\nabla f(\mathbf{x}) = A \mathbf{x} - \mathbf{b}$). In particular, when $P = D^{-1}$ the scheme (7) is written as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - hD^{-1} \nabla_{\mathbf{d}} f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)}), \quad (10)$$

where the i th component of the discrete gradient is calculated to be

$$\begin{aligned} (\nabla_{\mathbf{d}} f(\mathbf{x}, \mathbf{y}))_i &= \frac{f(x_1, \dots, x_i, y_{i+1}, \dots, y_n) - f(x_1, \dots, x_{i-1}, y_i, \dots, y_n)}{x_i - y_i} \\ &= \sum_{j < i} a_{ij} x_j + a_{ii} \frac{x_i + y_i}{2} + \sum_{j > i} a_{ij} y_j - b_i. \end{aligned}$$

Our previous work [13] proves the following equivalence result.

Theorem 1 ([13]). The SOR method and the scheme (10) are equivalent if

$$h = \frac{2\omega}{2 - \omega}. \quad (11)$$

In other words, under the relation (11) the sequences generated by the SOR method and the scheme (10) coincide with each other as long as the same initial vector \mathbf{x}_0 is used.

Therefore, the SOR method can be regarded as an algorithm solving the optimisation problem for the function (9).

Remark 2. It is well known that the SOR method applied to symmetric positive definite linear systems converges if and only if $\omega \in (0, 2)$. In the standard SOR theory, this condition is obtained by analysing the spectral radius of G_ω . However, this condition can also be proved in a completely different manner based on the above discussion. The convergence condition in terms of h is $h > 0$ (as discussed in the previous subsection), and $h > 0$ is equivalent to $\omega \in (0, 2)$ under the relation (11).

3 Adaptive SOR methods

Based on the interpretation of the SOR method discussed in the previous section, we develop some new adaptive SOR methods. The key idea is to control the step size instead of the relaxation parameter. In the following, the coefficient matrix A is always assumed to be symmetric positive definite.

Note that the step size could be adaptively controlled in many different ways. For example, several step size control techniques have been developed in the context of the numerical analysis of ordinary differential equations [6], and thus applying such techniques would be a possibility [6]. However, this merely provides an efficient strategy, because such step size control techniques aim to integrate ODEs as precisely as possible, but do not aim to find an equilibrium as quickly as possible.

Alternatively, we consider some line search strategies, developed in the context of unconditioned optimisation problems. At each iteration, after defining the step direction $\mathbf{p}^{(k)}$, for example as the direction of steepest descent $\mathbf{p}_k = -\nabla f(\mathbf{x}^{(k)})$, line search methods seek to determine the step size such that it minimises $f(\mathbf{x} + h^{(k)}\mathbf{p}^{(k)})$. However, because this task is often computationally expensive for a general function f , most line search methods seek to approximate the optimal step size at a low cost. Among these, methods employing the Armijo condition or Wolfe conditions are popular. We hope to apply such line search methods to the SOR method. However, such line search methods cannot be directly incorporated with the SOR method because the step direction, i.e., the discrete gradient $\nabla_d f(\mathbf{x}^{(n+1)}, \mathbf{x}^{(n)})$, depends on the choice of the step size (note that $\mathbf{x}^{(n+1)}$ depends on the step size). Therefore, slight modifications are required to apply the aforementioned line search methods to the SOR method. To achieve this task, we shall directly use the approach proposed in [17].

In the following, in order to simplify the presentation we assume without loss of generality that $D = I$. That is, the coefficient matrix A is preconditioned such that its diagonal matrix coincides with the identity matrix. This can be achieved by $D^{-1/2}AD^{-1/2}$.

3.1 Approach based on the locally optimal step size of the steepest descent method

Our previous work [13] considers an adaptive SOR method that employs the locally optimal step size of the steepest descent method. We review this method before presenting new approaches.

The locally optimal step size of the steepest descent method is obtained by solving

$$\min_{h^{(k)} > 0} f(\mathbf{x}^{(k)} + h^{(k)}\mathbf{p}^{(k)}), \quad \mathbf{p}^{(k)} = -\nabla f(\mathbf{x}^{(k)}). \quad (12)$$

For the function (9), the locally optimal step size is given explicitly by

$$h^{(k)} = \frac{(\mathbf{r}^{(k)})^\top \mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^\top A \mathbf{r}^{(k)}}, \quad (13)$$

where $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$. The main idea presented in our previous paper [13] is to adopt (13) as the step size of the SOR method. This approach is summarised in Algorithm 1.

Algorithm 1 Adaptive SOR method based on the steepest descent method

```

 $\mathbf{r}^{(0)} := \mathbf{b} - A\mathbf{x}^{(0)};$ 
for  $k = 0, 1, 2, \dots$  until  $\|\mathbf{r}^{(k)}\| \leq \varepsilon \|\mathbf{b}\|$  do
   $h^{(k)} := \frac{(\mathbf{r}^{(k)})^\top \mathbf{r}^{(k)}}{(\mathbf{r}^{(k)})^\top A \mathbf{r}^{(k)}};$ 
   $\omega^{(k)} := \frac{2h^{(k)}}{2 + h^{(k)}};$ 
   $\mathbf{x}^{(k+1)} := G_{\omega^{(k)}} \mathbf{x}^{(k)} + \mathbf{c};$ 
   $\mathbf{r}^{(k+1)} := \mathbf{b} - A\mathbf{x}^{(k+1)};$ 
end for

```

This algorithm does not contain any parameters to be predetermined. It is observed in [13] that this algorithm performs well for the Poisson equation: the number of iterations required for this algorithm is under two times that of the SOR method with the optimal parameter. It should be noted that there is only one additional matrix-vector product per iteration, i.e., the calculation of $A\mathbf{r}^{(k)}$ appearing in the denominator of (13). This is in contrast to some existing approaches, such as [1], for which additional matrix-vector products are required.

However, it turns out that this approach does not always work perfectly when tested with other linear systems, as illustrated in the next section. Moreover, it is hoped that the cost for estimating the parameter can be reduced further.

Remark 3. The computational effort for updating the step size (or the relaxation parameter) could be reduced if the frequency of the update is decreased, and this idea can be applied to any adaptive SOR method, including those presented in this paper.

3.2 New adaptive SOR methods

Instead of using the step size (12), we propose new strategies utilising the Armijo condition or Wolfe conditions with slight modifications.

Let us start the discussion by applying the Armijo condition, which is motivated by the recent paper [17]. As explained in [17], if $\mathbf{x}^{(k+1)}$ satisfies

$$f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + c_1 \nabla f(\mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \quad (14)$$

for a predetermined constant $c_1 \in (0, 1)$, then the step size is deemed to be good, and increased for the next iteration by a factor of $\lambda_1 > 1$: $h^{(k+1)} = \lambda_1 h^{(k)}$. If the condition (14) is not satisfied, then the step size is decreased for the next iteration by a factor of $\rho_1 \in (0, 1)$: $h^{(k+1)} = \rho_1 h^{(k)}$. This approach has the following distinguished features.

- In contrast to the standard application of the Armijo condition, the condition (14) is not used for calculating the current step size $h^{(k)}$, but rather for calculating the next step size $h^{(k+1)}$. This is because whether or not the condition is satisfied, the dissipation property $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)})$ always follows.
- Assume that the residual $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$ is calculated and checked at each iteration (note that this can be obtained without calculating additional matrix-vector products, because $L\mathbf{x}^{(k)}$ and $U\mathbf{x}^{(k)}$ are obtained in the preceding SOR iteration). Then, there are no additional matrix-vector products required to calculate $f(\mathbf{x}^{(k)})$ and $\nabla f(\mathbf{x}^{(k)}) = -\mathbf{r}^{(k)}$. Thus, the additional costs for checking the condition (14) consist of only a few inner product calculations.

Remark 4. In the condition (14), one might think that the gradient $\nabla f(\mathbf{x}^{(k)})$ should be replaced by the discrete gradient $\nabla_d f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})$. However, considering the condition

$$f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + c_1 \nabla_d f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$$

is meaningless, because this condition is always satisfied:

$$\begin{aligned} f(\mathbf{x}^{(k+1)}) &= f(\mathbf{x}^{(k)}) + \nabla_d f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \\ &\leq f(\mathbf{x}^{(k)}) + c_1 \nabla_d f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}). \end{aligned}$$

The equality follows thanks to (6), and the inequality follows from the fact that $f(\mathbf{x}^{(k+1)}, \mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \leq 0$ (see (8)) and the assumption $c_1 \in (0, 1)$.

Our preliminary numerical experiments show that the above approach performs well for many linear systems. However, the step size detected using the above approach may become close to zero or quite large, which delays the convergence. In other words, in terms of the relaxation parameter ω , $\omega^{(k)}$ sometimes takes a value close to 0 or 2. To avoid this situation, we need to set the maximum and minimum values (M_ω and ε_ω) that the relaxation parameter is allowed to take, and if this is violated we reset the relaxation parameter and the step size.

The above algorithm is summarised in Algorithm 2. Here, the step size is initially set to 2, which indicates that the relaxation parameter is initially set to 1, and if $\omega^{(k+1)} \notin (\varepsilon_\omega, M_\omega)$ then the step size and relaxation parameter are reset to 1 and 2, respectively.

Algorithm 2 Adaptive SOR method based on the Armijo condition

```

Set parameters  $c_1 \in (0, 1)$ ,  $\lambda_1 > 1$ ,  $\rho_1 \in (0, 1)$ ;
 $\mathbf{r}^{(0)} := \mathbf{b} - A\mathbf{x}^{(0)}$ ;
 $h^{(0)} := 2$ ;
 $\omega^{(0)} := 1$ ;
for  $k = 0, 1, 2, \dots$  until  $\|\mathbf{r}^{(k)}\| \leq \varepsilon \|\mathbf{b}\|$  do
   $\mathbf{x}^{(k+1)} := G_{\omega^{(k)}} \mathbf{x}^{(k)} + \mathbf{c}$ ;
   $\mathbf{r}^{(k+1)} := \mathbf{b} - A\mathbf{x}^{(k+1)}$ ;
  if  $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + c_1 \nabla f(\mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$  then
     $h^{(k+1)} := \lambda_1 h^{(k)}$ ;
  else
     $h^{(k+1)} := \rho_1 h^{(k)}$ 
  end if
   $\omega^{(k+1)} := \frac{2h^{(k+1)}}{2 + h^{(k+1)}}$ ;
  if  $\omega^{(k+1)} \notin (\varepsilon_\omega, M_\omega)$  then
     $h^{(k+1)} = 2$ ;
     $\omega^{(k+1)} = 1$ ;
  end if
end for

```

In the context of nonlinear optimisation problems, the Wolfe conditions, which consist of the Armijo condition and the so-called curvature condition, is often employed to avoid the step size becoming close to 0. In the following, we consider another algorithm, which is based on the Wolfe conditions.

Here, we consider the following curvature condition:

$$c_2 \nabla f(\mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \leq \nabla f(\mathbf{x}^{(k+1)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \quad (15)$$

for a predetermined constant $c_2 \in (c_1, 1)$. If the two conditions (14) and (15) are satisfied, then the step size for the next iteration is increased by $h^{(k+1)} = \lambda_1 h^{(k)}$, as in Algorithm 1. However, if the condition (15) is not satisfied, then the current step size is deemed too small and increased for the next iteration by a larger factor of $\lambda_2 > \lambda_1 (> 1)$: $h^{(k+1)} = \lambda_2 h^{(k)}$.

The above approach is summarised in Algorithm 3.

Algorithm 3 Adaptive SOR method based on the Wolfe conditions

```

Set parameters  $c_1 \in (0, 1)$ ,  $c_2 \in (c_1, 1)$ ,  $\lambda_1 > 1$ ,  $\lambda_2 > \lambda_1$ ,  $\rho_1 \in (0, 1)$ ;
 $\mathbf{r}^{(0)} := \mathbf{b} - A\mathbf{x}^{(0)}$ ;
 $h^{(0)} := 2$ ;
 $\omega^{(0)} := 1$ ;
for  $k = 0, 1, 2, \dots$  until  $\|\mathbf{r}^{(k)}\| \leq \varepsilon \|\mathbf{b}\|$  do
   $\mathbf{x}^{(k+1)} := G_{\omega^{(k)}} \mathbf{x}^{(k)} + \mathbf{c}$ ;
   $\mathbf{r}^{(k+1)} := \mathbf{b} - A\mathbf{x}^{(k+1)}$ ;
  if  $f(\mathbf{x}^{(k+1)}) \leq f(\mathbf{x}^{(k)}) + c_1 \nabla f(\mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$  then
    if  $c_2 \nabla f(\mathbf{x}^{(k)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \leq \nabla f(\mathbf{x}^{(k+1)})^\top (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$  then
       $h^{(k+1)} := \lambda_1 h^{(k)}$ ;
    else
       $h^{(k+1)} := \lambda_2 h^{(k)}$ ;
    end if
  else
     $h^{(k+1)} := \rho_1 h^{(k)}$ 
  end if
   $\omega^{(k+1)} := \frac{2h^{(k+1)}}{2 + h^{(k+1)}}$ ;
  if  $\omega^{(k+1)} \notin (\varepsilon_\omega, M_\omega)$  then
     $h^{(k+1)} = 2$ ;
     $\omega^{(k+1)} = 1$ ;
  end if
end for

```

4 Numerical experiments

In this section, the proposed adaptive SOR methods are tested for a variety of symmetric positive definite linear systems. The main aim of this section is to numerically verify the convergence of the proposed methods. Note that the computational costs required for updating the step size were discussed from the viewpoint of the number of matrix-vector products in the previous section, and the actual computation time depends on the frequency of the updates, as explained in Remark 3. Therefore, we do not check the actual computational time, and in all numerical experiments the step size is updated at every iteration, for a fair comparison with Algorithms 1, 2, and 3. We also note that a comparison with other types of adaptive SOR methods or linear solvers, such as the conjugate gradient method, is beyond the scope of this study, because of the difficulty in setting a fair gauge for a head-to-head comparison owing to quite different mathematical features. All numerical experiments are conducted after the coefficient matrix A is preconditioned such that its diagonal matrix coincides with the identity matrix. All the computations are performed in a computation environment with 3.5 GHz Intel Core i5, 8 GB memory, and OS X 10.10.5. We employ MATLAB (R2015a). Below, the results are often displayed after thinning out the data to highlight the behaviour of each algorithm and reduce the file size.

Note that in Algorithms 2 and 3 there are several constants and factors to be predetermined. Instead of finding optimal values for each linear system, which seems more difficult than determining the optimal relax-

ation parameter of the SOR method, we fix them as $(c_1, c_2, \lambda_1, \lambda_2, \rho_1) = (0.89, 0.95, 1.15, 1.4, 0.85)$, and this combination is employed for different linear systems.

4.1 Poisson equation

The first problem arises from the finite difference discretisation of the Poisson equation

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

where $\Omega = (0, 1)^2 \subset \mathbb{R}^2$, Δ denotes the Laplace operator and $f : \Omega \rightarrow \mathbb{R}$. The standard finite difference discretisation on a uniform mesh $N \times N$ leads to a symmetric positive definite linear system for which the coefficient matrix A is given by

$$A = \frac{1}{\Delta x^2} \left(-I \otimes I + \frac{1}{4} B \otimes I + \frac{1}{4} I \otimes B \right),$$

where $I \in \mathbb{R}^{(N-1) \times (N-1)}$ is the identity matrix,

$$B = \begin{bmatrix} 0 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 0 \end{bmatrix} \in \mathbb{R}^{(N-1) \times (N-1)},$$

and \otimes denotes the Kronecker product. The optimal relaxation parameter for this matrix A is well known, and is expressed in terms of $h = 1/(N+1)$ as

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \cos^2(\pi h)}}.$$

Fig. 1 illustrates the numerical results for a test problem $f(x, y) = \sin(\pi x) \sin(\pi y)$. The left figures show the relative residual 2-norm. The convergence of Algorithm 1 is slower than that of the SOR method with the optimal parameter, but is considerably faster than that of the Gauss-Seidel method. Furthermore, the number of iterations required for convergence is less than two times that of the SOR method with the optimal parameter. In this sense, as discussed in our previous report [13], Algorithm 1 appears to be flexible. However, as previously explained, the computational cost per iteration of Algorithm 1 is almost twice as expensive as that of the SOR method. Keeping this in mind, let us discuss the behaviour of Algorithms 2 and 3. For $N = 60$, both algorithms perform satisfactorily compared with Algorithm 1. However, for $N = 100$ and $N = 120$ there are substantial differences. While the results for Algorithm 3 still remain satisfactory, the convergence of Algorithm 2 appears to deteriorate. These differences can also be explained from the right-side figures, which show the variations of $\omega^{(k)}$. In particular, the difference is remarkable when $N = 100$. Here, $\omega^{(k)}$ of Algorithm 2 tends to 0, even after it is reset to 1 at around the 1500th iteration.

4.2 Additional examples

Table 1 presents the matrix data used in the numerical experiments, which were obtained from the Matrix Market (<https://math.nist.gov/MatrixMarket/>) and the ELSSES Matrix Library (<http://www.elses.jp/matrix/>). In the following numerical experiments, the right-hand vector was set to $\mathbf{b} = (1, \dots, 1)^\top$.

The results are illustrated in Figs. 2, 3, 4, and 5. In the left-hand figures, in addition to Algorithms 1, 2, and 3 we plot the results for the Gauss-Seidel method (dashed line), the SOR method with $\omega = 1.8$ (dotted line), and the SOR method, for which the convergence is the fastest in $\omega = 0.1, 0.2, \dots, 1.9$ (black line). In

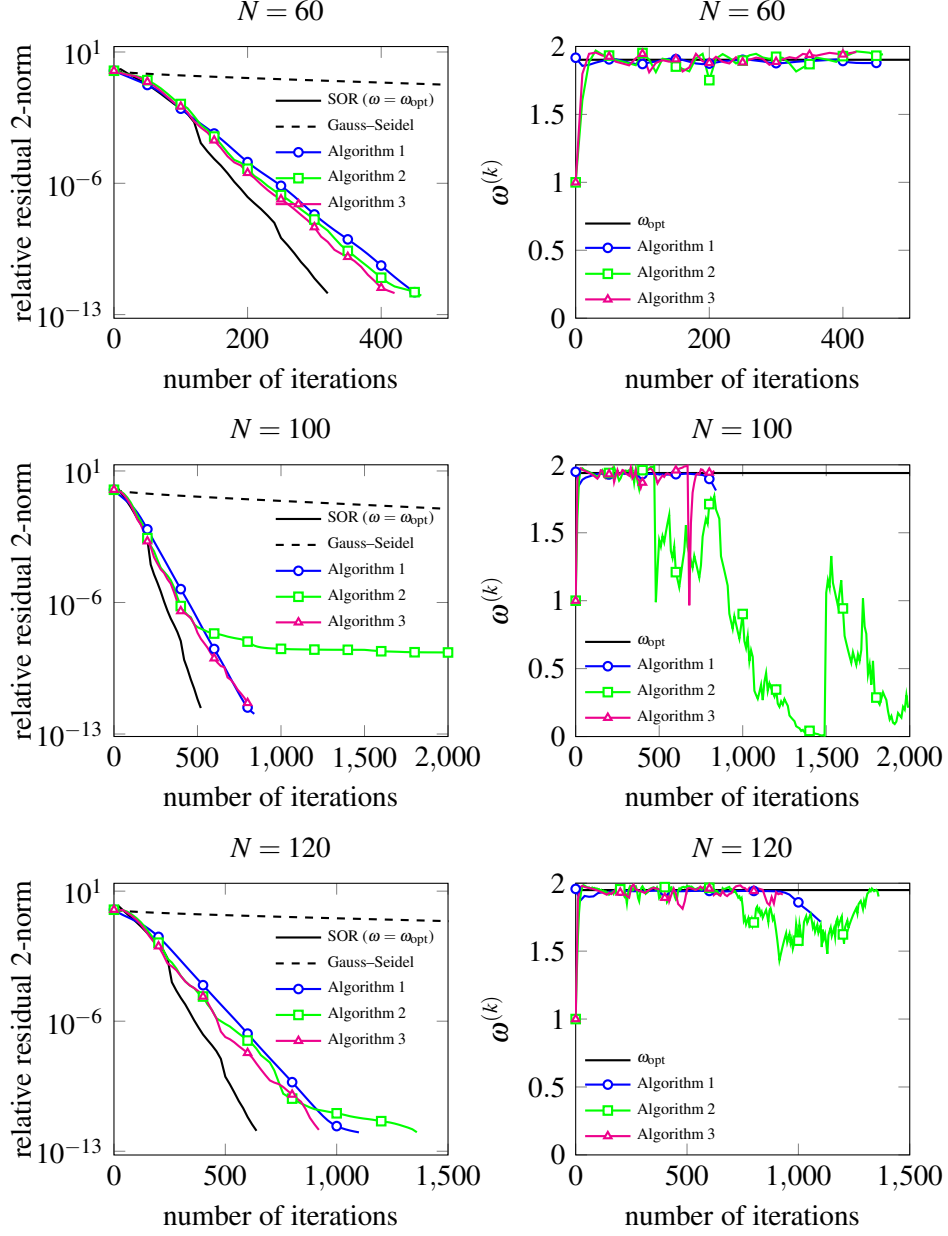


Figure 1: Numerical results for the Poisson equation. (LEFT) Relative residual 2-norm for the SOR method with the optimal relaxation parameter ω_{opt} , the Gauss–Seidel method (the SOR method with $\omega = 1$) and Algorithms 1, 2, and 3, (RIGHT) Variations of $\omega^{(k)}$ in Algorithms 1, 2, and 3. Numerical experiments are conducted for $N = 60, 100, 120$. In the right figures, the black lines express the optimal parameters ω_{opt} .

Table 1: Matrix data.		
Data	n	#nz
BCSSTK04	132	3648
BCSSTK05	153	2423
BCSSTK07	420	7860
ICNT1800	1800	574960

Fig. 5, the results for the Gauss–Seidel method are illustrated by the solid black line rather than the dashed one, because the Gauss–Seidel method performed best.

Although the results differ for each problem, Algorithm 3 is faster than Algorithms 1 and 2 under all settings. Note that from Fig. 5 one might infer that Algorithm 1 is the fastest, but if we take the computational cost at each iteration into account Algorithms 2 and 3 are superior. The number of iterations required to converge for Algorithm 3 is less than three times as big as that for the SOR method, for which the convergence is the fastest in $\omega = 0.1, 0.2, \dots, 1.9$. Furthermore, in the results shown in Fig. 4, Algorithm 3 is the fastest. We observed similar behaviours for the other problems obtained from the Matrix Market and the ELSSES Matrix Library.

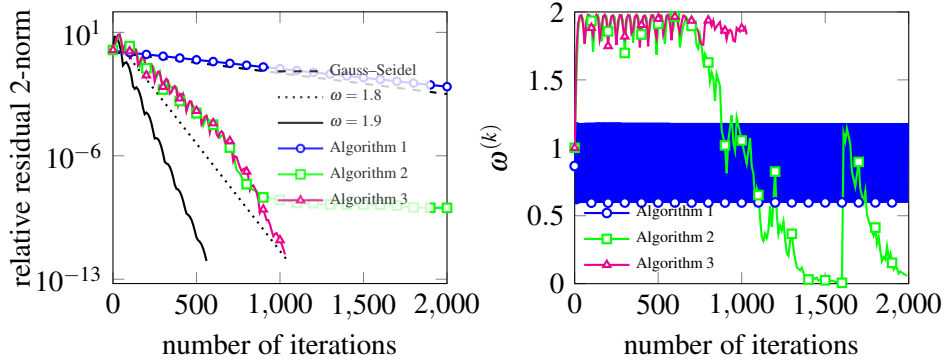


Figure 2: Numerical results for the test problem for BCSSTK04: (LEFT) relative residual 2-norm, and (RIGHT) variations of $\omega^{(k)}$.

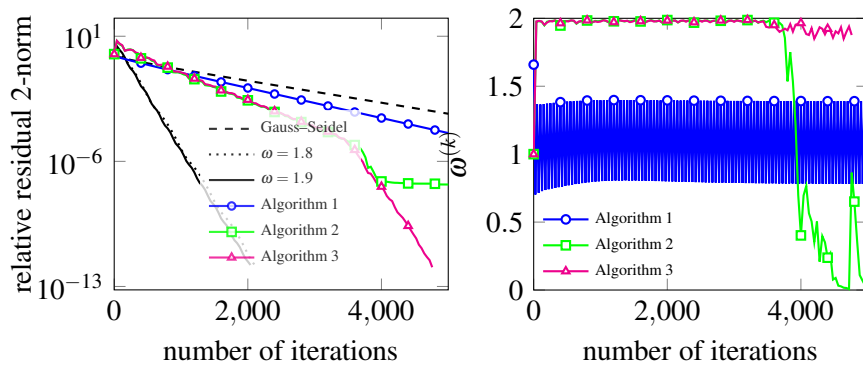


Figure 3: Numerical results for the test problem for BCSSTK05: (LEFT) relative residual 2-norm, and (RIGHT) variations of $\omega^{(k)}$.

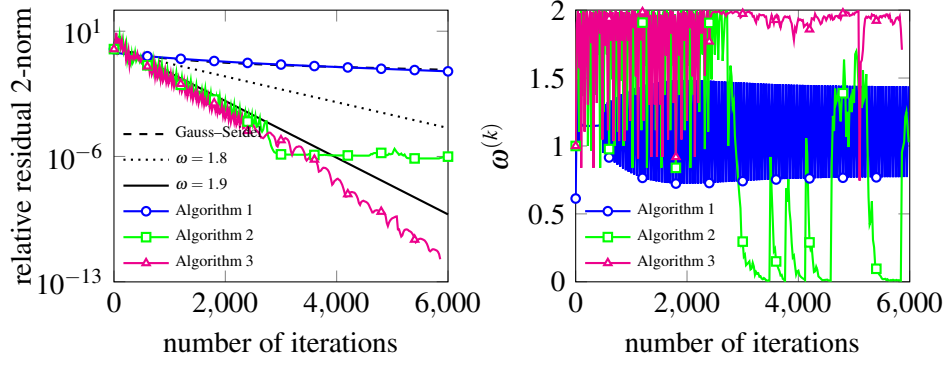


Figure 4: Numerical results for the test problem for BCSSTK07: (LEFT) relative residual 2-norm, and (RIGHT) variations of $\omega^{(k)}$.

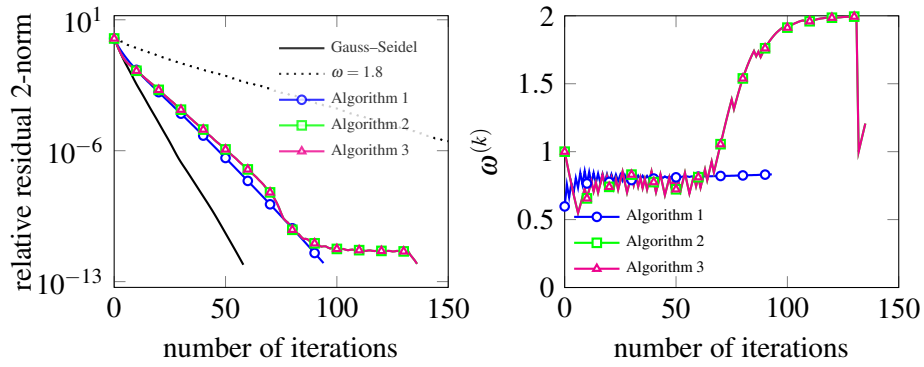


Figure 5: Numerical results for the test problem for ICNT1800: (LEFT) relative residual 2-norm, and (RIGHT) variations of $\omega^{(k)}$. In the left figure, the results by the Gauss-Seidel method is shown in not the dashed line but the black one, and the results for $\omega = 1.9$ are not displayed since the Gauss-Seidel method was the best.

5 Concluding remarks

In this paper, we proposed two adaptive SOR methods based on line search techniques. One is based on the Armijo condition, and the other on the Wolfe conditions. The features of the proposed methods are summarised as follows:

- They are applicable to any symmetric positive definite linear system, in the sense that the convergence condition is always satisfied.
- No additional matrix-vector products are required to update the relaxation parameter.
- There are several factors and parameters in the algorithms. While these should be predetermined and the optimal combination is difficult to obtain, for the adaptive SOR method based on the Wolfe conditions (Algorithm 3), the empirical combination $(c_1, c_2, \lambda_1, \lambda_2, \rho_1) = (0.89, 0.95, 1.15, 1.4, 0.85)$ employed in this paper can perform well for many symmetric positive definite linear systems.
- The convergence is slower than that of the SOR method using the optimal parameter, but is faster than naive choices such as $\omega = 1$ (the Gauss–Seidel method) and $\omega = 1.8$ in most cases. Furthermore, the number of iterations required for convergence is less than two times that of the SOR method with the optimal parameter in most cases, which indicates that the computational cost is also less than double that of the SOR method with the optimal parameter.

We mention several directions for future work. In this paper, among the infinite number of combinations of parameters that are to be predetermined, the results obtained by the chosen empirical combination were presented, but it would be interesting to study a strategy of tuning the combination to find broad applications. We are also currently attempting to extend the presented approach to more general linear systems.

References

- [1] Zhong-Zhi Bai and Xue-Bin Chi. Asymptotically optimal successive overrelaxation methods for systems of linear equations. *J. Comput. Math.*, 21:603–612, 2003.
- [2] O. Gonzalez. Time integration and discrete Hamiltonian systems. *J. Nonlinear Sci.*, 6:449–467, 1996.
- [3] V. Grimm, R. I. McLachlan, D. McLaren, G. R. W. Quispel, and C.-B. Schönlieb. Discrete gradient methods for solving variational image regularisation models. *J. Phys. A*, 50:295201, 2017.
- [4] A. Hadjidimos. Successive overrelaxation (SOR) and related methods. *J. Comput. Appl. Math.*, 123(1-2):177–199, 2000. Numerical analysis 2000, Vol. III. Linear algebra.
- [5] Louis A. Hageman and David M. Young. *Applied Iterative Methods*. Academic Press, New York, 1981.
- [6] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996.
- [7] Ernst Hairer and Christian Lubich. Energy-diminishing integration of gradient systems. *IMA J. Numer. Anal.*, 34:452–461, 2014.
- [8] Toshiaki Itoh and Kanji Abe. Hamiltonian-conserving discrete canonical equations based on variational difference quotients. *J. Comput. Phys.*, 76:85–102, 1988.
- [9] Takayasu Matsuo and Daisuke Furihata. A stabilization of multistep linearly implicit schemes for dissipative systems. *J. Comput. Appl. Math.*, 264:38–48, 2014.

- [10] Robert I. McLachlan, G. R. W. Quispel, and Nicolas Robidoux. Unified approach to Hamiltonian systems, Poisson systems, gradient systems, and systems with Lyapunov functions or first integrals. *Phys. Rev. Lett.*, 81:2399–2403, 1998.
- [11] Robert I. McLachlan, G. R. W. Quispel, and Nicolas Robidoux. Geometric integration using discrete gradients. *R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci.*, 357:1021–1045, 1999.
- [12] Guo-Yan Meng. A practical asymptotical optimal SOR method. *Appl. Math. Comput.*, 242:707–715, 2014.
- [13] Yuto Miyatake, Tomohiro Sogabe, and S Zhang. On the equivalence between SOR-type methods for linear systems and the discrete gradient methods for gradient systems. *J. Comput. Appl. Math.*, 342:58–69, 2018.
- [14] G. R. W. Quispel and D. I. McLaren. A new class of energy-preserving numerical integration methods. *J. Phys. A*, 41:045206, 2008.
- [15] G. R. W. Quispel and G. S. Turner. Discrete gradient methods for solving ODEs numerically while preserving a first integral. *J. Phys. A*, 29:L341–L349, 1996.
- [16] Luna Ren, Fujiao Ren, and Ruiping Wen. A selected method for the optimal parameters of the AOR iteration. *J. Inequal. Appl.*, 2016:279, 2016.
- [17] Torbjørn Ringholm, Jasmina Lazić, and Carola-Bibiane Schönlieb. Variational image regularization with Euler’s elastica using a discrete gradient scheme. *SIAM J. Imaging Sci.*, 11:2665–2691, 2018.
- [18] Richard S. Varga. *Matrix Iterative Analysis*. Springer-Verlag, Berlin, second edition, 2000.