Project: Solving (sparse) system of linear equations (SLE) with successive overrelaxation method (SOR)

Tomasz Chwiej

25th March 2025

1 Introduction

1.1 Discretization of Poisson equation

Sparse systems of linear equations can be efficently solved with iterative methods, because factorization of matrix is skipped while the computational cost of single iteration directly depends on matrix-vector multiplication which is minimized by taking into account only non-zero matrix elements. Successive over-relaxation method is one of such iterative method, in most cases it is used to solve the **Poisson equation**

$$\nabla^2 V(\vec{r}) = -\rho(\vec{r}) \tag{1}$$

where $V(\vec{r})$ is potential and $\rho(\vec{r})$ is the source of the field. We solve this problem in one dimension

$$\frac{d^2V(x)}{dx^2} = -\rho(x) \tag{2}$$

but first we must transform the differential problem onto the algebraic one. By first we define the size of computational box in which we wish to determine the solution $x \in [x_{min}, x_{max}]$, then define the number of nodes n, the distance between neighbouring nodes

$$\Delta = \frac{x_{max} - x_{min}}{n-1} \tag{3}$$

the position of nodes

$$x \to x_i = x_{min} + i \cdot \Delta, \quad i = 0, 1, 2, \ldots, n-1 \tag{4}$$

We are looking for the solution at these nodes so we define discretized vectors, for the density and potential

$$\rho(x) \to \rho(x_i) = \rho_i, \quad i = 0, 1, 2, \ldots, n-1 \tag{5}$$

$$V(x) \to V(x_i) = V_i, \quad i = 0, 1, 2, \ldots, n-1 \tag{6}$$

$$\tag{7}$$

Finally we replace the second derivative with symmetric three-term second-order finite difference formula

$$\frac{d^2V}{dx^2} \approx \frac{V_{i+1} - 2V_i + V_{i-1}}{\Delta^2} \qquad \left(O(\Delta^2)\right) \tag{8}$$

The discretized (algebraic) version of Poisson equation reads

$$\frac{V_{i+1} - 2V_i + V_{i-1}}{\Delta^2} = -\rho_i \tag{9}$$

By writing this equation for subsequent nodes we get the system of linear equation which, conveniently for further purposes, is expressed in matrix form

$$A\vec{v} = \vec{\rho} \tag{10}$$

A is **symmetric tridiagonal matrix** of the system with following non-zero entries $a_{i,j}$

$$a_{i,i} = -\frac{2}{\Delta^2} \tag{11}$$

$$a_{i,i\pm1} = \frac{1}{\Delta^2} \tag{12}$$

To solve Poisson equation we need to specify the boundary conditions at both, left and right ends. If we assume homogeneous boundary conditions $V_{-1} = V_n = 0$ then our SLE fullfills automatically these conditions, because these two elements do not enter to our SLE so the method assume zero-value solution at these points (see the lecture on solving SLE with direct method).

1.2 SOR method for sparse SLE

1.3 Single iteration

Single SOR iteration for SLE defined in matrix form reads

$$(D + \omega L)\vec{v}^{(k+1)} = [(1-\omega)D - \omega U]\,\vec{v}^{(k)} + \omega\vec{\rho} \tag{13}$$

where: L, D and U are the lower triangle, diagonal and upper triangle parts of matrix $A = L + D + U$, respectively, $\vec{v}^{(k)}$ and $\vec{v}^{(k+1)}$ are approximated solutions in two subsequent iterations. By shifting the term $\omega\, L\vec{x}^{(k+1)}$ from the left to the right side and keeping in mind that in SOR an old (k-th) and a new [(k+1)-th] elements are kept in the same vector

$$\vec{v} = [\underbrace{v_0, v_1, \ldots, v_{i-1}}_{\text{elements } \vec{v}^{(k+1)}}, \underbrace{v_i, v_{i+1}, \ldots, v_{n-1}}_{\text{elements } \vec{v}^{(k)}}] \tag{14}$$

we may rewrite equation (13) for i-th element element of $\vec{v}$ which shall be updated next

$$v_i \leftarrow v_i + \frac{\omega}{d_{i,i}}(\rho_i - A_{(i,*)}\vec{v}) \tag{15}$$

where $A_{(i,*)}$ means the i-th row of matrix A. To update the whole vector $\vec{v}$ we must sequentially iterate over its all elements. For matrix A written in CSR format: a[] - non-zero elements of A, col[] -indices of non-zero elements, row[] - indices of the first elements occuring in rows and additional array d[] - the diagonal elements of A, an update of the vector $\vec{v}$ can be performed as follows

```
input: d[], $\rho$[], v[], $\omega$, a[],col[],row[]

for i=0 to n-1 by 1 do
        !compute scalar product $c = A_{(i,*)}\vec{v}$
        c=0
        for l=row[i] to l<row[i+1] by 1 do
                !read column index
                j=col[l]
                !add contribution to scalar product
                c+=a[l]*v[j]
        end do
        !update v[i]
        v[i]=v[i]+$\frac{\omega}{d_i}$($\rho$[i]-c)
        end do
end do
```

1.4 Iterative algorithm

For SOR method we may use for example the following do-while loop. There are two stop conditions: (i) maximal number of iterations (*itmax*) prevents infinite loop iterations in case of lack of convergence and (ii) check if the norm of residual vector falls below some threshold e.g. $\varepsilon = 10^{-15}$

```
input: $\vec{b}$, $\vec{v}$ (initial), A, $\varepsilon$, itmax, k=0
do
        k++
        !update $\vec{v}$ with single-iteration SOR
        $\vec{v} \leftarrow SOR(\vec{v})$
        !compute residual vector
        $\vec{r} \leftarrow \vec{b} - A\vec{v}$
        !compute 2-norm of $\vec{r}$
        $norm = \|\vec{r}\|_2$
!check stopping conditions: repeat or stop
while k<itmax & norm>$\varepsilon$
```

2 Practical part

Tasks to do

1. Write the program solving SLE with SOR method for sparse matrix of the system encoded in CSR format. Besides the pseudocodes presented in Secs.1.3 and 1.4 you need two other routines: (i) for calculating the scalar product of two vectors and (ii) for matrix-vector multiplication with matrix stored in CSR format. The code for the latter is given in Sec. 3.

2. In calculations use parameters: $n = 500$, $\Delta = 1$, $itmax = 10^4$, $\varepsilon = 10^{-15}$. Elements of right-hand-side vector $\vec{\rho}$ fill with values

$$\rho_i = \frac{1}{10}\left[\sin\left(\frac{2\pi \cdot i}{n-1}\right)\right]^{10}, \quad i = 0, 1, 2, \ldots, n-1 \tag{16}$$

3. Perform calculations, i.e. solve SLE, for $\omega = 1$ and initial approximated solution filled with constant value

$$v_i = 1, \qquad i = 0, 1, \ldots, n-1 \tag{17}$$

   In each iteration write to file: index of iteration, the norm of approximated solution $\|\vec{v}\|_2$ and the norm of residual vector $\|\vec{r}\|_2$. Make separate plots showing the dependence of $\|\vec{v}\|_2$ and $\|\vec{r}\|_2$ on iteration index. Make plots of vector $\vec{\rho}$ and final solution $\vec{v}$ depending on spatial position $x_i = i \cdot \Delta$.

4. At home prepare a report including additional results, namely:

   - repeat calculations for different convergence factor values: $\omega = 1.5,\ 1.9,\ 1.99$
   - make single plot showing the changes of the residual vector norm depending on iteration index for all ω values

Try to answear the questions

- Is SOR effective in solving SLE? Comment on efficency of SOR in context of ω value.
- Perform additional tests. Set $\omega = 1.99$ and $\varepsilon = 10^{-8}$ and solve SLE for two initial approximated solutions

  $$v_i = 0, \quad i = 0, 1, \ldots, n-1 \tag{18}$$

  and randomly filled vector

  $$v_i = 100 * (double)rand()/RAND\_MAX, \quad i = 0, 1, \ldots, n-1 \tag{19}$$

  Does SOR method find the solution in both cases? How the intial solution influence on efficency (number of iterations)?

3 Computational hints

3.1 Matrix in CSR format

In project we use the tridiagonal matrix that we wish to store in CSR format. This needs previous declaration of three one-dimensional arrays: ***a, col, row***

- matrix elements: ***a***, number of elements ***nnz_max***
- column indices: ***col***, number of elements ***nnz_max***
- global indices of first entries in rows: ***row***, number of elements $n+1$

Value of ***nnz_max*** must be equal or greater than the number of real non-zero elements in matrix. It can be simply estimated, for tridiagonal matrix A we may assume $\mathbf{nnz\_max = 3 \cdot n}$. Next we may fill arrays with values remembering it must be done in row-wise order (sequentially from left to right) as follows

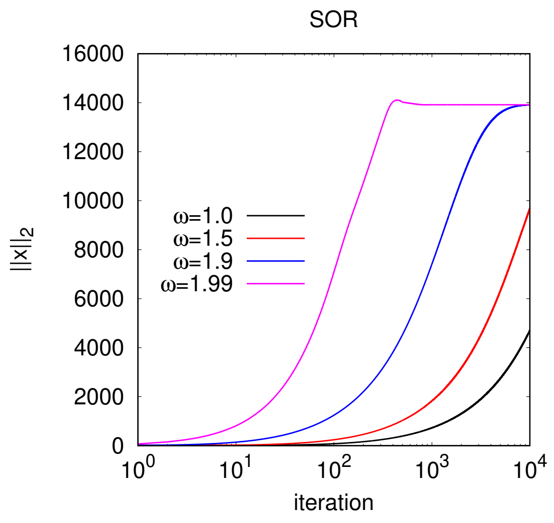
```
initialize: n, nnz_max, a[0:nnz_max-1], col[0:nnz_max-1], row[0:n]
!initialize number of non-zero values
nnz=0
for i=0 to n-1 by 1 do
        !indicate first entry in row is empty
        row[i]=-1

        !subdiagonal (left), column index: j=i-1
        if i > 0 then
                a[nnz]=$1/\Delta^2$
                col[nnz]=i-1
                row[i]=nnz
                nnz++
        end if

        !diagonal (middle of row), column index: j=i
                a[nnz]=$-2/\Delta^2$
                col[nnz]=i
                if row[i] < 0 then
                        row[i]=nnz
                end if
                nnz++

        !superdiagonal (right), column index: j=i+1
        if i < n-1 then
                a[nnz]=$1/\Delta^2$
                col[nnz]=i+1
                nnz++
        end if
end do
!add info about the number of non-zero elements
row[n]=nnz
```

3.2 Matrix-vector multiplication with matrix stored in CSR format

Pseudocode for (CSR)matrix-vector multiplication $\vec{y} = A\vec{x}$

```
input: x[],y[],a[],col[],row[]
for i=0 to n-1 by 1 do
        !empty the cell before summation operation
        y[i]=0
        !calculate scalar product
        for l=row[i] to row[i+1]-1 by 1 do
                !determine the column index
                j=col[l]
                !add contributions: $y_i = y_i + A_{i,j}\, x_j$
                y[i]=y[i]+a[l]*x[j]
        end do
end do
```

4 Example results



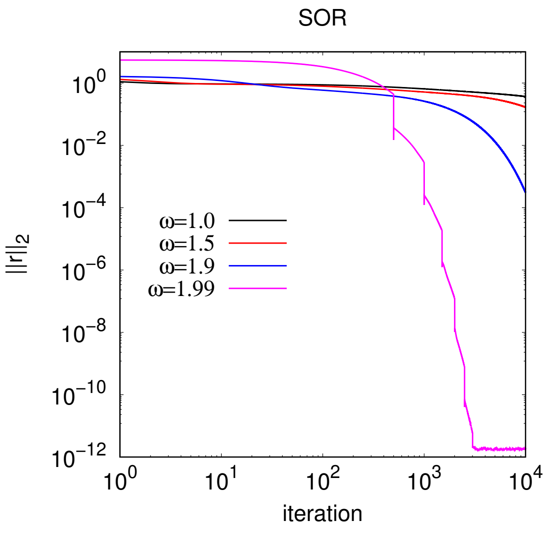


Figure 1: Convergence of solution norm (left) and norm of residual vector (right) in SOR method.