ITERATIVE METHODS

THANH-VAN HUYNH, MICHAEL THIELE, FLORIAN WOLF

Abstract. This is a report about the findings of Thanh-Van Huynh, Michael Thiele and Florian Wolf in the exercises accompanying Iterative Methods for Linear Systems. The lecture was held by Jun.-Prof. Dr. Gabriele Ciaramella with the assistance of Christian Jäckle in the winter term 2020/21 at the University of Constance.

Contents

1. Introduction

In the exercises we studied optimal control problems governed by the Laplace equation. We want to solve

$$\min_{\underline{y},\underline{u}\in\Omega} J(\underline{y},\underline{u}) := \frac{1}{2}\,||\underline{y}-\underline{y_d}||^2_{L^2(\Omega)} + \frac{\nu}{2}\,||\underline{u}||^2_{L^2(\Omega)} \tag{1.1}$$

$$\text{s.t.} \quad -\Delta y = f + u \text{ in } \Omega \tag{1.2}$$

$$y = 0 \text{ on } \partial\Omega \tag{1.3}$$

for a regularization parameter $\nu > 0$. We are looking for a control function u. With discretized norm

$$||\underline{x}||^2_{L^2(\Omega)} := h^2 \sum_{j=1}^{n} \underline{x}_j^2$$

such that

$$A\underline{y} = \underline{f} + \underline{u}$$

is satisfied. Notice that we do not have any boundary conditions. One possible way of finding a solution is the so-called reduced approach: We consider $\underline{y}$ as a function of $\underline{u}$, therefore obtaining the problem

$$\min_{\underline{u}\in\Omega} \hat{J}(\underline{u}) := J(\underline{y}(\underline{u}),\underline{u}).$$

2. Stationary Methods

2.1. Deriving the Optimality System.

We see that

$$||\underline{x}||^2_{L^2(\Omega)} = h^2\,||\underline{x}||^2_2$$

holds for arbitrary x. In combination with $\underline{y} = A^{-1}\left(\underline{f} + \underline{u}\right)$, we obtain

$$\begin{aligned}
\min_{\underline{u}\in\Omega} \hat{J}(\underline{u}) &= \frac{1}{2}\, h^2 || A^{-1}\left(\underline{f} + \underline{u}\right) - \underline{y_d}||_2^2 + \frac{\nu}{2}\, h^2 ||\underline{u}||_2^2 \\
&= \frac{1}{2}\, h^2 \langle A^{-1}\left(\underline{f} + \underline{u}\right) - \underline{y_d}, A^{-1}\left(\underline{f} + \underline{u}\right) - \underline{y_d}\rangle + \frac{\nu}{2}\, h^2 \langle \underline{u}, \underline{u}\rangle \\
&= \frac{1}{2}\, h^2 \left(A^{-1}\left(\underline{f} + \underline{u}\right) - \underline{y_d}\right)^\top \left(A^{-1}\left(\underline{f} + \underline{u}\right) - \underline{y_d}\right) + \frac{\nu}{2}\, h^2 \underline{u}^\top \underline{u}.
\end{aligned}$$

As we know that a solution $\underline{u}$ to the problem above has to satisfy $\nabla \hat{J}(\underline{u}) = 0$, this results in

$$\nabla \hat{J}(\underline{u}) = h^2 A^{-1}\left(A^{-1}(\underline{f} + \underline{u}) - \underline{y_d}\right) + \nu h^2 \underline{u} = 0.$$

We can do the following transformations

$$\begin{aligned}
h^2 A^{-1}\left(A^{-1}(\underline{f} + \underline{u}) - \underline{y_d}\right) + \nu h^2 \underline{u} &= 0 \\
A^{-1}\left(A^{-1}(\underline{f} + \underline{u}) - \underline{y_d}\right) + \nu \underline{u} &= 0 \\
A^{-1}A^{-1}(\underline{f} + \underline{u}) - A^{-1}\underline{y_d} + \nu \underline{u} &= 0 \\
A^{-1}A^{-1}\underline{f} + A^{-1}A^{-1}\underline{u} - A^{-1}\underline{y_d} + \nu \underline{u} &= 0 \\
A^{-1}A^{-1}\underline{u} + \nu \underline{u} &= A^{-1}\underline{y_d} - A^{-1}A^{-1}\underline{f} \\
\left(\nu \mathrm{I} + A^{-1}A^{-1}\right)\underline{u} &= A^{-1}\left(\underline{y_d} - A^{-1}\underline{f}\right)
\end{aligned}$$

and derive the optimality system

$$\left(\nu \mathrm{I} + A^{-2}\right)\underline{u} = A^{-1}\left(\underline{y_d} - A^{-1}\underline{f}\right) \tag{2.1}$$

or alternatively

$$\left(A^2\nu + \mathrm{I}\right)\underline{u} = A\underline{y_d} - \underline{f}. \tag{2.2}$$

2.2. **Laplace Matrices.** First, we implement a function `FD_Laplacian` to create the Finite Difference matrix representation of the Laplace operator in sparse. Depending on whether we choose $d = 1$ or $d = 2$, we obtain the 1D Laplace Matrix

$$T_1 = \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \end{pmatrix} \in \mathbb{R}^{m\times m}$$

or the 2D Laplace Matrix (by using the Kronecker product $\otimes$)

$$T_2 = \mathrm{I}_m \otimes T_1 + T_1 \otimes \mathrm{I}_m = \begin{pmatrix} T_1 & \mathrm{I}_m & & & \\ \mathrm{I}_m & T_1 & \mathrm{I}_m & & \\ & \ddots & \ddots & \ddots & \end{pmatrix} \in \mathbb{R}^{m^2\times m^2}.$$

. FD_Laplacian

```
Enter: matrix dimension $m \in \mathbb{N}$, $d = \{1,2\}$ choosing 1D or 2D
Return: Laplace matrix $T \in \mathbb{R}^{m\times m}$
diag = -2*sparse.identity($m$)
onesUpper = sparse.eye($m, k=1$)
onesLower = sparse.eye($m, k=-1$)
$T$ = diag + onesUpper + onesLower
if $d$ == 2:
    eye = sparse.eye($m$)
    $T$ = sparse.kron(eye, $T$) + sparse.kron($T$, eye)
return($T$)
```

2.3. **Fast Poisson Solver.** Now, we implement the Fast Poisson Solver `Fast_Poisson` to efficiently solve a linear system $Au = b$. As we do not have any boundary conditions, we have a simpler implementation:

. Fast_Poisson

```
Enter: matrix $V$ of eigenvectors of the matrix $A \in \mathbb{R}^{m\times m}$,
vector $\lambda$ of eigenvalues of the matrix $A$,
right-hand side $b$ of the linear system
Return: solution $u$ of the linear system $Au = b$
$B$ = reshape($b, (m,m)$)
$\tilde{B} = (V^\top (V^\top B)^\top)^\top$
for i in 0:m
    for j in 0:m
        $\tilde{u}_{ij} = (\tilde{B}_{i,j})/(\lambda_i + \lambda_j)$
$U = (V(V\tilde{U})^\top)^\top$
$u$ = reshape($U, m^2$)
return($u$)
```

2.4. **Convergence Analysis.** Now, we solve the system (2.1) by the stationary iteration

$$\underline{u}^{n+1} = -\frac{1}{\nu}A^{-2}\underline{u}^n + \frac{1}{\nu}A^{-1}(\underline{y}_d - A^{-1}\underline{f}) \tag{2.3}$$

and analyze its convergence behavior for different m and ν.

2.4.1. *Eigenvectors and eigenvalues of the laplace matrices.* As we have seen in the section regarding the stationary solver and we will see this again, if we consider solving (2.1) using the conjugated gradient method, the poisson solver is the key, to get a fast solution to the A^{-1} problem. The solver requires a decomposition of the matrix V in diagonalized form. In the beginning we achieved this task using `scipy.sparse.eigs`. Luckily the eigenvectors and eigenvalues of the one dimensional laplace matrix are easy to calculate by hand, exploiting its Toeplitz-structure. For the $m \times m$ one-dimensional laplace, we get the following eigenvalues

$$\lambda_k = -2 + 2\cos\left(\frac{\pi k}{n+1}\right), \quad k = 1, \dots, m$$

and the corresponding eigenvectors with entries

$$(\mathbf{v}_k)_i = \sin\left(\frac{i \cdot \pi k}{m+1}\right), \quad i, k = 1, \dots, m$$

Now we can include these manually to get a more exact and especially quicker solver.

To get the eigenvectors and eigenvalues of the two-dimensional laplace matrix, we use problem 10 of REFERENCE TO THE BOOK and proof its third bullet point. Therefore we use the formula $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$. For $j, k \in \{1, \dots, m\}$ arbitrary we get for $\lambda := \lambda_j + \lambda_k$ and $\mathbf{v} := \mathbf{v}_j \otimes \mathbf{v}_k$ that

$$\begin{aligned} A\mathbf{v} &= (\mathrm{I}_m \otimes T_1 + T_1 \otimes \mathrm{I}_m)\,\mathbf{v} \\ &= (\mathrm{I}_m \otimes T_1)\,(\mathbf{v}_j \otimes \mathbf{v}_k) + (T_1 \otimes \mathrm{I}_m)\,(\mathbf{v}_j \otimes \mathbf{v}_k) \\ &= (\mathbf{v}_j \otimes \lambda_k \mathbf{v}_k) + (\lambda_j \mathbf{v}_j \otimes \mathbf{v}_k) \\ &= \lambda_k\,(\mathbf{v}_j \otimes \mathbf{v}_k) + \lambda_j\,(\mathbf{v}_j \otimes \mathbf{v}_k) \\ &= \lambda \mathbf{v} \end{aligned}$$

Therefore the eigenvectors and eigenvalues of the two-dimensional laplace are obtained by using the kronecker product of the one-dimensional eigenvectors and summing up the one-dimensional eigenvalues. We will use the eigenvalues, to calculate the condition number of the matrix in the section about Krylov methods.

2.5. **Damped Jacobi.**

3. Krylov Methods

3.1. **Convergence Analysis.** In this part of the project we want to use the fact, that the matrix arising in our disretization is symmetric and positive definite. For this reason we want to apply the conjugated gradient (CG) method, to solve our systems (2.1) and (2.2). For the first system we use the already implemented fast poisson solver, to get a matrix free version of the left-hand side. In the second case we construct our sparse matrices and use the `scipy.sparse.linalg.LinearOperator` method, to create a linear operator of our left-hand side of the equation. So in both cases the solution process is completely matrix-free.

Looking at figure 1, we can see the convergence behaviour for the two systems. In the plots ν ist getting bigger going from top to bottom and m is getting bigger going from left to right. As you can directly see in the plots,

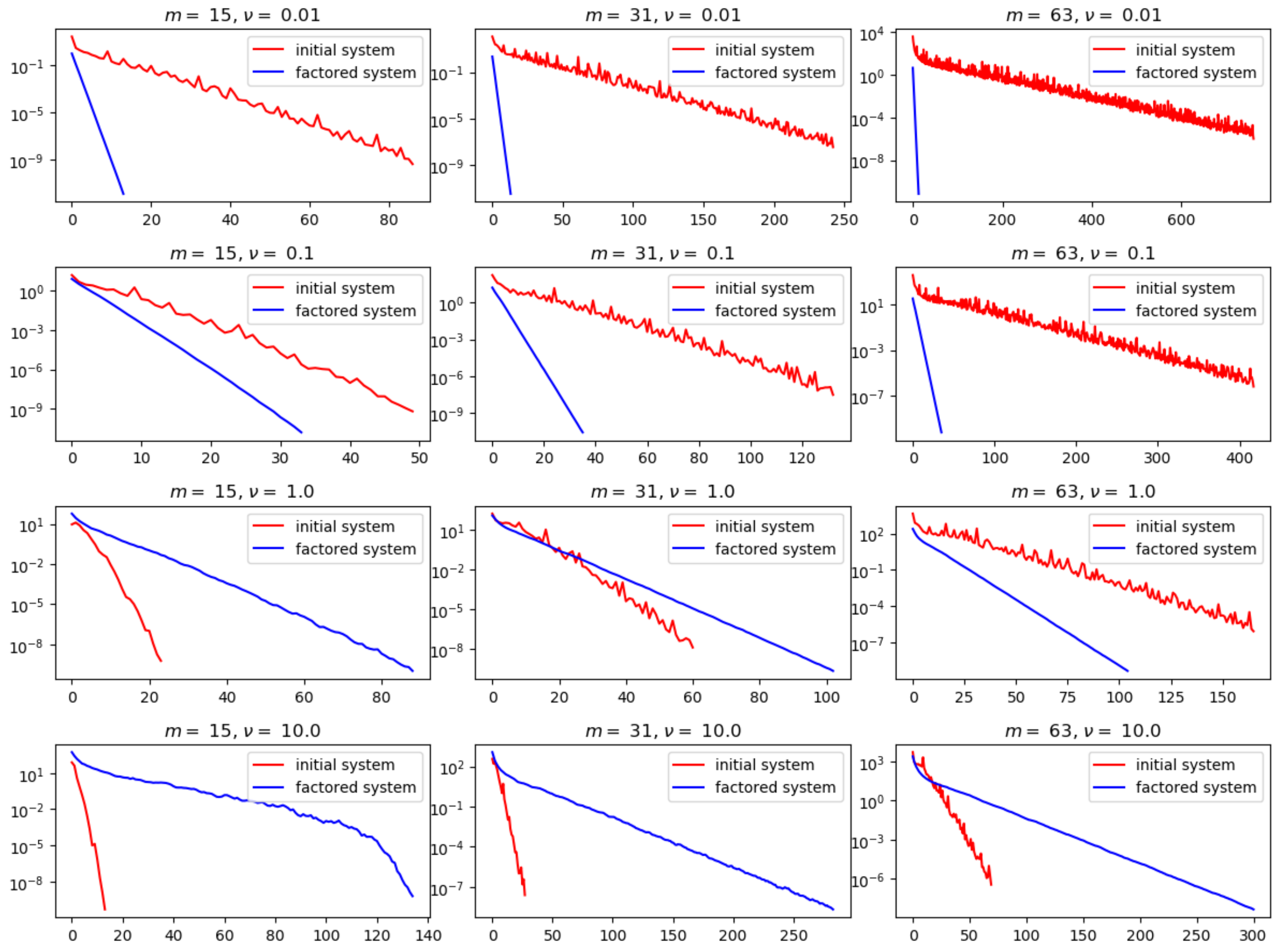

Figure 1. Convergence behaviour for different values of m and ν using CG to solve the systems. The x-axis is representing the number of iterations and the y-axis the 2-norm of the residuals.

the solution process of both systems is getting worse, as the size of the matrix increases. Meanwhile, the behaviour regarding different values of ν is more interesting. While the convergence of the initial optimality systems solution is getting slower and slower if ν increases, the convergence rate of the factored system is getting faster and faster. So the systems behave in an opposite way, regarding the size of ν.

3.2. **Conditional Number.** One can explain this, if we take a closer look at the conditon number of the systems for the upper combinations of ν and m. We can see this in figure 2. The three plots of the condition number explain really good, why we get the upper type of convergence behaviour. While both condition numbers get overall bigger from left to right (so with increasing matrix size due to bigger m), one can clearly see in each of the plots that the systems have opposed curves.

While the factored system (2.2) is worse conditioned for bigger ν, the condition number of the initial system gets better the bigger ν is. Looking at the formulas (2.1) and (2.2) this should now be really obvious. In (2.2) we factor

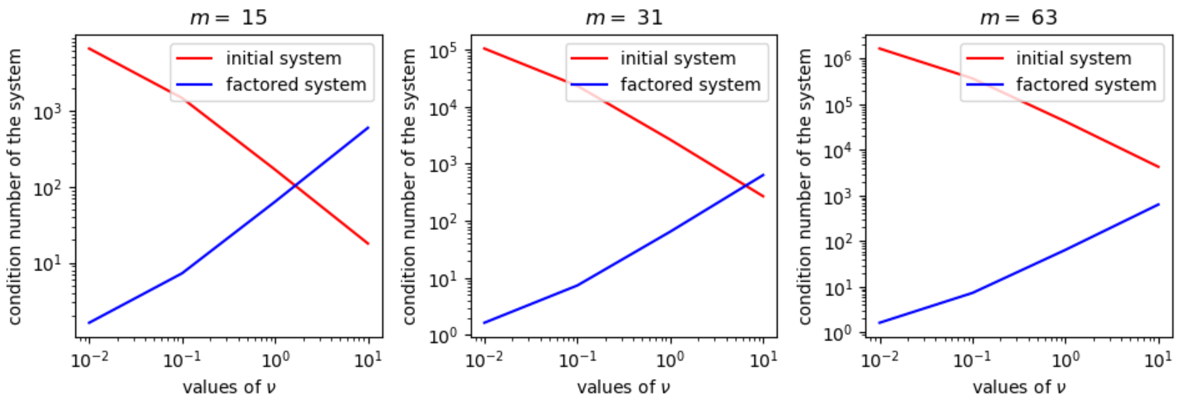

FIGURE 2. Condition numbers of both systems for different values of m and ν represented in a double-logarithmic plot.

our matrix A^2 by ν, so the bigger ν is, the worse the condition number gets. In the initial system (2.1) the identity matrix with factor ν dominates the diagonal of the left-hand side for bigger ν. So the condition number is getting better and better as ν increases.

4. MULTIGRID

4.1. **Stationary Method as Smoother.**

4.2. **Damped Jacobi as Smoother.** aoishdjfkljashgkfj

E-mail address: `thanh-van.huynh@uni-konstanz.de`, `michael.thiele@uni-konstanz.de`, `florian.2.wolf@uni-konstanz.de`