Reduced order model preconditioning for Finite Element Methods

Bastian Schmidt Jørgensen*, Martin Sæbye Carøe*

*Technical University of Denmark (DTU)

Introduction

When using the finite element method to solve PDE problems, one needs to solve a large system $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A}$ is a large sparse and square matrix. We will use the Conjugate Gradient method with Preconditioning (PCG) and compare different preconditioners for solving the heat equation in 2D. The preconditioners that we consider are based on Reduced Order Models (ROMs) where the problem is projected onto a small subspace, where solving the problem using a direct method is inexpensive.

Model

We consider the following test problem from [1] of solving the heat equation in 2D on a square domain, $\overline{\Omega} = [-2.5, 5.1] \times [-4.8, 1.1]$. Let Γ_2 be the lower and left edges and let Γ_1 be the upper and right edges. The model equations are

$$\begin{aligned} \Delta u &= 2\sin(x)\sin(y) \text{ in } \Omega \\ \frac{du}{dn} &= \mathbf{n} \cdot \nabla \sin(x)\sin(y) \text{ for } (x,y) \in \Gamma_1 \\ u(x,y) &= \sin(x)\sin(y) \text{ for } (x,y) \in \Gamma_2 \end{aligned} \tag{1}$$

where $\frac{d}{dn}$ is the normal derivative. One can derive a weak formulation of the problem on the form.

$$a(u,v) = l(v) \text{ for all } v \in \tilde{H}(\Omega).$$

where a, b and $H(\Omega)$ depend on the specifics of the problem. We obtain a symmetric positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{M \times M}$ of elements $a_{ij} = a(N_i, N_j)$ and a right hand side $\boldsymbol{b} \in \mathbb{R}^M$ of elements $b_i = l(N_i)$ where $\{N_i\}_{i=1}^M$ is a set of piecewise linear and continuous basis functions on a uniform grid of $\overline{\Omega}$ with M nodes. We thus obtain a sparse system $\boldsymbol{Ax} = \boldsymbol{b}$.

Preconditioned Conjugate Gradient (PCG)

The CG algorithm for solving $\boldsymbol{Ax} = \boldsymbol{b}$ with preconditioner [3],[4], M, finds

$$\boldsymbol{x}^{(k)} = \text{argmin}_{y \in \mathcal{K}_k} (y - x)^T \tilde{A} (y - x),$$

where $\mathcal{K}_k = \text{span}\{\tilde{b}, \tilde{A}\tilde{b}, \tilde{A}^2\tilde{b}, \ldots, \tilde{A}^{k-1}\tilde{b}\}$, $\tilde{A} = M^{-1}A$ and $\tilde{b} = M^{-1}b$.
The idea with preconditioning is to choose M so that $\kappa(\tilde{A}) << \kappa(A)$. This will ensure faster convergence. $M = I$ will yield the normal CG algorithm. Ideally M^{-1} should resemble A^{-1} while being "cheap".

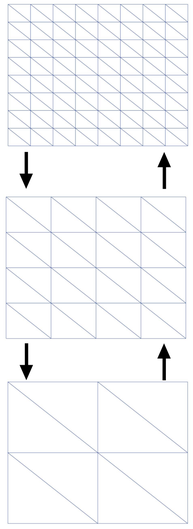
A ROM-Based preconditioner

We will construct a preconditioner by choosing an ONB $P = [p_1, \ldots, p_m]$ for a small subspace, $\mathcal{P}$ close to where we expect the true solution to be. The preconditioner is based on the fact that computing that $P^T AP$ resembles the operator A on $\mathcal{P}$, and that the inverse of $P^T AP$ is cheap. The details are given in [2] p. 1165. We can construct P by computing approximations to the first m left singular vectors of a matrix X whose columns are approximate solutions of the system. We will use previous iterates of the PCG to obtain X. This adaptive PCG method considerably speeds up convergence compared to the regular CG method. The SVD is approximated using random sampling, see [5].

Multigrid algorithm

The multigrid algorithm is a recursive generalization of the two-grid algorithm [1]. The idea is to solve the problem on a coarse grid, as this is cheap.

The maximum depth of the problem is the number of times the grid is coarsened until we solve the system directly.



- Let x be the initial guess.
- Apply Jacobi pre-smoothing to x
- Find residuals and coarsen these
- Solve system on coarse grid using multigrid algorithm (or with direct solver when maximum depth is reached).
- Interpolate to finer grid
- Apply post-smoothing

We can use one iteration of the multigrid algorithm as the preconditioner in the PCG algorithm.

Solution

We compare the following methods: Multigrid algorithm, CG method without preconditioning, adaptive PCG (APCG), PCG with the preconditioner based on SVD of the iterates from the (APCG), PCG with multigrid preconditioner (MGPCG).
The fastest method for solving the problem was MGPCG. The finite element solution is shown in figure 2.

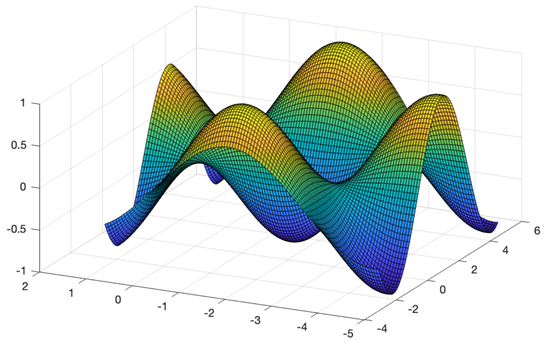


Figure 1: Finite element solution to problem (1) with MGPCG solver for the linear system.

Numerical results

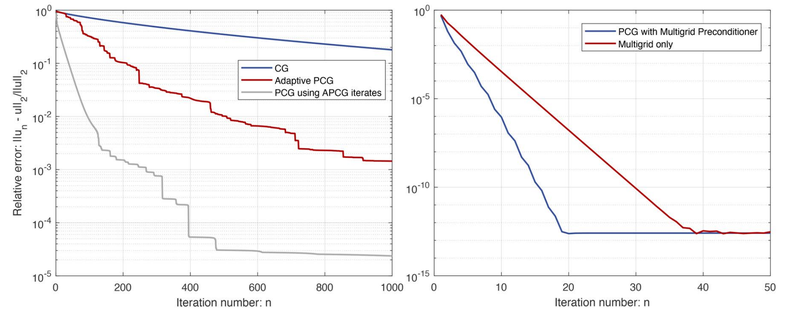


Figure 2: Relative algebraic error plotted versus iteration number, n for the CG, APCG, PCG using APCG iterates, MGPCG and multigrid only.

The MGPCG algorithm obtains an error below machine precision in only 18 iterations. The multigrid algorithm itself converges a little slower than the MGPCG. The APCG converges much slower than MGPCG but faster than CG with only little extra computational effort. Unsurprisingly, when using the iterates from the APCG to create a basis for the preconditioner, we see fast convergence initially.

For large problems in particular, we can obtain faster computing times than MATLABs direct solver for obtaining a relative error below 10^{-4}. With optimized code, this will be even more apparent.

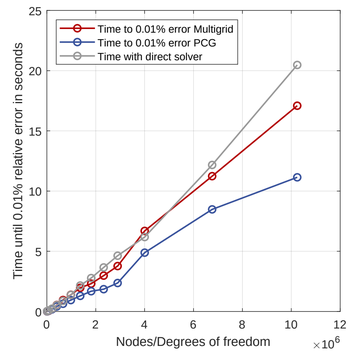


Figure 3: Time in seconds for the different methods to reach 0.01% relative error.

Environmental impacts

Implementing more efficient algorithms has a huge potential to reduce CO_2 emission in the world. This results in a positive impact on UN goal 12. To produce the solution in Fig. 1, we used $2.306 \cdot 10^{-8}$CO2e kg.

Further work

- Explore different multigrid methods
  - V-, W-, F-cycle
- Different smoothers: Jacobi, Gauss-Siegel
- "Train" the solver for many different initial (boundary) conditions.

References

[1] A. P. Engsig-Karup, *The Spectral/hp-Finite Element Method for Partial Differential Equations*, 2017
[2] Pasetto, Damiano Ferronato, Massimiliano Putti, Mario. (2016). *A reduced order model-based preconditioner for the efficient solution of transient diffusion equations: A ROM-BASED PRECONDITIONER FOR TRANSIENT DIFFUSION PDES.* International Journal for Numerical Methods in Engineering. 109.
[3] Jesse L. Barlow, The Preconditioned Conjugate Gradient Method, Lecture 20, Professor from Pennsylvania State University.
[4] Greg Fasshauer, Chapter 16: Preconditioning, Illinois Institute of Technology
[5] Antoine Liutkus (2023). randomized Singular Value Decomposition (https://www.mathworks.com/matlabcentral/fileexchange/47835-randomized-singular-value-decomposition), MATLAB Central File Exchange. Retrieved January 17, 2023.

Bastian: s194357@student.dtu.dk, Martin: s194317@student.dtu.dk