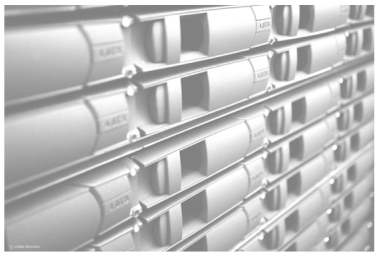
BACI Technical Group Meeting

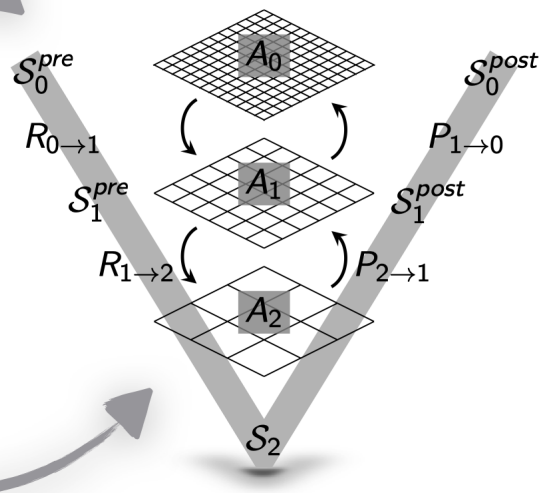
Institute for Mathematics and Computer-Based Simulation
Department for Civil Engineering and Environmental Sciences
April 13, 2022

der Bundeswehr Universität München

Introduction to linear solvers and preconditioning in BACI



$$\begin{bmatrix} a_{00} & a_{01} & \dots & \\ a_{10} & a_{11} & \dots & \\ \vdots & \vdots & \ddots & \vdots \\ & & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$$



Max Firmbach [1]

[1] Institute for Mathematics and Computer-Based Simulation, Universität der Bundeswehr München

Acknowledgements

Several people have contributed to these slides in some way or another (in alphabetical order):

- Max Firmbach
- Martin Kronbichler
- Matthias Mayr
- Tobias Wiesner

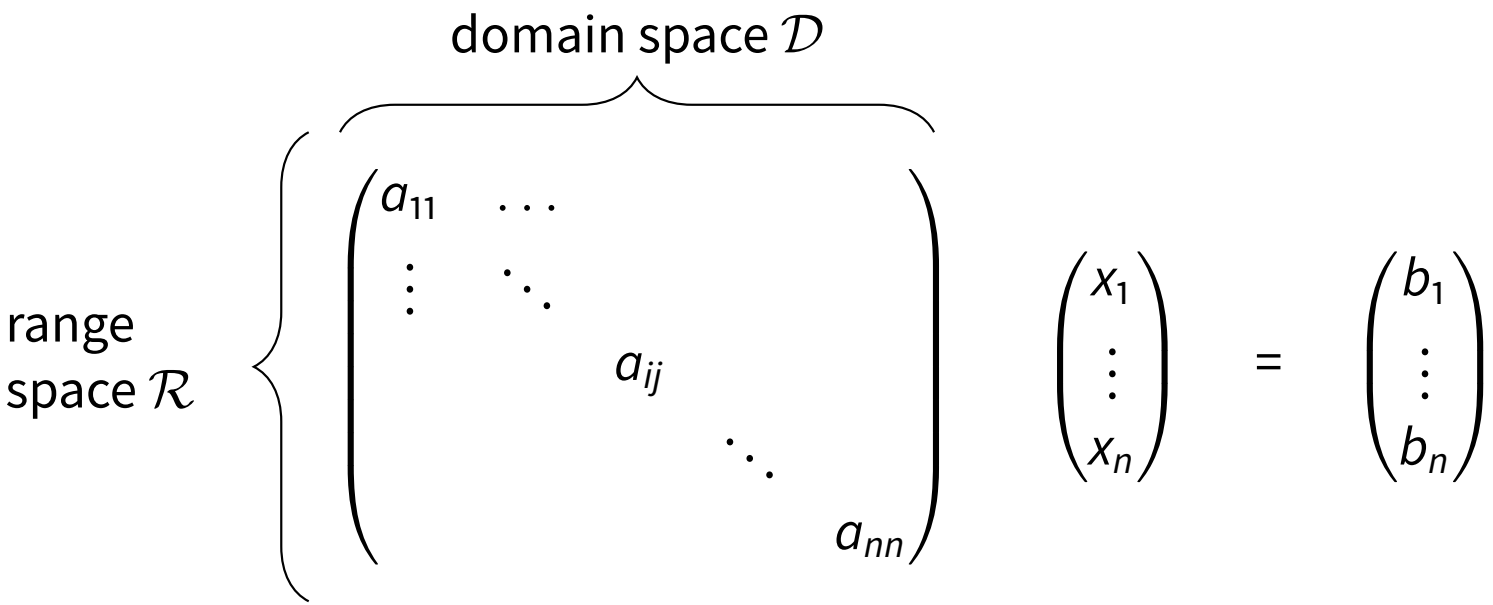
Agenda

- Introduction
- Backend
- Implementation
- Hands-on
- Discussion

Linear System

Abstract view: Linear system I

Linear system:



Task

Given (sparse) matrix $A \in \mathbb{R}^{n \times n}$ and vector $b \in \mathcal{R}$, find the solution vector $x \in \mathcal{D}$ such that $Ax = b$.

We choose $\mathcal{R} = \mathbb{R}^n$ and $\mathcal{D} = \mathbb{R}^n$.

Abstract view: Linear system II

Solution

Solution of the system is given by the inverse A^{-1}:

$$x = A^{-1}b.$$

Explicit calculation of the inverse is extremly expensive! Think about other ways how to solve the linear system.

Remarks:

- Matrix A needs to be non-singular.
- Matrix A can be either a “regular” matrix or a “block” matrix composed of multiple “regular” matrices,

$$A_{block} = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$$

- In FEM, matrix A is sparse.

Direct vs iterative solvers and preconditioning

Direct solvers

Direct method: factorize the matrix using (sparse) Gaussian elimination, i.e.

$$A = \begin{bmatrix} 4 & 2 & 4 \\ 1 & 2 & 1 \\ 3 & 3 & 9 \end{bmatrix} \quad \text{do LU factorization} \quad U = \begin{bmatrix} 4 & 2 & 4 \\ 0 & 1.5 & 0 \\ 0 & 0 & 6 \end{bmatrix}.$$

- Robust black-box solvers solve almost any reasonable (= well-posed) system $Ax = b$ accurately.
- But: direct solvers do not scale with increasing problem size.

You might know them from: `UMFPACK`, `Superlu`

Iterative methods

Iterative method: Improve initial guess x^0 iteratively by using matrix-vector products Ax^k, i.e.

$$x^{k+1} = \mathcal{P}\left(x^k, Ax^k, b\right).$$

such that $x^k \to x$ for $k \to \infty$.

Prerequisites

All iterative methods have certain prerequisites for convergence. If these are not fulfilled there is no guarantee to obtain a good approximation of the solution x.

Typical prerequisites:

- A has to be symmetric positive definite.
- A has to be diagonally dominant.
- A must not have zeros on the diagonal.

Iterative methods

Iterative method: Improve initial guess x^0 iteratively by using matrix-vector products Ax^k, i.e.

$$x^{k+1} = \mathcal{P}\left(x^k, Ax^k, b\right).$$

such that $x^k \to x$ for $k \to \infty$.

Basic usage:

- Only perform a finite number of iterations (possibly $k \leq C$ with $C \ll n$ a constant)
- How many iterations should we do — when to stop?

Preconditioning

- Slow progress in CG or GMRES for badly conditioned matrices (which are quite common: fluid flow, elasticity).
- Try to improve condition numbers (and hence performance) by solving a modified system.

$$P^{-1}Ax = P^{-1}b$$

  for some preconditioner P.
- Goal: $P^{-1}A \approx I$ and P^{-1} should be cheap to apply (so there is no use in setting $P^{-1} = A^{-1}$).
- Any (linear) operator can be used as preconditioner:
  - Jacobi or Gauss–Seidel iteration (e.g. $P^{-1} = (D + L)^{-1}$)
  - Incomplete factorizations (e.g. ILU: $A \approx \tilde{L}\tilde{U}$)
  - Multigrid
  - (CG with a loose tolerance)
- Choice and design of preconditioner highly affect performance.

The Trilinos Library

Trilinos Library

What is the Trilinos library exactly?

A collection of different packages, also including direct and iterative solvers as well as preconditioners.

What you might now already:

- Epetra $\rightarrow$ Sparse Linear Algebra
- Amesos $\rightarrow$ Direct Solver (UMFPACK, Superlu)
- AztecOO $\rightarrow$ Iterative Solver (CG, GMRES)
- Ifpack $\rightarrow$ Preconditioner (ILU, ICHOL, GS)
- ML $\rightarrow$ Multigrid-Preconditioner

Liner solver interface in BACI

Linver solver interface in BACI

Choose solver type

- Decision between direct or iterative solver
- If iterative method is choosen, additional options need to be set

```
enum SolverType
{
  aztec_msr,              // AztecOO (iterative)
  ...
  umfpack,                // Amesos – Umfpack (direct)
  amesos_klu_sym,         // Amesos – Klu (direct)
  amesos_klu_nonsym,      // Amesos – Klu (direct)
  superlu,                // Amesos – Superlu (direct)
  belos,                  // Belos (iterative)
  undefined               // no solver
};
```

Linver solver interface in BACI

Choose iterative type

- Solvers from AztecOO and Belos are available
- If iterative method is choosen, one should also specify a preconditioner

```
enum AzSolverType
 {
   azsolv_CG ,          // Aztec00/Belos – CG
   azsolv_GMRES ,       // Aztec00/Belos – GMRES
   ...
   azsolv_BiCGSTAB      // Aztec00/Belos – BiCGSTAB
 };
```

Linver solver interface in BACI

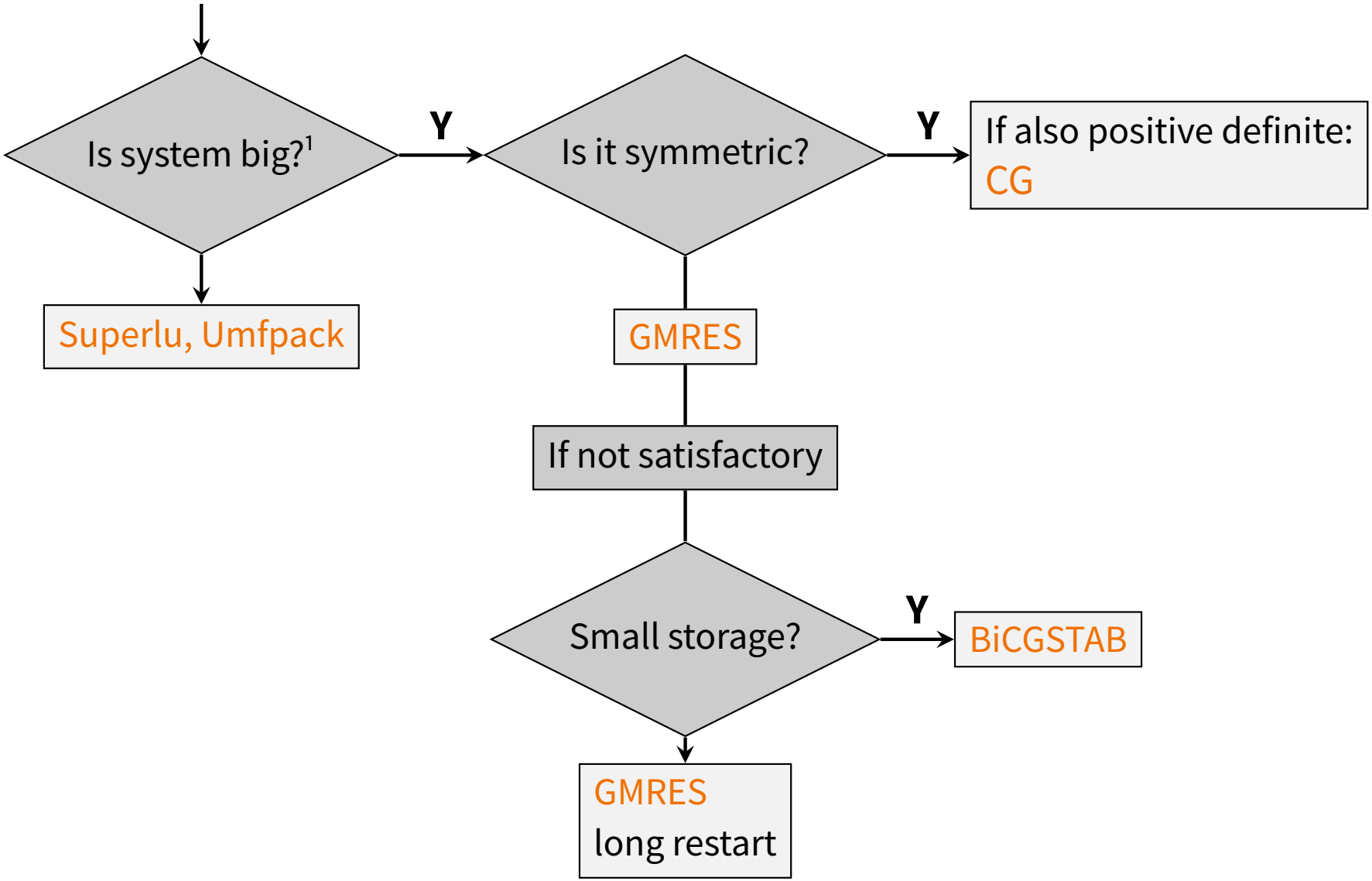
Choose preconditioner type

- Different options from several packages are available
- Setting no preconditioner is not advisable

```
enum AzPrecType
{
  azprec_none,          // no preconditioning
  ...
  azprec_ILU,           // AztecOO – LU
  azprec_ICC,           // AztecOO – Cholesky
  ...
  azprec_BGS2x2,        // BACI – block gauss seidel
  ...
  azprec_ML,            // ML – multigrid
  azprec_MueLuAMG,      // MueLu – multigrid
  azprec_AMGnxn,        // BACI – multigrid
};
```

Application and examples

Choosing a linear solver



[1] **around 50.000 degrees of freedom**.

Example 1: Fluid cavity

Description

Problem represented by a sparse matrix *A*, which now should be solved with a linear solver. Different solving strategies are possible!

- How to choose a respective linear solver?
- Which options to set for iterative solvers?

```
------------------------------------------------------SOLVER 1
NAME                 LinearSolver
SOLVER               ?
AZSOLVE              ?
AZPREC               ?
AZOUTPUT             ?
...
```

Discussion

Changes to usage of Trilinos packages

The packages from Trilinos used in BACI are old and not actively developed anymore! What could be done to use more recent implementations and features:

- Epetra $\rightarrow$ Tpetra / Kokkos / Teuchos (extremly difficult[2])
- Amesos $\rightarrow$ Amesos2 (easy)
- AztecOO $\rightarrow$ Belos (in the making[3])
- Ifpack $\rightarrow$ Ifpack2 (easy)
- ML $\rightarrow$ MueLu (difficult)

Switching to newer packages ensures long term compability with Trilinos and offers new possibilities regarding high performance computing ...

[2] **https://gitlab.lrz.de/baci/baci/-/issues/746**.
[3] **https://gitlab.lrz.de/baci/baci/-/issues/702**.

Changes to the BACI solver interface

Right now the solver parameters one can choose are quite overwhelming. One could think of reducing and simplifing the solver interface and input parameters and make them independent of Trilinos naming[4]:

```
----------------------------------------------------------------SOLVER 1
NAME                    LinearSolver
TYPE                    direct / iterative
METHOD                  umfpack / cg, gmres, ...
PREC                    ...
VERBOSITY               ...
...
```

Or switch to an xml input format like it is done in MueLu or NOX (nonlinear solver).

[4] **https://gitlab.lrz.de/baci/baci/-/issues/722**.

Questions & Remarks ?