SIAM EX14 Workshop
July 7, Chicago - IL

# Preliminary Investigations on Resilient Parallel Numerical Linear Algebra Solvers

Luc GIRAUD

joint work with
E. AGULLO, P. SALAS, E. F. YETKIN, M. ZOUNON
funded by ANR RESCUE and G8-ECS

HiePACS Inria Project
Joint Inria-CERFACS lab
INRIA Bordeaux Sud-Ouest

# Context

- HPC systems are not fault-free
- A faulty components (node, core, memory) loses all its data
- Simulations at exascale have to be resilient

Resilience: Ability to compute a correct output in presence of faults

- Context: Numerical linear algebra
- Goal: Keep converging in presence of fault
- Method: Recover-restart strategy without Checkpoint

# Outline

# Outline

# Framework

## Forecast for extreme scale systems

- Mean Time Between Failure (MTBF): less than one hour
- Checkpoint time might be larger than MTBF

# Framework

## Forecast for extreme scale systems

- Mean Time Between Failure (MTBF): less than one hour
- Checkpoint time might be larger than MTBF

## Objectives

- Explore fault-tolerant schemes with less/no overhead
- Numerical algorithms to deal with overhead issue

# Framework

## Forecast for extreme scale systems

- Mean Time Between Failure (MTBF): less than one hour
- Checkpoint time might be larger than MTBF

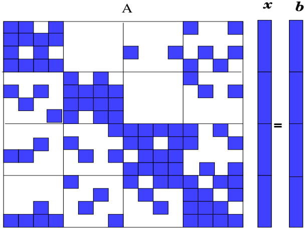
## Objectives

- Explore fault-tolerant schemes with less/no overhead
- Numerical algorithms to deal with overhead issue

## Faults in this presentation

- Detected corrupted memory space (node crashes, damaged memory pages, uncorrected bit-flip, . . . )

# Outline




$$Ax = b$$

We attempt to design fault tolerant solver for sparse linear system

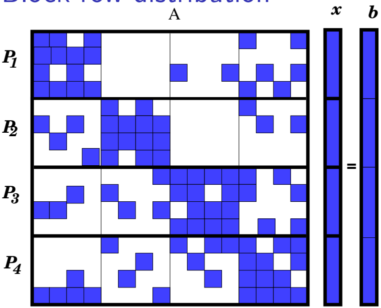
## Two classes of iterative methods

- Stationary methods (Jacobi, Gauss-Seidel, ...)
- Krylov subspace methods (CG, GMRES, Bi-CGStab, ...)

- Krylov methods have attractive potential for Extreme-scale

# Outline

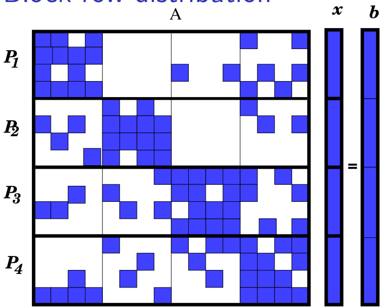
## Block row distribution




We distinguish two categories of data:

- Static data
- Dynamic data

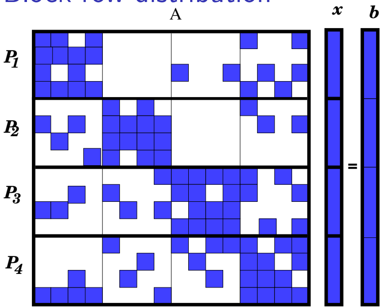
## Block row distribution




We distinguish two categories of data:

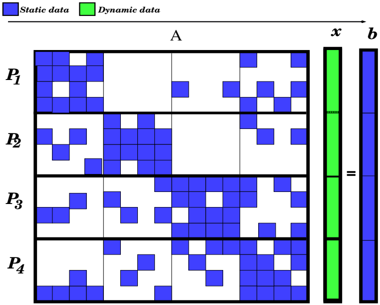
- Static data
- Dynamic data

# Block row distribution



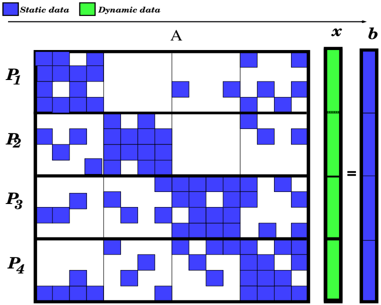

We distinguish two categories of data:

- Static data
- Dynamic data




We distinguish two categories of data:

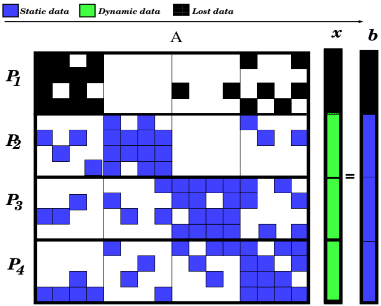
- Static data
- Dynamic data




We distinguish two categories of data:

- Static data
- Dynamic data

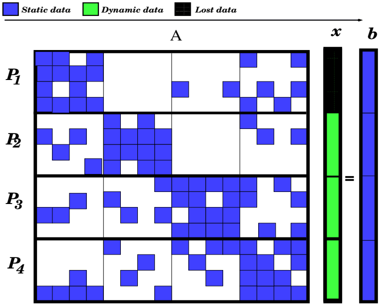
Let's assume that $P_1$ fails




We distinguish two categories of data:

- Static data
- Dynamic data

Let's assume that $P_1$ fails

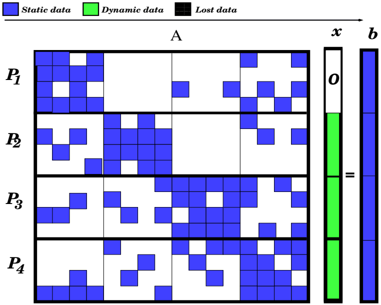



We distinguish two categories of data:

- Static data
- Dynamic data

Let's assume that $P_1$ fails

- Failed processor is replaced
- Static data are restored



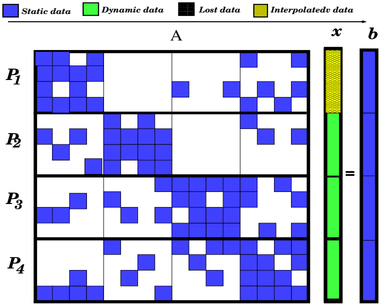

We distinguish two categories of data:

- Static data
- Dynamic data

Let's assume that $P_1$ fails

- Failed processor is replaced
- Static data are restored

Reset: Set $(x_1)$ to initial value




We distinguish two categories of data:

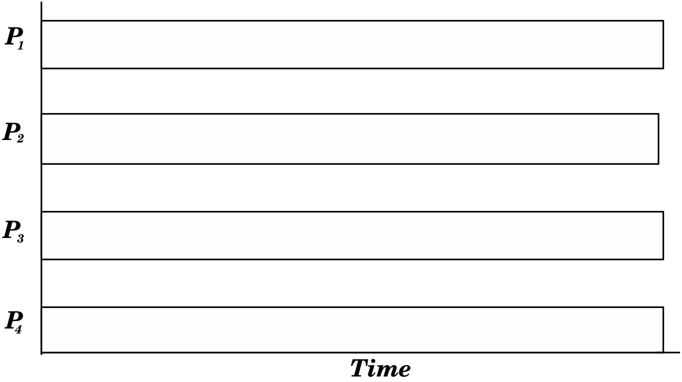
- Static data
- Dynamic data

Let's assume that $P_1$ fails

- Failed processor is replaced
- Static data are restored

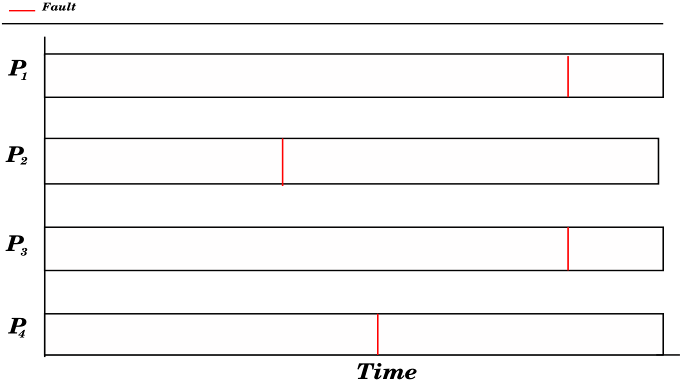
Our algorithms aim at recovering $x_1$ and restart

# Overview of our fault tolerant algorithm



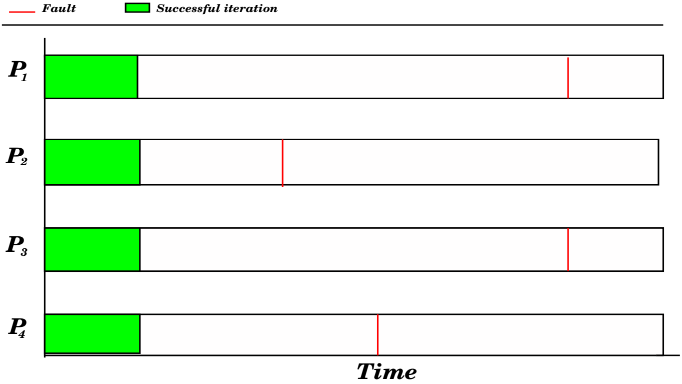

- Sequential simulations
- Simulation of parallel environment

## Overview of our fault tolerant algorithm



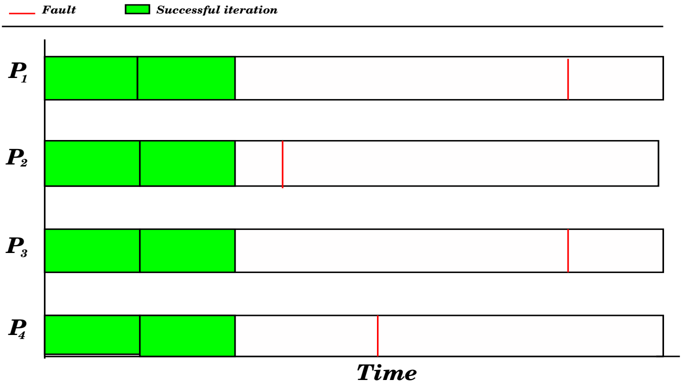

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



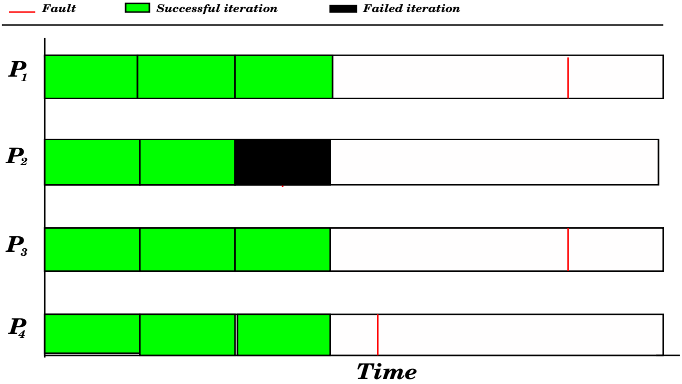

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



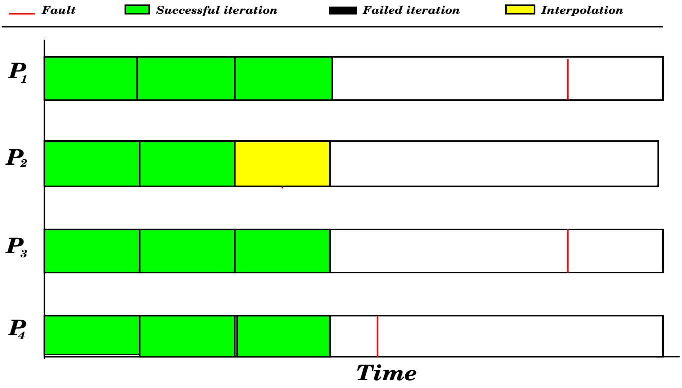

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



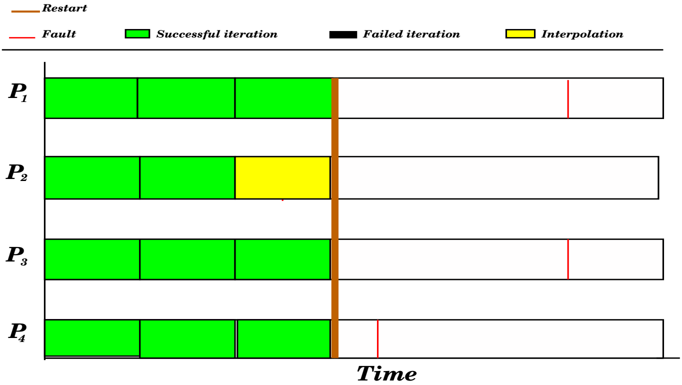

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

## Overview of our fault tolerant algorithm



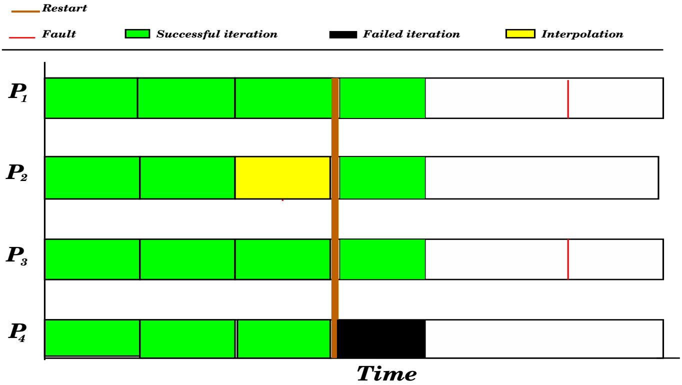

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



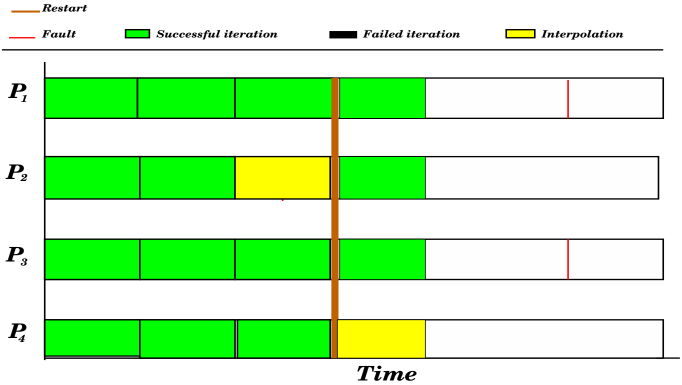

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



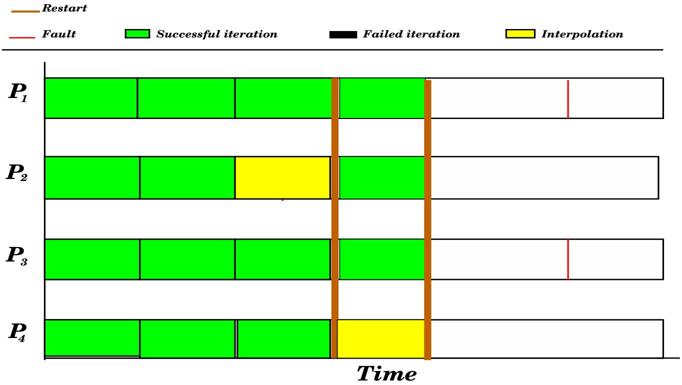

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm



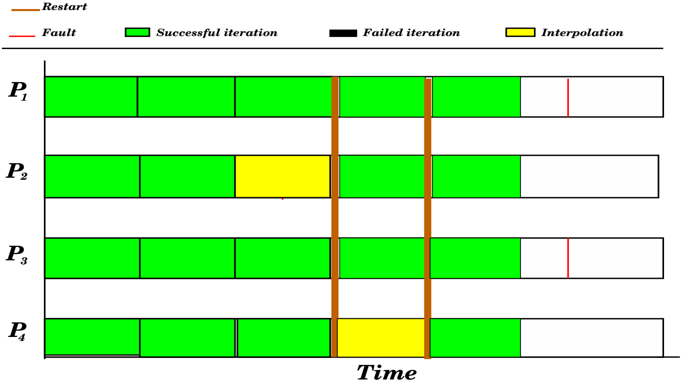

- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm




- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Overview of our fault tolerant algorithm




- Sequential simulations
- Simulation of parallel environment
- Generation of fault trace
- Realistic probability distribution

# Interpolation methods

## Fault in linear system

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

# Interpolation methods

Fault in linear system

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} ? \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$ How to recover $x_1$?

# Interpolation methods

Fault in linear system

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} ? \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$ How to recover $x_1$?

Linear Interpolation (LI) [Langou, Chen, Bosilca, Dongarra, SISC, 2007]

Solve $A_{11}x_1 = b_1 - A_{12}x_2$

# Interpolation methods

## Fault in linear system

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} ? \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$ How to recover $x_1$?

## Linear Interpolation (LI) [Langou, Chen, Bosilca, Dongarra, SISC, 2007]

Solve $A_{11}x_1 = b_1 - A_{12}x_2$

## Least Squares Interpolation (LSI)

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} x_1 + \begin{pmatrix} A_{21} \\ A_{22} \end{pmatrix} x_2 = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

$$x_1 = \underset{x}{argmin} \left\| \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} - \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} x - \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} x_2 \right\|_2$$

# Main properties - basic linear algebra

## Proposition

The initial guess generated by LI after a fault does ensure that the A-norm of the forward error associated with the iterates computed by restarted CG or PCG is monotonically decreasing

# Main properties - basic linear algebra

## Proposition

The initial guess generated by LI after a fault does ensure that the A-norm of the forward error associated with the iterates computed by restarted CG or PCG is monotonically decreasing
**[LI might not be defined for non-SPD matrices as diagonal blocks might be singular]**

# Main properties - basic linear algebra

## Proposition

The initial guess generated by LI after a fault does ensure that the A-norm of the forward error associated with the iterates computed by restarted CG or PCG is monotonically decreasing
**[LI might not be defined for non-SPD matrices as diagonal blocks might be singular]**

## Proposition

The initial guess generated by LSI after a fault does ensure the monotonic decrease of the residual norm of minimal residual Krylov subspace methods such as GMRES and MinRES after a restarting due to a failure

# Outline

# Impact of fault rate

## Preconditioned GMRES (Kim1 - 2 % data lost)

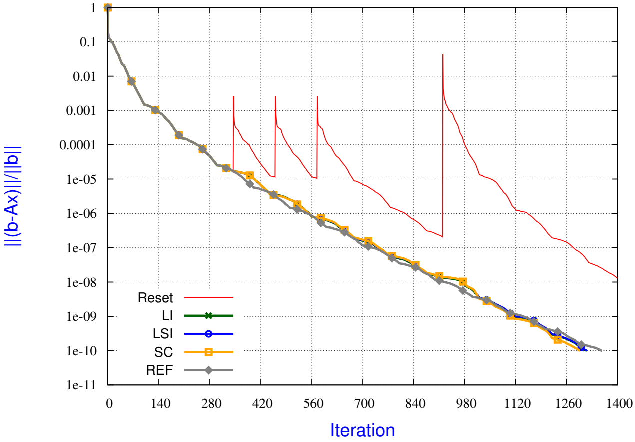



Figure: 4 faults

# Impact of fault rate

Preconditioned GMRES (Kim1 - 2 % data lost)

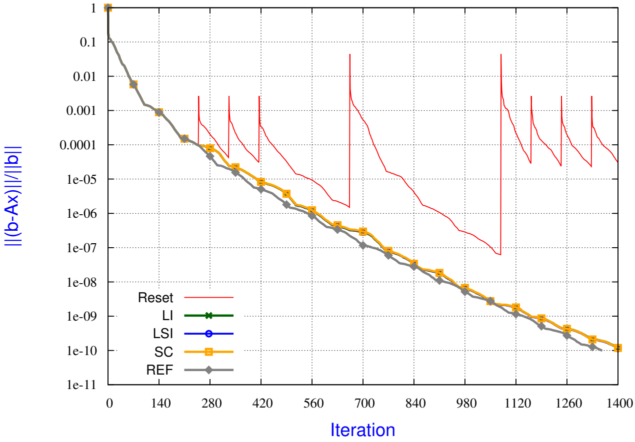



Figure: 8 faults

# Impact of fault rate

## Preconditioned GMRES (Kim1 - 2 % data lost)

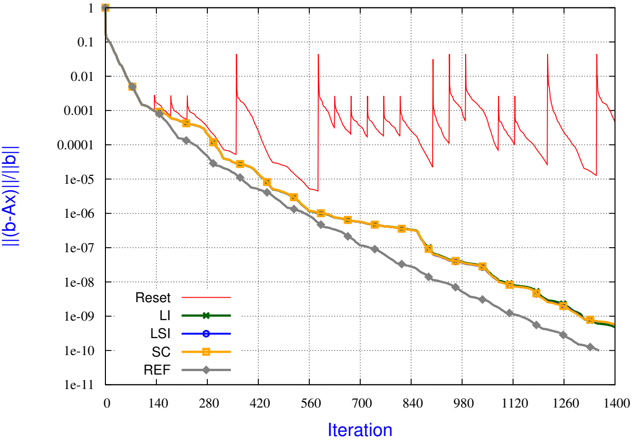



Figure: 17 faults

# Impact of fault rate

Preconditioned GMRES (Kim1 - 2 % data lost)

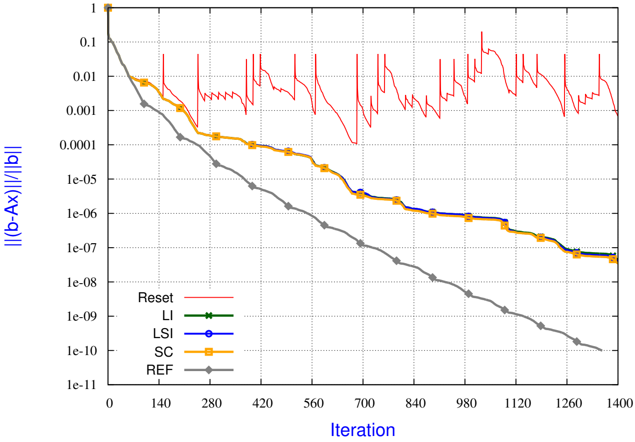



Figure: 40 faults

# Impact of lost data volume

## Preconditioned GMRES(100) (Averous/epb3 - 10 faults)

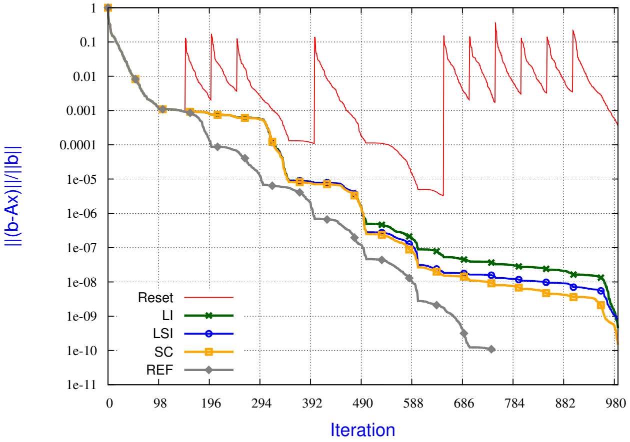



Figure: 3 % data lost

# Impact of lost data volume

Preconditioned GMRES(100) (Averous/epb3 - 10 faults)

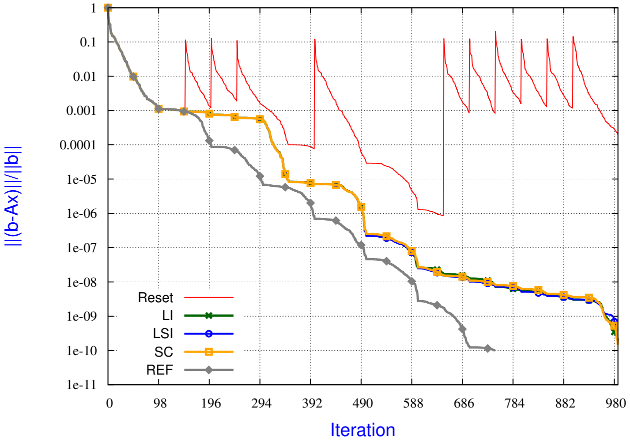



Figure: 0.8 % data lost

# Impact of lost data volume

## Preconditioned GMRES(100) (Averous/epb3 - 10 faults)

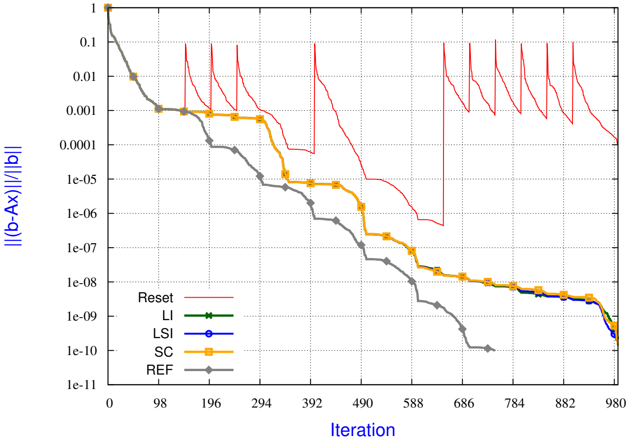



Figure: 0.2 % data lost

# Impact of lost data volume

## Preconditioned GMRES(100) (Averous/epb3 - 10 faults)

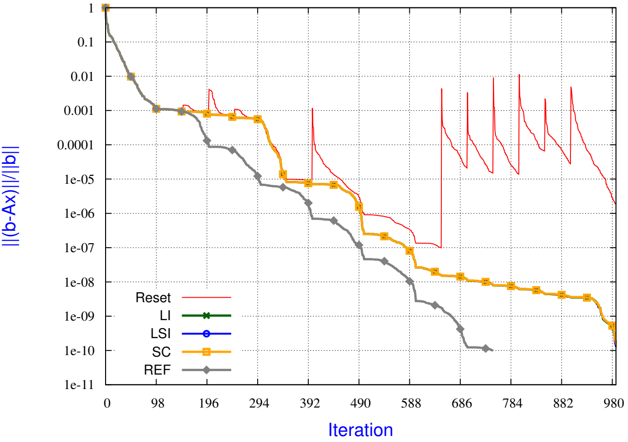



Figure: 0.001 % data lost

# Penalty of restart strategy

- Recover-restart strategy
- When restarting, we lose the Krylov subspace built before the fault
- Consequence: delay of convergence due to restart
- Restarting mechanism is naturally implemented in GMRES to reduce the computational resource consumption
- CG does not need to be restarted

# Penality of restart strategy on PCG

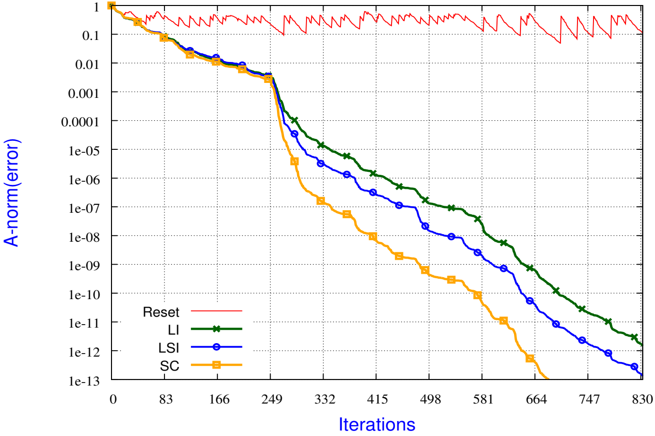



Figure: PCG on a 7-point stencil 3D Poisson equation with 70 faults - 5 % data lost

# Penality of restart strategy on PCG

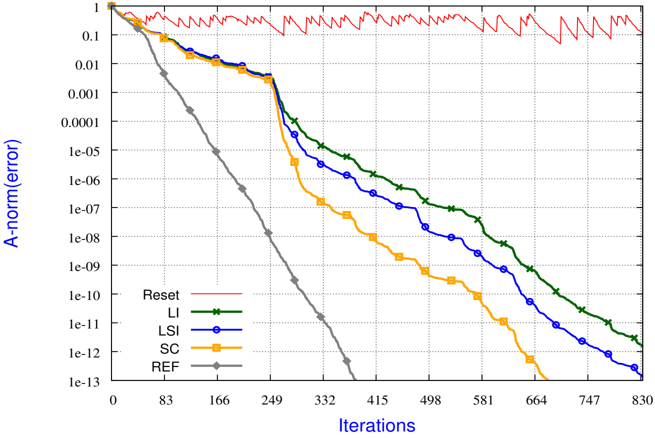



Figure: PCG on a 7-point stencil 3D Poisson equation with 70 faults - 5 % data lost

# Outline

# Recovery-restart for eigensolvers

## Fault in eigenproblem

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

# Recovery-restart for eigensolvers

## Fault in eigenproblem

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} ? \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} ? \\ x_2 \end{pmatrix}$$ How to recover $x_1$?

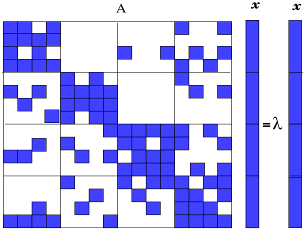
## Linear Interpolation (LI)

Solve the linear system $\left(A_{11} - \lambda I_1\right) x_1 = -A_{12} x_2$

## Least Squares Interpolation (LSI)

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} x_1 + \begin{pmatrix} A_{21} \\ A_{22} \end{pmatrix} x_2 = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$x_1 = \underset{x}{argmin} \left\| \begin{pmatrix} A_{11} - \lambda I_1 \\ A_{21} \end{pmatrix} x + \begin{pmatrix} A_{12} \\ A_{22} - \lambda I_2 \end{pmatrix} x_2 \right\|_2$$



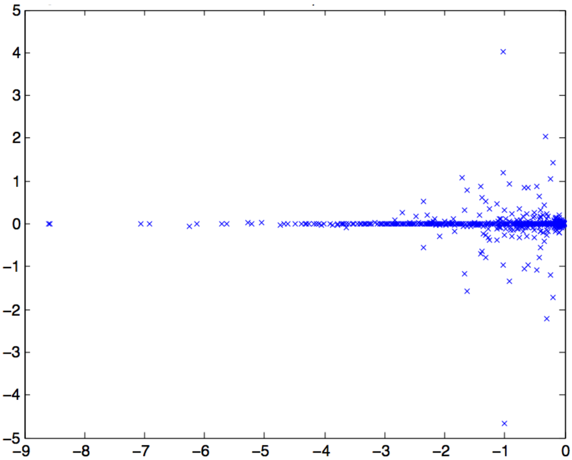

If $Ax = \lambda x$ with $x \neq 0$, where $A \in \mathbb{C}^{n \times n}$, $x \in \mathbb{C}^n$, and $\lambda \in \mathbb{C}$ , then,

- $\lambda$ : eigenvalue
- $x$ : eigenvector
- $(\lambda, x)$ : eigenpair

## Two classes of methods

- Fixed Point Methods (Power Method, Subspace iteration)
- Subpace Methods (Jacobi-Davidson, Arnoldi, IRA/Krylov Schur)

# Thermo-acoustic test example




(a few smallest eigenvalues)

# Jacobi-Davidson method

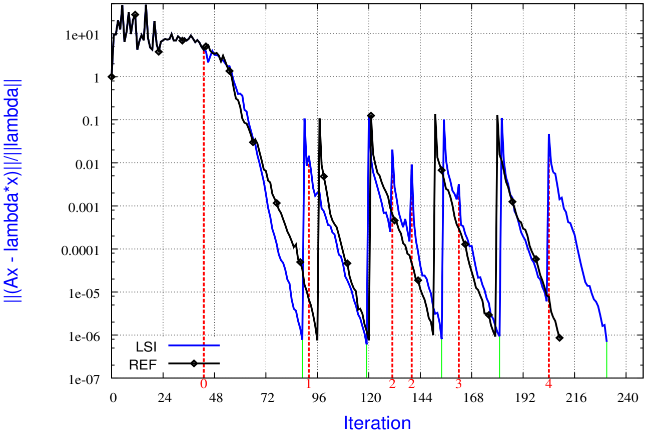



Figure: Jacobi-Davidson method with 5 faults - 1 % lost data. Convergence history using LSI and Checkpoint of current iterate

# Outline

# Concluding remarks

## Summary

- We have designed techniques to interpolate meaningfull lost data based on simple linear algebra tools
- Our techniques preserve some of the key monotonicy of Krylov solvers but lack of robustness of LI for non-SPD problems
- The restarting effect remains reasonable within the GMRES context
- No fault, no overhead
- These techniques can be adpated to multiple faults
- What about silent soft-error - CGPOP preliminary experiments ?

Merci for your attention

Questions ?

https://team.inria.fr/hiepacs/