Recovery policies for Krylov solver resiliency

Sparse day 2013
Toulouse, France.

Emmanuel AGULLO, Luc GIRAUD,
Abdou GUERMOUCHE, Jean ROMAN,
Mawussi ZOUNON

PROJECT-TEAM
HiePACS
Joint lab Inria-CERFACS
FRANCE

Agence Nationale de la Recherche
ANR

HARDWARE ERROR

- HPC systems are not fault-free.
- A faulty processor loose all its data.
- Applications have to be resilient.

Introduction

HARDWARE ERROR

- HPC systems are not fault-free.
- A faulty processor loose all its data.
- Applications have to be resilient.

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

HARDWARE ERROR

- HPC systems are not fault-free.
- A faulty processor loose all its data.
- Applications have to be resilient.

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

- Goal: Keep converging in presence of fault.
- Method: Re-generate lost data without Checkpoint/Restart strategy.
- Approach: Numerical algorithm.
- Context: Krylov solvers.

Outline

1. Faults in HPC Systems
2. Iterative methods for sparse linear systems
3. Our model assumptions
4. Interpolation methods
5. Numerical experiments
6. Concluding remarks and perspectives

Outline

Framework

Forecast for exascale systems

- Mean Time Between Failure (MTBF): less then one hour.
- Checkpoint overhead:
  - 30 minutes per checkpoint.
  - 1 Terabyte/second.

- Limitation of classical checkpointing.
- Explore fault-tolerant schemes with less/no overhead.
- Numerical algorithms to deal with overhead issue.

Faults in this presentation

- core crashes (memory, caches, network connections, . . .)

Framework

Forecast for exascale systems

- ⋆ Mean Time Between Failure (MTBF): less then one hour.
- ⋆ Checkpoint overhead:
  - 30 minutes per checkpoint.
  - 1 Terabyte/second.

- ⋆ Limitation of classical checkpointing.
- ⋆ Explore fault-tolerant schemes with less/no overhead.
- ⋆ Numerical algorithms to deal with overhead issue.

Faults in this presentation

- ⋆ core crashes (memory, caches, network connections, . . .)

Framework

Forecast for exascale systems

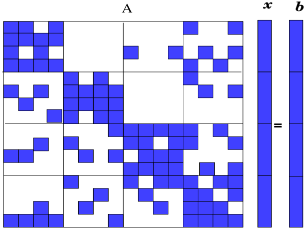
- ⋆ Mean Time Between Failure (MTBF): less then one hour.
- ⋆ Checkpoint overhead:
  - 30 minutes per checkpoint.
  - 1 Terabyte/second.

- ⋆ Limitation of classical checkpointing.
- ⋆ Explore fault-tolerant schemes with less/no overhead.
- ⋆ Numerical algorithms to deal with overhead issue.

Faults in this presentation

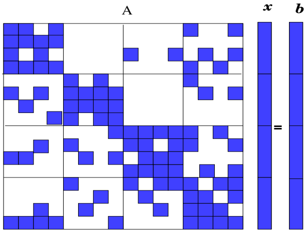
- ⋆ core crashes (memory, caches, network connections, . . .)

Outline



$$Ax = b.$$

We have to design fault tolerant solver for sparse linear system.

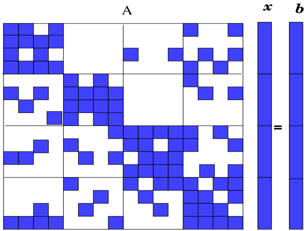


$$Ax = b.$$

We have 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, . . .).



$$Ax = b.$$

We have to design fault tolerant solver for sparse linear system.

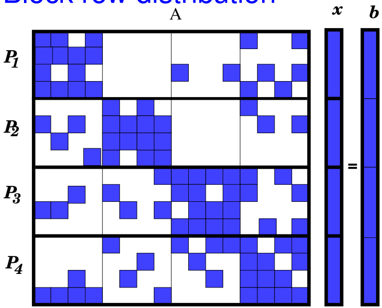
Two classes of iterative methods

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

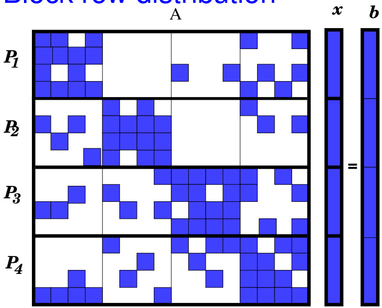
⋆ Krylov methods have attractive potential for Extreme-scale.

Outline

Block row distribution



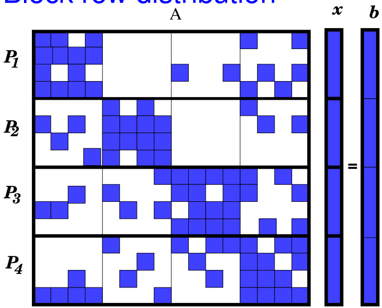
Block row distribution



We distinguish two categories of data:

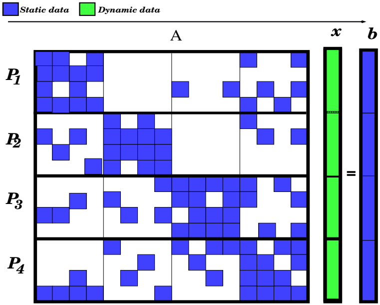
- Static data.
- Dynamic data.

Block row distribution



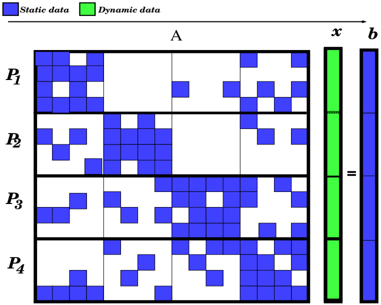
We distinguish two categories of data:

- ⋆ Static data.
- ⋆ Dynamic data.



We distinguish two categories of data:

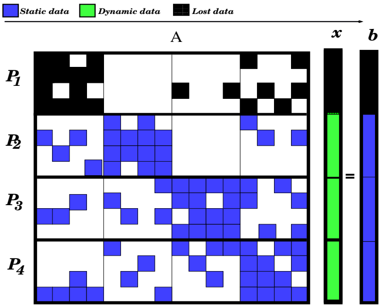
- ★ Static data.
- ★ Dynamic data.



We distinguish two categories of data:

- Static data.
- Dynamic data.

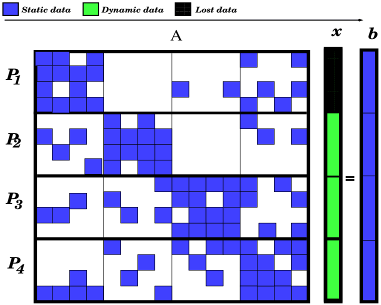
Let's Assume that P_1 fails.



We distinguish two categories of data:

- ★ Static data.
- ★ Dynamic data.

Let's Assume that P_1 fails.

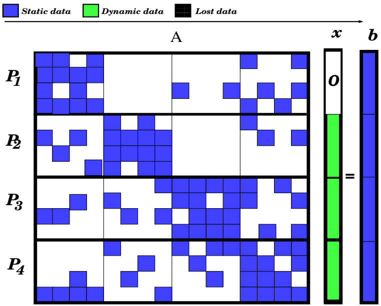


We distinguish two categories of data:

- ★ Static data.
- ★ Dynamic data.

Let's Assume that P_1 fails.

- ★ Failed processor is replaced.
- ★ Static data are restored.

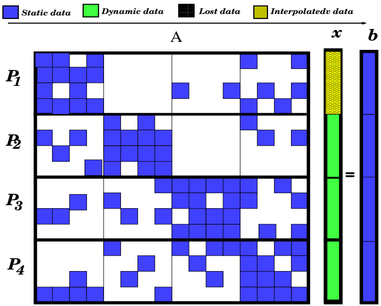


We distinguish two categories of data:

- ⋆ 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:

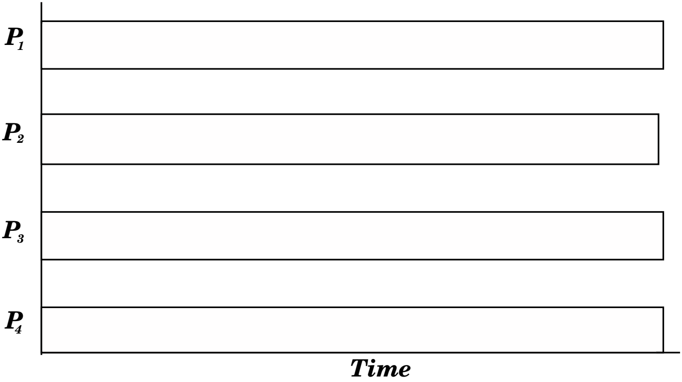
- Static data.
- Dynamic data.

Let's Assume that P_1 fails.

- Failed processor is replaced.
- Static data are restored.
- Reset: Set (x_1) to initial value.

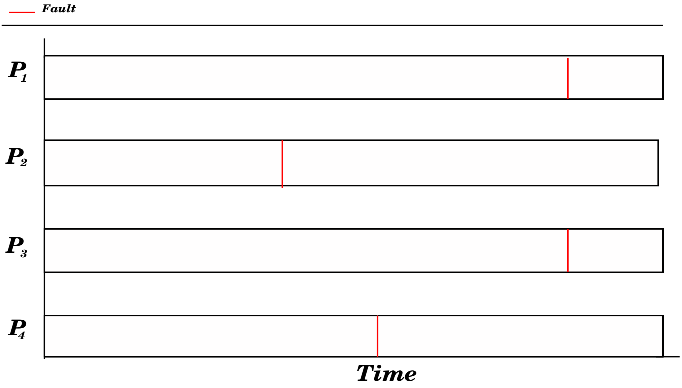
Our algorithms aim at recovering x_1.

Overview of our fault tolerant algorithm



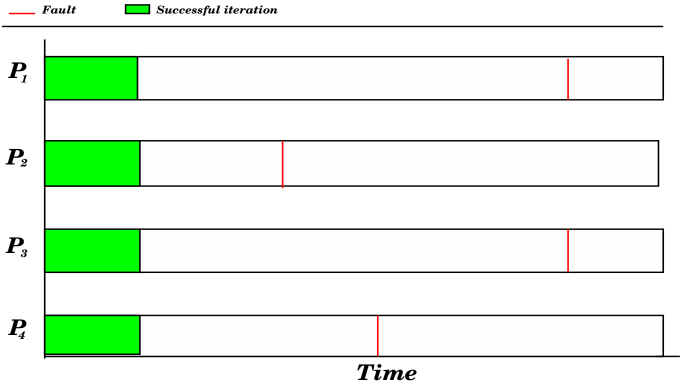
- Matlab prototype.
- Simulation of parallel environment.

Overview of our fault tolerant algorithm



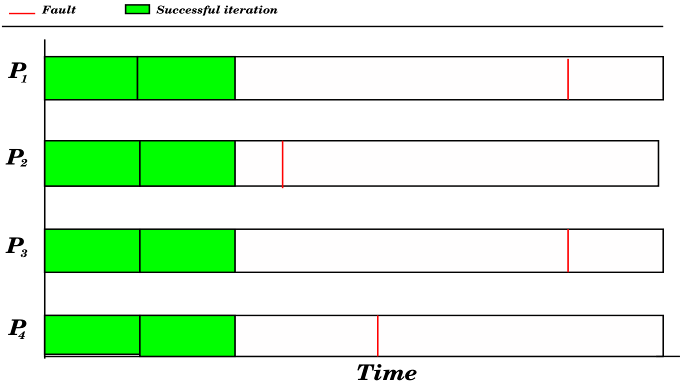
- Matlab prototype.
- Simulation of parallel environment.
- Generation of fault trace.
- Realistic probability distribution.

Overview of our fault tolerant algorithm



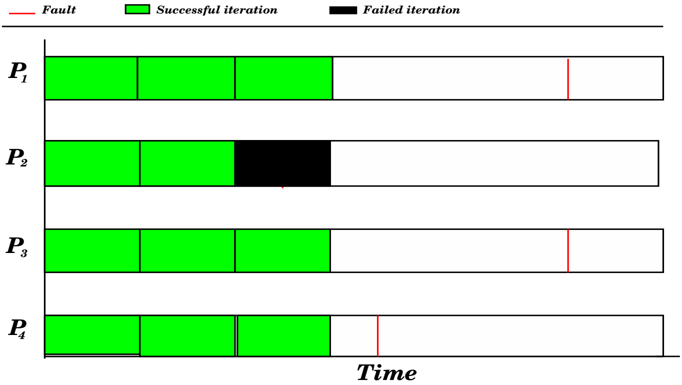
- Matlab prototype.
- Simulation of parallel environment.
- Generation of fault trace.
- Realistic probability distribution.

Overview of our fault tolerant algorithm



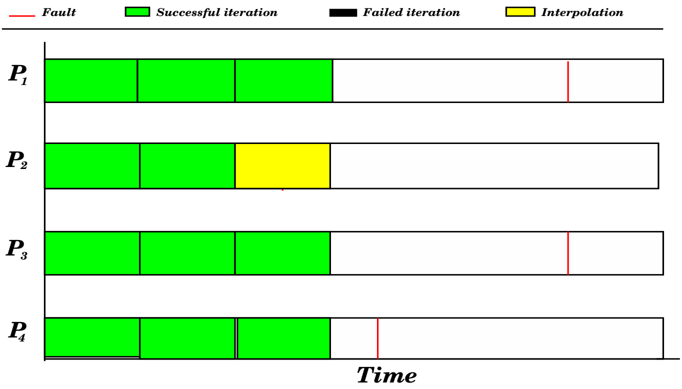
- Matlab prototype.
- Simulation of parallel environment.
- Generation of fault trace.
- Realistic probability distribution.

Overview of our fault tolerant algorithm



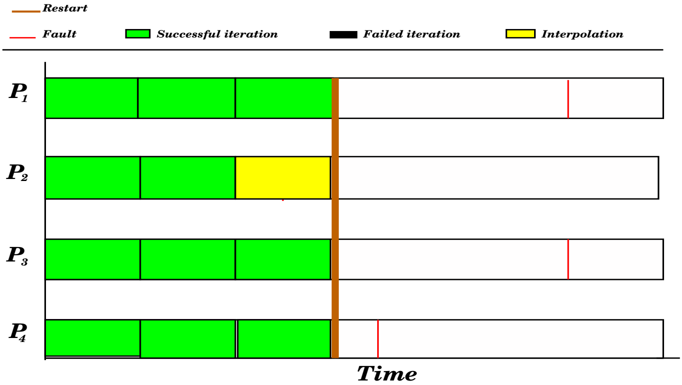
- ⋆ Matlab prototype.
- ⋆ Simulation of parallel environment.
- ⋆ Generation of fault trace.
- ⋆ Realistic probability distribution.

Overview of our fault tolerant algorithm



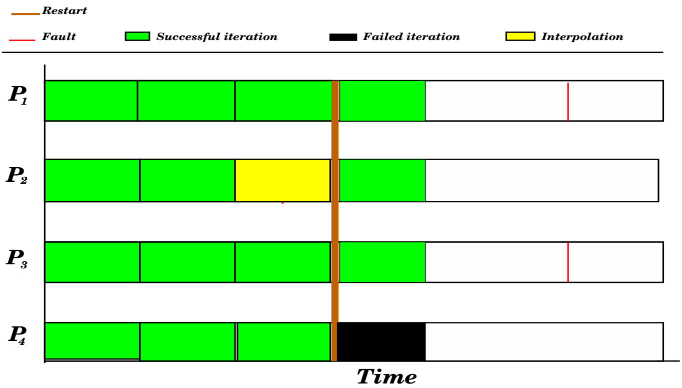
- Matlab prototype.
- Simulation of parallel environment.
- Generation of fault trace.
- Realistic probability distribution.

Overview of our fault tolerant algorithm



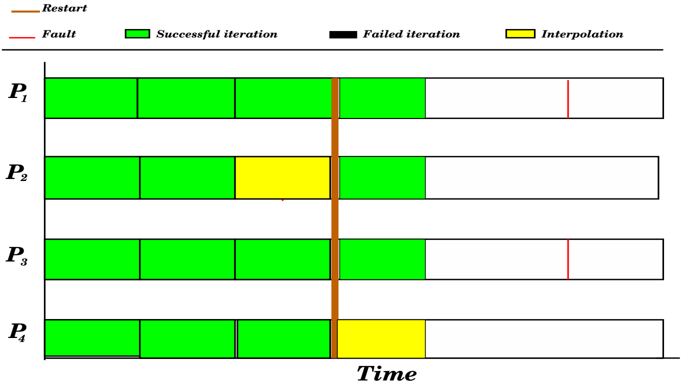
- ★ Matlab prototype.
- ★ Simulation of parallel environment.
- ★ Generation of fault trace.
- ★ Realistic probability distribution.

Overview of our fault tolerant algorithm



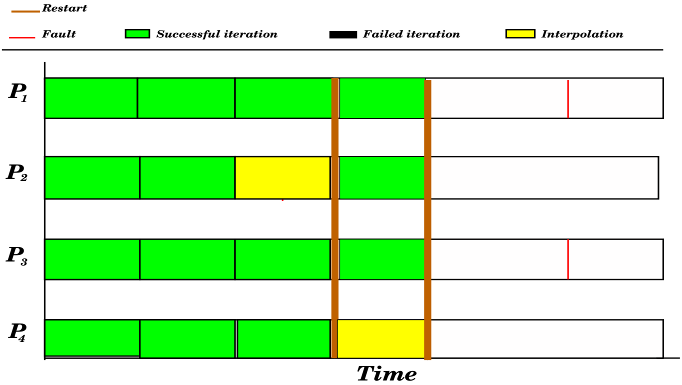
- ⋆ Matlab prototype.
- ⋆ Simulation of parallel environment.
- ⋆ Generation of fault trace.
- ⋆ Realistic probability distribution.

Overview of our fault tolerant algorithm



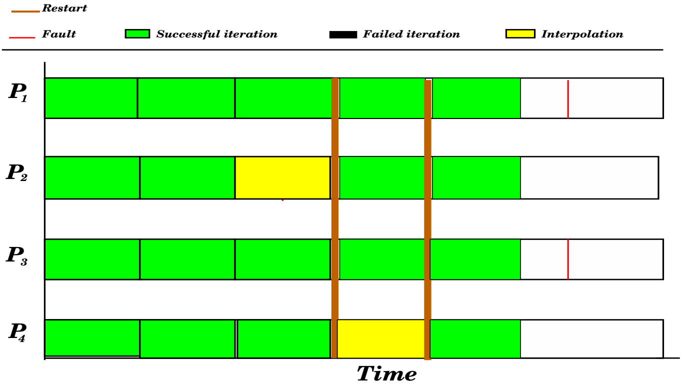
- Matlab prototype.
- Simulation of parallel environment.
- Generation of fault trace.
- Realistic probability distribution.

Overview of our fault tolerant algorithm



- ★ Matlab prototype.
- ★ Simulation of parallel environment.
- ★ Generation of fault trace.
- ★ Realistic probability distribution.

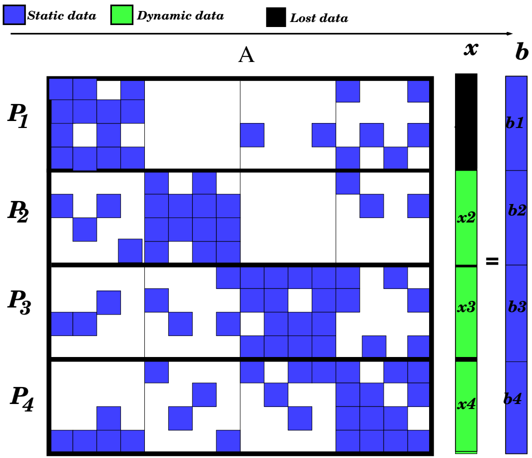
Overview of our fault tolerant algorithm



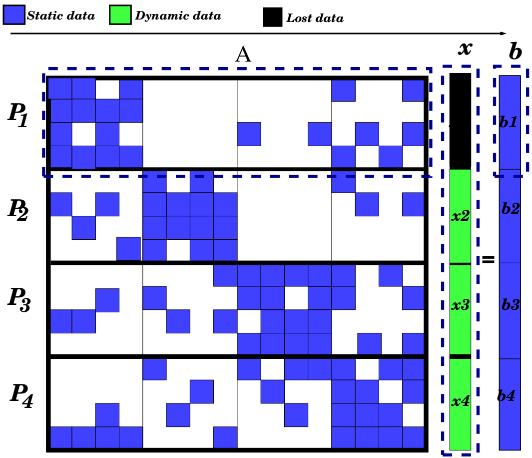
- ⋆ Matlab prototype.
- ⋆ Simulation of parallel environment.
- ⋆ Generation of fault trace.
- ⋆ Realistic probability distribution.

Outline

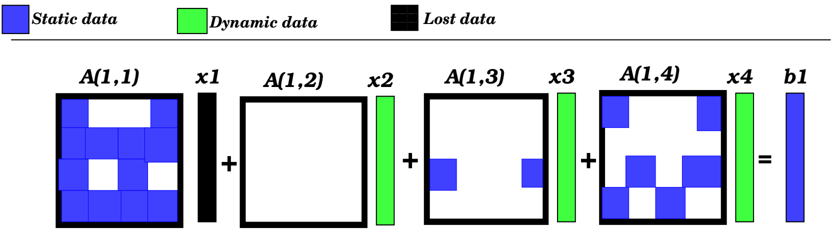
Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]



Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]

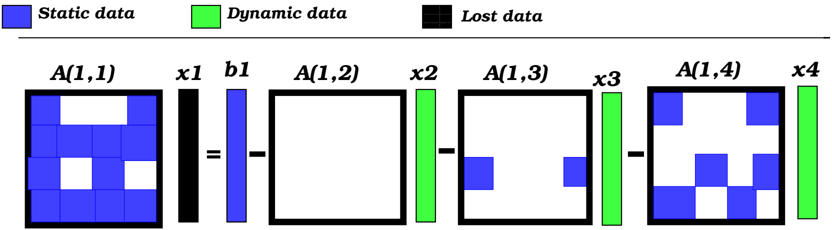


Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]



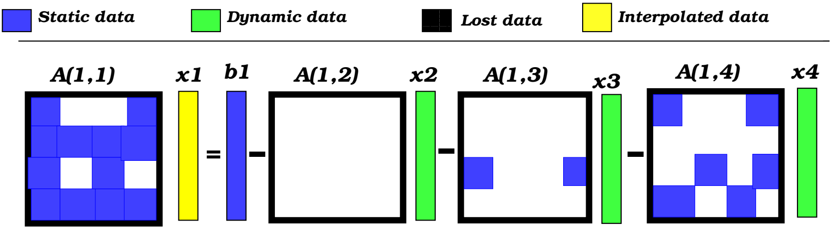
$$A_{(1,1)}x_1 + A_{(1,2)}x_2 + A_{(1,3)}x_3 + A_{(1,4)}x_4 = b_1.$$

Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]



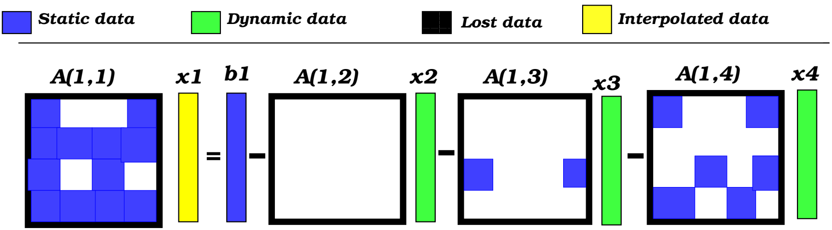
$$A_{(1,1)}x_1 = b_1 - A_{(1,2)}x_2 - A_{(1,3)}x_3 - A_{(1,4)}x_4.$$

Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]



$$A_{(1,1)}x_1 = b_1 - A_{(1,2)}x_2 - A_{(1,3)}x_3 - A_{(1,4)}x_4.$$

Linear Interpolation (LI) [Julien Langou et al, SIAM J. Sci, 2007]



$$A_{(1,1)}x_1 = b_1 - A_{(1,2)}x_2 - A_{(1,3)}x_3 - A_{(1,4)}x_4.$$

$$A_{(i,i)}x_i^{(new)} = b_i - \sum_{i \neq j} A_{(i,j)}x_j.$$

Linear Interpolation (LI)

Proposition

Let A be symmetric positive definite (SPD).The recovered entries defined by LI strategie are always uniquely defined. Furthermore, let $e^{(fail)} = x^{sol} - x^{(fail)}$, denotes the forward error associated with the current iterate, and $e^{(new)}$ be the forward error associated with the new initial guess recovered using the LI strategy, we have

$$\|e^{(new)}\|_A^2 \leq \|e^{(fail)}\|_A^2.$$

Linear Interpolation (LI)

Proposition

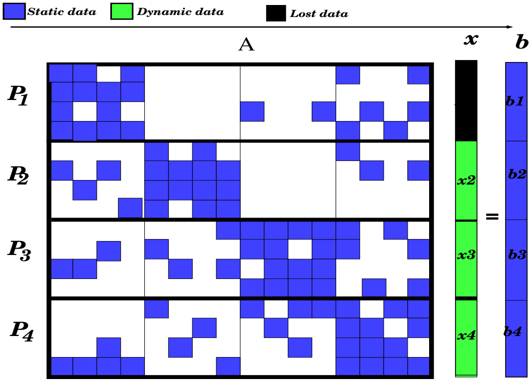
Let A be symmetric positive definite (SPD).The recovered entries defined by LI strategie are always uniquely defined. Furthermore, let $e^{(fail)} = x^{sol} - x^{(fail)}$, denotes the forward error associated with the current iterate, and $e^{(new)}$ be the forward error associated with the new initial guess recovered using the LI strategy, we have

$$\|e^{(new)}\|_A^2 \leq \|e^{(fail)}\|_A^2.$$

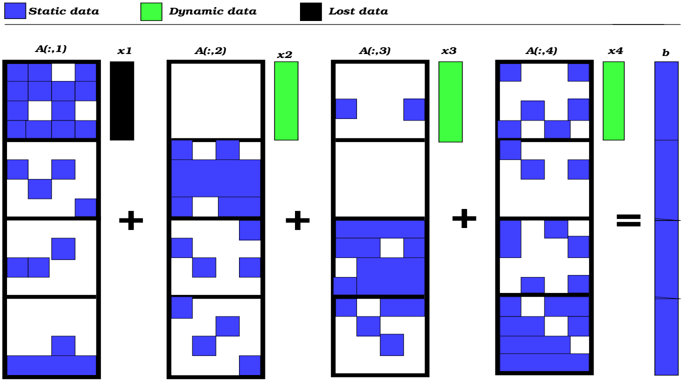
Corollary

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.

Least squares interpolation (LSI)

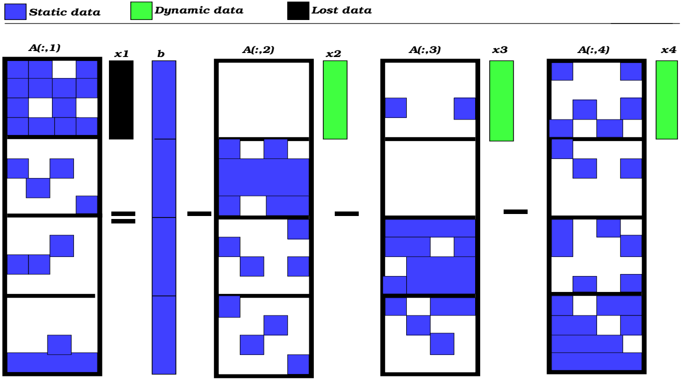


Least squares interpolation (LSI)



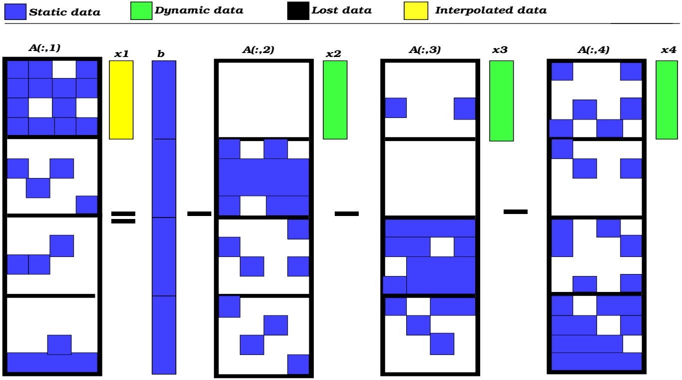
$$A_{(:,1)}x_1 + A_{(:,2)}x_2 + A_{(:,3)}x_3 + A_{(:,4)}x_4 = b.$$

Least squares interpolation (LSI)



$$x_1 = \underset{x}{argmin} \left\| \left(b - A_{(:,2)}x_2 - A_{(:,3)}x_3 - A_{(:,4)}x_4 \right) - A_{(:,1)}x \right\|_2.$$

Least squares interpolation (LSI)



$$x_1 = \underset{x}{argmin} \left\| \left(b - A_{(:,2)} x_2 - A_{(:,3)} x_3 - A_{(:,4)} x_4 \right) - A_{(:,1)} x \right\|_2.$$

Least squares interpolation (LSI)

Proposition

The recovered entries defined by LSI strategie are always uniquely defined. Furthermore Let $r^{(fail)} = b - Ax^{(fail)}$ denote the residual associated with the iterate when the fault occurs, and $r^{(new)}$ be the residual associated with the initial guess generated with the LSI strategy, we have

$$\|r^{(new)}\|_2 \leq \|r^{(fail)}\|_2.$$

Least squares interpolation (LSI)

Proposition

The recovered entries defined by LSI strategie are always uniquely defined. Furthermore Let $r^{(fail)} = b - Ax^{(fail)}$ denote the residual associated with the iterate when the fault occurs, and $r^{(new)}$ be the residual associated with the initial guess generated with the LSI strategy, we have

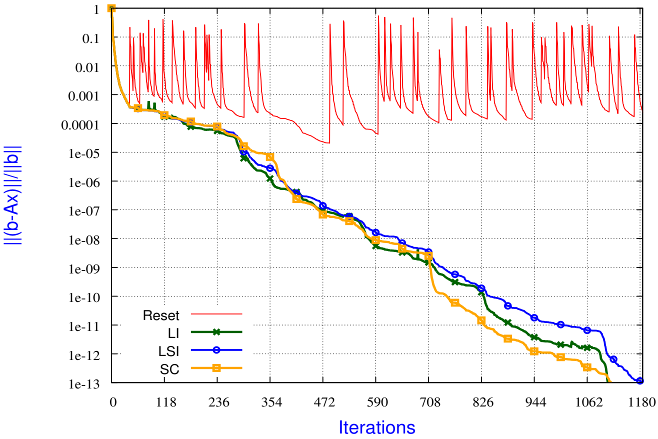
$$\|r^{(new)}\|_2 \leq \|r^{(fail)}\|_2.$$

Corollary

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.

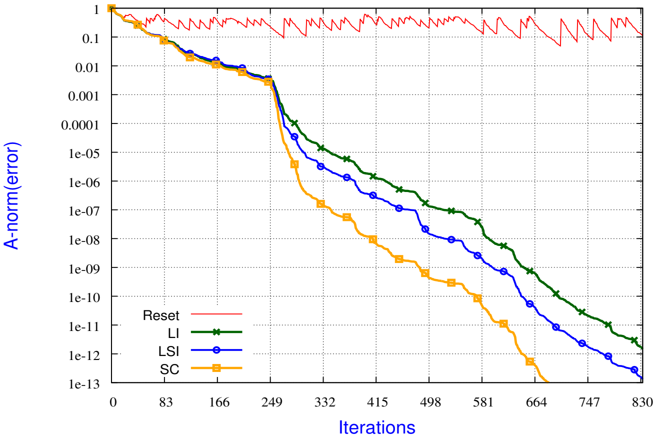
Outline

Preconditioned GMRES



Right block diagonal Preconditioned GMRES on UF Averous/epb0 using 16 cores with 44 faults

PCG

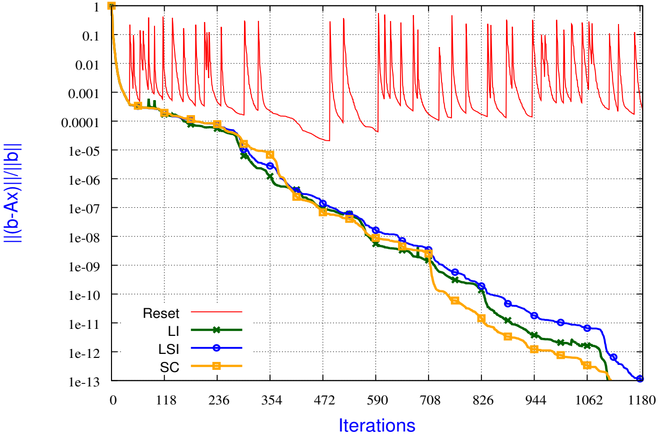


PCG on a 7-point stencil 3D Poisson equation using 16 cores with 70 faults

Impact of restart strategy

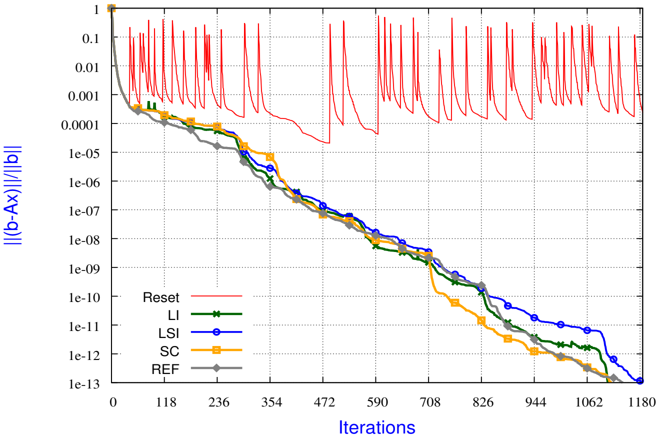
- Interpolate/restart strategy.
- When restarting, we loose krylov subspace built before fault.
- Consequence: Delay of convergence.
- Restarting mechanism is naturally implemented in GMRES to reduce the computational resource consumption.
- CG and BiCGStab do not need to be restarted.

Impact of restart strategy on GMRES



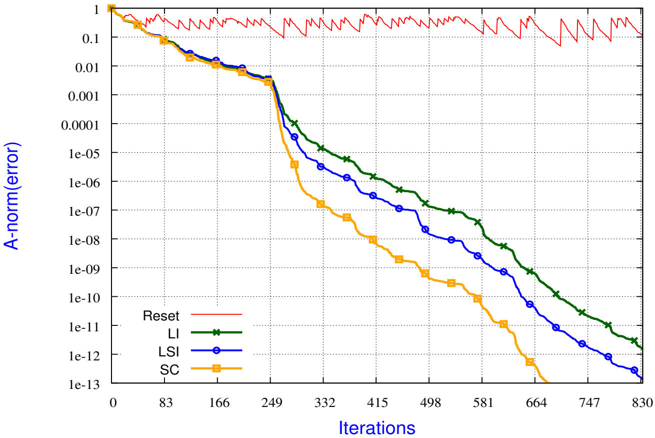
Right block diagonal Preconditioned GMRES on UF Averous/epb0 using 16 cores with 44 faults

Impact of restart strategy on GMRES



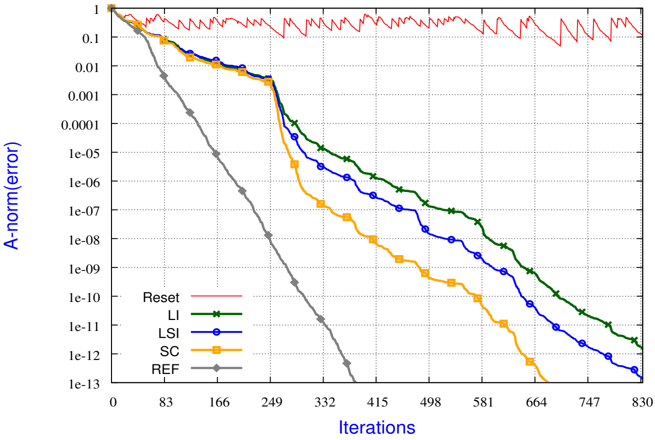
Right block diagonal Preconditioned GMRES on UF Averous/epb0 using 16 cores with 44 faults

Impact of restart strategy on PCG



PCG on a 7-point stencil 3D Poisson equation using 16 cores with 70 faults

Impact of restart strategy on PCG



PCG on a 7-point stencil 3D Poisson equation using 16 cores with 70 faults

Outline

Concluding remarks

Concluding remarks

- ⋆ We have designed techniques to interpolate meaningfull lost data.
- ⋆ Our techniques preserve some of the key monotonicy of Krylov solvers.
- ⋆ The restarting effect remains reasonable within the GMRES context.
- ⋆ No fault, no overhead.
- ⋆ Generalised to multiple faults.

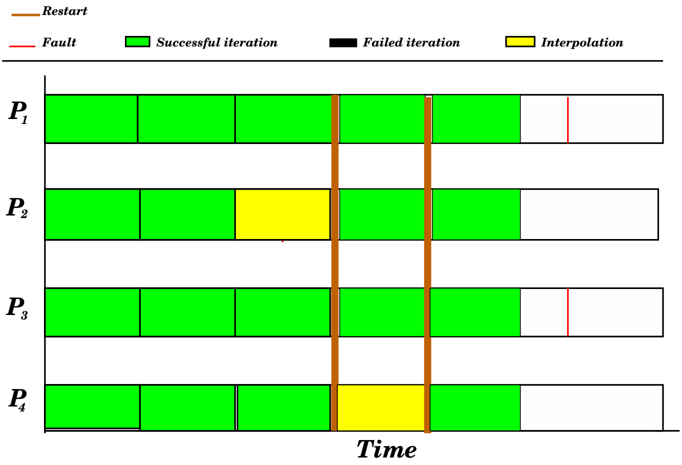
Perspectives

- ⋆ Study of others resilience schemes.
- ⋆ Best combination of interpolation and selective checkpoint.
- ⋆ Real implementation subjected to fault tolerant MPI implementation.

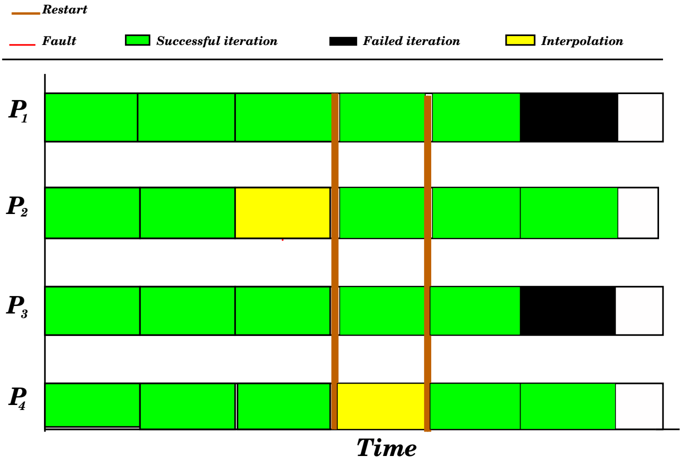
Thank you for listening

http://hiepacs.bordeaux.inria.fr/

Multiple Faults

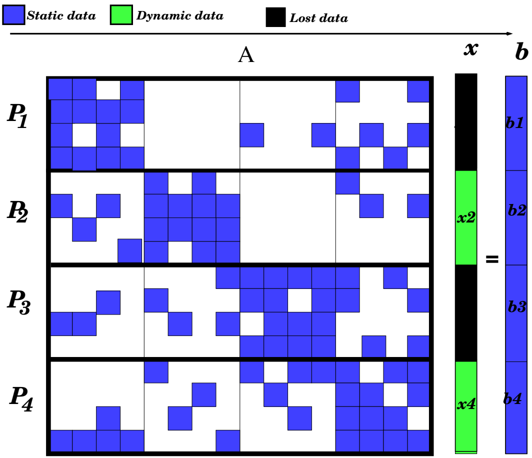


Multiple Faults



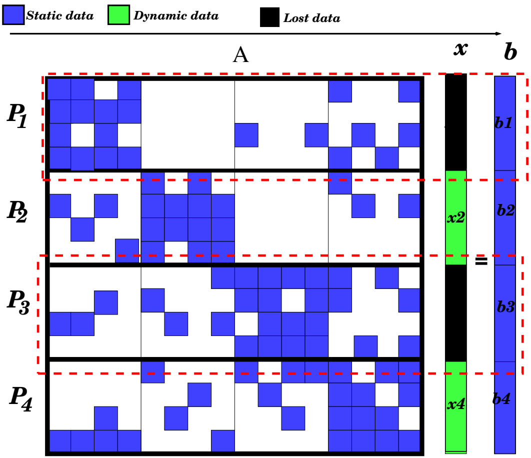
Multiple faults: more than one fault at the same iteration.

Multiple Faults

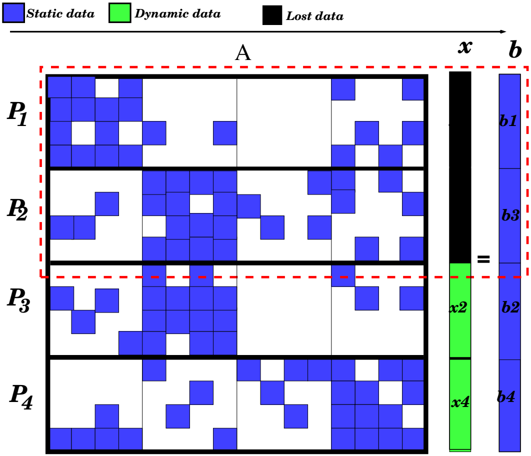


- ★ x_3 is needed to interpolate x_1, vice-versa.
- ★ How to deal with data dependency?

Assembled recovery: LI-A/LSI-A

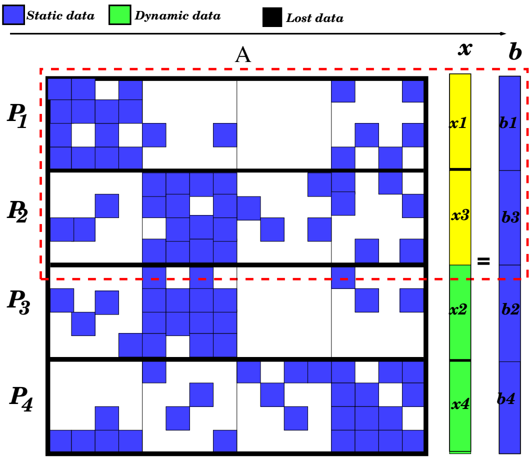


Assembled recovery: LI-A/LSI-A



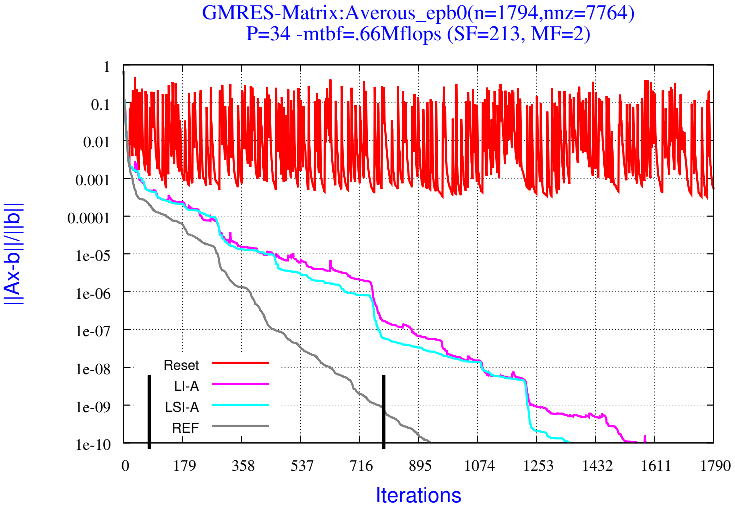
Failed blocks are assembled.

Assembled recovery: LI-A/LSI-A



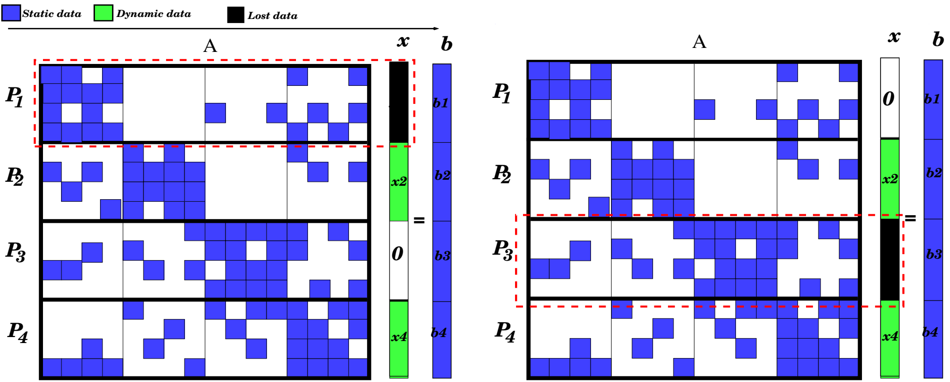
Failed blocks are assembled.

Assembled recovery: LI-A/LSI-A



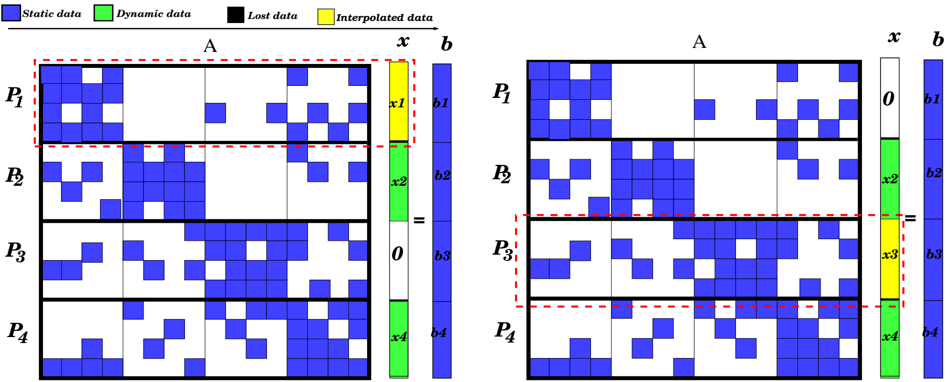
- 2 multiple faults.
- 56^{th} iteration and 784^{th} iteration.

Parallel recovery: LI-P/LSI-P



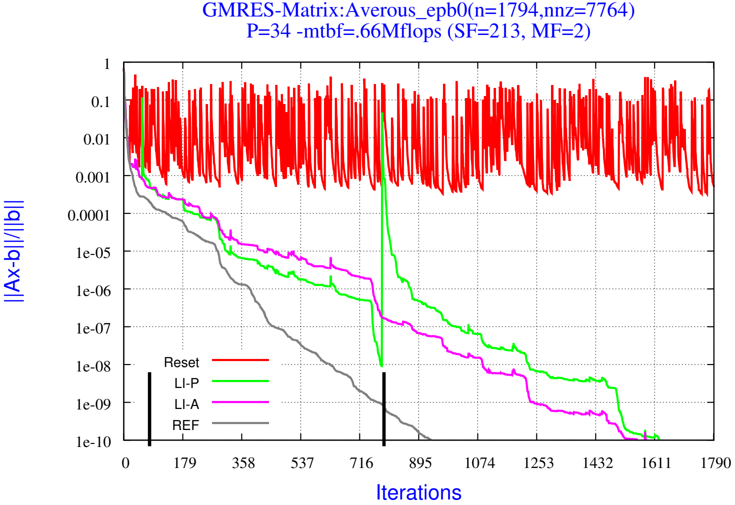
Interpolate x_3 assuming that x_1 is equal to zero subvector.
Interpolate x_1 assuming that x_3 is equal to zero subvector.

Parallel recovery: LI-P/LSI-P



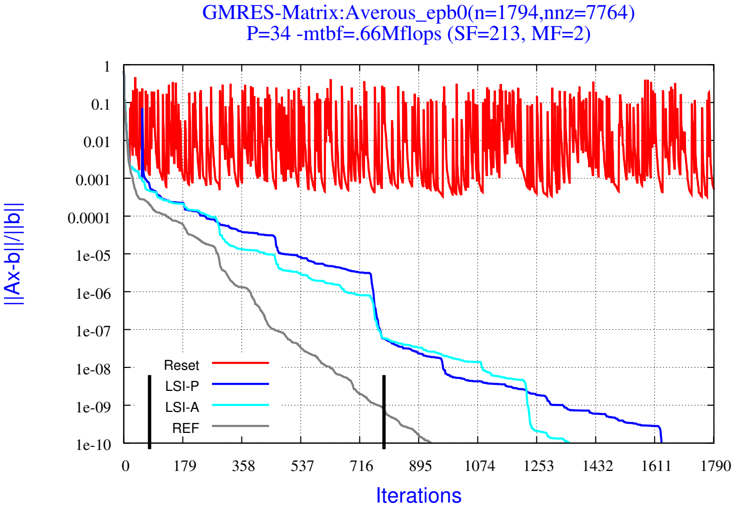
Interpolate x_3 assuming that x_1 is equal to zero subvector.
Interpolate x_1 assuming that x_3 is equal to zero subvector.

LI-P



- ⋆ 2 multiple faults.
- ⋆ 56^{th} iteration and 784^{th} iteration.

LSI-P



- 2 multiple faults.
- 56^{th} iteration and 784^{th} iteration.