Toward robust numerical linear solvers for large scale simulations

LUC GIRAUD

joint work with E. Agullo, A. Guermouche, J. Roman and M. Zounon

HiePACS
joint Inria-CERFACS Lab. on High Performance Computing
Inria / Univ. Bordeaux / CNRS (LaBRI)

Inria-NCSA workshop, Nov. 22, 2011

This work was supported in part by the ANR RESCUE project

Sparse linear systems

Linear system

Find x such that: $\mathcal{A}x = b$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- ⋆ $\mathcal{A}$ is an $n \times n$ real matrix.
- ⋆ b is a real n-vector.

Sparse linear systems

Linear system

Find x such that: $\mathcal{A}x = b$

$$\begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ a_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

- $\mathcal{A}$ is an $n \times n$ real matrix.
- b is a real n-vector.

Application consumers of linear solvers

Scientific and engieneering application areas

- ⋆ Accelerator physics
- ⋆ Chemical process simulations
- ⋆ Earth and environmental sciences including climate
- ⋆ Fluid flow
- ⋆ Fusion energy
- ⋆ Structural analysis
- ⋆ Structural biology

Iterative methods

Principle

Iterative methods for solving linear system $Ax = b$, begin with initial guess for solution and successively improve it until solution is as accurate as desired.

Two main classes

- Stationary methods (fixed point schemes: e.g., Jacobi, Gauss-Seidel, ...).
- Krylov subspace methods (CG [Hestenes, Stiefel, NIST, 52],. GMRES [Saad, Schultz, SISC, 86], Bi-CGStab [Vorst, SISC, 92], ...)

Iterative methods

Principle

Iterative methods for solving linear system $Ax = b$, begin with initial guess for solution and successively improve it until solution is as accurate as desired.

Two main classes

- ⋆ Stationary methods (fixed point schemes: e.g., Jacobi, Gauss-Seidel, ...).
- ⋆ Krylov subspace methods (CG [Hestenes, Stiefel, NIST, 52],. GMRES [Saad, Schultz, SISC, 86], Bi-CGStab [Vorst, SISC, 92], ...)

General framework

Faults in the literature

- ⋆ Soft errors.
- ⋆ Hard faults.

In this presentation

- ⋆ Hard fault: invalid core (memory, caches, network connections, ...).
- ⋆ Assumption: when a fault occurs we can start a new process on another core.

Outline

1. Fixed point schemes and resilience
2. Recovery strategies in Krylov subspace methods
3. Preliminary experimental results
4. Concluding remarks and perspectives

Mathematical models: fixed point iteration

$$x^* = F(x^*) \Rightarrow x^{k+1} = F(x^k)$$

Governing ideas to express parallelism

Split the problem in sub-problems, solve the subproblems in parallel with updates along the interfaces using available data (no-synchronization).

Chaotic/Asynchronous scheme definition

Let $x^0 \in E$ we consider the series of iterates defined by:

$$\forall k \in \mathbb{N},\ \forall p \in \{1, ..., m\},\ \ x_k^{k+1} = \begin{cases} x_p^k & \text{if} \quad p \notin s(k) \\ F_p(w) & \text{if} \quad p \in s(k) \end{cases}$$

where $w = (x_\ell^{c_\ell(k)})_{\ell=1,m}$, $w \in E; c_\ell(k) = k - d_\ell(k)$ accounts for delays and $s(p)$ defines the relaxation strategy
(e.g. $s(k) \equiv \{1, ..., m\}$and $d_\ell \equiv 0$ reduces to block Jacobi).

Convergence theory: contraction based

Theorem [J.C. Miellou, 75; F. Robert, 75]

Let assume that

> F is a J-contraction with respect to the fixed point x^*, that is there exists a nonnegative matrix $J \in \mathbb{R}^{m\times m}$ with $\rho(J) < 1$ such that
> $$\begin{pmatrix} \|F(x_1) - F(x_1^*)\|_1 \\ \vdots \\ \|F(x_m) - F(x_m^*)\|_m \end{pmatrix} < J. \begin{pmatrix} \|x_1 - x_1^*\|_1 \\ \vdots \\ \|x_m - x_m^*\|_m \end{pmatrix}.$$

Then chaotic relaxation scheme defines the iterates x^k converge to x^* the fixed point of F.

Brief overview on chaotic relaxation schemes

- Brief history (non-exhaustive):
  - Pioneer paper [D. Chazan and W. Miranker, LAA, 69].
  - Convergence analysis: contraction properties [D. Chazan and W. Miranker, LAA, 69],[F. Robert, LAA, 75], [L. Giraud, P. Spiteri, RAIRO, 91], order interval [J.C. Miellou, RAIRO, 75], [J.C. Miellou, D. El Baz, P. Spiteri, MathComp, 98].

    Recent book [J. Bahi,S. Contassot-Vivier, R. Couturier, Chapman & Hall, 2007].
  - Application areas: PDE [D. Amitai, A. Averbuch, M. Israeli, S. Itzikowitz, SISC, 98], DDM [A. Frommer, D. Szyld, JCAM, 97] inverse problems [V. Pereyra, ANM, 99], convex optimization [P. Tseng, SIOPT, 91], network flow [P. Tseng, D. P. Bertsekas, J. N. Tsitsiklis, SICON, 90], etc ...
  - Renewed interest with Grid computing (latency/bandwith network).

Asynchronous relaxation v.s. resilience

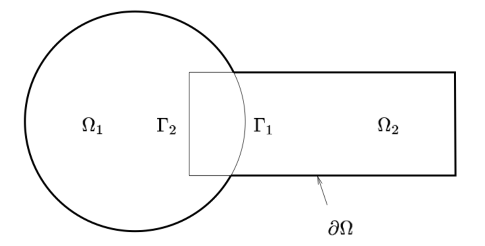
- By construction, the chaotic relaxation schemes are resilient to message loss.
- To comply with fault tolerance, the $c_k(p)$ non decreasing function of p implies uncoordinated local checkpointing of each core running on E_i (no synchronization).
- Good candidates for fault tolerance: classical fixed point iterations where convergence analysis is based on contraction properties

$$Ax = b$$

the scheme $x^{k+1} = x^k + B(b - Ax^k)$, will converge for any x^0 if $\rho(I - BA) < 1$;
e.g. classical Schwarz alternating method (1870).

Schwarz Alternating method [H.A. Schwarz, 1870]

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{in } \Omega, \\ u & = & 0 & \text{on } \partial\Omega. \end{array} \right.$$



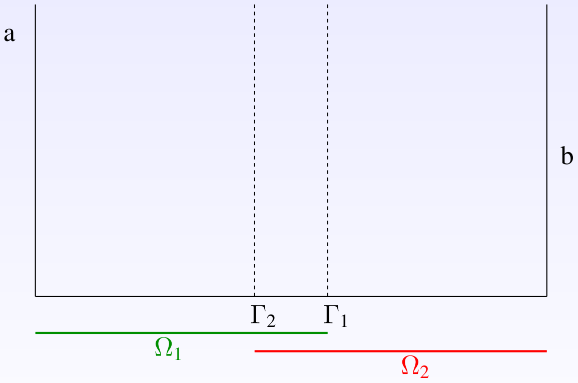
Fixed point iteration scheme

$$\left\{ \begin{array}{rcll} -\Delta u_1^{n+1} & = & f & \text{in } \Omega_1, \\ u_1^{n+1} & = & u_2^n|_{\Gamma_1} & \text{on } \Gamma_1, \end{array} \right. \quad \text{and} \quad \left\{ \begin{array}{rcll} -\Delta u_2^{n+1} & = & f & \text{in } \Omega_2, \\ u_2^{n+1} & = & u_1^{n+1}|_{\Gamma_2} & \text{on } \Gamma_1. \end{array} \right.$$

Convergence analysis based on contraction property of a product of projectors (also referred to as multiplicative Schwarz) [P.L. Lions, 88].

Schwarz Alternating method: 1D illustration

$$u'' = 0, u(0) = a, u(1) = b$$



- Contraction based analysis, $\forall u_0 \; \|u - u^k\| \leq \rho^k \|u - u^0\|$.
- *Matlab demo: visual evidence.*

Krylov subspace methods

General concept of Krylov subspace methods

Search for the slolution of a linear system of dimension n in a specific subspace of dimension k smaller than n. Basically, $x^k = F(x^0, ..., x^{k-1})$.

Krylov subspace

Let $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$, let $k \leq n$, the space denoted by $\kappa(b, A, k)$ with $\kappa(b, A, k) = Span\{b, Ab, ..., A^{k-1}b\}$ is referred to as the Krylov space of dimension k associated with A and b.

Krylov subspace methods

General concept of Krylov subspace methods

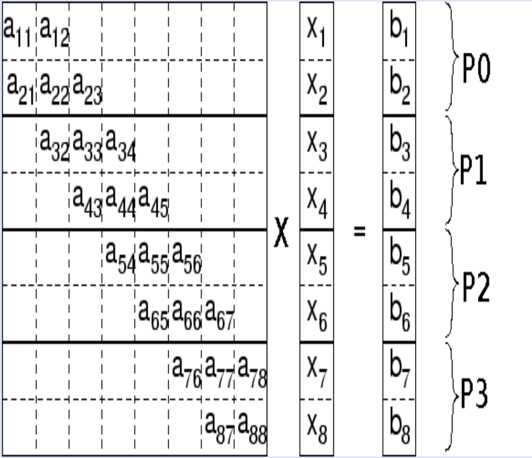
Search for the slolution of a linear system of dimension n in a specific subspace of dimension k smaller than n. Basically, $x^k = F(x^0, ..., x^{k-1})$.

Krylov subspace

Let $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$, let $k \leq n$, the space denoted by $\kappa(b, A, k)$ with $\kappa(b, A, k) = Span\{b, Ab, ..., A^{k-1}b\}$ is referred to as the Krylov space of dimension k associated with A and b.

Parallelization of the iterative methods

Distribution of a sparse matrix



Data lost when fault occurs

Two categories of lost data

- Static data:
  Matrix A, right-hand side b and possibly preconditioner M.
- Dynamic data:
  all the vectors and small matrices generated by Krylov solvers during the iterations.

Data lost when fault occurs

Assume the core number I fails at the k^{th} iteration

Before fault

$$\begin{pmatrix} \vdots \\ \hline x_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Data lost when fault occurs

Assume the core number I fails at the k^{th} iteration

Before fault

$$\begin{pmatrix} \vdots \\ \hline x_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

After fault

$$\begin{pmatrix} \vdots \\ \hline ? \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Data lost when fault occurs

Assume the core number I fails at the k^{th} iteration

Before fault

$$\left(\begin{array}{c} \vdots \\ \hline x_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{array}\right)$$

After fault

$$\left(\begin{array}{c} \vdots \\ \hline \color{blue}{?} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{array}\right)$$

Checkpoint

$$\left(\begin{array}{c} \vdots \\ \hline \color{blue}{x_I^{(k)}} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{array}\right)$$

Data lost when fault occurs

Assume the core number I fails at the k^{th} iteration

Before fault

$$\begin{pmatrix} \vdots \\ \hline x_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

After fault

$$\begin{pmatrix} \vdots \\ \hline ? \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Reset

$$\begin{pmatrix} \vdots \\ \hline 0_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Data lost when fault occurs

Assume the core number I fails at the k^{th} iteration

Before fault

$$\begin{pmatrix} \vdots \\ \hline x_I^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

After fault

$$\begin{pmatrix} \vdots \\ \hline ? \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Interpolation

$$\begin{pmatrix} \vdots \\ \hline x_{I_{new}}^{(k)} \\ \hline \vdots \\ \hline x_J^{(k)} \\ \hline \vdots \end{pmatrix}$$

Linear System (LS) [J. Langou et al., SISC, 2007]

Linear system

$$\left(\begin{array}{c|c|c} \ddots & A_{I-1,I} & \vdots \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \vdots & A_{I+1,I} & \ddots \end{array}\right) \times \left(\begin{array}{c} \vdots \\ \hline x_I \\ \hline \vdots \end{array}\right) = \left(\begin{array}{c} \vdots \\ \hline b_I \\ \hline \vdots \end{array}\right)$$

Linear System (LS) [J. Langou et al., SISC, 2007]

Linear system

$$\left(\begin{array}{c|c|c} \ddots & A_{I-1,I} & \vdots \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \vdots & A_{I+1,I} & \ddots \end{array}\right) \times \left(\begin{array}{c} \vdots \\ \hline x_I \\ \hline \vdots \end{array}\right) = \left(\begin{array}{c} \vdots \\ \hline b_I \\ \hline \vdots \end{array}\right)$$

Recovery

- ⋆ $A_{I,I}x_I = (b_I - A_{I,I-1}x_{I-1} - A_{I,I+1}x_{I+1})$.
- ⋆ Exact strategy for single fault, if the solver had converged.

Linear System (LS) [J. Langou et al., SISC, 2007]

Linear system

$$\left(\begin{array}{c|c|c} \ddots & A_{I-1,I} & \vdots \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \vdots & A_{I+1,I} & \ddots \end{array}\right) \times \left(\begin{array}{c} \vdots \\ \hline x_I \\ \hline \vdots \end{array}\right) = \left(\begin{array}{c} \vdots \\ \hline b_I \\ \hline \vdots \end{array}\right)$$

Recovery

- $A_{I,I}x_I = (b_I - A_{I,I-1}x_{I-1} - A_{I,I+1}x_{I+1})$.
- Exact strategy for single fault, if the solver had converged.

Linear Least Squares (LLS)

Linear least square problem

$$\left(\begin{array}{c|c|c} \begin{matrix} \cdot & \cdots \\ \vdots & \ddots \end{matrix} & A_{I-1,I} & \begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} & A_{I+1,I} & \begin{matrix} \ddots & \vdots \\ \cdots & \cdot \end{matrix} \end{array}\right) \times \left(\begin{array}{c} \cdot \\ \vdots \\ \hline x_I \\ \hline \vdots \\ \cdot \end{array}\right) = \left(\begin{array}{c} \cdot \\ \vdots \\ \hline b_I \\ \hline \vdots \\ \cdot \end{array}\right)$$

Recovery of x_I

- ⋆ $A_{:,I-1}x_{I-1} + A_{:,I}x_I + A_{:,I+1}x_{I+1} = b.$
- ⋆ $b_{new} = b - A_{:,I-1}x_{I-1} + A_{:,I+1}x_{I+1}.$

Linear Least Squares (LLS)

Linear least square problem

$$\left(\begin{array}{c|c|c} \begin{matrix} \cdot & \cdots \\ \vdots & \ddots \end{matrix} & A_{I-1,I} & \begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} & A_{I+1,I} & \begin{matrix} \ddots & \vdots \\ \cdots & \cdot \end{matrix} \end{array}\right) \times \left(\begin{array}{c} \cdot \\ \vdots \\ \hline x_I \\ \hline \vdots \\ \cdot \end{array}\right) = \left(\begin{array}{c} \cdot \\ \vdots \\ \hline b_I \\ \hline \vdots \\ \cdot \end{array}\right)$$

Recovery of x_I

- ⋆ $A_{:,I-1}x_{I-1} + A_{:,I}x_I + A_{:,I+1}x_{I+1} = b.$
- ⋆ $b_{new} = b - A_{:,I-1}x_{I-1} + A_{:,I+1}x_{I+1}.$

Linear Least Squares (LLS)

Linear least square problem

$$\left(\begin{array}{c|c|c} \begin{matrix} \vdots & \ddots \end{matrix} & A_{I-1,I} & \begin{matrix} \vdots & \vdots \end{matrix} \\ \hline A_{I,I-1} & A_{I,I} & A_{I,I+1} \\ \hline \begin{matrix} \vdots & \vdots \end{matrix} & A_{I+1,I} & \begin{matrix} \ddots & \vdots \end{matrix} \end{array}\right) \times \left(\begin{array}{c} \vdots \\ \hline x_I \\ \hline \vdots \end{array}\right) = \left(\begin{array}{c} \vdots \\ \hline b_I \\ \hline \vdots \end{array}\right)$$

Recovery of x_I

- $A_{:,I-1}x_{I-1} + A_{:,I}x_I + A_{:,I+1}x_{I+1} = b.$
- $b_{new} = b - A_{:,I-1}x_{I-1} + A_{:,I+1}x_{I+1}.$

Linear Least Squares (LLS)

Linear least square problem

$$\left(\begin{array}{cc|c|cc} \cdot & \cdots & & \cdot & \cdot \\ \vdots & \ddots & A_{I-1,I} & \vdots & \vdots \\ \hline \multicolumn{2}{c|}{A_{I,I-1}} & A_{I,I} & \multicolumn{2}{c}{A_{I,I+1}} \\ \hline \cdot & \cdot & & \ddots & \vdots \\ \vdots & \vdots & A_{I+1,I} & \cdots & \cdot \end{array}\right) \times \left(\begin{array}{c} \cdot \\ \vdots \\ \hline x_I \\ \hline \vdots \\ \cdot \end{array}\right) = \left(\begin{array}{c} \cdot \\ \vdots \\ \hline b_I \\ \hline \vdots \\ \cdot \end{array}\right)$$

Recovery of x_I

- ⋆ $A_{:,I-1}x_{I-1} + A_{:,I}x_I + A_{:,I+1}x_{I+1} = b.$
- ⋆ $b_{new} = b - A_{:,I-1}x_{I-1} + A_{:,I+1}x_{I+1}.$

Linear Least Squares (LLS)

Linear least square problem

$$
\left(\begin{array}{c|c|c}
\begin{matrix} \cdot & \cdots \\ \vdots & \ddots \end{matrix} & A_{I-1,I} & \begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} \\
\hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} \\
\hline
\begin{matrix} \cdot & \cdot \\ \vdots & \vdots \end{matrix} & A_{I+1,I} & \begin{matrix} \ddots & \vdots \\ \cdots & \cdot \end{matrix}
\end{array}\right)
\times
\left(\begin{array}{c}
\cdot \\ \vdots \\ \hline x_I \\ \hline \vdots \\ \cdot
\end{array}\right)
=
\left(\begin{array}{c}
\cdot \\ \vdots \\ \hline b_I \\ \hline \vdots \\ \cdot
\end{array}\right)
$$

Recovery of x_I

$$
x_I = \operatorname{argmin} \| b_{new} - \begin{pmatrix} A_{I-1,I} \\ A_{I,I} \\ A_{I+1,I} \end{pmatrix} x \|.
$$

Multiple Faults

More than one fault at the same iteration

Processors I and J failed

$$
\left(\begin{array}{c|c|c|c|c}
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} & A_{I,I+2} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\ \hline x_I \\ \hline \vdots \\ \hline x_J \\ \hline \vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\ \hline b_I \\ \hline \vdots \\ \hline b_J \\ \hline \vdots
\end{array}\right)
$$

Multiple Faults

More than one fault at the same iteration

Processors I and J failed

$$
\left(\begin{array}{c|c|c|c|c}
\begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix} \\ \hline
\color{blue}{A_{I,I-1}} & \color{blue}{A_{I,I}} & \color{blue}{A_{I,I+1}} & \color{blue}{A_{I,I+2}} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} \\ \hline
\begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix} \\ \hline
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} \\ \hline
\begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdot \\ \vdots & \vdots\end{smallmatrix} & \begin{smallmatrix}\cdot & \cdots \\ \vdots & \ddots\end{smallmatrix}
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\ \hline \color{blue}{x_I} \\ \hline \vdots \\ \hline x_J \\ \hline \vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\ \hline b_I \\ \hline \vdots \\ \hline b_J \\ \hline \vdots
\end{array}\right)
$$

Multiple Faults

More than one fault at the same iteration

Processors I and J failed

$$
\left(\begin{array}{c|c|c|c|c}
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} & A_{I,I+2} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\ \hline x_I \\ \hline \vdots \\ \hline x_J \\ \hline \vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\ \hline b_I \\ \hline \vdots \\ \hline b_J \\ \hline \vdots
\end{array}\right)
$$

Parallel recovery

Parallel recovery: handle each block independently

Recovery of x_I

$$
\left(\begin{array}{c|c|c|c|c}
\begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix} \\ \hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} & A_{I,I+2} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} \\ \hline
\begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix} \\ \hline
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} \\ \hline
\begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdot \\ \vdots & \vdots\end{matrix} & \begin{matrix}\cdot & \cdots \\ \vdots & \ddots\end{matrix}
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\ \hline {\color{blue}x_I} \\ \hline \vdots \\ \hline {\color{red}0_J} \\ \hline \vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\ \hline b_I \\ \hline \vdots \\ \hline b_J \\ \hline \vdots
\end{array}\right)
$$

Parallel recovery

Parallel recovery: handle each block independently

Recovery of x_J

$$
\left(\begin{array}{c|c|c|c|c}
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} & A_{I,I+2} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\
\hline
{\color{red}0_I} \\
\hline
\vdots \\
\hline
{\color{blue}x_J} \\
\hline
\vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\
\hline
b_I \\
\hline
\vdots \\
\hline
b_J \\
\hline
\vdots
\end{array}\right)
$$

Assembled recovery

Assembled recovery: assemble blocks that failed

Assembled recovery

$$
\left(\begin{array}{c|c|c|c|c}
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
A_{I,I-1} & A_{I,I} & A_{I,I+1} & A_{I,I+2} & \vdots \\
A_{J,J-2} & A_{J-1} & A_{J,J} & A_{J,J+1} & \vdots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots \\
\hline
\ddots & \vdots & \vdots & \vdots & \ddots
\end{array}\right)
\times
\left(\begin{array}{c}
\vdots \\
\hline
x_I \\
x_J \\
\hline
\vdots \\
\hline
\vdots
\end{array}\right)
=
\left(\begin{array}{c}
\vdots \\
\hline
b_I \\
b_J \\
\hline
\vdots \\
\hline
\vdots
\end{array}\right)
$$

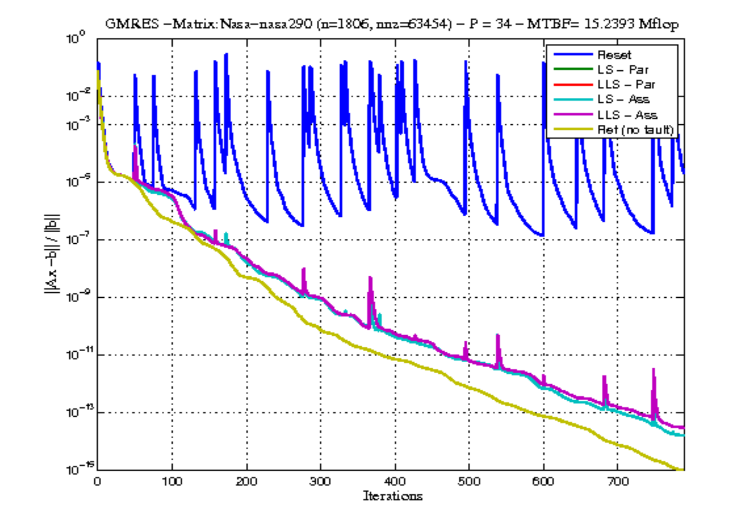
Experimental set up

Experimental environment

- ⋆ Matlab prototype.
- ⋆ Simulation of a parallel environment.
- ⋆ Weibull distribution.
- ⋆ Set Mean Time Between Fault (MTBF) of cores.
- ⋆ Assumption “instantaneous recovery” : study numerical behaviour only.

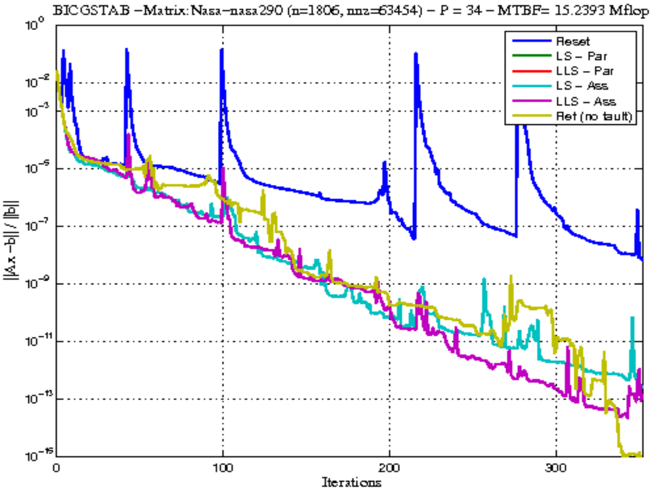
Single fault

GMRES- Matrix Nasa_nasa290 - P= 34 - (32 faults)



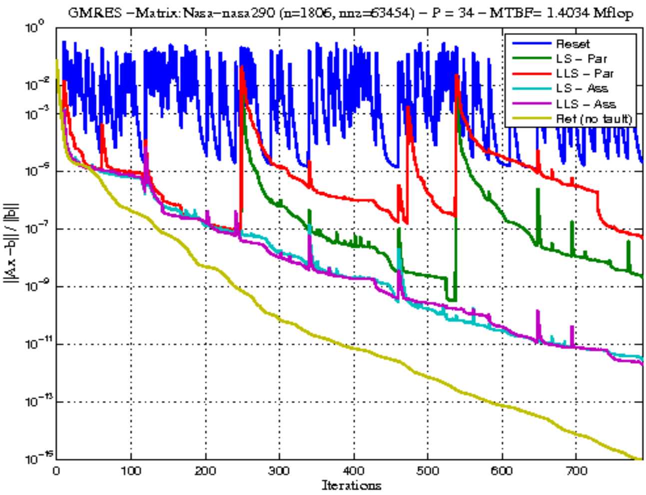
Single fault

BICGSTAB- Matrix Nasa_nasa290 - P= 34 - (9 faults)



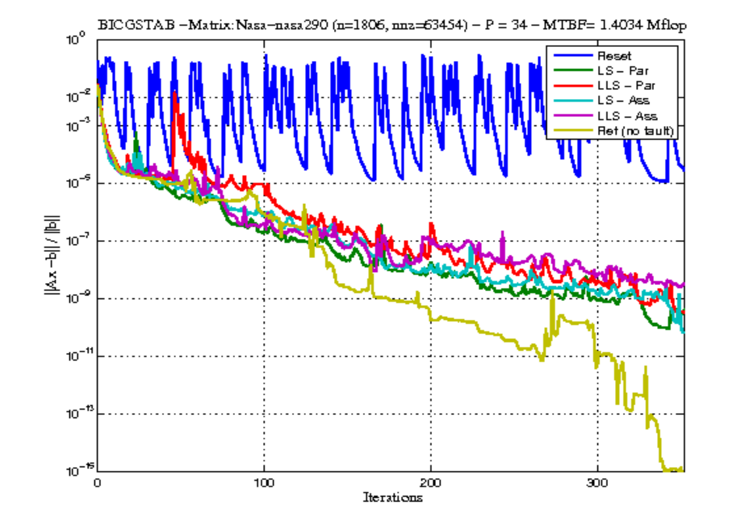
Multiple faults

GMRES- Matrix Nasa_nasa290 - P= 34 - (18 MF)



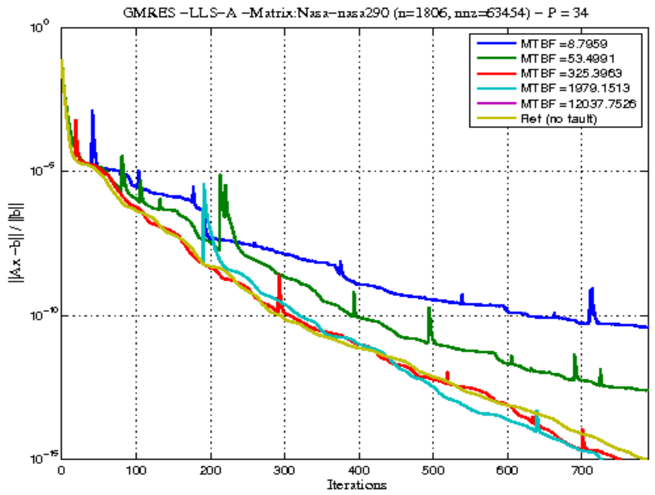
Multiple faults

BICGSTAB- Matrix Nasa_nasa290 - P= 34 - (29 MF)



Impact of the MTBF on the convergence rate

GMRES - LLS-A - Matrix Nasa_nasa290 - P= 34



Concluding remarks

Concluding remarks

- ⋆ Overhead free when no fault.
- ⋆ Reset strategy does not work.
- ⋆ The parallel recovery might be poor.
- ⋆ The assembled recovery is more robust, but more costly.
- ⋆ Convergence speed increases when fault rate decreases.

Concluding remarks

Concluding remarks

- ⋆ Overhead free when no fault.
- ⋆ Reset strategy does not work.
- ⋆ The parallel recovery might be poor.
- ⋆ The assembled recovery is more robust, but more costly.
- ⋆ Convergence speed increases when fault rate decreases.

Perspectives

- ⋆ Estimation of the recovery costs.
- ⋆ Inexact recovery strategies (iterative based).
- ⋆ ABFT variant, and possible hybrid.
- ⋆ Soft error recovery ?

ANR blanche RESCUE project (GRAND-LARGE, ROMA); G8 ECS project.

Concluding remarks

Concluding remarks

- ⋆ Overhead free when no fault.
- ⋆ Reset strategy does not work.
- ⋆ The parallel recovery might be poor.
- ⋆ The assembled recovery is more robust, but more costly.
- ⋆ Convergence speed increases when fault rate decreases.

Thank you pour votre attention. Questions ?

Perspectives

- ⋆ Estimation of the recovery costs.
- ⋆ Inexact recovery strategies (iterative based).
- ⋆ ABFT variant, and possible hybrid.
- ⋆ Soft error recovery ?

ANR blanche RESCUE project (GRAND-LARGE, ROMA); G8 ECS project.