Subsquares Approach - Simple Scheme for Solving OILS

Contents

Basic notation

- $\boldsymbol{A}, \boldsymbol{b}$ indicate an interval matrix and vector respectively
- A, b indicate a point real matrix and vector respectively
- $\boldsymbol{A} = [\underline{A}, \overline{A}]$, where $\underline{A}$ is called *lower bound* and $\overline{A}$ is called *upper bound*
- Also $\boldsymbol{A} = \langle A_c, A_\Delta \rangle$, where A_c is *midpoint matrix* and A_Δ is *radius matrix*

General interval linear system

Definition

$$\boldsymbol{A}x = \boldsymbol{b}, \quad \text{where}$$

- $\boldsymbol{A} \in \mathbb{IR}^{m \times n}$ (interval matrix)
- $\boldsymbol{b} \in \mathbb{IR}^{m \times 1}$ (interval vector)
- $\mathbb{IR}$ is the set of all real closed intervals

Overdetermined interval linear system

Definition

$\boldsymbol{A}x = \boldsymbol{b}$ with more equations than variables.

- From now on the word ”system” means overdetermined system.

Solution of an interval linear system

Definition

The solution set of $\boldsymbol{A}x = \boldsymbol{b}$ is

$$\Sigma = \{x \mid Ax = b \text{ for some } A \in \boldsymbol{A}, b \in \boldsymbol{b}\}.$$

That is not the least squares approach!

Our goal

- Computing tight *n*-dimensional box containing the solution set (*hull*).
  $\rightarrow$ NP-hard

- Computing ”nice” *n*-dimensional box containing the tight box (*enclosure*).
  $\rightarrow$ OUR GOAL

We call it solving a system.

Three ideas

Three ideas

Idea 1

There is plenty of algorithms for solving square systems.

Three ideas

Idea 1

There is plenty of algorithms for solving square systems.

Idea 2

The solution set of an OILS is contained in any square subsystem (*subsquare*).

Three ideas

Idea 1

There is plenty of algorithms for solving square systems.

Idea 2

The solution set of an OILS is contained in any square subsystem (*subsquare*).

Idea 3

Given some enclosure. Next subsquare can possibly shave it rapidly.

Example

Example

Let us take a randomly generated OILS $\boldsymbol{A}x = \boldsymbol{b}$, where

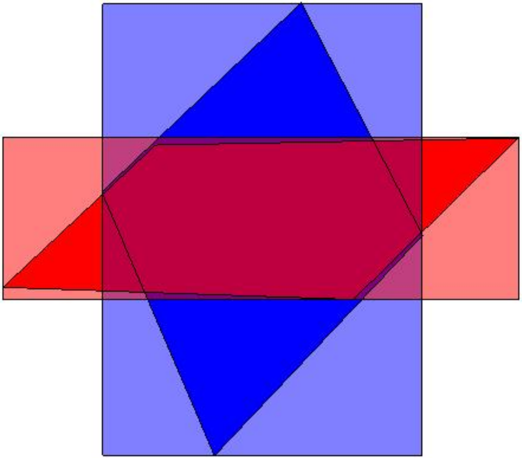
$$\boldsymbol{A} = \begin{bmatrix} [& -0.8, & 0.2 &] & [& -20.1, & -19.5 &] \\ [& -15.6, & -15.2 &] & [& 14.8, & 16.7 &] \\ [& 18.8, & 20.1 &] & [& 8.1, & 9.5 &] \end{bmatrix},$$

$$\boldsymbol{b} = \begin{bmatrix} [& 292.1, & 292.7 &] \\ [& -361.9, & -361.1 &] \\ [& 28.4, & 30.3 &] \end{bmatrix}.$$

- $\boldsymbol{A}_{\{1,2\}}x = \boldsymbol{b}_{\{1,2\}}$ denotes subsquare formed by 1st and 2nd equation of $\boldsymbol{A}x = \boldsymbol{b}$.
- Let us take a look at relation of $\boldsymbol{A}_{\{1,2\}}x = \boldsymbol{b}_{\{1,2\}}$ and $\boldsymbol{A}_{\{2,3\}}x = \boldsymbol{b}_{\{2,3\}}$ ...

Various subsquares

- $\boldsymbol{A}_{\{1,2\}} x = \boldsymbol{b}_{\{1,2\}}$
- $\boldsymbol{A}_{\{2,3\}} x = \boldsymbol{b}_{\{2,3\}}$



Subsquare scheme

Algorithm

1. Select certain amount of square subsystems of $\boldsymbol{A}x = \boldsymbol{b}$
2. Solve these subsystems and get together the enclosures

Method using this scheme will be called *Subsquare method*.

Simple algorithm

Simple algorithm

1. start with enclosure $\boldsymbol{x} = [-\infty, \infty]^n$
2. while not (terminal condition) do 3 and 4
3. choose a random subsquare and compute its enclosure $\boldsymbol{x}_{subsq}$
4. $\boldsymbol{x} := \boldsymbol{x} \cap \boldsymbol{x}_{subsq}$

Simple, but . . .

. . . we can get close to hull,

if we solve all (many) subsquares!

W – sum of interval widths of a vector
V – interval volume by vector
subsq – simple subsquares method
hull – hull computed by linear programming
ver – verifylss from INTLAB 6
av – average

system	$\mathrm{av}\left(\frac{\mathrm{W}(\boldsymbol{x}_{subsq})}{\mathrm{W}(\boldsymbol{x}_{hull})}\right)$	$\mathrm{av}\left(\frac{\mathrm{V}(\boldsymbol{x}_{subsq})}{\mathrm{V}(\boldsymbol{x}_{hull})}\right)$	$\mathrm{av}\left(\frac{\mathrm{W}(\boldsymbol{x}_{ver})}{\mathrm{W}(\boldsymbol{x}_{hull})}\right)$	$\mathrm{av}\left(\frac{\mathrm{V}(\boldsymbol{x}_{ver})}{\mathrm{V}(\boldsymbol{x}_{hull})}\right)$
5×3	1.0014	1.0043	1.1759	1.6502
9×5	1.0028	1.0140	1.1906	2.3831
13×7	1.0044	1.0316	1.2034	3.6733
15×9	1.0061	1.0565	1.1720	4.2902
25×21	1.0227	1.6060	1.0833	5.4266
30×29	1.0524	5.8330	1.0987	51.0466

Suitable systems for this method

- Number of subsquares is $\binom{m}{n}$
- But for some systems it is not too many

Moreover . . .

. . . we can reveal unsolvability,

if empty intersection in some iteration occurs!

rad – radius of interval coefficients of system

system	$rad = 0.01$	$rad = 0.001$	$rad = 0.0001$
15×10	2.1	2.0	2.0
25×21	2.2	2.0	2.0
35×23	2.2	2.0	2.0
50×35	2.4	2.0	2.0
73×55	2.9	2.1	2.0
100×87	7.1	2.1	2.0

Table shows number of iterations of simple algorithm needed to reveal unsolvability of a system.

We would like some improvement

Building better algorithm

- When selecting subsquares randomly, they usually overlap
- We can think of overlaps as a ”meeting points” of square subsystems
- We can use it to propagate a partial result of computation over one subsquare into computations over other subsquares
- $\Rightarrow$ Gauss-Seidel iteration

Solving each system

- Enclosure of some subsquare or from some different algorithm
- $$\begin{aligned} \boldsymbol{x}_i^{(k)} &= \frac{1}{\boldsymbol{A}_{ii}}\Big[\, \boldsymbol{b}_i - (\boldsymbol{A}_{i1}\boldsymbol{x}_1^{(k)} + \ldots + \boldsymbol{A}_{i(i-1)}\boldsymbol{x}_{i-1}^{(k)} + \\ &\quad + \boldsymbol{A}_{i(i+1)}\boldsymbol{x}_{i+1}^{(k-1)} + \ldots + \boldsymbol{A}_{in}\boldsymbol{x}_n^{(k-1)})\,\Big] \cap\ \boldsymbol{x}_i^{(k-1)} (GS) \end{aligned}$$
- In k-th iteration we provide k-th GS iteration step for all systems.
- We use preconditioning with midpoint systems

How to select subsquares

We state some desirable properties of a new algorithm inspired with four unfavorable features of the simple algorithm:

How to select subsquares

We state some desirable properties of a new algorithm inspired with four unfavorable features of the simple algorithm:

Idea 1

We do not want to solve too many subsquares.

How to select subsquares

We state some desirable properties of a new algorithm inspired with four unfavorable features of the simple algorithm:

Idea 1

We do not want to solve too many subsquares.

Idea 2

We want to cover the whole overdetermined system by subsystems.

How to select subsquares

We state some desirable properties of a new algorithm inspired with four unfavorable features of the simple algorithm:

Idea 1

We do not want to solve too many subsquares.

Idea 2

We want to cover the whole overdetermined system by subsystems.

Idea 3

The overlap of subsquares is not too low, not too high.

How to select subsquares

We state some desirable properties of a new algorithm inspired with four unfavorable features of the simple algorithm:

Idea 1

We do not want to solve too many subsquares.

Idea 2

We want to cover the whole overdetermined system by subsystems.

Idea 3

The overlap of subsquares is not too low, not too high.

Idea 4

We take subsquares that shave the resulting enclosure as much as possible.

Solution to ideas

- About 1 & 2 – Both can be solved by covering the system step by step using some overlap parameter
- About 3 – Open question, about $n/3$ is experimentally good
- About 4 – Open question, yet randomness serve us well

Selecting subsquares

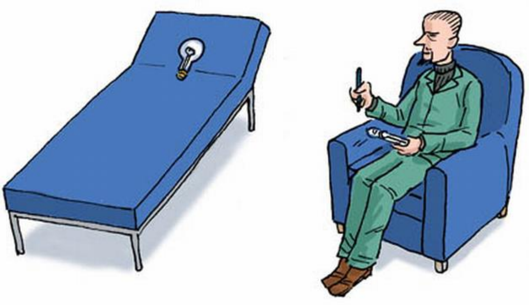
First and last system is handled separately.

Selecting subsquares

1. Divide equations into two sets *Covered* and *Waiting*
2. Choose random $(n - overlap)$ equations from *Waiting*
3. Choose random $(overlap)$ equations from *Covered*
4. We have a new subsquare, update *Covered* and *Waiting*
5. Repeat 2-4 until *Waiting* is not empty

Comparison

Well, my brother was always the bright one. I always felt rather dim by comparison. . .



Numerical testing 1

Sharpening enclosures produced by verifylss in INTLAB 6

system	overlap	rad	$\mathrm{av}\left(\frac{\mathrm{w}(\boldsymbol{x}_{subsq})}{\mathrm{w}(\boldsymbol{x}_{ver})}\right)$	$\mathrm{av}\left(\mathrm{best}\,\frac{\mathrm{w}(\boldsymbol{x}_{subsq})}{\mathrm{w}(\boldsymbol{x}_{ver})}\right)$	t_{ver}	t_{subsq}
15×10	3	0.1	0.99	0.94	0.006	0.06
15×10	3	0.25	0.97	0.86	0.007	0.07
15×10	3	0.35	0.93	0.79	0.008	0.09
15×10	3	0.5	0.87	0.66	0.01	0.12
25×13	5	0.1	0.99	0.98	0.006	0.09
25×13	5	0.25	0.99	0.94	0.007	0.12
25×13	5	0.35	0.98	0.92	0.008	0.14
25×13	5	0.5	0.94	0.79	0.012	0.20
37×20	7	0.1	0.99	0.98	0.008	0.11
37×20	7	0.25	0.99	0.95	0.011	0.19
37×20	7	0.35	0.97	0.90	0.015	0.29
37×20	7	0.5	0.87	0.38	0.016	0.72
50×35	11	0.1	0.99	0.98	0.014	0.16
50×35	11	0.25	0.97	0.84	0.023	0.51

Numerical testing 2

When midpoint system is solvable, we get better results.

system	overlap	*rad*	av. rat	best av. rat
15×10	3	0.1	0.98	0.89
15×10	3	0.25	0.85	0.59
15×10	3	0.35	0.76	0.40
25×13	5	0.1	0.99	0.96
25×13	5	0.25	0.93	0.74
25×13	5	0.35	0.76	0.33
37×20	7	0.1	0.99	0.97
37×20	7	0.25	0.89	0.34
50×35	11	0.1	0.98	0.82

Conclusion

- Simple idea of scheme
- Can be used when enclosure is needed to be shaved
- Both methods detect unsolvability
- Simple method easily parallelized
- So can be the second method

- Open problems: choosing systems, overlap, effective parallelitazion

Subsquares approach

Thank you very much for your attention. . .