Semidefinite Programming Relaxation vs Polyhedral Homotopy Method for Problems Involving Polynomials

Workshop on Advances in Optimization

Tokyo Institute of Technology, April 19-21, 2007

Masakazu Kojima

Tokyo Institute of Technology

- Numerical results

Contents

1. PHoMpara — Parallel implementation of the polyhedral homotopy method ([1] Gunji-Kim-Fujisawa-Kojima ’06)
2. SparsePOP — Matlab implementation of SDP relaxation for sparse POPs ([2] Waki-Kim-Kojima-Muramatsu ’05)
3. Numerical comparison between the SDP relaxation and the polyhedral homotopy method ([1]+[2]+[3] Mevissen-Kojima-Nie-Takayama)
4. Concluding remarks

SDP $=$ Semidefinite Program or Programming
POP $=$ Polynomial Optimization Problem

Contents

1. PHoMpara — Parallel implementation of the polyhedral homotopy method ([1] Gunji-Kim-Fujisawa-Kojima ’06)
2. SparsePOP — Matlab implementation of SDP relaxation for sparse POPs ([2] Waki-Kim-Kojima-Muramatsu ’05)
3. Numerical comparison between the SDP relaxation and the polyhedral homotopy method ([1]+[2]+[3] Mevissen-Kojima-Nie-Takayama)
4. Concluding remarks

SDP $=$ Semidefinite Program or Programming
POP $=$ Polynomial Optimization Problem

The polyhedral homotopy method

- Implementation on a single CPU:
  - PHCpack [Verschelde]
  - HOM4PS [Li-Li-Gao]
  - PHoM [Gunji-Kim-Kojima-Takeda-Fujisawa-Mizutani]

The polyhedral homotopy method

- Implementation on a single CPU:
  - PHCpack [Verschelde]
  - HOM4PS [Li-Li-Gao]
  - PHoM [Gunji-Kim-Kojima-Takeda-Fujisawa-Mizutani]
- Suitable for parallel computation — all isolated solutions can be computed independently in parallel.
  - PHoMpara [Gunji, Kim, Fujisawa and Kojima] — Next
  - Leykin, Verschelde and Zhuang

Numerical results: Hardware — PC cluster (AMD Athlon 2.0GHz)

Problem (#sol)	#CPUs	cpu time in second	speedup ratio
noon-10	1	62,672	1.0
(59,029)	40	1,797	34.9
eco-14	1	22,653	1.0
(4,096)	40	626	36.2

Numerical results: Hardware — PC cluster (AMD Athlon 2.0GHz)

Problem (#sol)	#CPUs	cpu time in second	speedup ratio
noon-10	1	62,672	1.0
(59,029)	40	1,797	34.9
eco-14	1	22,653	1.0
(4,096)	40	626	36.2

noon-12 (531,417)	40	49,458	
eco-16 (16,384)	40	12,051	

Contents

SDP = Semidefinite Program or Programming
POP = Polynomial Optimization Problem

SparsePOP (Waki-Kim-Kojima-Muramatsu ’06) = Lasserre’s SDP relaxation ’01 + “structured sparsity” — c-sparsity

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$.

Example: $f_0(\boldsymbol{x}) = \sum_{k=1}^{n} (-x_k^2)$

$f_j(\boldsymbol{x}) = 1 - x_j^2 - 2x_{j+1}^2 - x_n^2$ $(j = 1, \ldots, n-1)$.

$\boldsymbol{H} f_0(\boldsymbol{x})$: the $n \times n$ Hessian mat. of $f_0(\boldsymbol{x})$,

$\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$: the $m \times n$ Jacob. mat. of $\boldsymbol{f}_*(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T$,

$\boldsymbol{R}$: the csp matrix, the $n \times n$ density pattern matrix of $\boldsymbol{I} + \boldsymbol{H} f_0(\boldsymbol{x}) + \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$ (no cancellation in '+').

$[\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})]_{ij} \neq 0$ iff x_i and x_j are in a common constraint.

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$.

Example: $f_0(\boldsymbol{x}) = \sum_{k=1}^{n} (-x_k^2)$
$f_j(\boldsymbol{x}) = 1 - x_j^2 - 2x_{j+1}^2 - x_n^2$ $(j = 1, \ldots, n-1)$.

$\boldsymbol{H} f_0(\boldsymbol{x})$: the $n \times n$ Hessian mat. of $f_0(\boldsymbol{x})$,

$\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$: the $m \times n$ Jacob. mat. of $\boldsymbol{f}_*(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T$,

$\boldsymbol{R}$: the csp matrix, the $n \times n$ density pattern matrix of
$\boldsymbol{I} + \boldsymbol{H} f_0(\boldsymbol{x}) + \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$ (no cancellation in '$+$').

$[\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})]_{ij} \neq 0$ iff x_i and x_j are in a common constraint.

Example with n = 6:

the csp matrix $\boldsymbol{R} = \begin{pmatrix} \star & \star & 0 & 0 & 0 & \star \\ \star & \star & \star & 0 & 0 & \star \\ 0 & \star & \star & \star & 0 & \star \\ 0 & 0 & \star & \star & \star & \star \\ 0 & 0 & 0 & \star & \star & \star \\ \star & \star & \star & \star & \star & \star \end{pmatrix}$

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$.

Example: $f_0(\boldsymbol{x}) = \sum_{k=1}^{n} (-x_k^2)$
$f_j(\boldsymbol{x}) = 1 - x_j^2 - 2x_{j+1}^2 - x_n^2$ $(j = 1, \ldots, n-1)$.

$\boldsymbol{H} f_0(\boldsymbol{x})$: the $n \times n$ Hessian mat. of $f_0(\boldsymbol{x})$,

$\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$: the $m \times n$ Jacob. mat. of $\boldsymbol{f}_*(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T$,

$\boldsymbol{R}$: the csp matrix, the $n \times n$ density pattern matrix of $\boldsymbol{I} + \boldsymbol{H} f_0(\boldsymbol{x}) + \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$ (no cancellation in '+').

$[\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})]_{ij} \neq 0$ iff x_i and x_j are in a common constraint.

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$.

Example: $f_0(\boldsymbol{x}) = \sum_{k=1}^{n} (-x_k^2)$ ——— c-sparse

$f_j(\boldsymbol{x}) = 1 - x_j^2 - 2x_{j+1}^2 - x_n^2$ $(j = 1, \ldots, n-1)$.

$\boldsymbol{H} f_0(\boldsymbol{x})$: the $n \times n$ Hessian mat. of $f_0(\boldsymbol{x})$,

$\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$: the $m \times n$ Jacob. mat. of $\boldsymbol{f}_*(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))^T$,

$\boldsymbol{R}$: the csp matrix, the $n \times n$ density pattern matrix of

$\boldsymbol{I} + \boldsymbol{H} f_0(\boldsymbol{x}) + \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})$ (no cancellation in '+').

$[\boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})^T \boldsymbol{J} \boldsymbol{f}_*(\boldsymbol{x})]_{ij} \neq 0$ iff x_i and x_j are in a common constraint.

POP : c-sparse (correlatively sparse) $\Leftrightarrow$ The $n \times n$ csp matrix $\boldsymbol{R} = (R_{ij})$ allows a symbolic sparse Cholesky factorization (under a row & col. ordering like a symmetric min. deg. ordering).

Sparse (SDP) relaxation = Lasserre (2001) + c-sparsity

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$, c-sparse.

$$\Downarrow$$

A sequence of c-sparse SDP relaxation problems depending on the relaxation order $r = 1, 2, \ldots$;

Sparse (SDP) relaxation = Lasserre (2001) + c-sparsity

POP min. $f_0(\boldsymbol{x})$ s.t. $f_j(\boldsymbol{x}) \geq 0$ or $= 0$ $(j = 1, \ldots, m)$, c-sparse.

$$\Downarrow$$

A sequence of c-sparse SDP relaxation problems depending on the relaxation order $r = 1, 2, \ldots$;

(a) Under a moderate assumption,
opt. sol. of SDP $\rightarrow$ opt sol. of POP as $r \rightarrow \infty$.

(b) $r = \lceil$"the max. deg. of poly. in POP"$/2\rceil + 0 \sim 3$ is usually large enough to attain opt sol. of POP in practice.

(c) Such an r is unknown in theory except $\exists$ special cases.

(d) The size of SDP increases rapidly as $r \rightarrow \infty$.

Contents

1. PHoMpara — Parallel implementation of the polyhedral homotopy method ([1] Gunji-Kim-Fujisawa-Kojima ’06)
2. SparsePOP — Matlab implementation of SDP relaxation for sparse POPs ([2] Waki-Kim-Kojima-Muramatsu ’05)
3. Numerical comparison between the SDP relaxation and the polyhedral homotopy method ([1]+[2]+[3] Mevissen-Kojima-Nie-Takayama)
4. Concluding remarks

SDP $=$ Semidefinite Program or Programming
POP $=$ Polynomial Optimization Problem

A POP alkyl from globalib

$$\begin{array}{ll} \text{min} & -6.3x_5x_8 + 5.04x_2 + 0.35x_3 + x_4 + 3.36x_6 \\ \text{sub.to} & -0.820x_2 + x_5 - 0.820x_6 = 0, \end{array}$$

$$0.98x_4 - x_7(0.01x_5x_{10} + x_4) = 0,\ -x_2x_9 + 10x_3 + x_6 = 0,$$

$$x_5x_{12} - x_2(1.12 + 0.132x_9 - 0.0067x_9^2) = 0,$$

$$x_8x_{13} - 0.01x_9(1.098 - 0.038x_9) - 0.325x_7 = 0.574,$$

$$x_{10}x_{14} + 22.2x_{11} = 35.82,\ x_1x_{11} - 3x_8 = -1.33,$$

$$\text{lbd}_i \leq x_i \leq \text{ubd}_i\ (i = 1, 2, \ldots, 14).$$

- 14 variables, 7 poly. equality constraints with deg. 3.

A POP alkyl from globalib

min $\quad -6.3x_5x_8 + 5.04x_2 + 0.35x_3 + x_4 + 3.36x_6$

sub.to $\quad -0.820x_2 + x_5 - 0.820x_6 = 0,$

$0.98x_4 - x_7(0.01x_5x_{10} + x_4) = 0,\ -x_2x_9 + 10x_3 + x_6 = 0,$

$x_5x_{12} - x_2(1.12 + 0.132x_9 - 0.0067x_9^2) = 0,$

$x_8x_{13} - 0.01x_9(1.098 - 0.038x_9) - 0.325x_7 = 0.574,$

$x_{10}x_{14} + 22.2x_{11} = 35.82,\ x_1x_{11} - 3x_8 = -1.33,$

$\text{lbd}_i \leq x_i \leq \text{ubd}_i\ (i = 1, 2, \ldots, 14).$

- 14 variables, 7 poly. equality constraints with deg. 3.

	Sparse			Dense (Lasserre)		
r	ϵ_{obj}	ϵ_{feas}	cpu	ϵ_{obj}	ϵ_{feas}	cpu
2	1.0e-02	7.1e-01	1.8	7.2e-3	4.3e-2	14.4
3	5.6e-10	2.0e-08	23.0	out of	memory	

ϵ_{obj} = approx.opt.val. – lower bound for opt.val.
ϵ_{feas} = the maximum error in the equality constraints

Systems of polynomial equations

- Is the (sparse) SDP relaxation useful to solve systems of polynomial equations?
- The answer depends on:
  - how sparse the system of polynomial equations is,
  - the maximum degree of polynomials.

Systems of polynomial equations

- Is the (sparse) SDP relaxation useful to solve systems of polynomial equations?
- The answer depends on:
  - how sparse the system of polynomial equations is,
  - the maximum degree of polynomials.

- 2 types of systems of polynomial equations

(a) Benchmark test problems from Verschelde's homepage; Katsura, cyclic — not c-sparse

(b) Systems of polynomials arising from discretization of an ODE and a DAE (Differential Algebraic Equations) — c-sparse

Katsura n system of polynomial equations; $n = 8$ case

$$0 = -x_1 + 2x_9^2 + 2x_8^2 + 2x_7^2 + \cdots + 2x_2^2 + x_1^2,$$

$$0 = -x_2 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + \cdots + 2x_3x_2 + 2x_2x_1,$$

$$\cdots\cdots$$

not c-sparse

$$0 = -x_8 + 2x_9x_2 + 2x_8x_1 + 2x_7x_2 + 2x_6x_3 + 2x_5x_4,$$

$$1 = 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1.$$

Katsura n system of polynomial equations; $n = 8$ case

$$0 = -x_1 + 2x_9^2 + 2x_8^2 + 2x_7^2 + \cdots + 2x_2^2 + x_1^2,$$

$$0 = -x_2 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + \cdots + 2x_3x_2 + 2x_2x_1,$$

$$\cdots\cdots$$

not c-sparse

$$0 = -x_8 + 2x_9x_2 + 2x_8x_1 + 2x_7x_2 + 2x_6x_3 + 2x_5x_4,$$

$$1 = 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1.$$

- Numerical results on SparsePOP (WKKM 2004)

n	obj.funct.	relax. order r	cpu
8	$\sum x_i \uparrow$	1	0.08
8	$\sum x_i^2 \downarrow$	2	7.1
11	$\sum x_i \uparrow$	1	0.14
11	$\sum x_i^2 \downarrow$	2	101.3

- A formulation in terms of a POP

  max $\sum_{i=1}^n x_i$ or min $\sum_{i=1}^n x_i^2$

  sub.to Katsura n system , $-5 \leq x_i \leq 5 \ (i = 1, \ldots, n)$.

  - Different objective functions $\Rightarrow$ different solutions.

Katsura n system of polynomial equations; $n = 8$ case

$$0 = -x_1 + 2x_9^2 + 2x_8^2 + 2x_7^2 + \cdots + 2x_2^2 + x_1^2,$$

$$0 = -x_2 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + \cdots + 2x_3x_2 + 2x_2x_1,$$

$$\cdots\cdots$$

not c-sparse

$$0 = -x_8 + 2x_9x_2 + 2x_8x_1 + 2x_7x_2 + 2x_6x_3 + 2x_5x_4,$$

$$1 = 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1.$$

- Numerical results on SparsePOP (WKKM 2004)

n	obj.funct.	relax. order r	cpu
8	$\sum x_i \uparrow$	1	0.08
8	$\sum x_i^2 \downarrow$	2	7.1
11	$\sum x_i \uparrow$	1	0.14
11	$\sum x_i^2 \downarrow$	2	101.3

Katsura n system of polynomial equations; $n = 8$ case

$$0 = -x_1 + 2x_9^2 + 2x_8^2 + 2x_7^2 + \cdots + 2x_2^2 + x_1^2,$$

$$0 = -x_2 + 2x_9x_8 + 2x_8x_7 + 2x_7x_6 + \cdots + 2x_3x_2 + 2x_2x_1,$$

$$\cdots\cdots$$

not c-sparse

$$0 = -x_8 + 2x_9x_2 + 2x_8x_1 + 2x_7x_2 + 2x_6x_3 + 2x_5x_4,$$

$$1 = 2x_9 + 2x_8 + 2x_7 + 2x_6 + 2x_5 + 2x_4 + 2x_3 + 2x_2 + x_1.$$

- Numerical results on SparsePOP (WKKM 2004)

n	obj.funct.	relax. order r	cpu
8	$\sum x_i \uparrow$	1	0.08
8	$\sum x_i^2 \downarrow$	2	7.1
11	$\sum x_i \uparrow$	1	0.14
11	$\sum x_i^2 \downarrow$	2	101.3

- Numerical results on HOM4PS (Li-Li-Gao 2002)

n	cpu sec.	#solutions
8	1.9	256
11	209.1	2048

cyclic n system of polynomial equations: $n = 5$ case

$$0 = x_1 + x_2 + x_3 + x_4 + x_5,$$

$$0 = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1,$$

not c-sparse

$$0 = x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2,$$

$$0 = x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3,$$

$$0 = -1 + x_1x_2x_3x_4x_5.$$

- Numerical results on SparsePOP: obj.funct.+lbd, ubd on x_i

n	obj.funct.	relax. order r	cpu
5	$\sum x_i \uparrow$	3	1.83
6	$\sum x_i \uparrow$	4	753.2

cyclic n system of polynomial equations: $n = 5$ case

$$0 = x_1 + x_2 + x_3 + x_4 + x_5,$$

$$0 = x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_1,$$

not c-sparse

$$0 = x_1x_2x_3 + x_2x_3x_4 + x_3x_4x_5 + x_4x_5x_1 + x_5x_1x_2,$$

$$0 = x_1x_2x_3x_4 + x_2x_3x_4x_5 + x_3x_4x_5x_1 + x_4x_5x_1x_2 + x_5x_1x_2x_3,$$

$$0 = -1 + x_1x_2x_3x_4x_5.$$

- Numerical results on SparsePOP: obj.funct.+lbd, ubd on x_i

n	obj.funct.	relax. order r	cpu
5	$\sum x_i \uparrow$	3	1.83
6	$\sum x_i \uparrow$	4	753.2

- Numerical results on HOM4PS (Li-Li-Gao)

n	cpu sec.	#solutions
5	0.1	70
6	0.2	156

Discretization of Mimura's ODE with 2 unknowns $u,\ v : [0,5] \to \mathbb{R}$

$$
\begin{aligned}
u_{xx} &= -(20/9)(35 + 16u - u^2)u + 20uv, \\
v_{xx} &= (1/4)((1 + (2/5)v)v - uv), \\
&\underline{u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0},
\end{aligned}
$$

Discretize:

$$x_i = i\Delta x \ (i = 0, 1, 2, \ldots),\ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

Discretization of Mimura's ODE with 2 unknowns $u, \ v : [0,5] \to \mathbb{R}$

$$\begin{aligned} u_{xx} &= -(20/9)(35 + 16u - u^2)u + 20uv, \\ v_{xx} &= (1/4)((1 + (2/5)v)v - uv), \\ & \underline{u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0}, \end{aligned}$$

Discretize:

$$x_i = i\Delta x \ (i = 0, 1, 2, \dots), \ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

Discretized system of polynomials with $\Delta x = 1$:

$$\begin{aligned} f_1(\boldsymbol{u},\boldsymbol{v}) &= 76.8u_1 + u_3 + 35.6u_1^2 - 20.0u_1v_1 - 2.22u_2^3, \\ f_2(\boldsymbol{u},\boldsymbol{v}) &= -1.25v_1 + v_2 + 0.25u_1v_1 - 0.1v_1^2, \\ f_3(\boldsymbol{u},\boldsymbol{v}) &= u_1 + 75.8u_2 + u_3 + 35.6u_2^2 - 20.0u_2v_2 - 2.22u_2^3, \\ f_4(\boldsymbol{u},\boldsymbol{v}) &= v_1 - 2.25v_2 + v_3 + 0.25u_2v_2 - 0.1v_2^2, \\ f_5(\boldsymbol{u},\boldsymbol{v}) &= u_2 + 75.8u_3 + u_4 + 35.6u_3^2 - 20.0u_3v_3 - 2.22u_3^2, \\ f_6(\boldsymbol{u},\boldsymbol{v}) &= v_2 - 2.25v_3 + v_4 + 0.25u_3v_3 - 0.1v_3^2, \\ f_7(\boldsymbol{u},\boldsymbol{v}) &= u_3 + 76.8u_4 + 35.6u_4^2 - 20.0u_4v_4 - 2.22u_4^3, \\ f_8(\boldsymbol{u},\boldsymbol{v}) &= v_3 - 1.25v_4 + 0.25u_4v_4 - 0.1v_4^2. \end{aligned}$$

Here $u_i = u(x_i)$, $v_i = v(x_i)$ $(i = 0, 1, 2, 3, 4, 5)$,

$\underline{u_0 = u_1,\ u_5 = u_4,\ v_0 = v_1 \text{ and } v_5 = v_4}$.

$\Rightarrow$ c-sparse

Discretization of Mimura's ODE with 2 unknowns $u,\ v : [0,5] \to \mathbb{R}$

$$
\begin{aligned}
u_{xx} &= -(20/9)(35 + 16u - u^2)u + 20uv, \\
v_{xx} &= (1/4)((1 + (2/5)v)v - uv), \\
& \underline{u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0},
\end{aligned}
$$

Discretize:

$$x_i = i\Delta x \ (i = 0, 1, 2, \dots\ ),\ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

Discretization of Mimura's ODE with 2 unknowns $u,\ v : [0,5] \to \mathbb{R}$

$$u_{xx} = -(20/9)(35 + 16u - u^2)u + 20uv,$$
$$v_{xx} = (1/4)((1 + (2/5)v)v - uv),$$
$$u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0,$$

Discretize:

$$x_i = i\Delta x\ (i = 0, 1, 2, \dots),\ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

- Numerical results on SparsePOP

Δx	n	obj.funct.	relax. order r	cpu
1.0	8	$\sum r_i u(x_i) \uparrow$	3	11.3
0.5	18	$\sum r_i u(x_i) \uparrow$	3	57.8

Here $r_i \in (0,1)$: random numbers.

Discretization of Mimura's ODE with 2 unknowns $u,\ v : [0,5] \to \mathbb{R}$

$$
\begin{aligned}
u_{xx} &= -(20/9)(35 + 16u - u^2)u + 20uv, \\
v_{xx} &= (1/4)((1 + (2/5)v)v - uv), \\
& \underline{u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0,}
\end{aligned}
$$

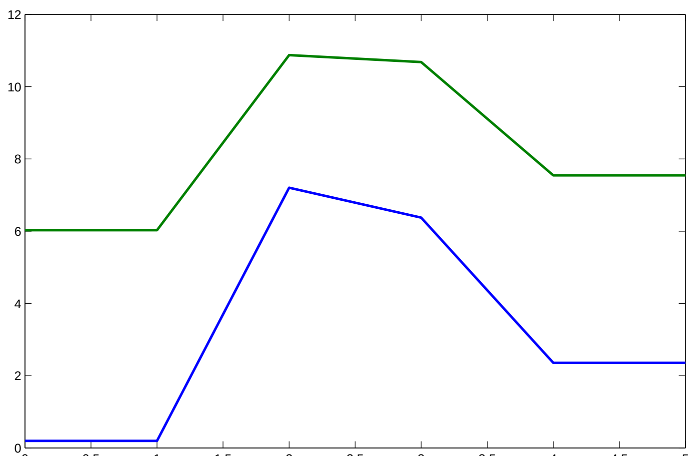
Discretize:

$$x_i = i\Delta x \ (i = 0, 1, 2, \ldots),\ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

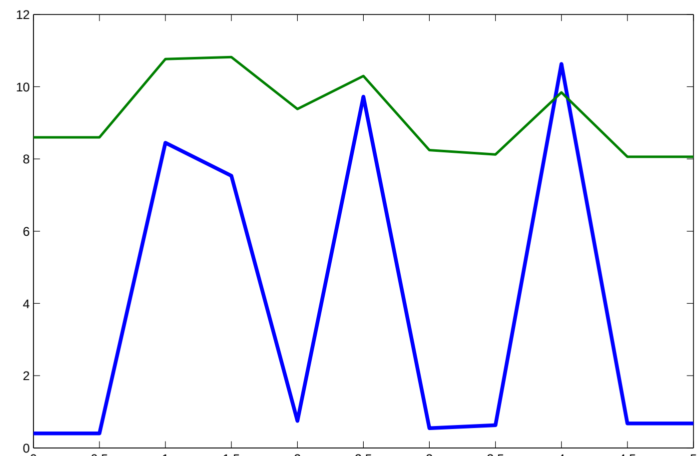
- Numerical results on SparsePOP

Δx	n	obj.funct.	relax. order r	cpu
1.0	8	$\sum r_i u(x_i) \uparrow$	3	11.3
0.5	18	$\sum r_i u(x_i) \uparrow$	3	57.8

Here $r_i \in (0,1)$: random numbers.



$\Delta x = 1.0$



$\Delta x = 0.5$

Discretization of Mimura's ODE with 2 unknowns $u, v : [0,5] \to \mathbb{R}$

$$
\begin{aligned}
u_{xx} &= -(20/9)(35 + 16u - u^2)u + 20uv, \\
v_{xx} &= (1/4)((1 + (2/5)v)v - uv), \\
& \underline{u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0,}
\end{aligned}
$$

Discretize:

$$x_i = i\Delta x \ (i = 0, 1, 2, \dots), \ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

- Numerical results on SparsePOP

Δx	n	obj.funct.	relax. order r	cpu
1.0	8	$\sum r_i u(x_i) \uparrow$	3	11.3
0.5	18	$\sum r_i u(x_i) \uparrow$	3	57.8

Here $r_i \in (0,1)$: random numbers.

Discretization of Mimura's ODE with 2 unknowns $u,\ v : [0,5] \to \mathbb{R}$

$$u_{xx} = -(20/9)(35 + 16u - u^2)u + 20uv,$$
$$v_{xx} = (1/4)((1 + (2/5)v)v - uv),$$
$$u_x(0) = u_x(5) = v_x(0) = v_x(5) = 0,$$

Discretize:

$$x_i = i\Delta x\ (i = 0, 1, 2, \dots),\ u_x(x_i) \approx (u(x_{i+1}) - u(x_{i-1}))/(2\Delta x).$$

- Numerical results on SparsePOP

Δx	n	obj.funct.	relax. order r	cpu
1.0	8	$\sum r_i u(x_i) \uparrow$	3	11.3
0.5	18	$\sum r_i u(x_i) \uparrow$	3	57.8

Here $r_i \in (0, 1)$: random numbers.

- Numerical results on HOM4PS

Δx	n	cpu sec.	#solutions	#real solutions
1.0	8	2.2	1296	222
0.5	18	167.7	10,077,696	not traced
		(M.vol.)	(M.vol.)	(M.cells=1089)

Discretization of DAE with 3 unknowns $y_1, y_2, y_3 : [0,2] \to \mathbb{R}$

$y_1' = y_3,\ 0 = y_2(1-y_2),\ 0 = y_1 y_2 + y_3(1-y_2) - t,\ y_1(0) = y_1^0.$

2 solutions : $y(t) = (t, 1, 1)$ and $y(t) = (y_1^0 + t_2^2, 0, t)$.

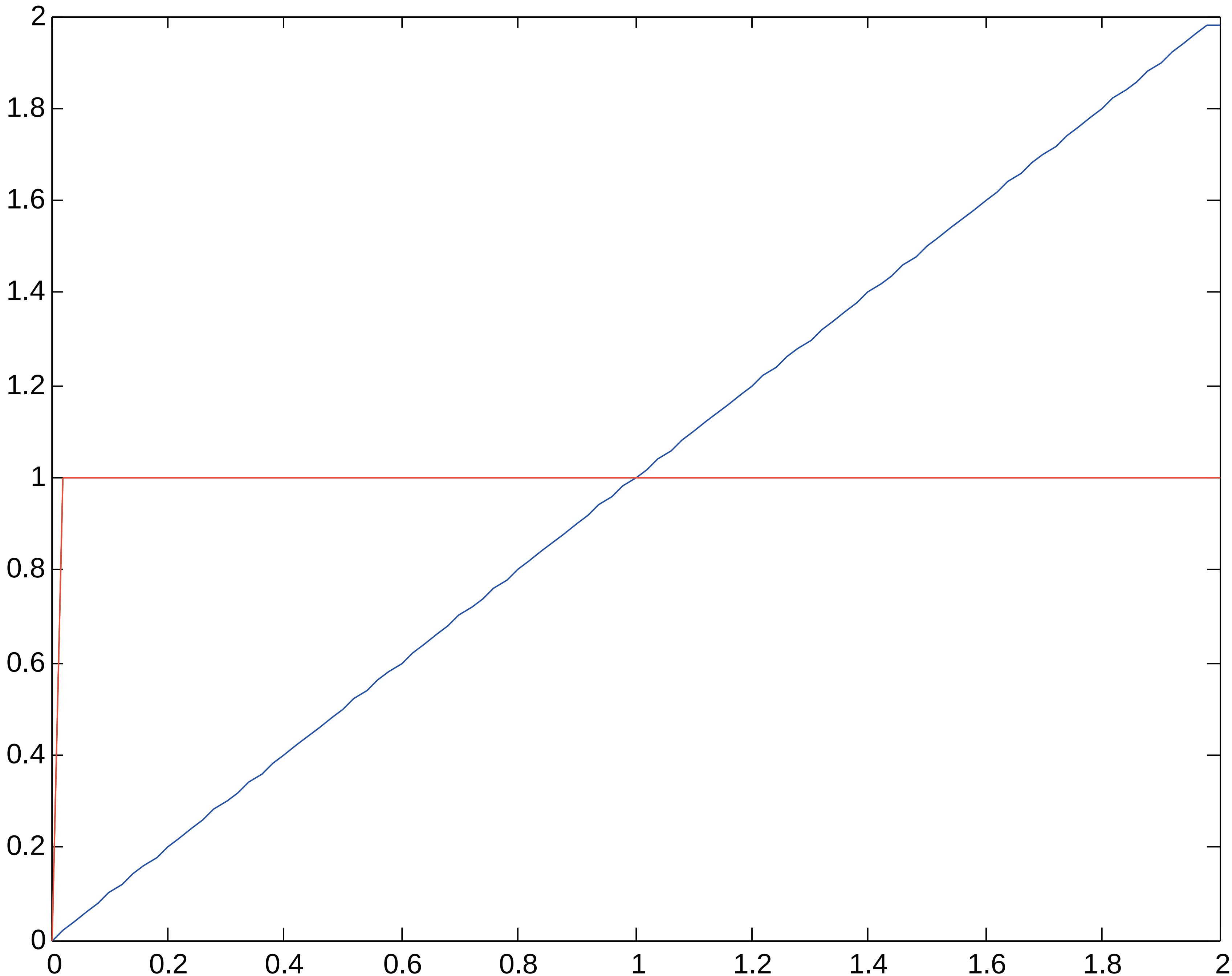
Discretization of DAE with 3 unknowns $y_1, y_2, y_3 : [0,2] \to \mathbb{R}$

$y_1' = y_3,\ 0 = y_2(1-y_2),\ 0 = y_1 y_2 + y_3(1-y_2) - t,\ y_1(0) = y_1^0.$

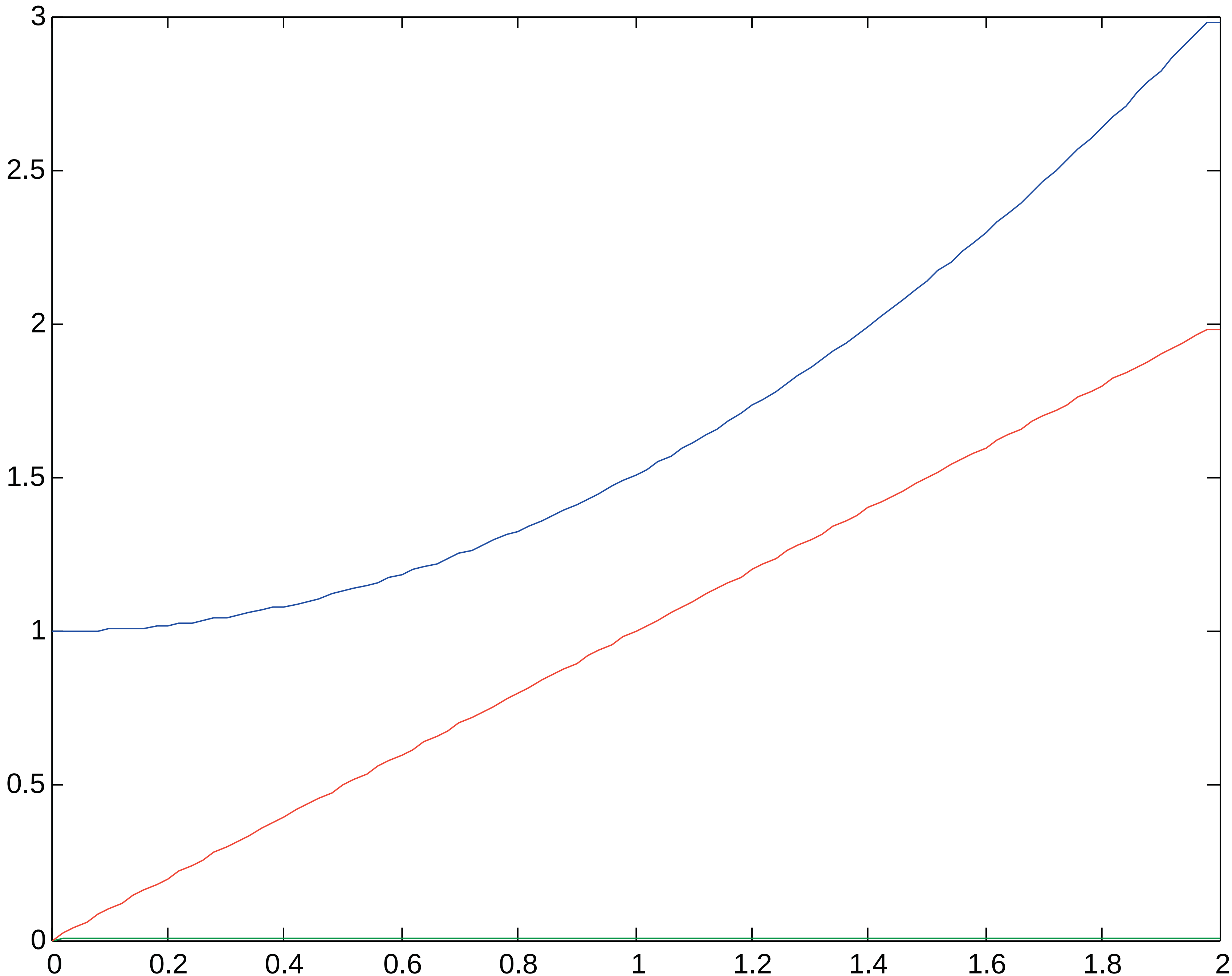
2 solutions : $y(t) = (t, 1, 1)$ and $y(t) = (y_1^0 + t_2^2, 0, t)$.

- Numerical results on SparsePOP — c-sparse

y_1^0	Δt	n	obj.funct.	relax. order r	cpu
0	0.02	297	$\sum y_2(t_i) \uparrow$	2	30.9
1	0.02	297	$\sum y_1(t_i) \uparrow$	2	33.9



Solution: $y(t) = (t, 1, 1)$



Solution: $y(t) = (y_1^0 + t_2^2, 0, t)$

Contents

1. PHoMpara — Parallel implementation of the polyhedral homotopy method ([1] Gunji-Kim-Fujisawa-Kojima ’06)
2. SparsePOP — Matlab implementation of SDP relaxation for sparse POPs ([2] Waki-Kim-Kojima-Muramatsu ’05)
3. Numerical comparison between the SDP relaxation and the polyhedral homotopy method ([1]+[2]+[3] Mevissen-Kojima-Nie-Takayama)

4. Concluding remarks

SDP $=$ Semidefinite Program or Programming
POP $=$ Polynomial Optimization Problem

- Some essential differences between Homotopy Continuation and (sparse) SDP Relaxation — 1:

- Some essential differences between Homotopy Continuation and (sparse) SDP Relaxation — 1:

(a) HC works on $\mathbb{C}^n$ while SDPR on $\mathbb{R}^n$.

(b) HC aims to compute all isolated solutions; in SDPR, computing all isolated solutions is possible but expensive.

(c) SDPR can process inequalities.

- Some essential differences between Homotopy Continuation and (sparse) SDP Relaxation — 2:

- Some essential differences between Homotopy Continuation and (sparse) SDP Relaxation — 2:

(d) SDPR is sensitive to degrees of polynomials of a POP because the SDP relaxed problem becomes larger rapidly as they increase.
$\Rightarrow$ SDPR can be applied to POPs with lower degree polynomials such as degree ≤ 4 in practice.

(e) HC fits parallel computation more than SDPR.

(f) The effectiveness of sparse SDPR depends on the c-sparsity; for example, discretization of ODE, DAE, Optimal control problem and PDE.

- Some essential differences between Homotopy Continuation and (sparse) SDP Relaxation — 2:

(d) SDPR is sensitive to degrees of polynomials of a POP because the SDP relaxed problem becomes larger rapidly as they increase.
$\Rightarrow$ SDPR can be applied to POPs with lower degree polynomials such as degree ≤ 4 in practice.

(e) HC fits parallel computation more than SDPR.

(f) The effectiveness of sparse SDPR depends on the c-sparsity; for example, discretization of ODE, DAE, Optimal control problem and PDE.

Thank you!