Solving semi-infinite programming problems by using an interface between MATLAB and SIPAMPL

A. Ismael F. Vaz
Edite M.G.P. Fernandes

Production and Systems Department
Engineering School
Minho University - Braga - Portugal
{aivaz,emgpf}@dps.uminho.pt

22-24 September, 2006

Outline

Outline

Semi-Infinite Programming

Problem

$$\begin{aligned} \min_{x \in R^n} \ & f(x) \\ s.t. \ \ & g_i(x,t) \leq 0, \ i = 1, ..., m \\ & h_i(x) \leq 0, \ i = 1, ..., o \\ & h_i(x) = 0, \ i = o+1, ..., q \\ & \forall t \in T \end{aligned}$$

Where $f(x)$ is the objective function, $h_i(x)$ are the finite constraint functions, $g_i(x,t)$ are the infinite constraint functions and $T \subset R^p$ is, usually, a cartesian product of intervals ($[\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \times \cdots \times [\alpha_p, \beta_p]$)

SIPAMPL environment

SIPAMPL was developed six years ago with the following purposes:

- To allow an easy and fast way to code SIP problems, using a (SIP)AMPL format
- To allow the interface between the coded problems and any solver
- To use AMPL software for automatic differentiation and its modeling language
- To provide a database with semi-infinite programming problems (to the image of AMPL or CUTE for finite problems)

SIPAMPL environment

SIPAMPL was developed six years ago with the following purposes:

- To allow an easy and fast way to code SIP problems, using a (SIP)AMPL format
- To allow the interface between the coded problems and any solver
- To use AMPL software for automatic differentiation and its modeling language
- To provide a database with semi-infinite programming problems (to the image of AMPL or CUTE for finite problems)

SIPAMPL environment

SIPAMPL was developed six years ago with the following purposes:

- To allow an easy and fast way to code SIP problems, using a (SIP)AMPL format
- To allow the interface between the coded problems and any solver
- To use AMPL software for automatic differentiation and its modeling language
- To provide a database with semi-infinite programming problems (to the image of AMPL or CUTE for finite problems)

SIPAMPL environment

SIPAMPL was developed six years ago with the following purposes:

- To allow an easy and fast way to code SIP problems, using a (SIP)AMPL format
- To allow the interface between the coded problems and any solver
- To use AMPL software for automatic differentiation and its modeling language
- To provide a database with semi-infinite programming problems (to the image of AMPL or CUTE for finite problems)

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- Select tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL today

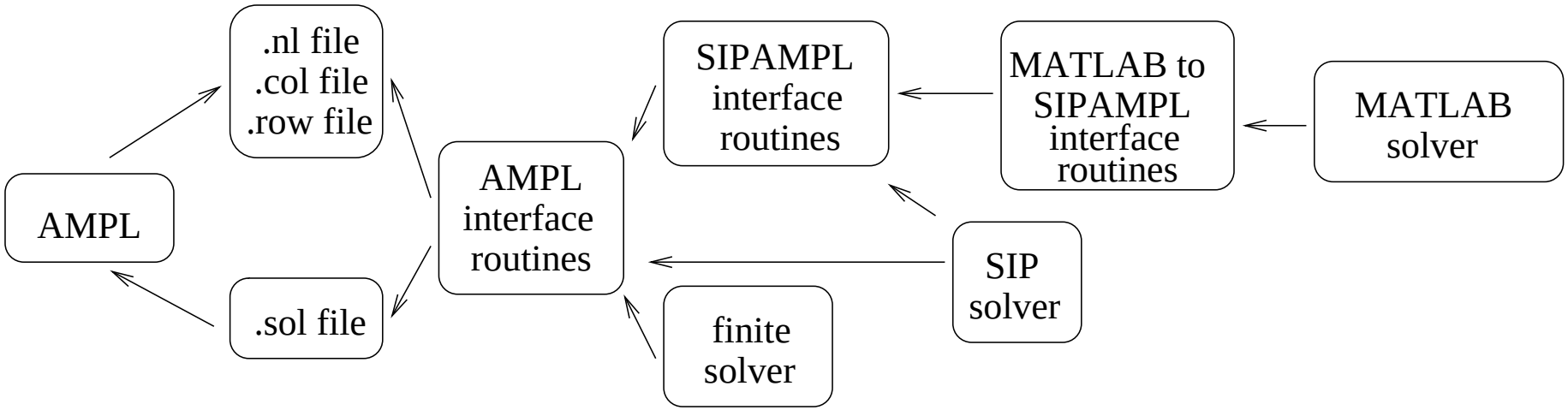
SIPAMPL is publicly available at

http://www.norg.uminho.pt/aivaz/

SIPAMPL provides:

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Dynamic fed-batch library (optimal control problems) - to be available soon
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL and a SIP solver interaction



Outline

MATLAB solver

- The user has to define the MATLAB functions to provide the objective, objective gradient and constraints function values;
- The user defines an initial sampling interval for the infinite constraints and the constraints function returns m vectors or matrices (of at most dimension two), which are the infinite constraints evaluated at an equally spaced grid of points.

Limitation

Since MATLAB expects vectors or matrices for the constraints, the use of SIPAMPL by MATLAB is limited to problems with 2 infinite constraints.

MATLAB solver

- The user has to define the MATLAB functions to provide the objective, objective gradient and constraints function values;
- The user defines an initial sampling interval for the infinite constraints and the constraints function returns m vectors or matrices (of at most dimension two), which are the infinite constraints evaluated at an equally spaced grid of points.

Limitation

Since MATLAB expects vectors or matrices for the constraints, the use of SIPAMPL by MATLAB is limited to problems with 2 infinite constraints.

MATLAB solver

- The user has to define the MATLAB functions to provide the objective, objective gradient and constraints function values;
- The user defines an initial sampling interval for the infinite constraints and the constraints function returns m vectors or matrices (of at most dimension two), which are the infinite constraints evaluated at an equally spaced grid of points.

Limitation

Since MATLAB expects vectors or matrices for the constraints, the use of SIPAMPL by MATLAB is limited to problems with 2 infinite constraints.

The MATLAB algorithm

Properties

- A quasi-Newton SQP algorithm with line search and a merit function;
- Identifies peaks in the discretized constraints function values and applies a quadratic or cubic interpolation to obtain an estimate of the maxima in the constraints;
- Since the number of maxima can change during the iterative process, the Lagrange multipliers are reallocated to the new set of maxima.

Implementation

The MATLAB solver is implemented in the `fseminf` function.

The MATLAB algorithm

Properties

- A quasi-Newton SQP algorithm with line search and a merit function;
- Identifies peaks in the discretized constraints function values and applies a quadratic or cubic interpolation to obtain an estimate of the maxima in the constraints;
- Since the number of maxima can change during the iterative process, the Lagrange multipliers are reallocated to the new set of maxima.

Implementation

The MATLAB solver is implemented in the `fseminf` function.

The MATLAB algorithm

Properties

- A quasi-Newton SQP algorithm with line search and a merit function;
- Identifies peaks in the discretized constraints function values and applies a quadratic or cubic interpolation to obtain an estimate of the maxima in the constraints;
- Since the number of maxima can change during the iterative process, the Lagrange multipliers are reallocated to the new set of maxima.

Implementation

The MATLAB solver is implemented in the `fseminf` function.

The MATLAB algorithm

Properties

- A quasi-Newton SQP algorithm with line search and a merit function;
- Identifies peaks in the discretized constraints function values and applies a quadratic or cubic interpolation to obtain an estimate of the maxima in the constraints;
- Since the number of maxima can change during the iterative process, the Lagrange multipliers are reallocated to the new set of maxima.

Implementation

The MATLAB solver is implemented in the `fseminf` function.

The MATLAB algorithm

Properties

- A quasi-Newton SQP algorithm with line search and a merit function;
- Identifies peaks in the discretized constraints function values and applies a quadratic or cubic interpolation to obtain an estimate of the maxima in the constraints;
- Since the number of maxima can change during the iterative process, the Lagrange multipliers are reallocated to the new set of maxima.

Implementation

The MATLAB solver is implemented in the `fseminf` function.

Outline

Interfaces

Previous interface

The previous interface between MATLAB and AMPL was mainly developed to allow the use of the `fseminf` function with problems coded in AMPL.

New interface

The new interface exports all SIPAMPL functions to MATLAB, allowing a greater flexibility in accessing from MATLAB the problem coded in AMPL.

Interfaces

Previous interface

The previous interface between MATLAB and AMPL was mainly developed to allow the use of the `fseminf` function with problems coded in AMPL.

New interface

The new interface exports all SIPAMPL functions to MATLAB, allowing a greater flexibility in accessing from MATLAB the problem coded in AMPL.

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

- provide the objective and constraints functions as M files
- call the `fseminf` function with the appropriate arguments

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

- provide the objective and constraints functions as M files
- call the `fseminf` function with the appropriate arguments

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

- provide the objective and constraints functions as M files
- call the `fseminf` function with the appropriate arguments

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

- provide the objective and constraints functions as M files
- call the `fseminf` function with the appropriate arguments

Some details

The new interface is composed by

- an external MEX file (`sipampl2`) written in the C programming language
- a set of M files that provides the problem data to MATLAB (by calling the MEX file `sipampl2`).

The user is requested to

- provide the objective and constraints functions as M files
- call the `fseminf` function with the appropriate arguments

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

MATLAB interface with SIPAMPL

`sip_init`	initializes and
`sip_end`	stops the SIPAMPL interface routines
`sip_objval/grd/hes`	objective function value, gradient and Hessian
`sip_conval`	values of all the constraints
`sip_contval/grd/hes`	infinite constraint value, gradient and Hessian
`sip_conxeqval/grd/hes`	finite equality constraint value, gradient and Hessian
`sip_conxineqval/grd/hes`	finite inequality constraint value, gradient and Hessian
`sip_jacval`	values of all Jacobians
`sip_usage`	usage of all the described functions

Example of MATLAB objective function

```
function [f,g]=mysipfun(x,s)
if nargin < 1 | nargout<1
          error('Invalid number of arguments');
end
f=sip_objval(x);
if nargout > 1
          g=sip_objgrd(x);
end
```

Example of MATLAB objective function

```
function [f,g]=mysipfun(x,s)
if nargin < 1 | nargout<1
          error('Invalid number of arguments');
end
f=sip_objval(x);
if nargout > 1
          g=sip_objgrd(x);
end
```

Example of MATLAB objective function

```
function [f,g]=mysipfun(x,s)
if nargin < 1 | nargout<1
          error('Invalid number of arguments');
end
f=sip_objval(x);
if nargout > 1
          g=sip_objgrd(x);
end
```

Example of MATLAB objective function

```
function [f,g]=mysipfun(x,s)
if nargin < 1 | nargout<1
          error('Invalid number of arguments');
end
f=sip_objval(x);
if nargout > 1
          g=sip_objgrd(x);
end
```

Outline

Test problems

Select tool

- Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.
- The *Select* tool can optionally provide a shell script, batch and `m`-file to run all the problems.

Due to approach limitation

Problems with at most two infinite variables.

Initial guess

Problems without initial guess provided by the user were not considered.

Test problems

Select tool

- Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.
- The *Select* tool can optionally provide a shell script, batch and `m`-file to run all the problems.

Due to approach limitation

Problems with at most two infinite variables.

Initial guess

Problems without initial guess provided by the user were not considered.

Test problems

Select tool

- Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.
- The *Select* tool can optionally provide a shell script, batch and `m`-file to run all the problems.

Due to approach limitation

Problems with at most two infinite variables.

Initial guess

Problems without initial guess provided by the user were not considered.

Test problems

Select tool

- Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.
- The *Select* tool can optionally provide a shell script, batch and `m`-file to run all the problems.

Due to approach limitation

Problems with at most two infinite variables.

Initial guess

Problems without initial guess provided by the user were not considered.

Test problems

Select tool

- Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.
- The *Select* tool can optionally provide a shell script, batch and `m`-file to run all the problems.

Due to approach limitation

Problems with at most two infinite variables.

Initial guess

Problems without initial guess provided by the user were not considered.

Selected problems

Problem	nx	nt	nxc	ntc	Problem	nx	nt	nxc	ntc
andreson1	3	2	0	1	blankenship1	2	1	0	1
coopeL	2	1	0	1	coopeM	2	1	1	1
coopeN	2	1	0	1	elke10	9	1	0	7
elke1std	9	1	0	19	elke2std	9	1	0	19
elke3std	9	1	0	19	elke4std	9	1	0	7
elke5std	9	1	0	19	elke6std	9	1	0	19
elke7std	9	1	0	19	elke8	9	1	0	7
elke9	9	1	0	7	fang1	50	1	0	1
fang2	50	1	0	1	fang3	50	1	0	1
ferris1	7	1	0	2	ferris2	7	1	0	1
gockenbach1	33	1	120	16	gockenbach10	33	1	120	16
gockenbach2	33	1	120	16	gockenbach3	33	1	120	16
gockenbach4	33	1	120	16	gockenbach5	33	1	120	16
gockenbach6	33	1	120	16	gockenbach7	33	1	120	16
gockenbach8	33	1	120	16	gockenbach9	33	1	120	16

Selected problems

Problem	nx	nt	nxc	ntc	Problem	nx	nt	nxc	ntc
goerner1	4	1	0	2	goerner2	5	1	0	2
goerner3	7	1	0	2	goerner4	7	2	0	2
goerner5	7	2	0	2	goerner6	16	2	0	2
goerner7	8	2	0	2	hettich10c	2	1	0	2
hettich5	3	2	0	2	leon1	4	1	0	2
leon10	3	1	0	2	leon11	3	1	0	2
leon12	2	1	0	1	leon13	2	1	0	1
leon14	2	1	0	1	leon15	2	1	0	1
leon16	3	1	0	1	leon17	3	1	0	1
leon18	2	1	0	1	leon19	5	1	0	1
leon2	6	1	0	2	leon3	6	1	0	2
leon4	7	1	0	2	leon5	8	1	0	2
leon6	5	1	0	2	leon7	5	1	0	2
leon8	7	1	0	2	leon9	7	1	0	2
li1	10	1	0	1	li2	6	1	0	1

Selected problems

Problem	nx	nt	nxc	ntc	Problem	nx	nt	nxc	ntc
lin1	6	2	0	1	matlab1	3	1	0	2
matlab2	3	2	0	1	powell1	2	1	0	1
priceK	2	1	0	1	random	4	2	4	4
tanaka1	2	1	1	1	teo1	3	1	0	1
teo2	3	1	0	1	watson1	2	1	0	1
watson10	3	2	0	1	watson11	3	2	0	1
watson12	3	2	0	1	watson13	3	2	0	1
watson14	2	1	0	1	watson2	2	1	0	1
watson3	3	1	0	1	watson4a	3	1	0	1
watson4b	6	1	0	1	watson4c	8	1	0	1
watson5	3	1	0	1	watson6	2	1	0	1
watson7	3	2	0	1	watson8	6	2	0	1
watson9	6	2	0	1	zhou1	2	1	0	1

Numerical results

Problem	nit	nf	fx	Problem	nit	nf	fx
andreson1	4	21	-0.333333	blankenship1	16	84	-0.000000
coopeL	7	34	0.343147	coopeM	4	20	1.000000
coopeN	5	21	0.000000	elke10[1]			
elke1std[1]				elke2std[1]			
elke3std[1]				elke4std[1]			
elke5std[1]				elke6std[1]			
elke7std[1]				elke8[1]			
elke9[1]				fang1	17	885	0.479429
fang2	41	2220	0.693170	fang3	78	4229	1.718288
ferris1	39	365	0.002192	ferris2	26	237	-1.782408

Problems not solved

[1] - Problem was stopped with a division by zero in the cubic interpolation (used to estimate the constraints peaks).

Problem	nit	nf	fx	Problem	nit	nf	fx
gockenbach1[2]				gockenbach10[2]			
gockenbach2[2]				gockenbach3[2]			
gockenbach4[2]				gockenbach5[2]	13	470	-0.007876
gockenbach6[2]				gockenbach7[2]			
gockenbach8[2]				gockenbach9[2]			
goerner1	15	98	0.004131	goerner2	15	123	0.004635
goerner3	18	167	0.000692	goerner4	9	85	0.052364
goerner5	17	183	0.027071	goerner6	52	1030	0.002309
goerner7	34	479	0.095039	hettich10c[1]			
hettich5	9	78	0.540096	leon1	33	277	0.005218
leon10	7	42	0.536725	leon11	55	302	1.847823
leon12	12	66	-0.999999	leon13[1]			
leon14[1]				leon15	6	25	-0.666667

Problems were interrupted

[2] - Problems were stopped after 10 hours of computation time.

Problem	nit	nf	fx	Problem	nit	nf	fx
leon16	18	151	1.856539	leon17	2	11	-2.000000
leon18[1]				leon19	24	187	0.785838
leon2	24	202	0.000042	leon3	14	143	0.004841
leon4	15	149	0.002603	leon5	32	380	0.014263
leon6	41	347	0.000138	leon7	32	245	0.001965
leon8	10	107	0.054449	leon9	23	268	0.203661
li1	62	1012	281497.529286	li2	22	199	39644.457962
lin1	22	189	-1.822943	matlab1	16	83	0.003573
matlab2	4	22	0.000200	powell1	12	66	-0.999999
priceK	7	30	-3.000000	random	1	7	-0.000001
tanaka1	17	114	-1.000000	teo1	16	81	0.167942
teo2	18	91	0.184545	watson1	12	53	-0.250162
watson10	3	16	0.275268	watson11	13	71	-4.386055
watson12	5	26	1.951452	watson13	13	74	1.950114
watson14	30	204	2.128976	watson2	7	29	2.430555
watson3	41	305	5.131713	watson4a	19	152	0.646698
watson4b	17	175	0.616780	watson4c	30	398	0.615752
watson5	7	55	4.301144	watson6	18	102	97.158852
watson7	7	36	1.000000	watson8	36	392	2.442798
watson9	38	412	-10.420355	zhou1[1]			

Outline

Conclusions

Conclusions

- SIPAMPL provides several coded SIP problems;
- SIPAMPL provides an interface between MATLAB and AMPL, allowing SIP problems coded in AMPL to be solved by MATLAB;
- MATLAB provides a solver for SIP;
- Numerical results shown with the MATLAB `fseminf` solver;

Conclusions

Conclusions

- SIPAMPL provides several coded SIP problems;
- SIPAMPL provides an interface between MATLAB and AMPL, allowing SIP problems coded in AMPL to be solved by MATLAB;
- MATLAB provides a solver for SIP;
- Numerical results shown with the MATLAB `fseminf` solver;

Conclusions

Conclusions

- SIPAMPL provides several coded SIP problems;
- SIPAMPL provides an interface between MATLAB and AMPL, allowing SIP problems coded in AMPL to be solved by MATLAB;
- MATLAB provides a solver for SIP;
- Numerical results shown with the MATLAB `fseminf` solver;

Conclusions

Conclusions

- SIPAMPL provides several coded SIP problems;
- SIPAMPL provides an interface between MATLAB and AMPL, allowing SIP problems coded in AMPL to be solved by MATLAB;
- MATLAB provides a solver for SIP;
- Numerical results shown with the MATLAB `fseminf` solver;

Conclusions

Conclusions

- SIPAMPL provides several coded SIP problems;
- SIPAMPL provides an interface between MATLAB and AMPL, allowing SIP problems coded in AMPL to be solved by MATLAB;
- MATLAB provides a solver for SIP;
- Numerical results shown with the MATLAB `fseminf` solver;

The End

Contacts

email: aivaz@dps.uminho.pt
emgpf@dps.uminho.pt
Web http://www.norg.uminho.pt/aivaz/