An interface between MATLAB and SIPAMPL for semi-infinite programming problems

A. Ismael F. Vaz
Edite M.G.P. Fernandes
Production and Systems Department
Engineering School
Minho University - Braga - Portugal
{aivaz,emgpf}@dps.uminho.pt

18-20 June, 2004

Contents

Semi-Infinite Programming

$$\min_{x \in R^n} \; f(x)$$

$$s.t. \;\; g_i(x,t) \leq 0, \; i = 1, ..., m$$

$$h_i(x) \leq 0, \; i = 1, ..., o$$

$$h_i(x) = 0, \; i = o+1, ..., q$$

$$\forall t \in T$$

$f(x)$ 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 - Motivations

SIPAMPL was developed four years ago with the following proposes:

- To allow an easy and fast way to code SIP problems, using a (SIP)AMPL format

SIPAMPL environment - Motivations

SIPAMPL was developed four years ago with the following proposes:

- 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

SIPAMPL environment - Motivations

SIPAMPL was developed four years ago with the following proposes:

- 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

SIPAMPL environment - Motivations

SIPAMPL was developed four years ago with the following proposes:

- 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

- More than 160 SIP problems coded

SIPAMPL is publicly available in http://www.norg.uminho.pt/aivaz/

SIPAMPL today

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)

SIPAMPL is publicly available in http://www.norg.uminho.pt/aivaz/

SIPAMPL today

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines

SIPAMPL is publicly available in http://www.norg.uminho.pt/aivaz/

SIPAMPL today

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines

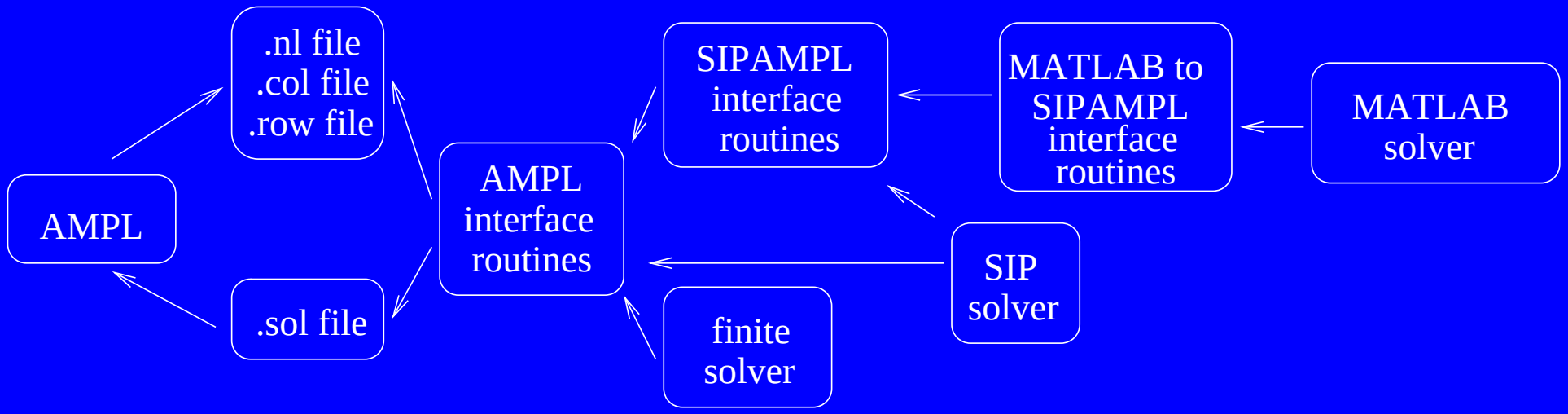
SIPAMPL is publicly available in http://www.norg.uminho.pt/aivaz/

SIPAMPL today

- More than 160 SIP problems coded
- Dynamic B- and C-Splines library (robotics problems)
- Interface routines between AMPL and any SIP solver (NSIPS) - SIPAMPL routines
- Interface routines between MATLAB and SIPAMPL routines
- *Select* tool

SIPAMPL is publicly available in http://www.norg.uminho.pt/aivaz/

SIPAMPL and a SIP solver interaction



Example of SIP problem ($hettich2$)

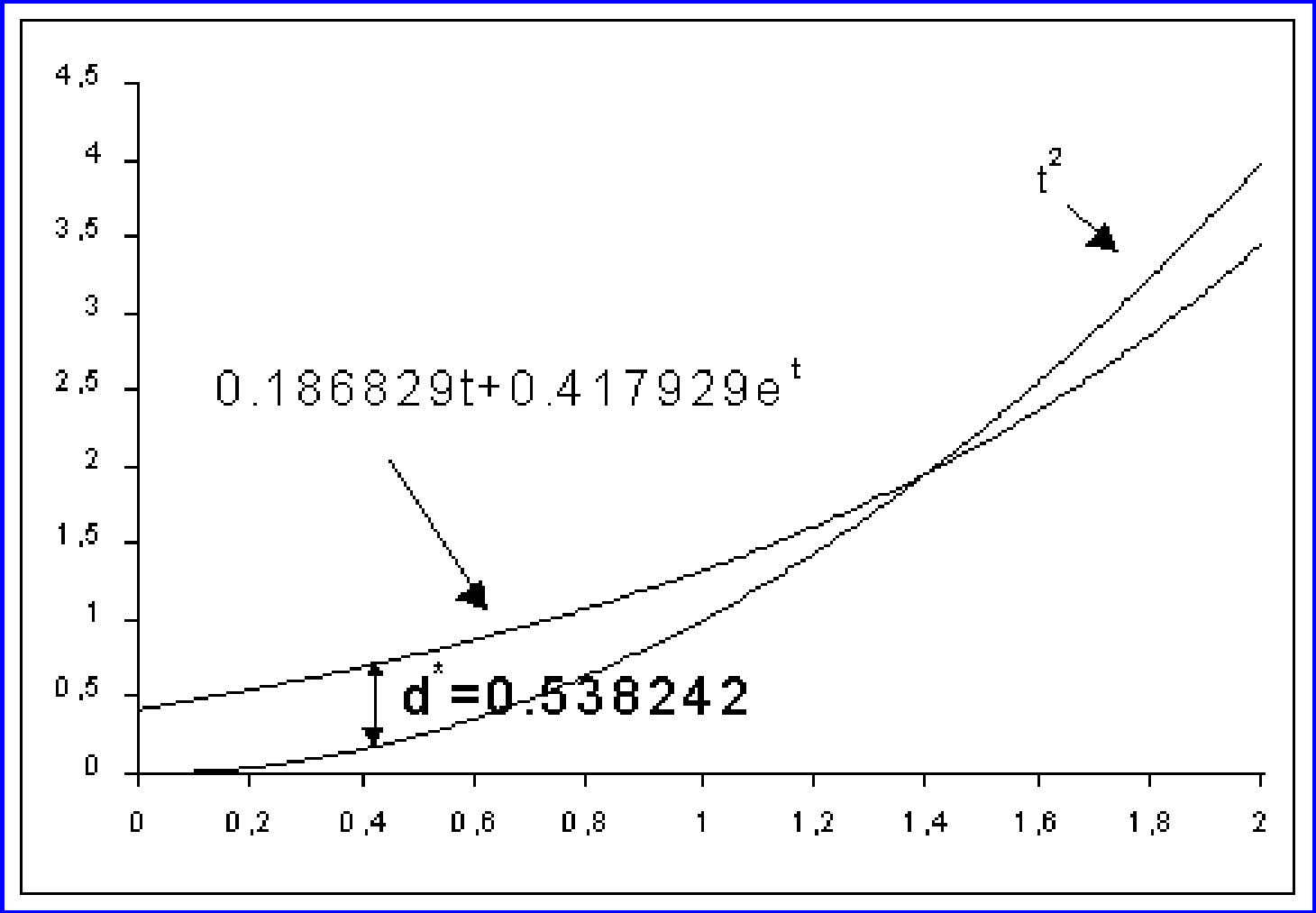
Approximation problem of t^2 by a combination of t and e^t. d is the minimum distance.

$$\min_{p,d \in R^{2+1}} \quad d$$

$$s.t. \quad |t^2 - (p_1 t + p_2 e^t)| - d \leq 0$$

$$\forall t \in [0, 2]$$

Problem solution



Example coded in AMPL ((SIP)AMPL format)

```
##############################################################
# Objective: Linear
# Constraints: Linear
##############################################################
#  Hettich set of SIP problems (Chebyshev approximation problems)
#   As described by R. Hettich, "A comparison of some numerical methods
#                                for semi-infinite programming",
#   in R. Hettich (eds.), "Semi-infinite Programming",
#    Lecture notes in Control and Information Sciences no. 15, pp. 112-125
#    Springer Verlag, Berlin (1979)
#
#   Example 2
#    Coded by A. Ismael F. Vaz 27/12/1999  aivaz@dps.uminho.pt
#    University of Minho, Portugal
##############################################################
var p {1..2};
var t {1..1};
var d;
```

```
minimize dp:
    d;

subject to tcons1:
    -(t[1]^2-(p[1]*t[1]+p[2]*exp(t[1])))-d <=0;
subject to tcons2:
    (t[1]^2-(p[1]*t[1]+p[2]*exp(t[1])))-d <=0;
subject to bounds {i in 1..1}:
    0 <= t[i] <= 2;

# don't forget to write .col and .row files
option nsips_auxfiles rc;
# this problem has no initial guess (starting point)
option reset_initial_guesses 1;
# change solver
option solver nsips;
# solve problem
solve;
```

```
printf "Solution found\n";
display p;
display dp;

# solution of problem presented by the authors
printf "Original Solution\n";
printf "p=(%lf,%lf) dp=%lf\n", 0.184, 0.418, 0.538;
```

MATLAB `fseminf` function

The MATLAB `fseminf` syntax is:

```
[x,fval,exitflag,output,lambda]=
    fseminf('mysipfun',x0,ntheta,'mysipcon',A,b,
                Aeq,beq,lb,ub,options,P1,P2,...).
```

Output arguments:

- `x` - solution found;
- `fval` - objective value at `x`;
- `exitflag` - exit condition of the algorithm;
- `output` - structure with algorithm information;
- `lambda` - Lagrange multipliers.

MATLAB `fseminf` function

The MATLAB `fseminf` syntax is:

```
[x,fval,exitflag,output,lambda]=
     fseminf('mysipfun',x0,ntheta,'mysipcon',A,b,
                    Aeq,beq,lb,ub,options,P1,P2,...).
```

Input arguments:

- `mysipfun` - objective function;

```
function [f,g] = mysipfun(x)
```

  where `f` and `g` are the objective function and gradient at `x`;

Input arguments:

- `x0` - initial guess;
- `ntheta` - number of infinite constraints;
- `mysipcon` - constraints function;

```
function [c,ceq,K1,K2,...,Kntheta,s] = mysipcon(x,s)
```

  where `c` and `ceq` are the finite inequality and equality constraints at `x`, `K1`, ..., `Kntheta` are vectors or matrices with the infinite constraints evaluated at an equally spaced (`s`) grid of points.

- `A`, `b`, `Aeq` and `beq` - inequality and equality linear constraints;
- `lb` and `ub` - lower and upper bounds on `x`;
- `options` - options to the algorithm;
- `P1`, `P2`, ... - extra arguments.

MATLAB solver

- The user has to define the MATLAB functions to provide the objective, objective gradient and constraints function values;

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 (`s`) for the infinite constraints and the constraints function returns `K1`, `K2`, ..., `Kntheta` vectors or matrices, which are the infinite constraints evaluated at an equally spaced grid of points.

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 (`s`) for the infinite constraints and the constraints function returns `K1`, `K2`, ..., `Kntheta` vectors or matrices, which are the infinite constraints evaluated at an equally spaced grid of points.

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

The MATLAB algorithm

- A quasi-Newton SQP algorithm with line search and a merit function;

The MATLAB algorithm

- 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;

The MATLAB algorithm

- 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.

MATLAB interface with SIPAMPL

sip_init sip_end	initialize and 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 constraint function

```
function [c,ceq,K1,K2,s]=mysipcon(x,s)

if nargin < 2 | nargout<5
  error('Invalid number of arguments');
end

if isnan(s(1,1)),
  s=[0.2 0; 0.2 0];
end

w1=1:s(1,1):100;
w2=1:s(2,1):100;

lw1=length(w1);
lw2=length(w2);

K1=zeros(lw1,1);
K2=zeros(lw2,1);

for i=1:lw1
  K1(i)=sip_contval(0,x,w1(i));
end

for i=1:lw2
  K2(i)=sip_contval(1,x,w2(i));
end

c=[]; ceq=[];

plot(w1,K1,'-',w2,K2,':'),

title('Semi-infinite constraints')

drawnow
```

A simple run

After producing the `matlab1.nl`, `matlab1.col`, `matlab1.row` (for SIPAMPL interface routines).

```
>> [x0, xbl, xbu, tbl, tbu]=sip_init('matlab1');
>> [x, fval]=fseminf('mysipfun',x0,2,'mysipcon')
Optimization terminated successfully:
 Search direction less than 2*options.TolX and
  maximum constraint violation is less than
  options.TolCon
Active Constraints:
     7
    10
x' =    0.6673    0.3013    0.4023
fval =
    0.0770
```

Test problems

Using the SIPAMPL *Select* tool to obtain the problems (files) names and the corresponding `.nl` files.

Test problems

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.

Test problems

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.

Problems with at most two infinite variables.

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
goerner1	4	1	0	2	goerner2	5	1	0	2
goerner3	7	1	0	2	goerner4	7	2	0	2

Selected problems

Problem	nx	nt	nxc	ntc	Problem	nx	nt	nxc	ntc
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
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

Selected problems

Problem	nx	nt	nxc	ntc	Problem	nx	nt	nxc	ntc
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

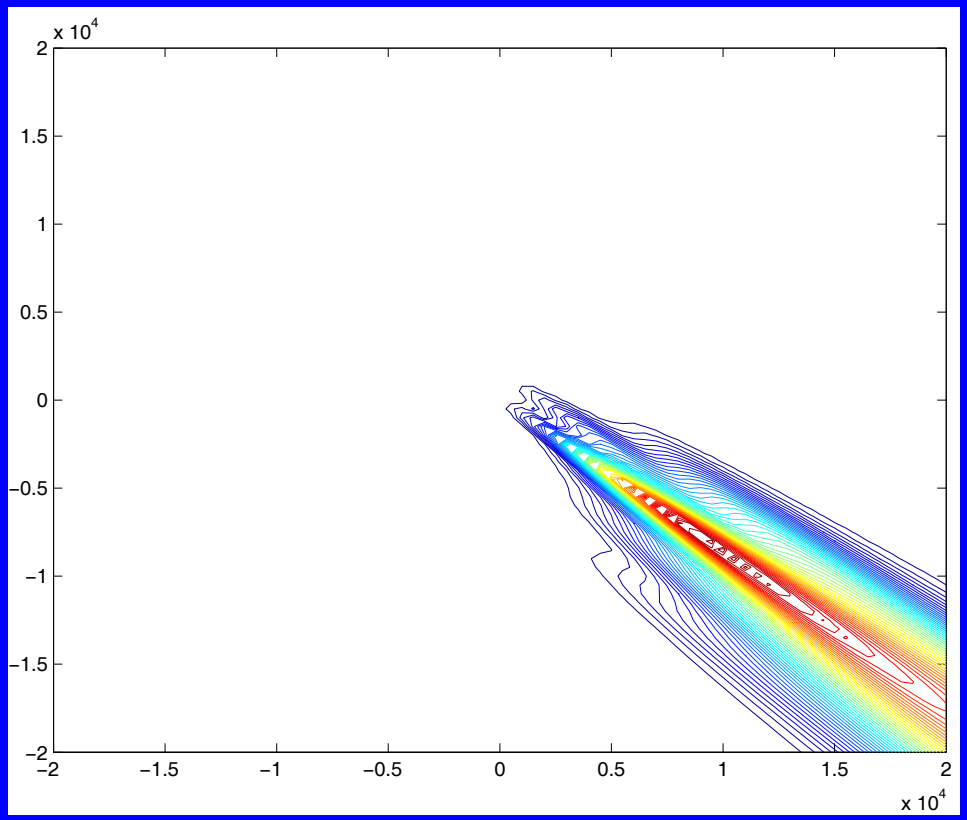
Problem	nit	nf	fx	Problem	nit	nf	fx
andreson1	4	21	-0.333333	blankenship1	16	84	-0.000000
coopeL	7	34	0.343147	coopeM	4	20	1.000000
coopeN	5	21	0.000000	elke10[1]			
elke1std[1]				elke2std[1]			
elke3std[1]				elke4std[1]			
elke5std[1]				elke6std[1]			
elke7std[1]				elke8[1]			
elke9[1]				fang1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

Problem	nit	nf	fx	Problem	nit	nf	fx
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
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

Problem	nit	nf	fx	Problem	nit	nf	fx
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]			

Plotting

The SIPAMPL interface routines also provide a great help in plotting.




```
>> [x0, xbl, xbu, tbl, tbu]=sip_init('vaz1');
>> t1=tbl(1):500:tbu(1);
>> t2=tbl(2):500:tbu(2);
>> for i=1:length(t1)
for j=1:length(t2)
gx(i,j)=sip_contval(0,x0,[t1(i) t2(j)]);
end
end
>> contour(t1,t2,gx,50);
>> sip_end('Free memory',x0);
```

Air pollution control problem `vaz1.mod`

Conclusions

- SIPAMPL provides several coded SIP problems;

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;

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;

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

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

First Page