SAND2006-7048C

Presentation at INFORMS Fall 2006, Pittsburgh

The AMPL® /Solver Interface Library

David M. Gay

Optimization and Uncertainty Estimation

dmgay@sandia.gov

+1-505-284-1456

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
Released as SAND2006-4717C.

Sandia National Laboratories

Outline

- Motivation and problems addressed
- Information available
- Sparsity
- Complementarity
- Suffixes
- Imported functions
- Turning Solver Knobs
- Forthcoming additions

Sandia National Laboratories

Motivation

AMPL® helps state, solve, analyze & manipulate finite-dimensional math. programming problems involving algebraic constraints and objectives.

AMPL (normally) uses separate *solvers* and must communicate with them.

ASL source is available from **`netlib`**.

Sandia National Laboratories

Problems Addressed

$$\text{Minimize (or maximize) } f(x)$$

$$\text{s.t. } \ell \leq c(x) \leq u$$

$$\text{and } x \in D$$

$$f : \mathcal{R}^n \rightarrow \mathcal{R}$$

$$c : \mathcal{R}^n \rightarrow \mathcal{R}^m$$

$D =$ Cartesian prod. of integer or real intervals.

Sandia National Laboratories

Variations in Solver Views

Some want standard form:

$$c(x) = 0, \quad \ell \leq x \leq u.$$

Some disallow range constraints:

$$\ell \leq c(x) \leq u$$

$$\text{with } -\infty < \ell < u < \infty.$$

Some forbid $\ell = -\infty$ or $u = \infty$.

Some require $\ell = -\infty$ or $u = \infty$.

More Variations in Solver Views

Some want only function values,
some want functions and gradients,
some want functions, gradients, and
the Lagrangian Hessian.

Some want to see linear and nonlinear
constraints separately.

Sandia National Laboratories

Still More Solver Variations

Some distinguish the numbers of nonlinear variables in objectives and constraints.

Only a few handle complementarity constraints — and differ in generality.

Some want funcs. component-wise; others want everything at once.

Sandia National Laboratories

Yet More Solver Variations

Some want Hessian-vector products; others want explicit Hessians

- entirely (dense or sparse)
- only a triangle (row- or col-wise).

Sign conventions for dual variables vary.

Interface Library + Solver Drivers

AMPL/Solver interface library (ASL) helps write solver *drivers*.

- `.nl` file has all info except names
- build structures for efficient eval. while reading `.nl` file
- write `.sol` file with "solve" results.

Sandia National Laboratories

Solver Driver Outline

```
 #include "asl.h"

....
ASL *asl = ASL_alloc(ASL_read_kind);
FILE *nl = jac0dim_ASL(asl, stub, 0);
      ....allocate some things....
kind_read_ASL(asl, nl, flags);
        ....solve the problem....
write_sol_ASL(asl, msg, x, y, 0);
ASL_free(&asl);
```

Places to Elide "asl" and "_ASL"

```
 #include "asl.h"

....
ASL *asl = ASL_alloc(ASL_read_kind);
FILE *nl = jac0dim_ASL(asl, stub, 0);
      ....allocate some things....
kind_read_ASL(asl, nl, flags);
        ....solve the problem....
write_sol_ASL(asl, msg, x, y, 0);
ASL_free(&asl);
```

#defines permit eliding "asl" and "_ASL"

```
 #include "asl.h"

....
ASL *asl = ASL_alloc(ASL_read_kind);
FILE *nl = jac0dim(stub, 0);
      ....allocate some things....
kind_read(nl, flags);
        ....solve the problem....
write_sol(msg, x, y, 0);
ASL_free(&asl);
```

Driver with Option Processing

```
 #include "getstub.h"

....
ASL *asl = ASL_alloc(ASL_read_kind);
char *stub = getstops(argv, &Oinfo);
FILE *nl = jac0dim(stub, 0);
      ....allocate some things....
kind_read(nl, flags);
        ....solve the problem....
write_sol(msg, x, y, &Oinfo);
ASL_free(&asl);
```

`ASL_read_`*kind* Choices

`ASL_read_f`	LP only
`ASL_read_fg`	f and ∇f
`ASL_read_pfgh`	p.s. f, ∇f, and $\nabla^2 f$

Assign `want_deriv = 0` for no ∇f.

Others (`fgh`, `pfg`) for debugging only.

Sandia National Laboratories

Problem Dimensions

`jac0dim` reads `.nl` *header*, providing

- `n_var` no. of variables
- `n_con` no. of constraints
- `n_obj` no. of objectives
- `n_cc` no. of complementarities

And much more (e.g., on nonlin.).

Sandia National Laboratories

Sample .nl Header

```
g3 2 1 0 # problem pg6
 10 19 1 0 0 # vars, constraints, objectives, ranges, e
 15 1 # nonlinear constraints, objectives
 0 0 # network constraints: nonlinear, linear
 10 10 10 # nonlinear vars in constraints, objectives,
 0 0 0 1 # linear network variables; functions; arith,
 0 0 0 0 0 # discrete variables: binary, integer, nonli
 53 10 # nonzeros in Jacobian, gradients
 0 0 # max name lengths: constraints, variables
 0 0 0 0 0 # common exprs: b,c,o,c1,o1
```

Misc. Features

`#defines` enhance readability:

`n_var = asl->i.n_var_`

Memory from `M1alloc(size_t len)`
freed implicitly by `ASL_free(&asl)`.

`M1zapalloc(len)`
= `M1alloc(len)` + zero-init.

Sandia National Laboratories

More Misc. Features

`Malloc` = `malloc` + fail check.

Assign `A_vals` before `f_read(...)` for col-wise LP constraint matrix.

To check for a QP, replace `fg_read` with `qp_read`, `nqpcheck`, and (before nonlin. eval's) `qpopify`.

Sandia National Laboratories

Bounds

Arrays $\underline{x}$, $\overline{x}$, $\underline{c}$, and $\overline{c}$ give bounds on variables and constraints:

$$\underline{x} \leq x \leq \overline{x}$$

$$\underline{c} \leq c(x) \leq \overline{c}.$$

Available as separate arrays or arrays of pairs.

Nonlinear Functions

`objval(i,x,e)` $= f_i(x)$.

`conival(i,x,e)` $= c_i(x)$.

`conval(x,r,e)` sets

$$\texttt{r} \leftarrow (c_0(x), c_1(x), ..., c_{m-1}(x))^T.$$

`objgrd(i,x,g,e)` sets $\texttt{g} \leftarrow \nabla f_i(x)$.

`conigrd(i,x,g,e)` sets $\texttt{g} \leftarrow \nabla c_i(x)$.

`conval(x,J,e)` sets

$\texttt{J} \leftarrow [\nabla c_0(x), \nabla c_1(x), ..., \nabla c_{m-1}(x)]^T$.

Hessians

$$H = \nabla_x^2 \left(f(x) + \sum_{i=0}^{m-1} y_i c_i(x) \right)$$

or more generally

$$H(w,x,y) = \nabla_x^2 \left(\sum_{i=0}^{\texttt{n\_obj}-1} w_i f_i(x) + \sigma \sum_{i=0}^{\texttt{n\_con}-1} y_i c_i(x) \right)$$

Can get Hv, (dense or sparse triangle of) H, and sparsity of H.

Complementarity

When `n_cc` > 0,

$j =$ `cvar[`i`]` $- 1 \geq 0 \;\Rightarrow\;$ constraint i is a complementarity condition:

$$x_j = \underline{x}_j \;\Rightarrow\; c_i(x) \geq 0;$$
$$x_j = \overline{x}_j \;\Rightarrow\; c_i(x) \leq 0;$$
$$\underline{x}_j < x_j < \overline{x}_j \;\Rightarrow\; c_i(x) = 0.$$

Suffixes $\longleftrightarrow$ Auxiliary Info.

- Initial, final basis
- Ray of unboundedness
- Solver-specific variable, constraint, or objective info
- SOS info (e.g., from nonconvex p-l)
- priorities

Imported Functions

- User-defined functions come from shared libraries.
- Same libs. work with AMPL and solvers.
- AMPL sets `$ampl_funclibs` to say which libs are used. (ASL looks at `$AMPLFUNC` if `$ampl_funclibs` = 0.)

Sandia National Laboratories

Some Imported Function Details

- Args numeric or string.
- If nonzero, `al->derivs` $\longleftarrow \nabla f$, `al->hes` $\longleftarrow \nabla^2 f$ w.r.t. num. args.
- `al->funcinfo` passed through.
- "`AmplExports *ae = al->AE`" + `#defines` give access to stdio, temp. mem., `at_exit`, `at_reset`, `getenv`

Turning Solver Knobs

```
stub = getstops(argv, &Oinfo);
```

or

```
stub = getstub(&argv, &Oinfo);
```

....read .nl header, open lic....

```
if (getopts(argv, &Oinfo))...
```

provide control, common options.

Solver Options

Command line...

```
minos pg6 outlev=2
```

or

```
minos_options='outlev=2' minos pg6
```

or from AMPL

```
option minos_options 'outlev=2';
```

Common Command-Line Options

`-?` show usage

`-=` summarize "`name=...`" options

`-s` write `.sol` files (without `-AMPL`)

— alternative to `wantsol=1`

`-u` report available imported functions

`-v` just report version, license

Sandia National Laboratories

Forthcoming (?)

- More constraint-programming facilities
- Random variables for stochastic prog.
- Update of "Hooking Your Solver to AMPL"

Sandia National Laboratories

More Info.

Hooking Your Solver to AMPL and pointers to source and examples:
`http://www.ampl.com/hooking.html`

Writing `.nl` *Files*: `http://endo.sandia.gov/~dmgay/nlwrite.pdf`

Sandia National Laboratories