Found an error or want to propose an extension?
Contact Jonas Asprion <asprionj@ethz.ch>

The MATLAB Interface for IPOPT

1. Function call

Syntax

```
[x, info] = ipopt (x0, funcs, options)
```

Arguments

`x0` Vector of initial values for the primal variables, length *N*.

`funcs` A struct with the fields

`funcs.objective`	
`funcs.gradient`	
`funcs.constraints`	(optional)
`funcs.jacobian`	(necessary if constraints are present)
`funcs.jacobianstructure`	(necessary if constraints are present)
`funcs.hessian`	(necessary if option `hessian_approximation = 'exact'`)
`funcs.hessianstructure`	(necessary if option `hessian_approximation = 'exact'`)
`funcs.iterfunc`	(optional)

The meaning of the individual functions is explained below.

`options` A struct with the fields

`options.lb`	
`options.ub`	
`options.cl`	
`options.cu`	
`options.auxdata`	(optional)
`options.zl`	(optional, for warm start)
`options.zu`	(optional, for warm start)
`options.lambda`	(optional, for warm start)
`options.ipopt`	

The meaning of the individual fields is explained below.

Return values

`x` Solution, i.e. optimised values of the primal variables.

`info` A struct with the fields

```
info.zl
info.zu
info.lambda
info.status
info.iter
info.cpu
```

The meaning of the individual fields is explained below.

2. The *funcs* struct

objective

Function handle for the objective function. Assignment of a function handle by `funcs.objective = @objective;` where `objective` is the name of a valid Matlab function. Signature:

```
function f = objective (x, auxdata)
```

`x` Current values of the primal variables. Format identical to `x0`.

`auxdata` (optional) User-defined data, cf. `options.auxdata`.

`f` Function value at the current point `x`.

gradient

Function handle for the objective gradient. Signature:

```
function g = gradient (x, auxdata)
```

`x` Current values of the primal variables. Format identical to `x0`.

`auxdata` (optional) User-defined data, cf. `options.auxdata`.

`g` Gradient (vector of partial derivatives w.r.t. `x`) of the objective at the current point `x`.

constraints

(optional) Function handle for the constraints function. Signature:

```
function c = constraints (x, auxdata)
```

The return value `c` is a vector of length *M* containing the values of the constraints at the current point x. Its size has to be equal to the lower and upper limits in `options.cl`and `options.cu`.

jacobian

(necessary if constraints are present) Function handle for the constraints Jacobian. Signature:

```
J = jacobian (x)
```

`J` is a *M*×*N* matrix in MATLAB's sparse format. It contains all first derivatives of the constraints w.r.t. the primal variables at the current point `x`.

Refer to the MATLAB help for details on the sparse format. Quick-guide:

```
A_sp = sparse(A);             % transform dense matrix A to sparse format
A    = full(A_sp);            % transform sparse matrix A_sp to dense format
A_sp = spalloc(4,3,8);        % allocate memory for a 4x3 sparse matrix, max. 8 non-zero entries
A_sp(2,3) = 3;                % normal indexing possible
```

Normal indexing may be slow since all data stored after the indexed element has to be shifted (if the indexed entry is changed from zero to nonzero). Most matrix operations between a dense and a sparse matrix result in a dense matrix. The MATLAB help provides details on both topics.

jacobianstructure

(necessary if constraints are present) Function handle for the structure of the constraints Jacobian. Signature:

```
function J_str = jacobianstructure (auxdata)
```

`J_str` is a sparse $M \times N$ matrix with ones wherever the constraint Jacobian potentially has nonzero entries.

hessian

(necessary if option `hessian_approximation = 'exact'`) Function handle for the Hessian of the Lagrangian. Signature:

```
function H = hessian (x, sigma, lambda, auxdata)
```

`x` Current values of the primal variables. Format identical to `x0`.

`sigma` Multiplier for the objective part of the Hessian.

`lambda` Vector of Lagrange multipliers of the constraints, length M.

`auxdata` (optional) User-defined data, cf. `options.auxdata`.

`H` The $N \times N$ Hessian matrix, i.e. second partial derivatives, of the Lagrangian w.r.t. the primal variables `x`.

Lagrangian: `L = sigma*objective + lambda'*c`

Hessian of the Lagrangian:
```
H = sigma*H_objective + lambda(1)*H_c(1) + ... +
                                          lambda(M)*H_c(M)
```

Here, `H_objective` is the Hessian of the objective and `H_c(i)` the Hessian of constraint i. Again, MATLAB's sparse matrix format has to be used. Since the Hessian is symmetric, only its lower triangular part has to be returned, i.e.

```
H = tril(H_full);
```

hessianstructure

(necessary if option `hessian_approximation = 'exact'`) Function handle for the structure of the Hessian of the Lagrangian. Signature:

```
function H_str = hessianstructure (auxdata)
```

`H_str` is a sparse, lower-triangular $N \times N$ matrix with ones wherever the lower-triangular part of the Hessian of the Lagrangian potentially has nonzero entries.

iterfunc

(optional) Function handle of a function which is called once each NLP iteration. Its arguments are the iteration count `nIter` and the current objective value `f`. If it returns `true`, the optimisation is continued; if it returns `false`, the optimisation is stopped after the current iteration. Signature:

```
function b = iterfunc(nIter, f, auxdata)     % b = {true, false}
```

3. The *options* struct

lb

Lower bounds for the primal variables `x`, vector of length *N*. If no lower bound exists for a specific variable, the corresponding entry in `options.lb` can be set to `-Inf`.

ub

Upper bounds for the primal variables `x`, vector of length *N*. If no upper bound exists for a specific variable, the corresponding entry in `options.ub` can be set to `Inf`.

cl / cu

Lower / upper bounds for the constraints, vectors of length *M*. If no bound is imposed on one side (inequality constraint), the corresponding entry can be set to `+/-Inf`. Equality constraints are imposed by setting `cl(i) = cu(i)`.

auxdata

(optional) This field accepts data in an arbitrary format. If the field is nonempty, this data is passed to the functions as last argument as indicated in Sec. 2. In this case, ALL functions concerned have to accept this additional argument.

Alternatively, global variables can be used to provide additional information to the functions.
[It has not been evaluated if there is a difference in the performances of the two alternatives.]

zl / zu / lambda

(optional, for warm start) Set initial values for the Lagrange multipliers for the lower / upper bounds on the variables and for the constraints. A warm start is enabled by setting the option `warm_start_init_point = 'yes';`

ipopt

This field is used to set all parameters for IPOPT, cf. the options reference at
http://www.coin-or.org/Ipopt/documentation/node59.html.

Some of the most important options are summarised below. Curly brackets denote the default setting, square brackets indicate a possibly good choice if different from the default (by experience on a specific, real-world problem).

```
% General
options.ipopt.max_iter          = 100;              % max. number of iterations
options.ipopt.tol               = 1e-8;             % accuracy to which the NLP is solved
options.ipopt.print_level       = 5;                % {6}, [5] is a nice compact form

% Barrier strategy
options.ipopt.mu_strategy       = 'adaptive';       % monotone (Fiacco-McCormick), {adaptive}
IPOPT_opts.ipopt.mu_oracle      = 'probing';        % {quality-function}, loqo, [probing]


% Constraint-multipliers update
IPOPT_opts.ipopt.alpha_for_y = 'primal-and-full';
    % {primal}, min, max, full, min-dual-infeas, safer-min-dual-infeas,
    % [primal-and-full], dual-and-full

% Hessian; quasi-Newton
IPOPT_opts.ipopt.hessian_approximation      = 'limited-memory';
IPOPT_opts.ipopt.limited_memory_update_type = 'bfgs';      % {bfgs}, sr1
IPOPT_opts.ipopt.limited_memory_max_history = 6;           % {6}

% Hessian; exact
IPOPT_opts.ipopt.hessian_approximation      = 'exact';

% Derivative test (set max_iter = 0 if ONLY the derivative check is to be performed)
IPOPT_opts.ipopt.derivative_test = 'only-second-order';    % {none}, first-order, ...
                                                           % second-order, only-second-order
IPOPT_opts.ipopt.derivative_test_perturbation = 3e-6;      % {1e-8}, [3e-6]
```

4. The *info* struct

status

This number contains information about the reason for termination:

```
   0  solved
   1  solved to acceptable level
   2  infeasible problem detected
   3  search direction becomes too small
   4  diverging iterates
   5  user requested stop

  -1  maximum number of iterations exceeded
  -2  restoration phase failed
  -3  error in step computation
 -10  not enough degrees of freedom
 -11  invalid problem definition
 -12  invalid option
 -13  invalid number detected

-100  unrecoverable exception
-101  non-IPOPT exception thrown
-102  insufficient memory
-199  internal error
```

zl / zu / lambda

Values of the Lagrange multipliers for the lower / upper bounds on the variables and the constraints, for the solution obtained. Can be used to warm start a continuation of the problem at hand or a similar/perturbed problem.

iter

Number of NLP iterations performed.

eval

Number of evaluations of the objective, the constriants, the objective gradient, the constraint Jacobian and the Hessian of the Lagrangian.

cpu

CPU time required by IPOPT.