EMSOL Subroutine Library: Version 0.2.n

J.A.J. Hall

1st March 2012

Contents

Chapter 1

Introduction

The Edinburgh Management Science Optimization Library (EMSOL) is an environment for solving large scale linear programming (LP) models and consists of a suite of FORTRAN subroutines. The following facilities are currently available.

- Read an LP model into EMSOL using an EMSOL file.
- Pass an LP model into EMSOL through a parameter list.
- Solve an LP model by the primal revised simplex method.
- Print the solution.
- Modify the LP model by adding rows/columns to the constraint matrix.
- Perform a matrix-vector product using the constraint matrix (or its transpose).
- Switch to another LP model already held within EMSOL.

EMSOL has been written with the aim of limiting the length of FORTRAN parameter lists and minimizing the requirement for the user to declare arrays whose length may be given by the dimension of the problem to be solved. This is achieved by the user declaring a single array of doublewords which is partitioned internally by EMSOL in order to store arrays of data for the models within the system.
Section 2 describes each of the subroutines in the EMSOL library, giving full details of the entries in their FORTRAN parameter list.

Chapter 2

Subroutines

This section describes the EMSOL subroutines. For each subroutine the following information is given.

- The purpose of the subroutine.
- The FORTRAN parameter list.
- A description of each item in the parameter list both on entry and on return.

ems_bcdo

This subroutine writes the current model to a file in EMSOL fixed format.

Specification

`call ems_bcdo(`*rtcod*, *dspace*, *unit*, *type*, *nfields*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

unit

is the unit number specifying where the basis file is read from.
Specified as: a fullword integer; $0 \leq unit \leq 99$.
Note: *unit* cannot be 0, 2 or 5.

type

is not currently used.
Specified as: a fullword integer. Its value must be 1.

nfields

is not currently used.
Specified as: a fullword integer. Its value must be 2.

On return

rtcod

is the return code for the subroutine, where:
If $rtcod = 0$, the subroutine completed successfully. Only informational messages were issued.
If $rtcod > 0$, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_dsca

This subroutine should be called once at the beginning of each application. It is the way to define the application to EMSOL, both in terms of the number of models in the application and and the amount of space available for the application.

Specification

`call ems_dsca`(*rtcod*, *dspace*, *ndwords*, *nmodel*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

ndwords

is the number of doublewords in *dspace*.
Specified as: a fullword integer.

nmodel

is the number of models in the application Specified as: a fullword integer; $nmodel \geq 1$.

On return

rtcod

is the return code for the subroutine, where:
If $rtcod = 0$, the subroutine completed successfully. Only informational messages were issued.
If $rtcod > 0$, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_iget

This subroutine requests current values of the integer control variables.

Specification

`call ems_iget(`*rtcod*, *dspace*, *iarray*, *num*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

num

is the number of integer control variables to be retrieved.
Specified as: a fullword integer; $0 \leq num \leq (length\ of\ iarray)$.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

iarray

is an array of integer control variables, where $iarray(n)$ is the nth integer control variable.
Specified as: a one-dimensional integer array of fullwords.

See Section 4.2 on page 32 for a list of integer control variables.

Notes

ems_init

This subroutine initialises EMSOL and *dspace*. It also resets the control variables as part of the initialisation. All calls to `ems_dsca` except the first one should be preceded by a call to `ems_init`.

Specification

`call ems_init`(*rtcod*, *dspace*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_iset

This subroutine sets the values of integer control variables.
`ems_iget` **must be called before calling** `ems_iset`**.**

Specification

call ems_iset(*rtcod*, *dspace*, *iarray*, *num*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

iarray

is an array of integer control variables, where $iarray(n)$ is the nth integer control variable.
Specified as: a one-dimensional integer array of fullwords.

num

is the number of integer control variables to be set.
Specified as: a fullword integer; $0 \leq num \leq (length\ of\ iarray)$.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* $= 0$, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.
See Section 4.2 on page 32 for a list of integer control variables.

Notes

ems_lmdl

This subroutine specifies the linear part of a model.

Specification

```
      call ems_lmdl(rtcod, dspace, type, nr, nc, nel, dobj, drlo, drup, dclo,
     &      dcup, mrow, mcol, dels)
```

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

type

is the format that the matrix is stored in, where:
If *type* = 1, the matrix is stored by indices.
If *type* = 2, the matrix is stored by columns.
If *type* = 3, the matrix is stored by rows.
Specified as: a fullword integer. Its value must be 1, 2 or 3.

nr

is the number of rows in the matrix
Specified as: a fullword integer.

nc

is the number of columns in the matrix
Specified as: a fullword integer.

nel

is the number of nonzero elements in the matrix
Specified as: a fullword integer.
$nel = mcol(nc{+}1){-}1$.

dobj

is the coefficients of the objective function.
Specified as: a one-dimensional real array of doublewords with *nc* entries.

drlo

is the lower bounds on the rows
Specified as: a one-dimensional real array of doublewords with *nr* entries.

drup

is the upper bounds on the rows
Specified as: a one-dimensional real array of doublewords with *nr* entries.

dclo

is the lower bounds on the columns
Specified as: a one-dimensional real array of doublewords with *nc* entries.

dcup

is the upper bounds on the columns
Specified as: a one-dimensional real array of doublewords with *nc* entries.

mrow

- If the matrix is stored by columns or indices:
  mrow is the row indices of the corresponding nonzero elements in *dels*.
  Specified as: a one-dimensional integer array of fullwords with *nel* entries.
- If the matrix is stored by rows:
  mrow is the start index in *dels* for each row.
  Specified as: a one-dimensional integer array of fullwords with nr+1 entries; $mrow(nr+1) = nel+1$.

mcol

- If the matrix is stored by columns:
  mcol is the start index in *dels* for each column.
  Specified as: a one-dimensional integer array of fullwords with nc+1 entries; $mcol(nc+1) = nel+1$.
- If the matrix is stored by indices or rows:
  mcol is the column indices of the corresponding nonzero elements in *dels*.
  Specified as: a one-dimensional integer array of fullwords with *nel* entries.

dels

is the values of the nonzero elements in the matrix.
Specified as: a one-dimensional real array of doublewords with *nel* entries.

On return

rtcod

is the return code for the subroutine, where:

If *rtcod* $= 0$, the subroutine completed successfully. Only informational messages were issued.

If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)

Returned as: a fullword integer.

dspace

is the user-provided work area.

Returned as: a one-dimensional real array of doublewords.

Notes

ems_mps

This subroutine reads a linear model that is in EMSOL format from a file.

Specification

`call ems_mps`(*rtcod*, *dspace*, *unit*, *type*, *intunit*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

unit

is the unit number specifying where the model file is read from.
Specified as: a fullword integer; $0 \leq unit \leq 99$.
Note: *unit* cannot be 1, 2 or 6.

type

is not currently used.
Specified as: a fullword integer. Its value must be 2.

intunit

is not currently used.
Specified as: a fullword integer; $0 \leq intunit \leq 99$.
Note: *intunit* cannot be 1, 2, 5 or 6.

On return

rtcod

is the return code for the subroutine, where:
If $rtcod = 0$, the subroutine completed successfully. Only informational messages were issued.
If $rtcod > 0$, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_mset

This subroutine allows message options to be set.

Specification

```
      call ems_mset(rtcod, dspace, strtnum, maxalw, maxprt, trace,
     &     usrexit, endnum, nonum)
```

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

strtnum

is the first message number to which *maxprt* applies
Specified as: a fullword integer. $0 \leq strtnum \leq 9999$.
If $strtnum = 0$ then no write statements will be executed by EMSOL.

maxalw

is not currently used.
Specified as: a fullword integer. Its value must be 0.

maxprt

controls whether messages should be printed or not.

If $maxprt < 0$, then messages will not be printed.
If $maxprt > 255$, then messages will be printed.
Specified as: a fullword integer; $maxprt < 0$ or $maxprt > 255$.

trace

is not currently used.
Specified as: a fullword integer. Its value must be 0.

usrexit

is not currently used.
Specified as: a fullword integer. Its value must be 0.

endnum

is the last message number to which *maxprt* applies
Specified as: a fullword integer. $1 \leq endnum \leq 9999$.

nonum

controls whether the message number should be printed or not.

If *nonum* $= 1$, the message number will not be printed.
If *nonum* $= 2$, the message number will be printed.
If *nonum* $\neq 1$ or 2, there is no change in the printing of message numbers.
Specified as: a fullword integer.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* $= 0$, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_nget

This subroutine provides pointers to various arrays used to store information within *dspace*.

Specification

`call ems_nget`(*rtcod*, *dspace*, *narray*, *num*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

num

is the number of pointers to be retrieved.
Specified as: a fullword integer; $0 \leq num \leq (length\ of\ narray)$.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

narray

is an array of pointers to arrays within *dspace*, where $narray(n)$ is the nth pointer.
Specified as: a one-dimensional integer array of fullwords.

See Chapter 5 on page 38 for a list of pointers.

Notes

ems_pkbs

This subroutine packs a basis into a user-supplied integer arrays. It is a companion to `ems_unpkbs` which unpacks a basis from this array.

Specification

```
 call ems_pkbs(rtcod, dspace, pk_bs_msk, usr_pk_bs_a_n_en, usr_pk_bs_a,
&      rq_pk_bs_a_n_en)
```

On entry

dspace
is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

pk_bs_msk
is not currently used.
Specified as: a fullword integer. Its value must be 1.

usr_pk_bs_a_n_en
is the number of entries in the user-supplied array *usr_pk_bs_a*.
Specified as: a fullword integer.

On return

rtcod
is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace
is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

usr_pk_bs_a
is the array which contains the packed basis.
Returned as: a one-dimensional integer array of fullwords with *usr_pk_bs_a_n_en* entries.

rq_bs_a_n_en

is the number of entries required to pack the basis in the user-supplied array *usr_pk_bs_a*.
Returned as: a fullword integer.

Notes

Depending on the proportion of rows and columns in the model with distinct finite bounds, the value of *rq_bs_a_n_en* will range between $1+(\mathit{Inumcols} + \mathit{Inumrows})/32$ and $1+(\mathit{Inumcols} + \mathit{Inumrows})/16$

ems_prts

This subroutine prints the status of the current model.

Specification

`call ems_prts(`*rtcod*, *dspace*`)`

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_rget

This subroutine requests current values of the real control variables.

Specification

`call ems_rget`(*rtcod*, *dspace*, *rarray*, *num*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

num

is the number of real control variables to be retrieved.
Specified as: a fullword real; $0 \leq num \leq (length\ of\ rarray)$.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* $= 0$, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

rarray

is an array of real control variables, where $rarray(n)$ is the nth real control variable.
Specified as: a one-dimensional real array of doublewords.

See Section 4.3 on page 36 for a list of real control variables.

Notes

ems_rgda

This subroutine examines an LP problem and determines the sensitivity of the solution to changes in objective function coefficients "costs" or (primal) activities of the variables. It finds the increase and decrease required in a particular cost or activity in order to force the basis to change. For non-basic variables, the bound corresponding to the activity of the variable is ignored, otherwise all bounds in the model are respected. For basic variables, the subroutine finds the extent to which the activity of that variable may be increased or decreased at least cost. The activity ranging information for non-basic variables corresponds to the traditional bound ranging information.

Specification

`call ems_rgda`(*rtcod*, *dspace*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

1. Before calling `ems_rgda`, you should have first found an optimal solution by calling ems_sslv.

2. `ems_rgda` sets the following control variables: *Nsobjupc*, *Nsobjdnc*, *Nsobjupv*, *Nsobjdnv*, *Nsobjupe*, *Nsobjdne*, *Nsobjupl*, *Nsobjdnl*, *Nsbndcupb*,

Nsbndcdnb, *Nsbndcupv*, *Nsbndcdnv*, *Nsbndcupe*, *Nsbndcdne*, *Nsbndcupl*, *Nsbndcdnl*, *Nsbndrupb*, *Nsbndrdnb*, *Nsbndrupv*, *Nsbndrdnv*, *Nsbndrupe*, *Nsbndrdne*, *Nsbndrupl* and *Nsbndrdnl*. Information about the results of ranging data can be obtained directly from these arrays or by calling `ems_prts`.

3. The integer control variable *Iprintsens* is set by `ems_rgda` so that any subsequent calls to `ems_prts` print the maximum amount of ranging data. You can prevent this information from being printed by setting *Iprintsens* after the call to `ems_rgda`, but before calling `ems_prts`.

4. As a special notation, if any of the entries in the arrays indexed by *Nsobjupe*, *Nsobjdne*, *Nsobjupl*, *Nsobjdnl*, *Nsbndcupe*, *Nsbndcdne*, *Nsbndcupl*, *Nsbndcdnl*, *Nsbndrupe*, *Nsbndrdne*, *Nsbndrupl* or *Nsbndrdnl* represent rows, the entries will be negated. For example, if row 5 would enter the basis if the cost on column 1 were increased then the entry *ispace*(*Nsobjupe*) would be -5.

ems_scal

This subroutine scales the coefficient matrix of the current model. Scaling frequently increases numerical stability of the solution process.

Specification

`call ems_scal`(*rtcod*, *dspace*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_sslv

This subroutine solves the current model.

Specification

`call ems_sslv(`*rtcod*, *dspace*, *alg*, *init*`)`

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

alg

is not currently used.
Specified as: a fullword integer. Its value must be 0, 1, or 2.

init

is the type of initialisation procedure to be used
If *init* = 0, 1 or 2, the initial basis is reset according to the primal and dual activities.
If *init* = 3, the initial basis is reset according to the status vector.
Specified as: a fullword integer. Its value must be 0, 1, 2 or 3.

On return

rtcod

is the return code for the subroutine, where:
If *rtcod* = 0, the subroutine completed successfully. Only informational messages were issued.
If *rtcod* > 0, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Notes

ems_unpkbs

This subroutine unpacks a basis from a user-supplied integer array. It is a companion to `ems_pkbs` which packs a basis into this array.

Specification

`call ems_unpkbs`(*rtcod*, *dspace*, *usr_pk_bs_a*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.is the array which contains the packed basis.

usr_pk_bs_a

Specified as: a one-dimensional integer array of fullwords.

On return

rtcod

is the return code for the subroutine, where:
If $rtcod = 0$, the subroutine completed successfully. Only informational messages were issued.
If $rtcod > 0$, *rtcod* is the return code associated with the highest severity message. (See Chapter 6 on page 45.)
Returned as: a fullword integer.

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

Chapter 3

User exit subroutines

This section describes the user exit subroutine.

ems_itru

This subroutine is called after every *Iiterufreq* iterations, or after resetting the current model.

Specification

`call ems_itru`(*dspace*, *ispace*, *reason*, *userrtcd*)

On entry

dspace

is the user-provided work area.
Specified as: a one-dimensional real array of doublewords.

ispace

is the integer version of the work area.
Specified as: a one-dimensional integer array of fullwords with twice the number of entries as *dspace*.
Note: *ispace*(1) and *dspace*(1) have the same address.

reason

is the situation where `ems_itru` is called.
If *reason* = 1, the subroutine is called after a primal iteration of `ems_sslv`.
If *reason* = 3, the subroutine is called after a resetting the current model during `ems_sslv`.
Specified as: a fullword integer.

usrrtcd

is the user return code from the user exit subroutine.
Specified as: a fullword integer.

On return

dspace

is the user-provided work area.
Returned as: a one-dimensional real array of doublewords.

ispace

is the integer version of the work area.
Returned as: a one-dimensional integer array of fullwords with twice the number of entries as *dspace*.
Note: *ispace*(1) and *dspace*(1) have the same address.

usrrtcd

is the user return code from the user exit subroutine, where:

If *usrrtcd* = 3, EMSOL acts as if the maximum number of iterations has been reached.
If *usrrtcd* = 99, then the current model is reset.
Returned as: a fullword integer.

Notes

Chapter 4

Control variables

This section contains a list of all of the control variables used by EMSOL grouped by type.
The subsections are ordered by index number. This is the index number used to reference the control variable in the array passed from or to the appropriate `ems_xget` or `ems_xset` subroutine. However, you may need to reference the control variables by name. Therefore, there is a table at the beginning of each of the sections of that lists the control variables alphabetically with their indices. You can find a control variable by name in the table, and then use its index to find the description in that section.

4.1 Bit masks

Some integer control variables are bit masks. A bit mask is a bit string that is used to specify one or more options by adding their values together. Consider integer control variable 32, *Iprtinfomask*. For `ems_prts` to print the names of variables and their status, the 4 and 8 bits of *Iprtinfomask* must be set. This is done adding these values together (4+8) and setting *Iprtinfomask* to be the sum, 12.

4.2 Integer control variables

The following is a table of the integer control variables used by EMSOL.

Name	Index	Name	Index
Idevexmode	17	*Inumcols*	28
Iiternum	4	*Inumdinf*	30
Iiterufreq	20	*Inumpinf*	29
Ilinelen	36	*Inumrows*	27
Ilogfreq	1	*Ipagelines*	26
Iloglevel	6	*Iprintunit*	2
Imaxcols	10	*Iprintsens*	49
Imaxfactor	3	*Iprobstat*	47
Imaxiter	5	*Iprtinfomask*	32
Imaxrows	9	*Isolmask*	33

Ilogfreq **Index:** 1 **Range:** 1, *maxint* **Default:** 999999
Settable: Yes
Description: The log frequency. If *Ilogfreq* $= n$ then log information is printed every n iterations. Log messages are also printed after refactorizations and switches in pricing strategy, unless messages have been turned off by an appropriate setting of *Iloglevel*.

Iprintunit **Index:** 2 **Range:** 0, 99 **Default:** 6
Settable: Yes
Description: The unit where output is directed for printing. *Iprintunit* cannot be 0, 2, or 5.

Imaxfactor **Index:** 3 **Range:** 0, 999 **Default:** 25
Settable: Yes
Description: The maximum number of iterations before a refactorization of the basis must be performed.

Iiternum **Index:** 4 **Range:** 0, *maxint* **Default:** 0
Settable: No
Description: The current number of simplex iterations that EMSOL has performed.

Imaxiter **Index:** 5 **Range:** 0, *maxint* **Default:** 999999
Settable: Yes
Description: The maximum number of simplex iterations that EMSOL will perform.

Iloglevel **Index:** 6 **Range:** 0, 31 **Default:** 16
Settable: Yes
Description: The simplex log detail bit mask. *Iloglevel* controls how much information is printed for each log message. *Iloglevel* is a bit mask. See Section 4.1 on page 31 for information on bit masks.

Iloglevel = 0 selects no information to be printed for the log message.
$1 \in$ *Iloglevel* prints a log line every *Ilogfreq* iterations and at final ems_sslv status.
$2 \in$ *Iloglevel* prints a log line at every refactorization and change in pricing method.
$4 \in$ *Iloglevel* prints pivoting information every *Ilogfreq* iterations.
$8 \in$ *Iloglevel* prints a log line at every change of Devex framework.
$16 \in$ *Iloglevel* reports minor computational errors.

Imaxrows **Index:** 9 **Range:** $-maxint$, *maxint* **Default:** 0
Settable: Yes
Description: The maximum number of rows allowed in the matrix. The value of *Imaxrows* can be set only once. Setting *Imaxrows* $= -n$ before a call to ems_mps or ems_lmdl yields n spare rows. Specifying spare rows in this way is necessary if rows are to be added to the model using ems_row.

Imaxcols **Index:** 10 **Range:** $-maxint$, *maxint* **Default:** 0
Settable: Yes
Description: The maximum number of columns allowed in the matrix. The value of *Imaxcols* can be set only once. Setting *Imaxcols* $= -n$ before a call to ems_mps or ems_lmdl yields n spare columns. Specifying spare columns in this way is necessary if columns are to be added to the model using ems_col.

Idevexmode **Index:** 17 **Range:** 0, 3 **Default:** 1
Settable: Yes
Description: The type of edge weight pricing strategy to be used by ems_sslv.

If *Idevexmode* = 0, then no edge weight pricing strategy is used.
If *Idevexmode* = 1 or 2, the Harris Devex strategy is used.
If *Idevexmode* = 3, the steepest edge strategy is used. This is the default.

Iiterufreq **Index:** 20 **Range:** 1, *maxint* **Default:** 1
Settable: Yes
Description: The frequency with which ems_itru is called.

Ipagelines **Index:** 26 **Range:** 10, *maxint* **Default:** 999999
Settable: Yes
Description: The number of lines on the output page.

Inumrows **Index:** 27 **Range:** 0, *Imaxrows* **Default:** 0
Settable: No
Description: The number of rows in the model.

Inumcols **Index:** 28 **Range:** 0, *Imaxcols* **Default:** 0
Settable: No
Description: The number of columns in the model.

Inumpinf **Index:** 29 **Range:** 0, *Inumrows* **Default:** n/a
Settable: No
Description: The current number of primal infeasibilities.

Inumdinf **Index:** 30 **Range:** 0, *Inumcols* **Default:** n/a
Settable: No
Description: The current number of dual infeasibilities.

Iprtinfomask **Index:** 32 **Range:** 0, 1023 **Default:** 3
Settable: Yes
Description: The model data bit mask for ems_prts. See Section 4.1 on page 31 for information on bit masks.

Iprtinfomask 0 then no information is selected for printing.
1∈*Iprtinfomask* selects the model dimension.
2∈*Iprtinfomask* selects the iteration count, current objective and model solution status.
4∈*Iprtinfomask* selects the names of variables.
8∈*Iprtinfomask* selects the status of variables.
16∈*Iprtinfomask* selects the primal activities of variables.
32∈*Iprtinfomask* selects the dual activities of variables.
64∈*Iprtinfomask* selects the lower bounds on variables.
128∈*Iprtinfomask* selects the upper bounds on variables.
256∈*Iprtinfomask* selects the costs of variables.
512∈*Iprtinfomask* selects the model matrix.

Isolmask **Index:** 33 **Range:** 0, 63 **Default:** 0
Settable: Yes
Description: The model status bit mask for ems_prts. See Section 4.1 on page 31 for information on bit masks.

Isolmask = 0 selects the whole of the model.
1∈*Isolmask* selects the rows.

2∈*Isolmask* selects the columns.
4∈*Isolmask* selects the rows with nonzero dual activity and columns with nonzero primal activity.
8∈*Isolmask* selects the variables which are infeasible.
16∈*Isolmask* selects the breakpoint variables.
32∈*Isolmask* selects the piecewise linear variables.

Ilinelen **Index:** 36 **Range:** 60, *maxint* **Default:** 80
Settable: Yes
Description: The maximum length of ouptput lines.

Iprobstat **Index:** 47 **Range:** -1, 6 **Default:** -1
Settable: No
Description: The model solution status.
If *Iprobstat* $= -1$, the solution status is unknown.
If *Iprobstat* $= 0$, the solution is optimal.
If *Iprobstat* $= 1$, the model is infeasible.
If *Iprobstat* $= 2$, the model is unbounded.
If *Iprobstat* $= 3$, ems_sslv has stopped on reaching the maximum number of iterations.
If *Iprobstat* $= 6$, ems_sslv has stopped because of lack of storage.

Iprintsens **Index:** 49 **Range:** 0, 1023 **Default:** 0
Settable: Yes
Description: The ranging information printing bit mask for ems_prts. See Section 4.1 on page 31 for information on bit masks. *Iprintsens* controls what information created by ems_rgda is printed by ems_prts.
If *Iprintsens* $= 0$, then no information is printed.
If *Iprintsens* ¿ 0, then all information is printed.

4.3 Real control variables

The following is a table of the real control variables used by EMSOL.

Name	Index	Name	Index
Rmaxmin	3	*Rsumdinf*	20
Robjvalue	18	*Rsumpinf*	19

Rmaxmin **Index:** 3 **Range:** -1.0, 1.0 **Default:** 1.0
Settable: Yes
Description: The weight of the linear objective.
If *Rmaxmin* = 1.0, then the objective function is minimized.
If *Rmaxmin* = 0.0, then the objective function is ignored and optimality is declared as soon as the current solution becomes feasible.
If *Rmaxmin* = -1.0, then the objective function is maximized.

Robjvalue **Index:** 18 **Range:** $-$*maxreal*, *maxreal* **Default:** n/a
Settable: No
Description: The value of the objective function.

Rsumpinf **Index:** 19 **Range:** $-$*maxreal*, *maxreal* **Default:** n/a
Settable: No
Description: The sum of the primal infeasibilities.

Rsumdinf **Index:** 20 **Range:** $-$*maxreal*, *maxreal* **Default:** n/a
Settable: No
Description: The sum of the dual infeasibilities.

4.4 Character control variables

The following is a table of the character control variables used by EMSOL.

Name	Index	Name	Index
Cbasis	6	*Cobjective*	2
Cbound	5	*Crange*	4
Cname	1	*Crhs*	3

Cname **Index:** 1 **Description:** The problem name in the EMSOL file

Cobjective **Index:** 2 **Description:** The objective function row name in the EMSOL file

Crhs **Index:** 3 **Description:** The RHS name in the EMSOL file

Crange **Index:** 4 **Description:** The range name in the EMSOL file

Cbound **Index:** 5 **Description:** The bound name in the EMSOL file

Cbasis **Index:** 6 **Description:** The basis name in the EMSOL file

Chapter 5

Pointers

The following is a table of the pointers into *dspace* or *ispace*.

Name	Index	Name	Index	Name	Index
Ncolaux	30	*Nrowscales*	14	*Nsbndrdnv*	50
Ncollower	6	*Nrowstat*	5	*Nsbndrupb*	47
Ncolnames	13	*Nrowupper*	3	*Nsbndrupe*	51
Ncolrcosts	9	*Nsbndcdnb*	40	*Nsbndrupl*	53
Ncolscales	15	*Nsbndcdne*	44	*Nsbndrupv*	49
Ncolsol	7	*Nsbndcdnl*	46	*Nsobjdnc*	32
Ncolstat	10	*Nsbndcdnv*	42	*Nsobjdne*	36
Ncolupper	8	*Nsbndcupb*	39	*Nsobjdnl*	38
Nobjective	11	*Nsbndcupe*	43	*Nsobjdnv*	34
Nrowacts	2	*Nsbndcupl*	45	*Nsobjupc*	31
Nrowaux	29	*Nsbndcupv*	41	*Nsobjupe*	35
Nrowduals	4	*Nsbndrdnb*	48	*Nsobjupl*	37
Nrowlower	1	*Nsbndrdne*	52	*Nsobjupv*	33
Nrownames	12	*Nsbndrdnl*	54		

Nrowlower **Index:** 1 **Description:** The index into *dspace* for the first element of row lower bounds.

Nrowacts **Index:** 2 **Description:** The index into *dspace* for the first element of row primal activities.

Nrowupper **Index:** 3 **Description:** The index into *dspace* for the first element of row upper bounds.

Nrowduals **Index:** 4 **Description:** The index into *dspace* for the first element of row dual activities.

Nrowstat **Index:** 5 **Description:** The index into *ispace* for the first element of the row status vector. *Nrowstat* and *Ncolstat* give the location of the EMSOL status vectors. These status vectors are one-dimensional integer arrays of fullwords. The numbering of the bits in these vectors follows the bit numbering conventions for VS FORTRAN. All 32 bits are used internally by EMSOL, but only three of these bits are particularly important to the user:

> If the 31 bit is set, the variable is basic (row logical for the row status vector and structural variable for the column status vector.) If it is set, then the remaining bits are ignored.
>
> If the 30 bit is set, then the row activity or column activity may go down.
>
> If the 29 bit is set, then the row activity or column activity may go up.

EMSOL will override any of these if they are invalid (for example, if the up bit is set, but the column (variable) is at or above its upper bound).
In practice this translates into:

> For basic variables
>
> > The variable will have the basic bit set.
>
> For nonbasic variables
>
> > A variable at its upper bound will have the down bit set.
> >
> > A variable at its lower bound will have the up bit set.
> >
> > A fixed variable (and at bound) will have no bits set.
> >
> > A free variable at zero will have the up and down bits set.
> >
> > Any variable between bounds will have both the up and down bits set.
> >
> > A variable above its upper bound will have the down bit set.
> >
> > A variable below its lower bound will have the up bit set.

Ncollower **Index:** 6 **Description:** The index into *dspace* for the first element of column lower bounds.

Ncolsol **Index:** 7 **Description:** The index into *dspace* for the first element of the column primal activities.

Ncolupper **Index:** 8 **Description:** The index into *dspace* for the first element of column upper bounds.

Ncolrcosts **Index:** 9 **Description:** The index into *dspace* for the first element of column dual activities.

Ncolstat **Index:** 10 **Description:** The index into *ispace* for the first element of the column status vector. See the description of *Nrowstat* for more information about *Ncolstat*.

Nobjective **Index:** 11 **Description:** The index into *dspace* for the first element of column costs.

Nrownames **Index:** 12 **Description:** The index into *dspace* for the first element of row names.

Ncolnames **Index:** 13 **Description:** The index into *dspace* for the first element of column names.

Nrowscales **Index:** 14 **Description:** The index into *dspace* for the first element of row scale factors. The scaling region contains scale factors that are applied to each matrix element as follows:

$$scaled\ a_{ij} = \frac{\mathrm{o}riginal\ a_{ij} \times rowscale_i}{colscale_j}$$

Ncolscales **Index:** 15 **Description:** The index into *dspace* for the first element of column scale factors. See the description of *Nrowscales* for more information about *Ncolscales*.

Nrowaux **Index:** 29 **Description:** The index into *dspace* for the first element of row auxiliary solve information. *Nrowaux* and *Ncolaux* give the location of EMSOL's auxiliary solve information. These regions are only set up if your model is infeasible or unbounded.

If the model is infeasible, then *Nrowaux* and *Ncolaux* are the Phase 1 reduced costs. The Phase 1 reduced costs are also referenced by *Nrowduals* and *Ncolrcosts*. These costs are set as if an objective coefficient of +1.0 has been given to each variable above its upper bound, and an objective coefficient of −1.0 has been given to each variable below its lower bound.

If the model is unbounded, then the values pointed to by *Nrowaux* and *Ncolaux* give a direction of unboundedness. (For example, for a minimization problem, a direction of unboundedness would be a vector you could travel along and continually decrease the objective function value.)

Ncolaux **Index:** 30 **Description:** The index into *dspace* for the first element of column auxiliary solve information. See the description of *Nrowaux*for more information about *Ncolaux*.

Nsobjupc **Index:** 31 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of upper cost limits. For each individual variable, this is the largest cost coefficient which maintains the current basis. An increase beyond this cost causes a basis change.

Nsobjdnc **Index:** 32 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of lower cost limits. For each individual variable, this is the smallest cost coefficient which maintains the current basis. A decrease below this cost causes a basis change.

Nsobjupv **Index:** 33 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the upper limits on cost coefficients indexed by *Nsobjupc*.

Nsobjdnv **Index:** 34 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the lower limits on cost coefficients indexed by *Nsobjdnc*.

Nsobjupe **Index:** 35 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the increased cost coefficients indexed by *Nsobjupc*. If the cost of an individual variable increases beyond the corresponding entry in the array indexed by *Nsobjupc*, the row or column indexed by *Nsobjupe* enters the basis.

Nsobjdne **Index:** 36 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the decreased cost coefficients indexed by *Nsobjdnc*. If the cost of an individual variable decreases below the corresponding entry in the array indexed by *Nsobjdnc*, the row or column indexed by *Nsobjdne* enters the basis.

Nsobjupl **Index:** 37 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the increased cost coefficients indexed by *Nsobjupc*. If the cost of an individual variable increases beyond the corresponding entry in the array indexed by *Nsobjupc*, the row or column indexed by *Nsobjupl* leaves the basis.

Nsobjdnl **Index:** 38 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the decreased cost coefficients indexed by *Nsobjdnc*. If the cost of an individual variable decreases below the corresponding entry in the array indexed by *Nsobjdnc*, the row or column indexed by *Nsobjdnl* leaves the basis.

Nsbndcupb **Index:** 39 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of upper column activity limits. For each individual variable, this is the largest activity which maintains the current basis. An increase beyond this activity causes a basis change.

Nsbndcdnb **Index:** 40 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of lower column activity limits. For each individual variable, this is the smallest column activity which maintains the current basis. A decrease below this column activity causes a basis change.

Nsbndcupv **Index:** 41 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the upper limits on column activities indexed by *Nsbndcupb*.

Nsbndcdnv **Index:** 42 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the lower limits on column activities indexed by *Nsbndcdnb*.

Nsbndcupe **Index:** 43 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the increased column activities indexed by *Nsbndcupb*. If the column activity of an individual variable increases beyond the corresponding entry in the array indexed by *Nsbndcupb*, the row or column indexed by *Nsbndcupe* enters the basis.

Nsbndcdne **Index:** 44 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the decreased column activities indexed by *Nsbndcdnb*. If the column activity of an individual variable decreases below the corresponding entry in the array indexed by *Nsbndcdnb*, the row or column indexed by *Nsbndcdne* enters the basis.

Nsbndcupl **Index:** 45 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the increased column activities indexed by *Nsbndcupb*. If the column activity of an individual variable increases beyond the corresponding entry in the array indexed by *Nsbndcupb*, the row or column indexed by *Nsbndcupl* leaves the basis.

Nsbndcdnl **Index:** 46 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the decreased column activities indexed by *Nsbndcdnb*. If the column activity of an individual variable decreases below the corresponding entry in the array indexed by *Nsbndcdnb*, the row or column indexed by *Nsbndcdnl* leaves the basis.

Nsbndrupb **Index:** 47 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of upper row activity limits. For each individual variable, this is the largest activity which maintains the current basis. An increase beyond this activity causes a basis change.

Nsbndrdnb **Index:** 48 **Description:** The index into *dspace* created by ems_rgda for the first element of the array of lower row activity limits. For each individual variable, this is the smallest row activity which maintains the current basis. A decrease below this row activity causes a basis change.

Nsbndrupv **Index:** 49 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the upper limits on row activities indexed by *Nsbndrupb*.

Nsbndrdnv **Index:** 50 **Description:** The index into *dspace* created by ems_rgda for the first element of ranges of the objective function values corresponding to the lower limits on row activities indexed by *Nsbndrdnb*.

Nsbndrupe **Index:** 51 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the increased row activities indexed by *Nsbndrupb*. If the row activity of an individual variable increases beyond the corresponding entry in the array indexed by *Nsbndrupb*, the row or column indexed by *Nsbndrupe* enters the basis.

Nsbndrdne **Index:** 52 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of entering rows or columns corresponding to the decreased row activities indexed by *Nsbndrdnb*. If the row activity of an individual variable decreases below the corresponding entry in the array indexed by *Nsbndrdnb*, the row or column indexed by *Nsbndrdne* enters the basis.

Nsbndrupl **Index:** 53 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the increased row activities indexed by *Nsbndrupb*. If the row activity of an individual variable increases beyond the corresponding entry in the array indexed by *Nsbndrupb*, the row or column indexed by *Nsbndrupl* leaves the basis.

Nsbndrdnl **Index:** 54 **Description:** The index into *ispace* created by ems_rgda for the first element of the array of leaving rows or columns corresponding to the decreased row activities indexed by *Nsbndrdnb*. If the row activity of an individual variable decreases below the corresponding entry in the array indexed by *Nsbndrdnb*, the row or column indexed by *Nsbndrdnl* leaves the basis.

Nsobjupact **Index:** 80 **Description:** The index into *dspace* created by ems_rgda for the first element of primal activity values corresponding to the upper limits on cost coefficients indexed by *Nsobjupc* for the basis change indicated by *Nsbndcupe* and *Nsbndcupl*.

Nsobjdnact **Index:** 81 **Description:** The index into *dspace* created by ems_rgda for the first element of primal activity values corresponding to the dower limits on cost coefficients indexed by *Nsobjdnc* for the basis change indicated by *Nsbndcdne* and *Nsbndcdnl*.

Chapter 6

Return codes

Each message issued by an EMSOL subroutine has an associated return code. The value of *rtcod* which is returned by an EMSOL subroutines is the return code associated with the message of the highest severity that was issued by the subroutine. indicates the the level of success with which it was completed. Return code values have the following meaning.
If $0 \leq rtcod < 100$ then only informational messages were issued.
If $100 \leq rtcod < 200$ then only warning messages were issued.
If $200 \leq rtcod < 300$ then only error messages were issued.
If $300 \leq rtcod$ then serious error messages were issued.

Index