C interfaces to GALAHAD BSC

Jari Fowkes and Nick Gould
STFC Rutherford Appleton Laboratory

Sun Apr 16 2023

Chapter 1

GALAHAD C package bsc

1.1 Introduction

1.1.1 Purpose

Given matrices A and (diagonal) D, **build the "Schur complement"** $S = ADA^T$ in sparse co-ordinate (and optionally sparse column) format(s). Full advantage is taken of any zero coefficients in the matrix A.

Currently, only the control and inform parameters are exposed; these are provided and used by other GALAHAD packages with C interfaces.

1.1.2 Authors

N. I. M. Gould, STFC-Rutherford Appleton Laboratory, England.

C interface, additionally J. Fowkes, STFC-Rutherford Appleton Laboratory.

Julia interface, additionally A. Montoison and D. Orban, Polytechnique Montréal.

1.1.3 Originally released

October 2013, C interface January 2022.

1.1.4 Call order

To solve a given problem, functions from the bsc package must be called in the following order:

- bsc_initialize - provide default control parameters and set up initial data structures
- bsc_read_specfile (optional) - override control values by reading replacement values from a file
- bsc_import - set up matrix data structures for A.
- bsc_reset_control (optional) - possibly change control parameters if a sequence of problems are being solved
- bsc_form - form the Schur complement S
- bsc_information (optional) - recover information about the process
- bsc_terminate - deallocate data structures

1.2 Further topics

1.2.1 Unsymmetric matrix storage formats

An unsymmetric m by n matrix A may be presented and stored in a variety of convenient input formats.

Both C-style (0 based) and fortran-style (1-based) indexing is allowed. Choose `control.f_indexing` as `false` for C style and `true` for fortran style; the discussion below presumes C style, but add 1 to indices for the corresponding fortran version.

Wrappers will automatically convert between 0-based (C) and 1-based (fortran) array indexing, so may be used transparently from C. This conversion involves both time and memory overheads that may be avoided by supplying data that is already stored using 1-based indexing.

1.2.1.1 Dense storage format

The matrix A is stored as a compact dense matrix by rows, that is, the values of the entries of each row in turn are stored in order within an appropriate real one-dimensional array. In this case, component $n * i + j$ of the storage array A_val will hold the value A_{ij} for $0 \leq i \leq m-1$, $0 \leq j \leq n-1$.

1.2.1.2 Dense storage format

The matrix A is stored as a compact dense matrix by columns, that is, the values of the entries of each column in turn are stored in order within an appropriate real one-dimensional array. In this case, component $m * j + i$ of the storage array A_val will hold the value A_{ij} for $0 \leq i \leq m-1$, $0 \leq j \leq n-1$.

1.2.1.3 Sparse co-ordinate storage format

Only the nonzero entries of the matrices are stored. For the l-th entry, $0 \leq l \leq ne-1$, of A, its row index i, column index j and value A_{ij}, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$, are stored as the l-th components of the integer arrays A_row and A_col and real array A_val, respectively, while the number of nonzeros is recorded as A_ne = ne.

1.2.1.4 Sparse row-wise storage format

Again only the nonzero entries are stored, but this time they are ordered so that those in row i appear directly before those in row i+1. For the i-th row of A the i-th component of the integer array A_ptr holds the position of the first entry in this row, while A_ptr(m) holds the total number of entries. The column indices j, $0 \leq j \leq n-1$, and values A_{ij} of the nonzero entries in the i-th row are stored in components l = A_ptr(i), . . ., A_ptr(i+1)-1, $0 \leq i \leq m-1$, of the integer array A_col, and real array A_val, respectively. For sparse matrices, this scheme almost always requires less storage than its predecessor.

1.2.1.5 Sparse column-wise storage format

Once again only the nonzero entries are stored, but this time they are ordered so that those in column j appear directly before those in column j+1. For the j-th column of A the j-th component of the integer array A_ptr holds the position of the first entry in this column, while A_ptr(n) holds the total number of entries. The row indices i, $0 \leq i \leq m-1$, and values A_{ij} of the nonzero entries in the j-th columnsare stored in components l = A_ptr(j), . . ., A_ptr(j+1)-1, $0 \leq j \leq n-1$, of the integer array A_row, and real array A_val, respectively. As before, for sparse matrices, this scheme almost always requires less storage than the co-ordinate format.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 galahad_bsc.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "galahad_precision.h"
#include "galahad_cfunctions.h"
```

Data Structures

- struct bsc_control_type
- struct bsc_inform_type

3.1.1 Data Structure Documentation

3.1.1.1 struct bsc_control_type

control derived type as a C struct

Data Fields

bool	f_indexing	use C or Fortran sparse matrix indexing
int	error	error and warning diagnostics occur on stream error
int	out	general output occurs on stream out
int	print_level	the level of output required is specified by print_level
int	max_col	maximum permitted number of nonzeros in a column of A; -ve means unlimit
int	new_a	how much has A changed since it was last accessed: • 0 = not changed, • 1 = values changed, • 2 = structure changed • 3 = structure changed but values not required

Data Fields

int	extra_space_s	how much extra space is to be allocated in S above that needed to hold the Schur complement
bool	s_also_by_column	should s.ptr also be set to indicate the first entry in each column of S
bool	space_critical	if .space_critical true, every effort will be made to use as little space as possible. This may result in longer computation time
bool	deallocate_error_fatal	if .deallocate_error_fatal is true, any array/pointer deallocation error will terminate execution. Otherwise, computation will continue
char	prefix[31]	all output lines will be prefixed by .prefix(2:LEN(TRIM(.prefix))-1) where .prefix contains the required string enclosed in quotes, e.g. "string" or 'string'

3.1.1.2 struct bsc_inform_type

inform derived type as a C struct

Data Fields

int	status	return status. See SBLS_form_and_factorize for details
int	alloc_status	the status of the last attempted allocation/deallocation
char	bad_alloc[81]	the name of the array for which an allocation/deallocation error ocurred
int	max_col_a	the maximum number of entries in a column of A
int	exceeds_max_col	the number of columns of A that have more than control.max_col entries
real_wp_	time	the total CPU time spent in the package
real_wp_	clock_time	the total clock time spent in the package