



## C interfaces to GALAHAD FDC

Jari Fowkes and Nick Gould  
STFC Rutherford Appleton Laboratory  
Fri Mar 18 2022



<b>1 GALAHAD C package fdc</b>	<b>1</b>
1.1 Introduction	1
1.1.1 Purpose	1
1.1.2 Authors	1
1.1.3 Originally released	1
1.1.4 Method	1
1.1.5 Call order	2
1.1.6 Array indexing	2
<b>2 File Index</b>	<b>3</b>
2.1 File List	3
<b>3 File Documentation</b>	<b>5</b>
3.1 fdc.h File Reference	5
3.1.1 Data Structure Documentation	5
3.1.1.1 struct fdc_control_type	5
3.1.1.2 struct fdc_time_type	6
3.1.1.3 struct fdc_inform_type	6
3.1.2 Function Documentation	7
3.1.2.1 fdc_initialize()	7
3.1.2.2 fdc_read_specfile()	7
3.1.2.3 fdc_find_dependent_rows()	8
3.1.2.4 fdc_terminate()	9
<b>4 Example Documentation</b>	<b>13</b>
4.1 fdct.c	13
4.2 fdctf.c	14
<b>Index</b>	<b>15</b>



# Chapter 1

## GALAHAD C package fdc

### 1.1 Introduction

#### 1.1.1 Purpose

Given an under-determined set of linear equations/constraints  $a_i^T x = b_i, i = 1, \dots, m$  involving  $n \geq m$  unknowns  $x$ , this package **determines whether the constraints are consistent, and if so how many of the constraints are dependent**; a list of dependent constraints, that is, those which may be removed without changing the solution set, will be found and the remaining  $a_i$  will be linearly independent. Full advantage is taken of any zero coefficients in the vectors  $a_i$ .

#### 1.1.2 Authors

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

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

#### 1.1.3 Originally released

August 2006, C interface January 2021

#### 1.1.4 Method

A choice of two methods is available. In the first, the matrix

$$K = \begin{pmatrix} \alpha I & A^T \\ A & 0 \end{pmatrix}$$

is formed and factorized for some small  $\alpha > 0$  using the GALAHAD package SLS—the factors  $K = PLDL^T P^T$  are used to determine whether  $A$  has dependent rows. In particular, in exact arithmetic dependencies in  $A$  will correspond to zero pivots in the block diagonal matrix  $D$ .

The second choice of method finds factors  $A = PLUQ$  of the rectangular matrix  $A$  using the GALAHAD package ULS. In this case, dependencies in  $A$  will be reflected in zero diagonal entries in  $U$  in exact arithmetic.

The factorization in either case may also be used to determine whether the system is consistent.

### 1.1.5 Call order

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

- `fdc_initialize` - provide default control parameters and set up initial data structures
- `fdc_read_specfile` (optional) - override control values by reading replacement values from a file
- `fdc_find_dependent_rows` - find the number of dependent rows and, if there are any, whether the constraints are independent
- `fdc_terminate` - deallocate data structures

See Section 4.1 for examples of use.

### 1.1.6 Array indexing

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; add 1 to input integer arrays if fortran-style indexing is used, and beware that return integer arrays will adhere to this.

## Chapter 2

# File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

<a href="#">fdc.h</a> . . . . .	5
---------------------------------	---





## Chapter 3

# File Documentation

### 3.1 `fdc.h` File Reference

```
#include <stdbool.h>
#include "galahad_precision.h"
#include "sls.h"
#include "uls.h"
```

#### Data Structures

- struct [fdc\\_control\\_type](#)
- struct [fdc\\_time\\_type](#)
- struct [fdc\\_inform\\_type](#)

#### Functions

- void [fdc\\_initialize](#) (void \*\*data, struct [fdc\\_control\\_type](#) \*control, int \*status)
- void [fdc\\_read\\_specfile](#) (struct [fdc\\_control\\_type](#) \*control, const char specfile[])
- void [fdc\\_find\\_dependent\\_rows](#) (struct [fdc\\_control\\_type](#) \*control, void \*\*data, struct [fdc\\_inform\\_type](#) \*inform, int \*status, int m, int n, int A\_ne, const int A\_col[], const int A\_ptr[], const real\_wp\_ A\_val[], const real\_wp\_ b[], int \*n\_depen, int depen[])
- void [fdc\\_terminate](#) (void \*\*data, struct [fdc\\_control\\_type](#) \*control, struct [fdc\\_inform\\_type](#) \*inform)

#### 3.1.1 Data Structure Documentation

##### 3.1.1.1 struct `fdc_control_type`

control derived type as a C struct

#### Examples

[fdct.c](#), and [fdctf.c](#).

## Data Fields

bool	f_indexing	use C or Fortran sparse matrix indexing
int	error	unit for error messages
int	out	unit for monitor output
int	print_level	controls level of diagnostic output
int	indmin	initial estimate of integer workspace for sls (obsolete)
int	valmin	initial estimate of real workspace for sls (obsolete)
real_wp_	pivot_tol	the relative pivot tolerance (obsolete)
real_wp_	zero_pivot	the absolute pivot tolerance used (obsolete)
real_wp_	max_infeas	the largest permitted residual
bool	use_sls	choose whether SLS or ULS is used to determine dependencies
bool	scale	should the rows of A be scaled to have unit infinity norm or should no scaling be applied
bool	space_critical	if space is critical, ensure allocated arrays are no bigger than needed
bool	deallocate_error_fatal	exit if any deallocation fails
char	symmetric_linear_solver[31]	symmetric (indefinite) linear equation solver
char	unsymmetric_linear_solver[31]	unsymmetric linear equation solver
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'
struct sls_control_type	sls_control	control parameters for SLS
struct uls_control_type	uls_control	control parameters for ULS

## 3.1.1.2 struct fdc\_time\_type

time derived type as a C struct

## Data Fields

real_wp_	total	the total CPU time spent in the package
real_wp_	analyse	the CPU time spent analysing the required matrices prior to factorizatio
real_wp_	factorize	the CPU time spent factorizing the required matrices
real_wp_	clock_total	the total clock time spent in the package
real_wp_	clock_analyse	the clock time spent analysing the required matrices prior to factorizat
real_wp_	clock_factorize	the clock time spent factorizing the required matrices

## 3.1.1.3 struct fdc\_inform\_type

inform derived type as a C struct

## Examples

[fdct.c](#), and [fdctf.c](#).

## Data Fields

int	status	return status. See <code>FDC_find_dependent</code> 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 occurred
int	factorization_status	the return status from the factorization
int	factorization_integer	the total integer workspace required for the factorization
int	factorization_real	the total real workspace required for the factorization
real_wp_	non_negligible_pivot	the smallest pivot which was not judged to be zero when detecting linear dependent constraints
struct <a href="#">fdc_time_type</a>	time	timings (see above)
struct <code>sls_inform_type</code>	sls_inform	SLS inform type.
struct <code>uls_inform_type</code>	uls_inform	ULS inform type.

## 3.1.2 Function Documentation

3.1.2.1 `fdc_initialize()`

```
void fdc_initialize (
    void ** data,
    struct fdc\_control\_type * control,
    int * status )
```

Set default control values and initialize private data

## Parameters

in, out	<i>data</i>	holds private internal data
out	<i>control</i>	is a struct containing control information (see <a href="#">fdc_control_type</a> )
out	<i>status</i>	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): <ul style="list-style-type: none"> <li>• 0. The import was succesful.</li> </ul>

## Examples

[fdct.c](#), and [fdctf.c](#).

3.1.2.2 `fdc_read_specfile()`

```
void fdc_read_specfile (
    struct fdc\_control\_type * control,
    const char specfile[] )
```

Read the content of a specification file, and assign values associated with given keywords to the corresponding control parameters

#### Parameters

in, out	<i>control</i>	is a struct containing control information (see <a href="#">fdc_control_type</a> )
in	<i>specfile</i>	is a character string containing the name of the specification file

### 3.1.2.3 fdc\_find\_dependent\_rows()

```
void fdc_find_dependent_rows (
    struct fdc_control_type * control,
    void ** data,
    struct fdc_inform_type * inform,
    int * status,
    int m,
    int n,
    int A_ne,
    const int A_col[],
    const int A_ptr[],
    const real_wp_ A_val[],
    const real_wp_ b[],
    int * n_depen,
    int depen[] )
```

Find dependent rows and, if any, check if  $Ax = b$  is consistent

#### Parameters

in	<i>control</i>	is a struct containing control information (see <a href="#">fdc_control_type</a> )
in, out	<i>data</i>	holds private internal data
out	<i>inform</i>	is a struct containing output information (see <a href="#">fdc_inform_type</a> )

## Parameters

in, out	<i>status</i>	<p>is a scalar variable of type int, that gives the entry and exit status from the package. Possible exit are:</p> <ul style="list-style-type: none"> <li>• 0. The run was succesful.</li> <li>• -1. An allocation error occurred. A message indicating the offending array is written on unit control.error, and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively.</li> <li>• -2. A deallocation error occurred. A message indicating the offending array is written on unit control.error and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively.</li> <li>• -3. The restrictions <math>n &gt; 0</math> and <math>m &gt; 0</math> or requirement that a type contains its relevant string 'dense', 'coordinate', 'sparse_by_rows', 'diagonal', 'scaled_identity', 'identity', 'zero' or 'none' has been violated.</li> <li>• -5. The constraints appear to be inconsistent.</li> <li>• -9. The analysis phase of the factorization failed; the return status from the factorization package is given in the component inform.factor_status</li> <li>• -10. The factorization failed; the return status from the factorization package is given in the component inform.factor_status.</li> </ul>
in	<i>m</i>	is a scalar variable of type int, that holds the number of rows of $A$ .
in	<i>n</i>	is a scalar variable of type int, that holds the number of columns of $A$ .
in	<i>A_ne</i>	is a scalar variable of type int, that holds the number of nonzero entries in $A$ .
in	<i>A_col</i>	is a one-dimensional array of size <i>A_ne</i> and type int, that holds the column indices of $A$ in a row-wise storage scheme. The nonzeros must be ordered so that those in row $i$ appear directly before those in row $i+1$ , the order within each row is unimportant.
in	<i>A_ptr</i>	is a one-dimensional array of size $n+1$ and type int, that holds the starting position of each row of $A$ , as well as the total number of entries plus one.
in	<i>A_val</i>	is a one-dimensional array of size <i>a_ne</i> and type double, that holds the values of the entries of the $A$ ordered as in <i>A_col</i> and <i>A_ptr</i> .
in	<i>b</i>	is a one-dimensional array of size $m$ and type double, that holds the linear term $b$ in the constraints. The $i$ -th component of $b$ , $i = 0, \dots, m-1$ , contains $b_i$ .
out	<i>n_depen</i>	is a scalar variable of type int, that holds the number of dependent constraints, if any.
out	<i>depen</i>	is a one-dimensional array of size $m$ and type int, whose first <i>n_depen</i> components contain the indices of dependent constraints.

## Examples

[fdct.c](#), and [fdctf.c](#).

## 3.1.2.4 fdct\_terminate()

```
void fdct_terminate (
    void ** data,
```

```
struct fdc_control_type * control,  
struct fdc_inform_type * inform )
```

Deallocate all internal private storage

## Parameters

<code>in, out</code>	<i>data</i>	holds private internal data
<code>out</code>	<i>control</i>	is a struct containing control information (see <a href="#">fdct_control_type</a> )
<code>out</code>	<i>inform</i>	is a struct containing output information (see <a href="#">fdct_inform_type</a> )

## Examples

[fdct.c](#), and [fdctf.c](#).





## Chapter 4

# Example Documentation

### 4.1 fdct.c

This is an example of how to use the package to solve a quadratic program. A variety of supported Hessian and constraint matrix storage formats are shown.

Notice that C-style indexing is used, and that this is flagged by setting `control.f_indexing` to `false`.

```
/* fdct.c */
/* Full test for the FDC C interface using C sparse matrix indexing */
#include <stdio.h>
#include <math.h>
#include "fdc.h"

int main(void) {
    // Derived types
    void *data;
    struct fdc_control_type control;
    struct fdc_inform_type inform;
    // Set problem data
    int m = 3; // number of rows
    int n = 4; // number of columns
    int A_ne = 10; // number of nonzeros
    int A_col[] = {0, 1, 2, 3, 0, 1, 2, 3, 1, 3}; // column indices
    int A_ptr[] = {0, 4, 8, 10}; // row pointers
    double A_val[] = {1.0, 2.0, 3.0, 4.0, 2.0, -4.0, 6.0, -8.0, 5.0, 10.0};
    double b[] = {5.0, 10.0, 0.0};
    // Set output storage
    int depen[m]; // dependencies, if any
    int n_depen;
    int status;
    printf(" C sparse matrix indexing\n");
    // Initialize FDC
    fdc_initialize( &data, &control, &status );
    // Set user-defined control options
    control.f_indexing = false; // C sparse matrix indexing
    // Start from 0
    fdc_find_dependent_rows( &control, &data, &inform, &status, m, n, A_ne,
                           A_col, A_ptr, A_val, b, &n_depen, depen );
    if(status == 0){
        if(n_depen == 0){
            printf("FDC_find_dependent - no dependent rows, status = %li\n",
                  status);
        }else{
            printf("FDC_find_dependent - dependent rows(s):" );
            for( int i = 0; i < n_depen; i++) printf(" %i", depen[i]);
            printf(", status = %i\n", status);
        }
    }else{
        printf("FDC_find_dependent - exit status = %li\n", status);
    }
    // Delete internal workspace
    fdc_terminate( &data, &control, &inform );
}
```

## 4.2 fdctf.c

This is the same example, but now fortran-style indexing is used.

```

/* fdctf.c */
/* Full test for the FDC C interface using Fortran sparse matrix indexing */
#include <stdio.h>
#include <math.h>
#include "fdc.h"
int main(void) {
    // Derived types
    void *data;
    struct fdc_control_type control;
    struct fdc_inform_type inform;
    // Set problem data
    int m = 3; // number of rows
    int n = 4; // number of columns
    int A_ne = 10; // number of nonzeros
    int A_col[] = {1, 2, 3, 4, 1, 2, 3, 4, 2, 4}; // column indices
    int A_ptr[] = {1, 5, 9, 11}; // row pointers
    double A_val[] = {1.0, 2.0, 3.0, 4.0, 2.0, -4.0, 6.0, -8.0, 5.0, 10.0};
    double b[] = {5.0, 10.0, 0.0};
    // Set output storage
    int depen[m]; // dependencies, if any
    int n_depen;
    int status;
    printf(" Fortran sparse matrix indexing\n");
    // Initialize FDC
    fdc_initialize( &data, &control, &status );
    // Set user-defined control options
    control.f_indexing = true; // Fortran sparse matrix indexing
    // Start from 0
    fdc_find_dependent_rows( &control, &data, &inform, &status, m, n, A_ne,
                           A_col, A_ptr, A_val, b, &n_depen, depen );
    if(status == 0){
        if(n_depen == 0){
            printf("FDC_find_dependent - no dependent rows, status = %i\n",
                  status);
        }else{
            printf("FDC_find_dependent - dependent rows(s):" );
            for( int i = 0; i < n_depen; i++) printf(" %i", depen[i]);
            printf(", status = %i\n", status);
        }
    }else{
        printf("FDC_find_dependent - exit status = %i\n", status);
    }
    // Delete internal workspace
    fdc_terminate( &data, &control, &inform );
}

```

# Index

- fdc.h, [5](#)
  - fdc\_find\_dependent\_rows, [8](#)
  - fdc\_initialize, [7](#)
  - fdc\_read\_specfile, [7](#)
  - fdc\_terminate, [9](#)
- fdc\_control\_type, [5](#)
- fdc\_find\_dependent\_rows
  - fdc.h, [8](#)
- fdc\_inform\_type, [6](#)
- fdc\_initialize
  - fdc.h, [7](#)
- fdc\_read\_specfile
  - fdc.h, [7](#)
- fdc\_terminate
  - fdc.h, [9](#)
- fdc\_time\_type, [6](#)