C interfaces to GALAHAD LMS

Jari Fowkes and Nick Gould
STFC Rutherford Appleton Laboratory
Tue May 2 2023

Chapter 1

GALAHAD C package lms

1.1 Introduction

1.1.1 Purpose

Given a sequence of vectors $\{s_k\}$ and $\{y_k\}$ and scale factors $\{\delta_k\}$, **obtain the product of a limited-memory secant approximation** H_k **(or its inverse) with a given vector**, using one of a variety of well-established formulae.

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

July 2014, C interface January 2022.

1.1.4 Method

Given a sequence of vectors $\{s_k\}$ and $\{y_k\}$ and scale factors $\{\delta_k\}$, a limited-memory secant approximation H_k is chosen so that $H_{\max(k-m,0)} = \delta_k I$, $H_{k-j} s_{k-j} = y_{k-j}$ and $\|H_{k-j+1} - H_{k-j}\|$ is `‘small'' for $j = \min(k-1, m-1), \ldots, 0$. Different ways of quantifying `‘small'' distinguish different methods, but the crucial observation is that it is possible to construct H_k quickly from s_k, y_k and δ_k, and to apply it and its inverse to a given vector v. It is also possible to apply similar formulae to the `‘shifted'' matrix $H_k + \lambda_k I$ that occurs in trust-region methods.

1.1.5 Reference

The basic methods are those given by

R. H. Byrd, J. Nocedal and R. B. Schnabel (1994) Representations of quasi-Newton matrices and their use in limited memory methods. Mathematical Programming, **63(2)** 129-156,

with obvious extensions.

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_lms.h File Reference

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

Data Structures

- struct lms_control_type
- struct lms_time_type
- struct lms_inform_type

3.1.1 Data Structure Documentation

3.1.1.1 struct lms_control_type

control derived type as a C struct

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	memory_length	limited memory length
int	method	limited-memory formula required (others may be added in due course): • 1 BFGS (default). • 2 Symmetric Rank-One (SR1). • 3 The inverse of the BFGS formula. • 4 The inverse of the shifted BFGS formula. This should be used instead of .method = 3 whenever a shift is planned.

Data Fields

bool	any_method	allow space to permit different methods if required (less efficient)
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	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 lms_time_type

time derived type as a C struct

Data Fields

real_wp_	total	total cpu time spent in the package
real_wp_	setup	cpu time spent setting up space for the secant approximation
real_wp_	form	cpu time spent updating the secant approximation
real_wp_	apply	cpu time spent applying the secant approximation
real_wp_	clock_total	total clock time spent in the package
real_wp_	clock_setup	clock time spent setting up space for the secant approximation
real_wp_	clock_form	clock time spent updating the secant approximation
real_wp_	clock_apply	clock time spent applying the secant approximation

3.1.1.3 struct lms_inform_type

inform derived type as a C struct

Data Fields

int	status	the return status. Possible values are: • 0 the update was successful. • -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. • -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. • -3. One of the restrictions n $>$ 0, delta $>$ 0, lambda $>$ 0 or s^T y $>$ 0 has been violated and the update has been skipped. • -10. The matrix cannot be built from the current vectors {s_k} and {y_k} and values delta_k and lambda_k and the update has been skipped. • -31. A call to the function lhs_apply has been made without a prior call to lhs_form_shift or lhs_form with lambda specified when control.method = 4, or lhs_form_shift has been called when control.method = 3, or lhs_change_method has been called after control.any_method = false was specified when calling lhs_setup.
int	alloc_status	the status of the last attempted allocation/deallocation
int	length	the number of pairs (s,y) currently used to represent the limited-memory matrix.
bool	updates_skipped	have (s,y) pairs been skipped when forming the limited-memory matrix?
char	bad_alloc[81]	the name of the array for which an allocation/deallocation error occurred.
struct lms_time_type	time	timings (see above)