LibHSL: the ultimate collection for large-scale scientific computation

Alexis Montoison, Dominique Orban,

Andrew Lister, Jaroslav Fowkes

July 20th, 2024

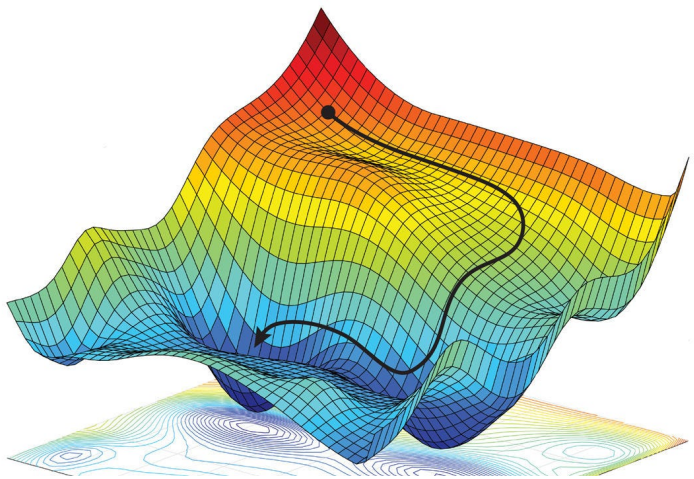
JuMP-dev – Montréal

Introduction

The HSL Mathematical Software Library [1] is a key resource for **numerical computations**, offering robust and efficient routines for **sparse linear systems** and **eigenvalue problems**.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \qquad \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \lambda \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

These routines are **essential tools** to address complex problems, including those arising in **continuous optimization** and **partial differential equations**.



1. Harwell Subroutine Library – `https://www.hsl.rl.ac.uk`

What is libHSL ?

- **LibHSL** is a collection of more than **160 HSL packages**.

- It aims to facilitate the use of HSL in **Julia** as well as **Fortran**, **C**, **C++**, and **Python** such as Ipopt, GALAHAD and CasADi.

- LibHSL provides the **source code** of the included HSL packages, **pre-built binaries** with **shared libraries**, and a Julia package named **HSL_jll.jl**.

How to use HSL_jll.jl in the Julia ecosystem

To **install** the package **HSL_jll.jl**, you only need the following commands in the Julia REPL :

```
julia> ]
pkg> dev path_to_hsl_jll
```

HSL_jll.jl is a **registered Julia package** and can be added as a **dependency** of any Julia package.

How to use HSL_jll.jl in Julia packages

- A version of HSL_jll.jl based on a dummy LibHSL was precompiled with Yggdrasil [1].

```
using HSL_jll

function LIBHSL_isfunctional()
    @ccall libhsl.LIBHSL_isfunctional()::Bool
end

bool = LIBHSL_isfunctional()
```

- In the dummy version, **LIBHSL_isfunctional** returns the boolean `false`. In the full-featured version provided by LibHSL, this C function returns the boolean `true`.

```
if LIBHSL_isfunctional()
  ... # solve a symmetric linear system with HSL_MA57
else
  ... # solve a symmetric linear system with SuiteSparse
end
```

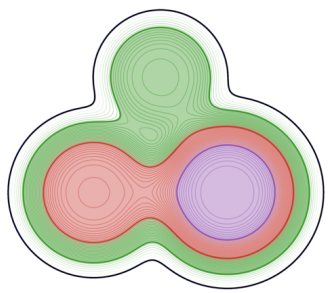
1. https://github.com/JuliaPackaging/Yggdrasil

HSL.jl

- The package **HSL.jl**[1] provides wrappers for all HSL packages with a C interface or those written in Fortran 77.

- **Wrappers** are Julia functions that call C and Fortran routines with the macro `@ccall`.

- The combination of **HSL.jl** and **HSL_jll.jl** enables users to **abstract away from low-level C and Fortran languages**, allowing them to utilize most HSL packages as if they were native Julia packages.



1. `https://github.com/JuliaSmoothOptimizers/HSL.jl`

How to use HSL_jll.jl in Ipopt.jl

For the Julia interface Ipopt.jl[1], HSL_jll.jl must be used directly.

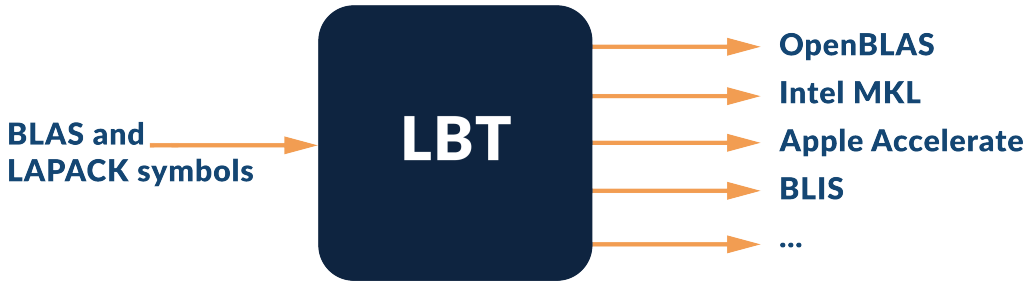
```
using JuMP, Ipopt, HSL_jll

# An optimization problem
model = Model(Ipopt.Optimizer)
@variable(model, x)
@objective(model, Min, (x - 2)^2)

# Use the linear solver MA57
set_attribute(model, "linear_solver", "ma57")
optimize!(model)

# Use the linear solver MA97
set_attribute(model, "linear_solver", "ma97")
optimize!(model)
```

The available HSL linear solvers are `"ma27"`, `"ma57"`, `"ma77"`, `"ma86"` and `"ma97"`.

1. https://github.com/jump-dev/Ipopt.jl

BLAS and LAPACK demuxing

HSL_jll.jl is compiled with **libblastrampoline**[1] (LBT) and offers the possibility to switch between *BLAS* and *LAPACK* backends in Julia.




```
# Load OpenBLAS
using OpenBLAS32_jll
BLAS.lbt_forward(libopenblas)

# Load MKL
using MKL

# Load AppleAccelerate
using AppleAccelerate

# BLAS and LAPACK backends loaded
BLAS.lbt_get_config()
```

1. https://github.com/JuliaLinearAlgebra/libblastrampoline

HSL vs MUMPS

```
This is Ipopt version 3.14.14, running with linear solver X.

Number of nonzeros in equality constraint Jacobian...:  3194444
Number of nonzeros in inequality constraint Jacobian.:   756090
Number of nonzeros in Lagrangian Hessian.............:  5708220

Total number of variables............................:   674143
                     variables with only lower bounds:        0
                variables with lower and upper bounds:   595665
                     variables with only upper bounds:        0
Total number of equality constraints.................:   661017
Total number of inequality constraints...............:   378045
        inequality constraints with only lower bounds:        0
   inequality constraints with lower and upper bounds:   126015
        inequality constraints with only upper bounds:   252030
```

Linear solver	Elapsed time (in seconds)
MUMPS	1445.870
MA27	302.216
MA57	319.198

1. https://github.com/JuliaLinearAlgebra/libblastrampoline

Future improvements

- Support for 64-bit integers : A planned enhancement is to add support for **64-bit integers** in HSL packages. This upgrade would enable the **solution** of even **larger-scale problems**.

- GPU-accelerated linear solvers : We anticipate integrating **GPU support** for BLAS libraries, such as CUBLAS for **NVIDIA GPUs**, rocBLAS for **AMD GPUs**, or oneMKL for **Intel GPUs**. This integration could significantly **enhance performance** in linear solvers.

Conclusion

- **LibHSL** provides **everything** that you could need in **HSL packages**;
- **HSL_jll.jl** is a pre-built version of LibHSL to be **readily used in the Julia ecosystem**;
- HSL_jll.jl is **precompiled** for various **operating systems** (Windows, Mac, Linux, FreeBSD) and **architectures** (x64, arm64, ppc64);
- The combination of HSL.jl and HSL_jll.jl allows one to **abstract away from the low-level C and Fortran languages**;
- LibHSL is **free** if you can request an **academic licence**!

Downloading libHSL

https://licences.stfc.ac.uk/product/libhsl