QRMumps.jl

`QRMumps.jl` is a Julia package designed to solve large, sparse linear systems of equations (Ax=b)

Technology

It acts as a wrapper for the QR_MUMPS library, which uses a multifrontal QR factorization method. This method is known for its numerical stability, especially with ill-conditioned matrices.

Audience

The software is targeted at computational scientists, engineers, and researchers using Julia for tasks that involve sparse matrices, such as numerical optimization.

Repository: github.com/JuliaQR/QRMumps.jl

Age: Started in 2017

Community: 6 lifetime contributors, with discussions on GitHub Issues

Activity Level: Mature and stable with a low commit frequency, which indicates a complete feature set rather than neglect.

Code Example

Setup

```julia
using SparseArrays, LinearAlgebra

# Small 4×4 sparse matrix
A = sparse([
    4.0   1.0   0.0   0.0;
    1.0   3.0   1.0   0.0;
    0.0   1.0   3.0   1.0;
    0.0   0.0   1.0   2.0
])

b = [15.0, 10.0, 10.0, 10.0]

display(A)
display(b)
```

```
4×4 SparseMatrixCSC{Float64, Int64} with 10 stored entries:
 4.0  1.0   ⋅    ⋅ 
 1.0  3.0  1.0   ⋅ 
  ⋅   1.0  3.0  1.0
  ⋅    ⋅   1.0  2.0
4-element Vector{Float64}:
 15.0
 10.0
 10.0
 10.0
```

Solve

```
using QRMumps, LinearAlgebra

qrm_init()

spmat = qrm_spmat_init(A)

spfct = qrm_spfct_init(spmat)

# Analyze sparsity, factorize, and solve
qrm_analyse!(spmat, spfct)
qrm_factorize!(spmat, spfct)
x = qrm_solve(spfct, b)

cond_number = cond(Matrix(A))

println("Solution x = ", x)
println("Residual norm = ", norm(A * x - b))
println("Condition Number = ", cond_number)
```

```
Solution x = [-3.001433187162955, 2.215685929711668, 5.988795543912927, -10.90406868485484]
Residual norm = 35.80278984429186
Condition Number = 4.2072442645144745
```

Question: How does QR factorization compare to the default LU for ill-conditioned systems?

Julia's default sparse solver uses LU factorization which is typically faster than QR.

However, QR factorization is known to be more numerically stable. For matrices that have a high condition number, this stability could be crucial for accuracy.

Proposed Experiment: Quantify the performance and accuracy trade-off between `QRMumps.jl` and Julia's default sparse solver.

Experimental Steps

1. **Generate Matrices:** Create a set of sparse matrices with increasing condition numbers. This systematically increases the difficulty of the problem.
2. **Solve:** For each matrix, solve the linear system Ax=b using both methods
3. **Measure:** Record two key metrics for each solve: How long did it take, and how accurate is the solution?