BLAS Implementation Blueprint for STDLIB

Breakdown of BLAS routines (according to the current tracking issue #2039)

Total no. of packages:

Level 1:

- Double Precision : 14
- Single Precision : 14
- Complex Double Precision: 11
- Complex Single Precision: 11

Total no. of Level 1 Packages: 50

Level 2:

- Double Precision : 16
- Single Precision : 16
- Complex Double Precision: 17

- Complex Single Precision: 17

Total no. of Level 2 Packages: 66

Level 3:

- Double Precision : 6
- Single Precision : 6
- Complex Double Precision: 9
- Complex Single Precision: 9

Total no. of Level 3 Packages: 30

Total : 146 Packages

Implemented packages:

Only Javascript Implemented:

Level 1:

- Double Precision : 1
- Single Precision : 1
- Complex Double Precision: 1
- Complex Single Precision: 1

Total no. of Level 1 Packages: 4

Level 2:

- Double Precision : 8
- Single Precision : 8
- Complex Double Precision: 0
- Complex Single Precision: 0

Total no. of Level 2 Packages: 16

Level 3:

- Double Precision : 1
- Single Precision : 1
- Complex Double Precision: 0
- Complex Single Precision: 0

Total no. of Level 3 Packages: 2

Total : 22 Packages

Fully Implemented(Javascript/C/FORTRAN):

Level 1:

- Double Precision : 12
- Single Precision : 12
- Complex Double Precision: 4
- Complex Single Precision: 5

Total no. of Level 1 Packages: 33

Level 2:

- Double Precision : 0
- Single Precision : 0
- Complex Double Precision: 0
- Complex Single Precision: 0

Total no. of Level 2 Packages: 0

Level 3:

- Double Precision : 0
- Single Precision : 0
- Complex Double Precision: 0
- Complex Single Precision: 0

Total no. of Level 3 Packages: 0

Total : 33 Package

WebAssembly packages:

This is the list of packages that already have a full implementation but do not have a Web Assembly implementation.

Has Pull Request:

- scasum

Does not have Pull Request:

- sswap
- sdsdot
- scnrm2
- dsdot
- dznrm2

- zscal

Note:

The above list does not take existing open pull requests into account. Currently, we have 21 open pull requests to the tracking issue(including only the feature pull requests)

LAPACK Dependencies(according to issue #2464):

This is the list of BLAS Routines that are the dependencies of the LAPACK routines.

Only Javascript Implemented:

Has Pull Request:

No such packages

Does not have Request:

- dspr
- dtpmv
- dgemm
- dtrmv

No Implementation:

Has Pull Request:

- dger
- dtpsv
- dtpmv
- dsymv
- dspmv

Does not have Request:

- dtbsv
- dtrsm
- lsame (needs discussion)

Note:

I have marked lsame as needs discussion because I am not sure about what the plan is behind that package, as it has a FORTRAN implementation in stdlib, and what the approach is for Javascript and C.

Implementation Plan for BLAS Routines

This will be the blueprint for how I am going to approach the implementation. I have come up with this after making several R&D with the maintainers, and I am open to feedback on this.

Priority Order(Based on programming language):

Javascript > C > FORTRAN

Priority Order(Based on levels):

Level1 > Level 2 > Level 3

Priority Order(Based on Precision):

Double Precision > Single Precision > Double Precison Complex > Single Precision Complex

Phase 1: (Implementation of LAPACK Dependencies)

During this phase, we will be focusing on the implementation of packages that LAPACK routines need to have. This will ensure that there is a smooth flow of implementations in LAPACK routines, too.

First, we will be implementing the packages that need Javascript implementations and then heading towards C/FORTRAN implementations.

Phase 2: (Implementation of Level 1 Routines)

In Level 1, there are only C/FORTRAN implementations for drotg, drotmg, srotg, srotmg (including the open pull requests. There is a draft pull request #4762 which we need to look at.

After we need to get into implementing Javascript implementations first for double precision and then single precision complex, and then their C/FORTRAN implementations.

Phase 3: (Implementation of Level 2 (Real Routines))

At this phase, we will move on to the Level 2 routines, specifically double precision and single precision ones, same as others, first the Javascript ones and then C and FORTRAN

Phase 4: (Implementation of Level 3 (Real Routines))

At this phase, we will move to implement double precision and single precision of Level 3 with the same priority order of languages. This priority is chosen because of the wide usage of real values over complex ones.

Phase 5: (Implementation of Level 2 (Complex Routines))

At this phase, we will implement double precision complex and single precision complex ones with the same priority order as the others.

Phase 6: (Implementation of Level 3 (Complex Routines))

At this phase, we will implement double precision complex and single precision complex of Level 3 with the same priority order of languages.

Additional Works:

- Add Web Assembly Implementations
- Improvement of implemented JSDoc to current stdlib standards
- Add documented fixtures as a reference for the user's view of matrices

- Need to know more about how cdotu and zdotu should be implemented based on higher-order implementation or lower-order implementation.

Note:

- The information given here is what I observed till the 29th March. The information may vary because of contributions after that.
- Currently, FORTRAN implementations for level 2 and level 3 have been blocked, so we need to implement pure C after javascript, which enables us to implement the WebAssembly for that particular package.