# progress towards porting EISPACK to forth

*Krishna Myneni*

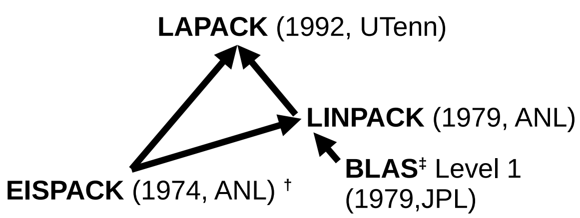
EuroForth 2022

V1.0



- numerical linear algebra source libraries for scientific computing
- numerical linear algebra modules in the FSL
- basic forms of linear systems problems I
- basic forms of linear systems problems II
- overview of EISPACK
- goals of porting EISPACK to Forth
- challenges of translating unstructured Fortran to Forth
- status of EISPACK port to Forth port
- what remains to be done?

## numerical linear algebra source libraries for scientific computing




- EISPACK and LINPACK were developed in 1970s – 1980s to provide *well-documented*, *well-tested* source libraries for scientific computing.
- EISPACK solves eigensystems of equations. Code is translated from *Algol*[¥] *→ Fortran (→ Forth)*.
- LINPACK solves linear systems of equations. Uses BLAS (Basic Linear Algebra Subprograms) Level 1.
- LAPACK combines functionality of LINPACK and EISPACK. It factors core matrix and vector computations (BLAS Level 3). Matlab, R, and other software use LAPACK.

† T. Haigh, *"An interview with Jack J. Dongarra,"* 26 April 2004, Soc. Industr. Appl. Math.; http://history.siam.org/pdfs2/Dongarra_ returned_SIAM_copy.pdf
‡ C. L. Lawson, et al., ACM Transactions on Math. Software 5, pp 308–323 (1979); https://doi.org/10.1145/355841.355847
¥ J. H. Wilkinson and C. Reinsch, Handbook for Automatic Computation: vol II Linear Algebra, Part 2, Springer-Verlag, New York 1972.

## numerical linear algebra routines in the Forth Scientific Library†

| module | description |
|---|---|
| **lufact** | factor a matrix **A** into a product of lower triangular (**L**) and upper triangular (**U**) matrices. |
| **dets** | find determinant of a matrix which has been factored in LU form. |
| **backsub** | solve linear system of equations using LU factorization: **A X** = **B**, where **A = L U** |
| **invm** | find the inverse of a matrix using LU factorization. |
| **gaussj** | provides tools for matrix arithmetic, finding inverse, solving linear system of equations and least-squares problems. |
| **svd** | solve matrix equations involving nearly singular matrices. |

- **FSL** provides some **LINPACK** functionality.
- **EISPACK** functionality is completely missing from the **FSL**!

† The Forth Scientific Library, https://www.taygeta.com/fsl/scilib.html

## basic forms of linear systems problems I

**I. A X** = **B** : **A** and **B** are given; solve for **X**

**simple case:**

$2x_0 + 3x_1 = -6$

$4x_0 + 8x_1 = 10$

**matrix form:**

$$\begin{pmatrix} 2 & 3 \\ 4 & 8 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \begin{pmatrix} -6 \\ 10 \end{pmatrix}$$

solve using Gauss-Jordan elimination with FSL **gaussj**:

```
 2 2 float matrix a{{
 2 1 float matrix b{{
  2.0e0  3.0e0
  4.0e0  8.0e0 2 2 a{{ }}fput   \ init matrix a{{
-6.0e0 10.0e0 2 1 b{{ }}fput   \ init matrix b{{
 a{{ b{{ 2 1 gaussj .          \ solve and print error (0 = no error)
 2 1 b{{ }}fprint              \ print solution x0 and x1:
                               \ -19.5
                               \ 11
```

## basic forms of linear systems problems II

**II. A X** = λ**X** : **A** is given; solve for λ's and corresponding **X**'s

**simple case:**

$2x_0 + 3x_1 = \lambda\, x_0$

$3x_0 + 4x_1 = \lambda\, x_1$

**matrix form:**

$$\begin{pmatrix} 2 & 3 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} = \lambda \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$$

solve using matrix tridiagonalization and QL reduction with EISPACK **tred2** and **imtql2** :

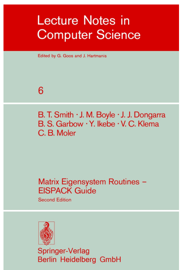
```
2 2 float matrix A{{
 2.0e0  3.0e0  3.0e0  4.0e0   2 2 A{{ }}fput

2 float array diag{  2 float array subdiag{  2 2 float matrix ot{{

2 2 A{{ diag{ subdiag{ ot{{ tred2  \ tridiagonalize the matrix
2 2 diag{ subdiag{ ot{{ imtql2 .   \ find λs and eigenvectors; print error code
2 diag{ }fprint                    \ print eigenvalues (λs): -0.162278 6.16228
2 2 ot{{ }}fprint                  \ print corresponding eigenvectors:
                                   \ 0.811242 0.58471
                                   \ -0.58471 0.811242
```

## overview of EISPACK

"EISPACK is a systematized collection of [Fortran] subroutines† which compute the eigenvalues and/or eigenvectors of six classes of matrices…"




1 *complex* general
2 *complex* Hermitian
3 *real* general
4 *real* symmetric
5 *real* symmetric tridiagonal
6 *real* special tridiagonal

EISPACK Guide‡ provides recommended calling sequence of routines, the "EISPACK path", for a given problem class, e.g.

```
call balanc( … )
call elmhes( … )
call eltran( … )
call hqr2( … )
call balbak( … )
```

to find all eigenvalues and eigenvectors for a real general matrix.

† Fortran source library from August 1983 release is at https://netlib.org/eispack/
‡ B. T. Smith, et al., Matrix Eigensystem Routines – EISPACK Guide 2nd ed., Springer-Verlag 1976.

## goals of porting EISPACK routines to Forth

- provide library of eigensystems solvers to support scientific computing in Forth
- provide source library in Forth for portability, ease of use, ease of debugging, and ability to modify/adapt the code
- translate unstructured Fortran to structured Forth for improved source comprehensibility
- work with FSL style matrices and arrays
- provide test code and examples in library
- check results of Forth computations against "original" Fortran

## challenges of translating unstructured Fortran to Forth

```
      subroutine imtql1(n,d,e,ierr)
   :
   :
c     look for small sub-diagonal element
  105    do 110 m = 1, n
   :
            if (r .eq. 0.0d0) go to 210
   :
  200    continue
c
         d(l) = d(l) - p
         e(l) = g
         e(m) = 0.0d0
         go to 105
c     recover from underflow
  210    d(i+1) = d(i+1) - p
         e(m) = 0.0d0
         go to 105
c     order eigenvalues
  215    if (l .eq. 1) go to 250
c     for i=l step -1 until 2 do
         do 230 ii = 2, l
            i = l + 2 - ii
            if (p .ge. d(i-1)) go to 270
            d(i) = d(i-1)
  230    continue
c
  250    i = 1
  270    d(i) = p
  290 continue
```

→

```
: imtql1 ( n d e -- ierr )
   :
   :
\  look for small sub-diagonal element
       BEGIN
         N I DO
   :
         uflow IF
\  recover from underflow
           d{ ii 1+ } f@ p f@ f- d{ ii 1+ } f!
           false to uflow
         ELSE
           d{ I } f@ p f@ f- d{ I } f!
           g f@ e{ I } f!
         THEN
         0.0e0 e{ m } f!
       REPEAT

\  order eigenvalues
       I 0 = IF
         p f@ d{ 0 } f!
       ELSE
\ for i=l step -1 until 2 do
         I 2+ 1 DO
           J 1+ I - to ii
           p f@ d{ ii 1- } f@ f>= IF
             LEAVE
           THEN
           d{ ii 1- } f@ d{ ii } f!
         LOOP
         p f@ d{ ii } f!
       THEN
     LOOP  \ end main loop
```

## status of EISPACK port to Forth

| word | description | In Pr | C/N.T. | C/T | Demo |
|---|---|---|---|---|---|
| `balanc` | balance a real matrix | ✓ | | | |
| `balbak` | form eigenvectors of a general real balanced matrix | | ✓ | | |
| `elmhes` | reduce submatrix of real general matrix to upper Hessenberg form | | ✓ | | |
| `eltran` | accumulate similarity transforms for reduction of real general matrix | | ✓ | | |
| `hqr2` | find eigenvalues and eigenvectors of real general matrix | ✓ | | | |
| `htribk` | form eigenvectors of complex Hermitian matrix | | | ✓ | `cherm-01.4th` |
| `htridi` | reduce complex Hermitian matrix to real symmetric tridiagonal | | | ✓ | `cherm-01.4th` |
| `imtql1` | find the eigenvalues of a real symmetric tridiagonal matrix | | | ✓ | `rsymm-01.4th` |
| `imtql2` | find eigenvalues and eigenvectors of real symmetric tridiag matrix | | | ✓ | `rsymm-02.4th` |
| `tred1` | reduce real symmetric matrix to tridiagonal matrix | | | ✓ | `rsymm-01.4th` |
| `tred2` | reduce real symmetric matrix to symmetric tridiagonal matrix | | | ✓ | `rsymm-02.4th` |

**In Pr** = *Fortran* → *Forth* translation in progress
**C/N.T.** = Completed translation, **not tested**
**C/T** = Completed translation, **tested**

**C/T** Forth code may be found at
https://github.com/mynenik/kForth-64/tree/master/forth-src/eispack

*complex* general
*complex* Hermitian ✓
*real* general
*real* symmetric ✓
*real* symmetric tridiagonal ✓
*real* special tridiagonal

## what remains to be done?

- complete translation and testing of words needed to solve eigensystems for *real* general matrices (target date: end of 2022)
- begin translation of words for solution of *complex* general matrices (2023)
- write demo programs to test and illustrate use of EISPACK in Forth (2023 – 2024)
- testing, testing, testing …

***applications are the payoff!***