Oak Leaf

The Guide

Filip Hroch

Brno 2021

Introduction

Oak Leaf is a computer library which implements robust statistical methods.

```
https://integral.physics.muni.cz/oakleaf
```

1 QR factorisation

This chapter describes an implementation of QR factorisation and related algorithms by Oak Leaf. The implementation is based on ideas, the spirit, and a framework developed in the book Golub, G. & Van Loan, C. *Matrix Computations, 4th edition* (Johns Hopkins University Press, 2013) (henceforth cited as GvL). The fundamental principles of QR decomposition, the algorithm by John G.F. Francis[1], can be found in references, see Section 1.9.

I have been studying QR factorisation in the hope I will find genuinely deep understanding of principles, and properties, of the approach. It has helped me to select of effective algorithms for optimisation problems solved in Oak Leaf rather than to apply of general forms of the algorithms. Another benefit is the implementation in modern Fortran equipped by vectorisation computing capabilities; that permits writing of a clean and computationally efficient code. By the way, I had been able to drop dependency of Oak Leaf on external libraries, however compilers itselfs utilises routines by Lapack ([2]), the well-known project on base of GvL.

I had developed both computation effective and debug variants of the routines. A source code for the effective way can be found in `src/qrhouse.f08`, whilst `src/qrdebug.f08` is the basic, naive, implementation intended for verification of algorithms at all. Run-time tests contains the files `test/qr/testqrhouse.f08`, and `test/qr/testqrdebug.f08`.

1.1 QR factorisation

QR algorithm factorise (decompose) of a matrix A on: an upper diagonal triangular matrix R, and Q matrix having orthogonal columns

$$A = QR. \tag{1.1}$$

The decomposition is a wise way how to solve systems of linear equations, to derive an inversion, or to find both eigenvalues and eigenvectors of a matrix. QR factorisation can be applied on non-square matrices, or rank deficient systems of linear equations; the problems so much important for real world applications. All these astonishing abilities raises from the

[1] `https://en.wikipedia.org/wiki/John_G._F._Francis`.

[2] `http://www.netlib.org/lapack/index.html`.

extraordinary coincidence: a linear system with triangular matrix R can be solved trivialy, and the inversion of an orthogonal matrix is the matrix transpose $Q^{-1} = Q^T$.

QR factorisation can be applied on the matrix $A^{m \times n}$ having real elements $a_{i,j} \in \mathcal{R}$ arranged in m rows and n columns. i is an index for rows, j for columns. The result matrices has dimensions $m \times n$ for R, and $m \times m$ for Q. A can be non-square $m \neq n$, singular, and asymmetric. Q is a matrix having orthogonal columns

$$
\begin{aligned}
Q_i^T \cdot Q_j = 0, &\quad i \neq j, \\
Q_i^T \cdot Q_i \neq 0, &\quad i = j,
\end{aligned}
$$

or $Q^T Q = QQ^T = I$. The generalisation on complex elements of A is straightforward, but out of scope of our interest.

QR factorisation for elements of 3×3 matrix can be written as

$$
\begin{pmatrix} a_{11}, & a_{12}, & a_{13}, \\ a_{21}, & a_{22}, & a_{23}, \\ a_{31}, & a_{32}, & a_{33} \end{pmatrix} = \begin{pmatrix} q_{11}, & q_{12}, & q_{13}, \\ q_{21}, & q_{22}, & q_{23}, \\ q_{31}, & q_{32}, & q_{33} \end{pmatrix} \begin{pmatrix} r_{11}, & r_{12}, & r_{13}, \\ 0, & r_{22}, & r_{23}, \\ 0, & 0, & r_{33} \end{pmatrix}. \tag{1.2}
$$

The principle of QR algorithm is to find such equivalence transformation of A which will nullify $r_{ij} = 0$ under-diagonal elements $i < j$ of R together with keeping Q orthogonal. Householder reflection, represented by the symmetric $m \times m$ projection matrix

$$
P = I - \beta\, v \otimes v^T, \quad \beta = \frac{2}{v^T \cdot v}, \tag{1.3}
$$

satisfies the requirement.[3] The determination of Householder vector v, the heart of the factorisation, will be described in Section 1.3.

Since this point, Fortran notation for element block selection of matrices and vectors will be used: the plain (:) means whole row or column, $j{:}m$ is a part with starting index j and the last index m (including the interval points).

A simple choice for the annihilation of all sub-diagonal elements of the column A_j is to get j-th column of A: $x = A_j \in \mathcal{R}^m$, and to set the Householder vector as

$$
v(j{:}m) = x(j{:}m) \pm \|x(j{:}m)\|_2\, e_j \tag{1.4}
$$

for the elements of x with indexes $j \dots n$, zero otherwise. The vector is a reflection around the unit vector e_j. Householder matrix (1.3) for $j = 2$ is derived from $v^{(2)} = (0, v_2, v_3)^T$ in the

[3] The outer and the inner (scalar) product of vectors $u, v \in \mathrm{R}^m$ are defined as

$$
u \otimes v^T = u_i v_j, \; i, j = 1, \dots, m, \quad \text{and} \quad u^T \cdot v = \sum_{i=1}^{m} u_i v_i.
$$

form

$$P^{(2)} = \begin{pmatrix} 1, & 0, & 0, \\ 0, & p_{22}, & p_{23}, \\ 0, & p_{32}, & p_{33} \end{pmatrix}. \tag{1.5}$$

The annihilation process is designed by this way: the matrix multiplication $P^{(1)}A$ eliminates the first sub-column giving the first iteration of $R^{(1)}$, the updated matrix is used to the eliminate next column $R^{(2)} = P^{(2)}R^{(1)}$, and the machinery is repeated until finish in n steps.

$$R^{(j)} = P^{(j)}R^{(j-1)}, \quad j = 1 \dots n. \tag{1.6}$$

The sequence has start from the full matrix $R^{(0)} = A$,

The orthogonal matrix Q collects the projection matrices during the process.

$$Q^{(j)} = Q^{(j)}P^{(j-1)}, \quad j = 1 \dots n \tag{1.7}$$

The sequence has start from a unitary matrix $Q^{(0)} = I$ of dimension m.

Algorithm 1.1 summarises our development. One can be implemented via matrix operations.

Let $A, R \in \mathcal{R}^{m\times n}$, $Q, I, P \in \mathcal{R}^{m\times m}$, $v \in \mathcal{R}^{m}$, $\beta \in \mathcal{R}$
$R = A$
$Q = I$
for $j = 1 \dots n$
  $v, \beta =$ housevec(R_j)
  $P = I - \beta\, v \otimes v^T$
  $R = PR$
  $Q = QP$
end for
return factored A into R, Q

Algorithm 1.1: QR factorisation

The R can be computed using of the sub-matrix $R(j{:}m, j{:}n) = P(j{:}m, j{:}m), R(j{:}m, j{:}n)$; this way of acceleration is not valid for Q.

There are more ways how to derive QR factorisation. Householder projection is the way having best numerical stability and properties. Givens rotations, or modified Gram-Schmidt process, are another methods useful in particular problems.

1.2 An effective QR factorisation

The formulation of QR factorisation in previous section 1.1 can be improved in terms of both speed up and memory economy.

1.2.1 Speed up

The computations of P in (1.3) are expensive due the outer product of Householder vector $v \in \mathcal{R}^m$, followed by the additional matrix multiplication.

A way how to significantly improve speed of the approach comes from the observation (GvL, Sect. 5.1.4) that the projection matrix P is commonly used as part of matrix products. In such case, P can be eliminated in behalf of v:

$$PA = (I - \beta\, v \otimes v^T)\, A = A - (\beta v) \otimes (v^T A) \tag{1.8}$$

as well as

$$AP = A\,(I - \beta\, v \otimes v^T) = A - (Av) \otimes (\beta v)^T. \tag{1.9}$$

The re-arrange removes one step in the above algorithm. Another step reduction is possible, when Q is not required explicitly.

1.2.2 Room sharing

The second improvement evolves the previous idea: if we need only v for successive operations, we can drop to keep matrix Q which can be recovered from vectors v, if need. That approach is very handy; otherwise, we should keep both Q and a set of v in order to be able do any operations with factored matrices.

Now, we will keep only vectors v, and recover Q later. The vectors v can be stored in an additional matrix $(v^{(j)})$, but there is a better way. All the vectors $v^{(j)}$ has the structure

$$(0, \dots, v_j, v_{j+1}, \dots, v_m)^T.$$

with $j - 1$ null elements. The determination of P (1.3) is independent on norm of v, so it becomes useful to introduce the normalisation

$$v = \left(0, \dots, 1, \frac{v_{j+1}}{v_j}, \dots, \frac{v_m}{v_j}\right)^T. \tag{1.10}$$

Only the elements $j + 1, \dots, n$ carries desired information, and we can use under diagonal elements, the unfilled place, of matrix R to store parts of the vectors.

The final structure of matrix R, which can share memory with A being overwritten sequentially, will be

$$\begin{pmatrix} r_{11}, & r_{12}, & r_{13}, & r_{14}, \\ v_2^{(1)}, & r_{22}, & r_{23}, & r_{24}, \\ v_3^{(1)}, & v_3^{(2)}, & r_{33}, & r_{34}, \\ v_4^{(1)}, & v_4^{(2)}, & v_4^{(3)}, & r_{44} \end{pmatrix}. \tag{1.11}$$

1.2.3 Some notes

It may be disputable, if this way to save some memory is rational today, but the related speed up should be important for applications.

The algorithm here is not intended for solution of very large problems with sparse matrices. The expected matrix should be dense and small. For that matrices, the Householder transformation offers numerical stability — it is the most stable known method.

The execution speed can be improved by a parallel implementation. Unfortunately, the Householder matrix should be applied in an exact sequence, the parallelization is difficult but possible via a recursion (see Golub, G. & Van Loan, C. *Matrix Computations, 4th edition* (Johns Hopkins University Press, 2013), henceforth cited as GvL Sect. 5.2.4). The more parallel friendly way is to replace Householder transformations by Givens rotations. Unfortunately, they brings worse numerical stability so they are recommended for large sparse problems only.

The effective way, described in this section, can speed-up computations, but it will work only for compiled computer languages like Fortran, Go, or Julia. The scripting languages, Matlab or Python, will benefits from the matrix formulation; per-element operations are too slow due the additional interactive layer between the operator and the physical machine.

And finally, there is a question of choice: Do we want to save some memory, to be fast, or to be easy to understand?

1.2.4 The effective QR factorisation

This section describes algorithms for the effective QR factorisation as has been linked above. The algorithms are designed for a heavy load processing.

The effective QR factorisation in Algorithm 1.2 is a transcription of Algorithm 5.2.1 by GvL. The function housevec is defined in Algorithm 1.5. The function housemul is a convenient implementation of (1.8). Having Householder vectors v stored inside the factored matrix, we can easy prepare the orthogonal matrix Q at any moment; Algorithm 1.4 (the equation (5.1.5) due GvL) is such representation.

All the presented algorithms are only slightly changed prototypes of the equivalents of GvL intentionally keeping notations of the book. Only a few important differences are here: Householder vector v is starting from j-th index, rather than from the first index, to be consistent in notation. The difference is observable on last line of Algorithm 1.2 where $A(j+1{:}m, j) = v(j+1{:}m)$ is replaced by $A(j+1{:}m, j) = v(1{:}m-j+1)$. Another difference is to keep of the vector β for later use, rather than recovering it during subsequent applications to save computation time.

Also, all the algorithms are arranged such way, to by easy implemented in Fortran (see `src/qrhouse.f08`). Just for comparison, the particular implementation of Algorithm 1.3 in Fortran can be found in listing 1.1.

function qrfac(A, β)
  Let $A \in \mathcal{R}^{m \times n}, \beta \in \mathcal{R}^{n}$
  A contains a matrix to factorise; the upper triangle of R and Householder vectors v on output. β is an output vector storing computed normalisation.
  An auxiliary variable: $v \in \mathcal{R}^{m}$
  for $j = 1 \ldots n$
    $v(j{:}m), \beta_j =$ housevec($A(j{:}m, j)$)
    $A(j{:}m, j{:}n) =$ housemul($v(j{:}m), \beta_j, A(j{:}m, j{:}n)$)
    if $j < m$
      $A(j+1{:}m, j) = v(j+1{:}m)$
    end if
  end for
  Returns factored A and β, see section. 1.2.2.
end function

Algorithm 1.2: The effective QR factorisation

function housemul(v, β, A)
  Let $A \in \mathcal{R}^{m \times n}, v \in \mathcal{R}^{n}, \beta \in \mathcal{R}$
  Auxiliary variables: $u, w \in \mathcal{R}^{m}$
  $u(j{:}m) = \beta v(j{:}m)$
  for $j = 1 \ldots n$
    $w_j = v^T \cdot A_j$
  end for
  for $i = 1 \ldots m, j = 1 \ldots m$
    $A_{ij} = A_{ij} - u_j w_i$
  end for
  Returns RP stored on place.
end function

Algorithm 1.3: Householder matrix multiplication

function qform(A, β)
  Let $A \in \mathcal{R}^{m\times n}, \beta \in \mathcal{R}, Q \in \mathcal{R}^{m\times m}$
  Auxiliary variables: $v, u, w \in \mathcal{R}^m$
  $Q = I_{nn}$
  for $j = n, n-1, \dots, 1$
    $v(j{:}m) = (1, A(j+1{:}m, j))^T$
    $u(j{:}m) = \beta_j v(j{:}m)$
    for $l = j \dots n$
      $w_l = v(j{:}m)^T \cdot Q(j{:}m, l)$
    end for
    for $k = j \dots m, l = j \dots n$
      $Q_{kl} = Q_{kl} - w_l u_k$
    end for
  end for
  Returns Q
end function

Algorithm 1.4: Evaluate Q from v stored under diagonal of R

Listing 1.1: Householder matrix multiplication in Fortran

```
subroutine housemul(v,beta,A)

  real, dimension(:), intent(in) :: v
  real, intent(in) :: beta
  real, dimension(:,:), intent(in out) :: A

  real, dimension(size(v)) :: u, w
  integer :: i,j,m,n

  m = size(A,1)
  n = size(A,2)
  u = beta*v
  forall(j = 1:m) w(j) = dot_product(v,A(:,j))
  forall(i = 1:n, j = 1:m) A(i,j) = A(i,j) - u(i)*w(j)

end subroutine housemul
```

1.3 Householder reflection

To nullify the desired elements of R during j-th iteration by the numerically stable way, Householder vector $v^{(j)}$ is defined via the corresponding column of A as $x^{(j)} = A_j$ (j-th column of A)

$$v_i^{(j)} = \begin{cases} 0, & i < j, \\ v_j, & i = j, \\ x_j, & i > j, \end{cases} \tag{1.12}$$

where

$$v_j = x_j \pm \operatorname{sign}(x_j) \|x(j{:}m)\|_2. \tag{1.13}$$

The sign in (1.13) as well as the computation of v_j are source of a disputation. GvL proposes negative sign (to maximise norm) which is keeping the numerical precision. If x is close to a positive multiple of e, we can improve the precision by the way

$$v = x - \|x\|_2 = \frac{x_1^2 - \|x\|_2}{x_1 + \|x\|_2} = -\frac{x_2^2 + \ldots + x_m^2}{x_1 + \|x\|_2}.$$

Finally, we define

$$v_j = \begin{cases} -\dfrac{x_{j+1}^2 + \ldots + x_m^2}{x_j + \|x(j{:}m)\|_2}, & x_j \geq 0, \\ x_j - \|x(j{:}m)\|_2, & x_j < 0. \end{cases} \tag{1.14}$$

In the important case of a rank deficient matrix, Householder vector will $v = 0$, and we should add the condition $\beta = 0$.

Example 1.1 Householder vector

We will compute Householder vector v and matrix P for the first column of the matrix (example[4] by wiki , our procedure reflects the distinct definition):

$$\begin{pmatrix} 12, & -51, & 4, \\ 6, & 167, & -68, \\ -4, & 24, & -41. \end{pmatrix} \tag{1.15}$$

For $j = 1$, we have $v^{(1)} = (1, -3, 2)^T, \beta = 1/7$ from (1.12), (1.14) and (1.3)

$$H^{(1)} = \begin{pmatrix} 1, & 0, & 0, \\ 0, & 1, & 0, \\ 0, & 0, & 1. \end{pmatrix} - \frac{1}{7} \begin{pmatrix} 1, & -3, & 2, \\ -3, & 9, & -6, \\ 2, & -6, & 4. \end{pmatrix} = \begin{pmatrix} 6/7, & 3/7, & -2/7, \\ 3/7, & -2/7, & 6/7, \\ -2/7, & 6/7, & 3/7. \end{pmatrix} \tag{1.16}$$

(to be continued). ◇

[4] `https://en.wikipedia.org/wiki/QR_decomposition`.

function housevec(x)
    $x, v \in \mathcal{R}^m, \beta \in \mathcal{R}$
    $v(2{:}m) = x(2{:}m)$
    $\sigma^2 = x(2{:}m)^T \cdot x(2{:}m)$
    if $\sigma^2 > 0$
        $d = \sqrt{x_1^2 + \sigma^2}$
        if $x_1 > 0$
            $v_1 = -\sigma^2/(x_1 + d)$
        else
            $v_1 = x_1 - d$
        end if
        if $|v_1| > 0$
            $v = v/v_1$
            $\beta = 2/(v^T \cdot v)$
        else
            $\beta = 0, v = 0$
        end if
    else
        $v_1 = 1$
        if $x_1 \geq 0$
            $\beta = 0$
        else
            $\beta = 2$
        end if
    end if
    return v, β
end function

Algorithm 1.5: Householder vector

1.4 QR factorisation with pivoting

QR factorisation including pivoting is important for the numerical stability, and significantly extends field of usage of the developed methods. It permits to find a solution in such real cases when a system of linear equations is badly conditioned or singular. It also suppress accumulation of numerical rounding errors emerged from long summations and products, the most frequent operations executed on matrices.

The implemented pivoting uses the same principles as the column pivoting known for Gauss elimination technique. The pivots are represented by a matrix Π which collects permutations of rows of A. Π has all elements as zeros except the ones on the appropriate places

$$A\Pi = QR. \tag{1.17}$$

A common used storage of the permutation matrix is a vector $p \in \mathcal{N}_0^m$. It is implemented in Algorithm 1.6 which is a rich variant of Algorithm 1.2. The permutation matrix Π can be easy recovered from p as Algorithm 1.7 demonstrates.

1.5 A solution of a system of linear equations

A solution $x \in \mathcal{R}^m$ of a system of linear equations

$$Ax = b \tag{1.18}$$

with a matrix A factored on Q, R, and a right side $x \in \mathcal{R}^m$, is straightforward thanks to the identity

$$Q^{-1} = Q^T. \tag{1.19}$$

The full system A is transformed on the system having triangle matrix R

$$Rx = q, \tag{1.20}$$

and the right side $q = Q^T b$. To solve one, we can apply the back-substitution under the condition $|R_{ii}| \neq 0$ (see Algorithm 1.8):

$$x_i = \frac{1}{R_{ii}} \left(q_i - \sum_{j=i+1}^{m} R_{ij} x_j \right), \qquad i = m, \dots, 1. \tag{1.21}$$

Note that Q can be again restored from vectors v stored in columns of R by applications of the trick presented in section 1.2.1. Algorithm 1.9 materialises the idea.

This way is only valid for full square matrices with dimensions $m \times m$. This is, in fact, an equivalent to direct Gaussian elimination or its variants as LU or LL^T factorisation, notwithstanding a numerical class of the matrices is wider (GvL, Sect. 5.3).

function qrpivot(A, β, p)
    Let $A \in \mathcal{R}^{m \times n}, \beta \in \mathcal{R}^n, p \in \mathcal{N}_0^m, m \geq n$.
    A contains a matrix to factorise; the upper triangle of R and Householder vectors v and factors β on output. p is vector of permutations.
    An auxiliary variables: $c, v \in \mathcal{R}^m, \tau \in \mathcal{R}$
    $p = (\{1, \ldots, n\})^T$
    for $j = 1 \ldots n$
        $c(j) = A(1{:}m, j)^T \cdot A(1{:}m, j)$
    end for
    for $r = 1, \ldots, n$
        Find the smallest index $r \leq k \leq n$ for which $c(k)$ is the maximum.
        if not $(c(k) > 0)$
            exit loop over r
        end if
        if $r \neq k$
            $p(r) \leftrightarrow p(k)$
            $c(r) \leftrightarrow c(k)$
            $A(1{:}m, r) \leftrightarrow A(1{:}m, k)$
        end if
        $v(r{:}m), \beta(r) =$ housevec($A(r{:}m, j)$)
        $A(r{:}m, r{:}n) =$ housemul($v(r{:}m), \beta(r), A(r{:}m, r{:}n)$)
        $A(r+1{:}m, r) = v(r+1{:}m)$
        for $l = r+1 \ldots n$
            $c(l) = c(l) - A(r, l)^2$
        end for
    end for
    return A factored, β, p
end function

Algorithm 1.6: QR factorisation with column pivoting

$\Pi = 0$
for $j = 1, \ldots, m$
    $\Pi(p(j), j) = 1$
end for

Algorithm 1.7: Compilation of the permutation matrix

function rsol(A, q)
  Let $A \in \mathcal{R}^{m\times n}, q, u \in \mathcal{R}^n$
  for $i = m, \ldots, 1$
    $u(i) = [q(i) - A(i, i+1{:}n)^T \cdot x(i+1{:}n)]/A(i,i)$
  end for
  return u
end function

Algorithm 1.8: Solution of $Rx = q$ with back-substitution

The essential advantage of QR factorisation is revealed for over-determined systems having more equations than variables ($m > n$), and under-determined systems having less equations than variables ($m < n$). The particular important case are rank deficient singular matrices.

Next paragraphs briefly describes solutions in the difficult cases, which covers Algorithm 1.10.

1.5.1 A least square solution of a system of linear equations

The least square solution minimises of Pythagorean norm, a generalised distance, between a computed Ax, and expected b, solutions

$$\min ||Ax - b||_2, \tag{1.22}$$

where $A \in \mathcal{R}^{m\times n}, m \geq n, b \in \mathcal{R}^m, x \in \mathcal{R}^n$. The problem can be solved with help of QR factorisation which decompose the system on the two parts:

$$Q^T A = R = \begin{pmatrix} R_1, \\ 0 \end{pmatrix}, \quad Q^T b = \begin{pmatrix} c, \\ d \end{pmatrix}. \tag{1.23}$$

$R_1 \in \mathcal{R}^{n\times n}$ is a square matrix. The lower part of factorised R is nullified. $c \in \mathcal{R}^n$ part corresponds to the solution, while $d \in \mathcal{R}^{m-n}$ stores residuals.

$$||Ax - b||_2^2 = ||Q^T Ax - Q^T b||_2^2 = ||R_1 x - c||_2^2 + ||d||_2^2. \tag{1.24}$$

The solution in sense of least squares is than

$$R_1 x_0 = c, \tag{1.25}$$

and the residual sum is

$$||Ax_0 - b||_2 = ||d||_2. \tag{1.26}$$

function houseapp(A, β, x)
    Let $A \in \mathcal{R}^{m\times n}$, $v, \beta \in \mathcal{R}^n$
    for $j = 1, \ldots, m$
        $v(j{:}m) = (1, A(j+1{:}m, j))^T$
        $x(j{:}m) = x(j{:}m) - \beta_j[v(j{:}m)^T \cdot x(j{:}m)]v(j{:}m)$
    end for
    return x
end function

Algorithm 1.9: Householder matrix application

function qrsol(A, b, x)
    Let $A, R \in \mathcal{R}^{m\times n}$, $x, b \in \mathcal{R}^n$
    Auxiliary variables: $u, q, \beta \in \mathcal{R}^m$
    if $m > n$
        The least-square solution, Section 1.5.1
        $R, \beta =$ qrfac(A)
        $q =$ houseapp(R, β, b)
        return $x =$ rsol($R(1{:}n, 1{:}n), q$)
    else
        $R, \beta, \Pi =$ qrpivot(A)
        $q = Q^T b =$ houseapp(R, β, b)
        Set $\delta > 0$, $\delta = \tau\varepsilon|R(1,1)|$, $R(1,1) \neq 0$
        Determine rank: $r = \{i, |R(i,i)| > \Delta\}$
        if $r = m$
            A square matrix Section 1.5
            $z =$ rsol(R, q)
            return $x = \Pi^T z$
        else if $r < m$
            A rank deficient matrix, Section 1.5.3
            Basic solution: $u(1{:}r) =$ rsol($R(1{:}r, 1{:}r), q(1{:}r)$), $x_B = \Pi^T u$
            $S(1{:}n, 1{:}r), \beta'(1{:}r) =$ qrfac($R^T(1{:}r, 1{:}n)$)
            $U(1{:}n, 1{:}n) =$ qform($S(1{:}n, 1{:}r), \beta'(1{:}r)$)
            $z(1{:}r) =$ lsol($S(1{:}r, 1{:}m), q(1{:}r)$)
            return $x = \Pi^T U(:, 1{:}r)z(1{:}r)$
        end if
    end if
end function

Algorithm 1.10: Linear equations solver

1.5.2 A basic solution of linear system with a rank deficient matrix

We know that rank deficient matrix has no unique solution. The class of the possible solutions is usually described with help of a slack variable, known as a parameter, for analytical solutions.

The parametric solution approach is not suitable for a numerical solution. It is reason than one adds an additional information to select some particular solution. A basic solution is developed in this section, while next section describes a solution with the minimal norm.

A solution of rank deficient problems has started by column pivoted QR factorisation $A\Pi = QR, Q^{-1} = Q^T$. The pivoting process sorts columns by size of diagonal elements $|R_{11}| \geq |R_{22}| \geq \ldots \geq |R_{mm}|$. Potentially linearly dependent columns has the diagonal elements close to the machine epsilon ε. In such case, a condition ($\tau > 1$), says

$$|R_{ii}| < \tau\varepsilon|R_{11}| \tag{1.27}$$

can be used to reveal a numerical rank of the matrix. We will denote the rank, an index of a last non-zero diagonal element in numerical sense, by r,

Let $r \in \mathcal{N}$ is the rank of matrix A, the last index for which $|R_{rr}| > \varepsilon|R_{11}|$. The pivoting process will split the matrix R, and the system, on two parts

$$A\Pi = Q\begin{pmatrix} R_{11}, & R_{12}, \\ 0 & 0 \end{pmatrix}. \tag{1.28}$$

with $R_{11} \in \mathcal{R}^{r\times r}$, $R_{12} \in \mathcal{R}^{r\times m-r}$. The least square approach requires

$$\|Ax - b\|_2^2 = \|(Q^T A\Pi)(\Pi x) - (Q^T b)\|_2^2 = \|R_{11}y(c - R_{12}z)\|_2^2 + \|d\|_2^2, \tag{1.29}$$

where

$$\Pi^T x = \begin{pmatrix} y, \\ z \end{pmatrix}, \qquad Q^T b = \begin{pmatrix} c, \\ d \end{pmatrix}. \tag{1.30}$$

The solution is

$$x_B = \Pi\begin{pmatrix} R_{11}^{-1}(c - R_{12}z), \\ z \end{pmatrix}. \tag{1.31}$$

Ones calls x_B solution as a basic solution. The basic solution is no solution in least square sense, except case $R_{12} = 0$.

1.5.3 A solution in least square sense by a complete orthogonal decomposition

The least square solution of rank deficient matrix can be found by application of a dark magic trick. If we continue with the revealed information from previous paragraph 1.5.2, the rectangular matrix has dimensions $r \times m$. We can transform the matrix on the least square solution studied in Section 1.5.1 by a matrix transposition having dimensions $m \times r$, $r \leq m$.

By the way, we apply again the QR factorisation on already factored transposed triangle of R: $R^T = US$

$$\begin{pmatrix} R_{11}^T, \\ R_{12}^T \end{pmatrix} = U \begin{pmatrix} S_1, \\ 0 \end{pmatrix} \tag{1.32}$$

and so the system $A\Pi x = b$ can be rewritten on $R\Pi x = Q^T b$, and than on

$$S_1 z = Q^T b \tag{1.33}$$

and so

$$\Pi x_{\mathrm{LS}} = Uz \tag{1.34}$$

Finally, the permutation should be inverted $x = u(p)$.

This way factorises a matrix on two orthogonal spaces described by U, V matrices, or more exactly

$$A = U \begin{pmatrix} T_{11}, & 0 \\ 0, & 0 \end{pmatrix} V^{-1} \tag{1.35}$$

where $V = \Pi Q$, and we are identifying $T_{11} = S_1^T$. The process is known as the complete orthogonal decomposition.

This framework gives exactly the same solution as well-known SVD, by significantly cheaper way: The number of operations required for a single QR run is $2n^3/3$ compared to $12n^3$ (GvL, p. 293). The drawback of SVD is the interactive algorithm and more deep matrix analysis — singular values (eigenvalues) and eigenvectors are hard to find. By the way, a natural choice for a quick solution of systems with a (potentially) rank-deficient matrix is the described complete orthogonal decomposition.

function $\mathsf{lsol}(L, q)$
  Let $L \in \mathcal{R}^{m\times n}$, $q, x \in \mathcal{R}^n$
  for $i = 1, \ldots, m$
    $x(i) = [q(i) - L(1{:}i-1, i)^T \cdot x(1{:}i-1)]/L(i,i)$
  end for
  return x
end function

Algorithm 1.11: Solution of $Lx = q$ with back-substitution

1.6 About a matrix inversion

Let $A \in \mathcal{R}^{m\times m}$ is a square matrix. The inverse matrix satisfies these conditions

$$AA^{-1} = I, \quad A^{-1}A = I. \tag{1.36}$$

function rank(R, τ)
    Let $R \in \mathcal{R}^{m \times n}, \tau > 0 \in \mathcal{R}$
    $r = 0$
    for $i = 1, \dots, m$
        if $|R(i,i)| < \tau$
            return r
        end if
        $r = i$
    end for
    return r
end function

Algorithm 1.12: Determine rank of a matrix

Let A is a full rank matrix. A way for computing of inverse matrix is a solution of series of linear equations with unit vectors. The unitary matrix I has unit vectors e_j in columns (rows). By the way, for j-th column of $a_j = A_j^{-1}$, we have this linear system

$$Aa_j = e_j, \qquad j = 1, \dots, n.$$

which can be solved for n times to fill $a_j = A_j^{-1}$. This is just a direct generalisation of methods mentioned in previous section 1.5.

Note that solving systems for coordinate vectors e_i removes the need for the matrix multiplication of right side: all elements e_j are zeros except the one in j-th row:

$$Q^T e_j = Q^T(:, j),$$

so the operation just selects the appropriate column of Q

The rank deficient problems, or problems with non-square matrix, can be maintained by similar way as we study in the solutions of Sect. 1.5.3. The result will no more satisfy conditions (1.36); they are replaced by the conditions:

$$AA^+A = A,\ A^+AA^+ = A^+,\ (AA^+)^T = AA^+,\ (A^+A)^T = A^+A.$$

Such matrix is known as pseudo-inverse matrix A^+ having minimal Frobenius norm

$$\|AA^+ - I\|_F.$$

The pseudo-inverse is very suitable to estimate of a covariance (sub-)matrix. The final algorithm 1.13 works for both full-rank and rank deficient systems.

function qrinv(A, A^{-1}, τ)
  Let $A, A^{-1}, A^{+} \in \mathcal{R}^{m\times n}, x, b \in \mathcal{R}^{m}, \tau > 1 \in \mathcal{R}$
  Auxiliary variables: $Q, U, W \in \mathcal{R}^{m\times n}, \Pi \in \mathcal{N}^{m\times m}, z, q, \beta, \beta' \in \mathcal{R}^{m}$
  $R, \beta, \Pi =$ qrpivot(A)
  $Q =$ qform(R, β)
  $r =$ rank($A, \tau\varepsilon|R(1,1)|$)
  if $r = m$
    for $i = 1, \ldots, m$
      $z =$ rsol($R, Q(i, :)$)
      $A^{-1}(:, i) = \Pi^T z$
    end for
    return A^{-1}
  else if $r < m$
    $S(1{:}n, 1{:}r), \beta'(1{:}r) =$ qrfac($R(1{:}r, 1{:}n)^T$)
    $U(1{:}n, 1{:}n) =$ qform($S(1{:}n, 1{:}r), \beta'(1{:}r)$)
    for $i = 1, \ldots, n$
      $z(1{:}r) =$ lsol($S(1{:}r, 1{:}r), Q(i, 1{:}r)$)
      $A^{+}(:, i) = \Pi^T U(:, 1{:}n) z$
    end for
    return A^{+}
  end if
end function

Algorithm 1.13: (Pseudo-)Inverse of a matrix

1.7 Eigenvalues and eigenvectors of a symmetric matrix

The framework of QR factorisation can be utilised also on a problem of determination of eigenvalues and eigenvectors of a symmetric square matrix $A \in \mathcal{R}^{n\times n}$. Eigenvalues λ are roots of a characteristics polynomial taken from determinant of a matrix

$$|A - \lambda I| = 0. \tag{1.37}$$

Counts of the roots is exactly the order of the polynomial n. The roots are real numbers $\lambda_i \in \mathcal{R}$ for symmetric real matrices.

An eigenvector $z_i \in \mathcal{R}^n$ is a vector, which is associated to every eigenvalue via a solution of the system of equations

$$Az_i = \lambda_i z_i, \quad i = 1, \dots n. \tag{1.38}$$

The eigenvectors are orthogonal each other $z_i \perp z_j$, $z_i^T \cdot z_j = 0$.

The characteristic polynomial has analytic solution only up to order three. All others roots should be, in principle, found by an iterative technique. That is the reason, why I am limiting only on symmetric problems having efficient methods developed; fortunately, only such kind of problems is encountered in field of optimisation.

The basic Algorithm 1.14 for determination of eigenvalues is on base of the power method (cite?): reiteration of QR factorisation. Eigenvalues are then computed by direct solution of (1.38); the system is singular by definition, so I use the parametric solution for the demonstration.

The drawbacks of the algorithm 1.14 are: the slow convergence, and the needs for additional eigenvectors computations. There is again an effective way how to compute both eigenvalues and vectors, like in case of QR factorisation. The algorithm, described in this section, computes both values and vectors simultaneously within a minimal number of iterations.

1.8 An effective algorithm for determination of eigenvalues and eigenvectors of a symmetric matrix on base of QR factorisation

Today convenient algorithms are on base of application of elementary rotations which nullifies off-diagonal elements: if the elemental rotation are applied repeatedly, they effective transforms an original matrix onto diagonal form:

$$D = Z^T A Z. \tag{1.39}$$

The matrices Z contains eigenvectors, while eigenvalues are on the diagonal. As for QR factorisation, it is possible to save a computer memory and store appropriate matrices in which are overwrote.

Let $A, Q, R, \Lambda, Z \in \mathcal{R}^{m \times n}$, $b \in \mathcal{R}^{n}$
$\Lambda = A$
for $k = 1, \dots$
  $Q, R = \mathsf{qrfac}(\Lambda)$
  $\Lambda = QR$
  $d = \max\{|\Lambda_{\div}|\}$, $\Lambda_{\div}$ denotes off-diagonal elements
  Finish when d is under a tolerance
end for
for $i = 1, \dots, n$
  $Q, R = \mathsf{qrfac}(A - \lambda_i I)$
  $Z_{ni} = \|\{R_{ii}, i = 1, \dots, n\}\|_2$
  for $j = n-1, \dots, 1$
    $Z_{ji} = -[R(j, j+1{:}n)^T \cdot Z(j, j+1{:}n)]/R_{jj}$
  end for
  $Z_i = Z_i / \|Z_i\|_2$, for a non-zero vector
end for
$\lambda_i = \{\Lambda_{ii}, i = 1, \dots, n\}$
return λ, Z

Algorithm 1.14: Outline for determination of eigenvalues and vectors

The elementary rotations — known as Givens transformation — are the matrices:

$$\begin{pmatrix} 1, & \cdots & 0, & \cdots & 0, & \cdots & 0, \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0, & \cdots & c, & \cdots & s, & \cdots & 0, \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0, & \cdots & -s, & \cdots & c, & \cdots & 0, \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0, & \cdots & 0, & \cdots & 0, & \cdots & 1 \end{pmatrix}. \tag{1.40}$$

If we know the coordinates x, y, the rotation, the angle in $-\pi \dots \pi$ can be determined as Algorithm 1.15 summarises.

1.8.1 Householder tri-diagonalisation

Every symmetric matrix can be transformed by application of Householder rotations onto tri-diagonal matrix. The transformation applies equivalent matrices which does not changes eigenvalues. If we are able to collect the transformations, we can determine eigenvectors. The tri-diagonalisation significantly simplifies consecutive iteration process via QR factorisation.

```
function givens(x, y)
    if |y| < ε
        c = 1, s = 0
    else
        if |y| > |x|
            τ = −x/y, s = 1/√(1 + τ²), c = sτ
        else
            τ = −y/x, c = 1/√(1 + τ²), s = cτ
        end if
    end if
    return c, s
end function
```

Algorithm 1.15: Givens rotation

The result is a symmetric tri-diagonal matrix and a matrix storing the transformations. The economy use of memory follows ideas developed for QR factorisation. The tri-diagonal matrix is can be stored in two vectors $a \in \mathcal{R}^n, b \in \mathcal{R}^{n-1}$:

$$\begin{pmatrix} a_1, & b_1, & 0, & 0 & \dots, & 0 \\ b_1, & a_2, & b_2, & 0 & \dots, & 0 \\ 0, & b_2, & a_3, & b_3, & \dots, & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0, & \dots, & 0, & b_{n-2} & a_{n-1}, & b_{n-1} \\ 0, & \dots, & 0, & 0 & b_{n-1}, & a_n \end{pmatrix}. \tag{1.41}$$

1.8.2 Wilkinson shift

Algorithm 1.17 is adapted for the tri-diagonal matrix discussed in 1.8.1. The key operation is the Givens rotation applied on the matrix:

$$\begin{pmatrix} c, & s \\ s, & c \end{pmatrix} \begin{pmatrix} T_{kk}, & T_{k,k+1} \\ T_{k+1,k}, & T_{k+1,k+1} \end{pmatrix} \begin{pmatrix} c, & s \\ -s, & c \end{pmatrix}.$$

The application make a new non-zero element out of the tri-diagonal; unfortunately, the immediate step will nullify then. The anomaly is moved from up to down and will vanish after the last operation, leaving clear tri-diagonal.

If the a, b representation is used, the non-zero element should by treated separately. The transformation

$$\begin{pmatrix} 1, & 0, & 0, & 0 \\ 0, & c, & -s, & 0 \\ 0, & s, & c, & 0 \\ 0, & 0, & 0, & 1 \end{pmatrix} \begin{pmatrix} a_{k-1}, & b_{k-1}, & z, & 0 \\ b_{k-1}, & a_k, & b_k, & 0 \\ z, & b_k, & a_{k+1}, & b_{k+1} \\ 0, & 0, & b_{k+1}, & a_{k+1} \end{pmatrix} \begin{pmatrix} 1, & 0, & 0, & 0 \\ 0, & c, & s, & 0 \\ 0, & -s, & c, & 0 \\ 0, & 0, & 0, & 1 \end{pmatrix} \tag{1.42}$$

function housetridig(A)
  Let $A, T \in \mathcal{R}^{n\times n}, a \in \mathcal{R}^n, b \in \mathcal{R}^{n-1}$,
  An auxiliary variables: $p, v, w, u, \beta \in \mathcal{R}^n$
  $T = A$
  for $k = 1, \dots, n-2$
    $v(k+1{:}n), \beta_k = \text{housevec}(T(k+1{:}n, k))$
    $p(k+1{:}n) = \beta_k T(k+1{:}n, k+1{:}n) \cdot v(k+1{:}n)$
    $r = \beta_k [p(k+1{:}n)^T \cdot v(k+1{:}n)]/2$
    $w(k+1{:}n) = p(k+1{:}n) - rv(k+1{:}n)$
    $T_{k+1,k} = T_{k,k+1} = \|T(k+1{:}n, k)\|_2$
    for $i = k+1, \dots, n, j = k+1, \dots, n$
      $T_{i,j} = T_{i,j} - v_i w_j - w_i v_j$
    end for
    if $k > 1$
      $T(k+1{:}n, k-1) = u(k+1{:}n)$
    end if
    $u(k+1{:}n) = v(k+1{:}n)$
  end for
  if $n > 2$
    $T_{n,n-2} = u_n$
  end if
  $a = \{T_{ii}, i = 1, \dots, n\}$
  $b = \{T_{i,i+1}, i = 1, \dots, n-1\}$
  $T(n-1{:}n, n-1{:}n) = I$
  for $k = n-2, \dots, 1$
    $v(k+1{:}n) = (1, T(k+2{:}n, k))^T$
    $T_{kk} = 1$
    $T(k+1{:}n, k) = T(k, k+1{:}n) = 0$
    $w(k+1{:}n) = \{[T(k+1{:}n, i)^T \cdot v(k+1{:}n)], i = k+1, \dots, n\}$
    $u(k+1{:}n) = \beta_k v(k+1{:}n)$
    for $i = k+1, \dots, n, j = k+1, \dots, n$
      $T_{i,j} = T_{i,j} - w_j u_i$
    end for
  end for
  Returns a, b, T
end function

Algorithm 1.16: Householder tri-diagonalisation

function qrstep(a, b, Q)
  Let $Q \in \mathcal{R}^{n\times n}, a \in \mathcal{R}^{n}, b \in \mathcal{R}^{n-1}$,
  $d = (a_{n-1} - a_n)/2$
  $\mu = a_n - b_{n-1}^2/[d + \text{sign}(d)\|(d, b_{n-1})\|_2]$
  $x = a_1 - \mu$
  $y = b_1$
  for $k = 1, \ldots, n-1$
    $c, s =$ givens(x, y)
    $u = a_k, v = a_{k+1}, w = 2csb_k$
    $a_k = c^2u + s^2v - w$
    $a_{k+1} = c^2v + s^2u + w$
    $b_k = (c^2 - s^2)b_k + cs(u - v)$
    $b_{k-1} = cx - sy$, if $k > 1$
    if $k < n-1$
      $x = b_k, y = -sb_{k+1}, b_{k+1} = cb_{k+1}$
    end if
    $Q(1{:}n, k{:}k+1) = Q(1{:}n, k{:}k+1)\begin{pmatrix} c, & s, \\ -s, & c \end{pmatrix}$
  end for
  Returns modified a, b, Q
end function

Algorithm 1.17: Implicit symmetric Francis QR step with Wilkinson shift leads to the resultant matrix

$$\begin{pmatrix} a_{k-1}, & cb_{k-1} - sz, & sb_{k-1} + zc, & 0 \\ cb_{k-1} - sz, & a'_k, & b'_k, & -sb_{k+1} \\ sb_{k-1} + cz, & b'_k, & a'_{k+1}, & cb_{k+1} \\ 0, & -sb_{k+1}, & cb_{k+1}, & a_{k+2} \end{pmatrix}, \tag{1.43}$$

where the updated elements of the tri-diagonal are

$$a'_k = c^2 a_k + s^2 a_{k+1} - 2csb_k, \tag{1.44}$$

$$a'_{k+1} = c^2 a_{k+1} + s^2 a_k + 2csb_k, \tag{1.45}$$

$$b'_k = (c^2 - s^2) b_k + cs(a_k - a_{k+1}). \tag{1.46}$$

We also see that there are conditions

$$b'_{k-1} = cb_{k-1} - sz, \qquad b'_{k+1} = cb_{k+1}. \tag{1.47}$$

1.8.3 A deflation

A selecting of suitable sub-matrix during iterative process is know as deflation or bounding. The off-diagonal elements of tri-diagonal matrix represented by b are nullified from the end or begin. An immediate matrix looks like

$$\begin{pmatrix} a_1, & < \varepsilon, & 0, & 0 & \dots, & 0 \\ < \varepsilon, & a_2, & b_2, & 0 & \dots, & 0 \\ 0, & b_2, & a_3, & b_3, & \dots, & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0, & \dots, & 0, & < \varepsilon & a_{n-1}, & < \varepsilon \\ 0, & \dots, & 0, & < \varepsilon & a_{n-1}, & < \varepsilon \\ 0, & \dots, & 0, & 0 & < \varepsilon, & a_n \end{pmatrix}. \tag{1.48}$$

With removed the already diagonalised leading and tailing sub-matrices, the computation speed can be improved, and numerical errors suppressed.

1.9 Bibliography notes

The textbook of Ralston & Rabinowitz (2012) gives an introduction into whole field of numerical mathematics including matrix decomposition itself. GvL is a comprehensive summary of general computation methods of linear algebra; Lapack computer library[5] is also based on the book.

There are many excellent on-line resources: blogs, documents or forums. Mr. Higham[6] provides the valuable collection of related articles on his website. Curses[7] by Mr. Arbenz was

[5] `http://www.netlib.org/lapack/index.html`.

[6] `https://nhigham.com/`.

[7] `https://people.inf.ethz.ch/arbenz/`.

```
function qreig(A, λ, Z)
    Let A, Q ∈ ℛ^{n×n}, λ, a ∈ ℛ^n, b ∈ ℛ^{n−1},
    a, b, Q = housetridig(A)
    l = 1
    m = n
    for k = 1, ...
        for i = 2, ..., n − 1
            δ(i) = ε(|a_i| + |a_{i+1}|)
        end for
        for i = 2, ..., n − 1
            if |b_i| > δ_i and |b_{i−1}| < δ_{i−1}
                l = i
            end if
            if |b_i| < δ_i and |b_{i−1}| > δ_{i−1}
                l = i
            end if
        end for
        if m > l
            qrstep(a(l:m), b(l:m − 1), Q(1:n, l:m − 1))
        end if
        Finish, if m − l = 1
    end for
    λ = a
    Returns modified λ, Q
end function
```

Algorithm 1.18: Determines eigenvalues and eigenvectors.

been very helpful for me during the implementation phase.

References

1. `https://en.wikipedia.org/wiki/John_G._F._Francis`.
2. `http://www.netlib.org/lapack/index.html`.
3. `https://en.wikipedia.org/wiki/QR_decomposition`.
4. `https://nhigham.com/`.
5. `https://people.inf.ethz.ch/arbenz/`.

GvL. Golub, G. & Van Loan, C. *Matrix Computations, 4th edition* (Johns Hopkins University Press, 2013).

6. Ralston, A. & Rabinowitz, P. *A First Course in Numerical Analysis: Second Edition* (Dover Publications, 2012).

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "**publisher**" means any person or entity that distributes copies of the Document to the public.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `https://www.gnu.org/licenses/`.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with … Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.