

## Matlab's Inverses of Hilbert Matrices

*Hilbert* matrices arise from least-squares fitting of polynomials to arbitrary functions over the interval  $0 \leq \tau \leq 1$  as follows:

Given a function  $y(\tau)$ , find coefficients  $\xi_j$  of polynomial  $p(\tau) := \sum_{0 \leq j < N} \xi_j \cdot \tau^j$  that minimize  $\int_0^1 (p(\tau) - y(\tau))^2 \cdot d\tau$ .

The minimizing coefficients form a column vector  $x := [\xi_0, \xi_1, \dots, \xi_{N-1}]^T$  that satisfies the

$$\text{Normal Equations} \quad H \cdot x = b$$

in which column  $b := [\beta_0, \beta_1, \dots, \beta_{N-1}]^T$  has elements  $\beta_i := \int_0^1 \tau^i \cdot y(\tau) \cdot d\tau$ , and matrix  $H$  has elements  $\theta_{ij} := \int_0^1 \tau^{i+j} \cdot d\tau = 1/(i+j+1)$  in row  $\#(i+1)$  and column  $\#(j+1)$ . This  $H$  is an  $N$ -by- $N$  *Hilbert* matrix. Matlab has a function to generate  $H$  from the statement  $H = \text{hilb}(N)$ :

```
function H = hilb(n)
%HILB Hilbert matrix.
% HILB(N) is the N by N matrix with elements 1/(i+j-1),
% which is a famous example of a badly conditioned matrix.
% See INVHILB for the exact inverse.
%
% This is also a good example of efficient MATLAB programming
% style where conventional FOR or DO loops are replaced by
% vectorized statements. This approach is faster, but uses
% more storage.
%
% C. Moler, 6-22-91.
% Copyright (c) 1984-96 by The MathWorks, Inc.
% $Revision: 5.4 $ $Date: 1996/08/15 21:52:09 $
%
% I, J and E are matrices whose (i,j)-th element
% is i, j and 1 respectively.

J = 1:n;
J = J(ones(n,1),:);
I = J';
E = ones(n,n);
H = E./(I+J-1);
```

Hilbert matrix  $H$  is “badly conditioned” because it is close to a singular matrix, much closer as  $N$  gets big. Consequently, tiny perturbations like those that occur when reciprocals  $1/(i+j-1)$  are rounded off cause the computed inverse of  $H$  to change drastically, more so as  $N$  gets big. To avoid those perturbations, function  $\text{hilbl}(N)$  computes  $L \cdot H$  for the smallest positive integer  $L$  such that all quotients  $L/(i+j-1)$  are integers computed exactly.

```
function [ H, L ] = hilbl(n,z)
% H = HILBL(N) is a scaled N-by-N hilbert matrix H(i,j) = L/(i+j-1)
% with scale-factor L so chosen that all entries are integers.
% H = HILBL(N, z) is another scaled N-by-N hilbert matrix with
% elements H(i,j) = L/(i+j-1+z) for any nonnegative integer z .
% Restriction: 2N+z < 44 , to ensure that no rounding error occurs.
% See also HILB and INVHILB. W. Kahan, 9 March 1996
if ( nargin < 2 ), z = 0 ; end
```

```

if (z < 0) | (z ~= round(z)),
    error( ' HILBL(N, z) requires a nonnegative integer z .' )
end
L = 2*n+z-1 ; if ( L > 42 ),
    error( ' HILBL(N, z) will not run with 2N+z > 43 .' )
end % It's the best I care to do without an Inexact flag.
% --- 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,...
LLLL = [1,2,3,2,5,1,7,2,3, 1,11, 1,13, 1, 1, 2,17, 1,19, 1, 1, 1,23,...
        1, 5, 1, 3, 1,29, 1,31, 2, 1, 1, 1, 1,37, 1, 1, 1,41, 1] ;
% --- 24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42.
LLLL = cumprod(LLLL(1:L)) ; % LLLL(i) = LCM(1:i) for i = 1:L .
% ( LCM(1:43) would change when rounded to 53 sig. bits.)
L = LLLL(L) ;
x = [1:n] ; y = (x+(z-1)).' ;
H = L./( x(ones(n,1),:) + y(:,ones(1,n)) ) ;

```

In principle,  $H^{-1} = L(L \cdot H)^{-1}$  could be computed using Matlab's `inv(...)` function, but this incurs rounding errors that cause about as much damage as would rounding off  $H$ 's elements. To avoid that damage, a special function `invhilb(N)` can be used instead of `inv(hilb(N))`:

```

function H = invhilb(n)
%INVHILB Inverse Hilbert matrix.
% INVHILB(N) is the inverse of the N by N matrix with elements
% 1/(i+j-1), which is a famous example of a badly conditioned
% matrix. The result is exact for N less than about 15.
%
% See also HILB.
%
% C. Moler, 4-30-87.
% Copyright (c) 1984-96 by The MathWorks, Inc.
% $Revision: 5.3 $ $Date: 1996/08/15 21:52:11 $

p = n;
H = zeros(n);
for i = 1:n
    if i > 1, p = ((n-i+1)*p*(n+i-1))/(i-1)^2; end
    r = p*p;
    H(i,i) = r/(2*i-1);
    for j = i+1:n
        r = -((n-j+1)*r*(n+j-1))/(j-1)^2;
        H(i,j) = r/(i+j-1);
        H(j,i) = r/(i+j-1);
    end
end
end

```

As supplied by Matlab, the foregoing program's claim "exact for  $N$  less than about 15" is not quite right; it may have been valid for some old machine with wider floating-point precision than is standard nowadays. Now the correct claim is "exact for  $N \leq 12$ ." For larger dimensions  $N$  the elements of  $H^{-1}$ , though all integers, are so big that they have to be rounded off to different integers that fit into 53 sig. bits.

The algorithm used by `invhilb(...)` is based upon the observation that  $H^{-1} = DHD$  for a diagonal matrix  $D$ . It can be computed in several ways; here is another that is more general:

```

function W = invhilb(N, z)
% INVHILB(N, z) is the inverse of the N-by-N matrix H = HILB(N, z)
%   whose elements are H(i,j) = 1/(i+j-1+z) . If omitted, z = 0 ;
%   then H is the notoriously ill-conditioned hilbert matrix.
%   INVHILB is computed not from inv(H) but from a comparatively
%   simple formula, published by Sam Schechter in MTAC (1959),
%   that gives accurate results despite ill-condition. If z is a
%   nonnegative integer, so are all elements of abs(INVHILB(N, z))
%   except for roundoff, which interferes only for (N, z) beyond
%   (12,0) - (12,2) - (11,5) - (10,7) - (9,10) - (8,16) - (7,19) -
%   (6,39) - (5,67) - (4,165) - (3,573) - (2,13774) - (1, 94906265)
%   when computed to 53 sig. bits in IEEE 754 Double ( REAL*8 ).
%
%                                     (C)   W. Kahan, Feb. 19, 1994
if ( nargin < 2 ), z = 0 ; end
x = [0:N-1] ; u = z+1 ; y = (x+u).' ;
if ( z == 0 ) ,
    p = N ; % ... The most common case is the fastest:
    elseif ((0 < z) & (z == round(z)) & (z < N)),
        p = N*(N+1) ; for j = 2:z , p = (p/j)*(N+j) ; end
    else, % ... The most general case is the slowest:
        p = u ; for j = 1:N-1, p = (p/j)*(u+j) ; end
    end
u = [ p , zeros(1,N-1) ] ;
for j = 1:N-1, u(j+1) = (( u(j)/(j+z))*(j-N) )/j)*(N+j+z) ; end
W = (u.'*u)./( x(ones(N,1),:) + y(:,ones(1,N)) ) ;
if ((0 <= z) & (z == round(z))), W = round(W) ; end

```

Let  $W$  be an estimate of  $H^{-1}$  computed somehow. How can we ascertain  $W$ 's (in)correctness using arithmetic no more accurate than was used to compute  $W$ ? When its elements should all be integers, a non-integer is an error revealed in Matlab by computing  $W - \text{round}(W)$ . If this turns out to be all zeros, further tests must be performed to ascertain whether  $W$ 's elements are the correct integers. What tests will not be confounded by roundoff?

Let the elements of  $M$  be the reciprocals, all small integers, of the corresponding elements of  $H$ ; in Matlab compute  $M = \text{round}(1.0 ./ H)$ . (The  $\text{round}(\dots)$  operation is required to clean out rounding errors only for  $N > 24$ .) Then  $v = w./M$  should have elements all exactly integers, and therefore computed exactly by Matlab, if the alleged inverse  $W$  is correct; and then  $\text{sum}(v)$  should turn out to be a row of 1's, each also computed exactly as a sum of integers of alternating signs. Do you see why roundoff does not contaminate these sums?

**Exercise:** Use the foregoing tests to compare different estimates for  $H^{-1}$  computed by Matlab from each of

$$\text{inv}(\text{hilb}(N)) , \quad \text{round}(\text{inv}(\text{hilb}(N))) , \quad L * \text{inv}(\text{hilb}(N)) , \\ \text{round}(L * \text{inv}(\text{hilb}(N))) , \quad \text{and} \quad \text{invhilb}(N)$$

for  $N = 1, 2, 3, \dots$  in turn to determine when each kind of estimate breaks down.

Recall that the singular matrix  $S$  nearest  $H$  is distant from it by  $\|H-S\| = 1/\|H^{-1}\|$ . (This is proved as Theorem 7 in the note "Huge Generalized Inverses of Rank-Deficient Matrices", <http://www.cs.berkeley.edu/~wkahan/MathH110/GI1ite.pdf>). The norm  $\|\dots\|$  here is Matlab's  $\text{norm}(\dots)$ . Even a rough computation of  $H^{-1}$  reveals how near  $H$  is to singular; any estimate  $W \approx H^{-1}$  provides an estimate  $\|H-S\| \approx \|W\|$ . How widely can estimates  $W$  vary?

Programs like Matlab's `inv(...)` compute inverses by Gaussian Elimination or, equivalently, triangular factorization in which roundoff causes  $\text{inv}(H)$  to differ from  $H^{-1}$  by not much more than  $(H+\Delta H)^{-1}$  can differ if  $\|\Delta H\| \approx N \cdot \varepsilon \cdot \|H\|$ , although  $\text{inv}(H) \neq (H+\Delta H)^{-1}$  for any such small perturbation  $\Delta H$ . Here  $\varepsilon = \text{eps} = 1/2^{52}$  is Matlab's roundoff threshold. In other words, although a computed  $w = \text{inv}(H)$  is not necessarily the inverse of any matrix  $H+\Delta H$  differing from  $H$  by at most about  $N$  rounding errors in each element of  $H$ , the computed inverse  $W$  is not much farther from  $H^{-1}$  than the inverse  $(H+\Delta H)^{-1}$  of such a perturbed  $H$ . And

$$(H+\Delta H)^{-1} - H^{-1} = -H^{-1} \cdot \Delta H \cdot (H+\Delta H)^{-1}, \text{ so}$$

$$\|(H+\Delta H)^{-1} - H^{-1}\| \leq \|H^{-1}\| \cdot \|\Delta H\| \cdot \|(H+\Delta H)^{-1}\| \approx N \cdot \varepsilon \cdot \|H^{-1}\| \cdot \|H\| \cdot \|(H+\Delta H)^{-1}\|.$$

Therefore any estimate  $W$  of  $H^{-1}$  obtained from `inv(...)` should satisfy roughly

$$\|W - H^{-1}\| \leq N \cdot \varepsilon \cdot \|H\| \cdot \|W\|^2 \text{ provided } N \cdot \varepsilon \cdot \|H\| \ll 1/\|W\| \approx \|H-S\|.$$

The last proviso says “ $H$  is much farther than  $N$  rounding errors from the nearest singular matrix  $S$ .”

But if  $\|H-S\|$  is not much bigger than  $N \cdot \varepsilon \cdot \|H\|$ , then  $w = \text{inv}(H)$  may well be computed far too inaccurately for the bound upon  $\|W - H^{-1}\|$  above to be trusted. Usually Matlab issues a

Warning: Matrix is close to singular or badly scaled.

Results may be inaccurate. RCOND = 3.572099e-17.”

The displayed value  $\text{RCOND} = 1/(\|H\| \cdot \|W\|)$  is Matlab's estimate of  $\|H-S\|/\|H\|$ , but it may be wrong by an order of magnitude because  $\|H+\Delta H - S\|$  may vary by that much or more as  $\Delta H$  varies by several rounding errors per element of  $H$ . Consequently  $\|(H+\Delta H)^{-1}\| = 1/\|H+\Delta H - S\|$  may vary that much, and therefore so may  $\|W\|$ , and consequently so may  $W$ , depending upon the rounding errors occurring during its computation.

The last “consequently ...” follows from the observation that the norm of a matrix changes by no more than the norm of its change:  $|\|W+\Delta W\| - \|W\|| \leq \|\Delta W\|$ ; can you see why?

In summary, if  $H$  is too nearly singular, different computations may yield estimates  $W$  of  $H^{-1}$  that look very different. However, provided all the estimates are no worse than  $(H+\Delta H)^{-1}$  for roundoff-like perturbations  $\Delta H$ , the diverse estimates  $W$  usually have this in common:

*They are all nearly scalar multiples one of another.*

This is explained on pp. 3-4 of the note “Jacobi's Formula for the Derivative of a Determinant”,

<http://www.cs.berkeley.edu/~wkahan/MathH110/JACOBI.pdf>.

An alternative explanation in terms of the singular values of  $H+\Delta H$  is more illuminating but requires a little more technical machinery; it's a story for another day.

**Exercise:** Use different formulas, like those near the bottom of the previous page, to compute diverse estimates  $W_1, W_2, \dots$  for the inverses of  $N$ -by- $N$  Hilbert matrices with  $N$  big enough to elicit Matlab's “close to singular” warning, and display their elementwise ratios  $w_i./w_j$  in Matlab's `format short` to see how nearly these estimates are scalar multiples of each other.