Examples:

Hilbert Matrix H with $H_{ij} = \frac{1}{i+j-1}$

Pascal Matrix P with $p_{ij} = \binom{i+j-2}{i-1}$

```
julia> H5=[1/(i+j-1) for i=1:5,j=1:5]
5×5 Matrix{Float64}:
 1.0       0.5       0.333333  0.25      0.2
 0.5       0.333333  0.25      0.2       0.166667
 0.333333  0.25      0.2       0.166667  0.142857
 0.25      0.2       0.166667  0.142857  0.125
 0.2       0.166667  0.142857  0.125     0.111111
```

5x5 Hilbert matrix

sum of these

$\|H_5\| = \max_j \sum_{i=1}^{5} |a_{ij}|$

```
julia> 1+1/2+1/3+1/4+1/5
2.283333333333333

julia> opnorm(H5,1)
2.283333333333333
```

$$\frac{\|x - x^*\|}{\|x\|} \lesssim 10^k \cdot 10^{-15} = 10^{k-15}$$

If $\kappa(A) \approx 10^k$ then we can only see $15-k$ significant digits of x using residual error...

```
julia> using LinearAlgebra

julia> cond(H5)
476607.25024259434
```

$\approx 10^5$

expect to lose about 5 digits precision...

Idea: solve a problem for which we know the answer as a test.

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad \text{Set } b = H_5 x$$

```
julia> x=ones(5)
5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0
```

```
julia> b=H5*x
5-element Vector{Float64}:
 2.283333333333333
 1.45
 1.0928571428571427
 0.8845238095238095
 0.7456349206349207
```

Solve $H_5 x = b$ using Gaussian elimination. The algorithm w/partial pivoting is built in to julia (also Matlab) and called left division.

$$x = H_5 \backslash b$$

left matrix division operator
it does Gaussian elimination (if the matrix is square).

```
julia> xapp=H5\b
5-element Vector{Float64}:
 0.9999999999999578
 1.0000000000006084
 0.9999999999978405
 1.0000000000027782
 0.9999999999988143
```

digits are wrong...

```
julia> H10=[1/(i+j-1) for i=1:10,j=1:10];

julia> x=ones(10);

julia> b=H10*x;

julia> xapp=H10\b
10-element Vector{Float64}:
 1.0000000004766838
 0.9999999590501241
 1.0000008692519378
 0.9999921141445542
 1.0000375643596116
 0.999896823042628
 1.0001691902590395
 0.9998365530003722
 1.000085790575106
 0.9999811355093464
```

← 13 digits are wrong...

```
julia> cond(H10)
1.6024416992541715e13
```

This is big cond # for such a small matrix...

```
julia> cond(H10)
1.6024416992541715e13

julia> bapp=H10*xapp;

julia> norm(bapp-b)/norm(b)
1.224858843104039e-16
```

The relative residual error is on the order 10^{-15} which is the best one can hope for with double precision.

In particular, even if this were zero... all it means is that the relative residual error is within 10^{-15} of 0.

Try Pascal's matrix

```
julia> P5=[binomial(i+j-2,i-1) for i=1:5,j=1:5]
5×5 Matrix{Int64}:
 1  1   1   1   1
 1  2   3   4   5
 1  3   6  10  15
 1  4  10  20  35
 1  5  15  35  70
```

$\|P_5\|_1 = \max_j \sum_{i=1}^{m} |A_{ij}|$

Largest sum

```
julia> 1+5+15+35+70
126

julia> opnorm(P5,1)
126.0
```

```
julia> cond(P5)
8517.524361138405
```

10^4

```
julia> b=P5*x
5-element Vector{Float64}:
   5.0
  15.0
  35.0
  70.0
 126.0
```

```
julia> xapp=P5\b
5-element Vector{Float64}:
 1.0
 1.0
 1.0
 1.0
 1.0
```

this is the same as x

```
julia> P10=[binomial(i+j-2,i-1) for i=1:10,j=1:10]
10×10 Matrix{Int64}:
 1   1   1    1    1     1     1      1      1      1
 1   2   3    4    5     6     7      8      9     10
 1   3   6   10   15    21    28     36     45     55
 1   4  10   20   35    56    84    120    165    220
 1   5  15   35   70   126   210    330    495    715
 1   6  21   56  126   252   462    792   1287   2002
 1   7  28   84  210   462   924   1716   3003   5005
 1   8  36  120  330   792  1716   3432   6435  11440
 1   9  45  165  495  1287  3003   6435  12870  24310
 1  10  55  220  715  2002  5005  11440  24310  48620
```

```
julia> x=ones(10);

julia> b=P10*x;

julia> xapp=P10\b
10-element Vector{Float64}:
 0.9999999965777964
 1.0000000276874015
 0.9999998993027817
 1.0000002156453363
 0.9999997008425456
 1.0000002784245863
 0.9999998263305007
 1.0000000699546696
 0.9999999834984266
 1.0000000017359554
```

10 digits wrong

```
julia> cond(P10)
4.155205690095196e9
```

almost 10^{10}

How to compute condition number $\kappa(A) = \|A\| \, \|A^{-1}\|$ without finding A^{-1}?

Easier with Euclidean norm ...

How to find $\|A\|_2 = \max\{\sigma_i : \sigma_i \text{ is a singular value of } A\}$

$= \max\{\lambda_i^{1/2} : \lambda_i \text{ is an eigenvalue of } A^T A\}.$

$\|A^{-1}\|_2 = \max\{\lambda_i^{1/2} : \lambda_i \text{ is an eigenvalue of } (A^{-1})^T A^{-1}\}$

Goal: compute largest eigenvalue of $A^T A$ and $(A^{-1})^T A^{-1}$.