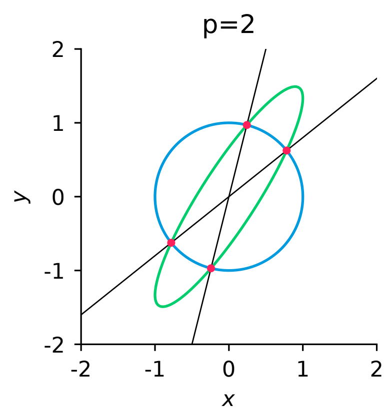
AM205 HW2. Numerical linear algebra. Solution

P1. Equations with vector norms

See solution code in [p1_norms.py].

(a)



When b is written as (x, y) $\|b\|_2 = 1$ would require $x^2 + y^2 = 1$ and $\|Ab\|_2 = 2$ would imply $(4x - 3y)^2 + (2x)^2 = 4$ equivalent to $20x^2 + 9y^2 - 24xy = 4$. We can subtract 4 times first equation from the second equation and obtain

$$16x^2 - 24xy + 5y^2 = 0.$$

Using the quadratic formula, we can find x as function of y. First solution would be

$$x = \frac{(24y + \sqrt{576y^2 - 320y^2})}{32}$$

$$x = \frac{24y + \sqrt{256}y}{32} = \frac{24 + \sqrt{256}}{32} = \frac{40}{32}y = \frac{5}{4}y$$

The other solution is

$$x = \frac{24 - \sqrt{256}}{32}y = \frac{1}{4}y$$

Now, let's go back to equation $x^2 + y^2 = 1$.
Using the second solution would require:

$$\frac{1}{16}y^2 + y^2 = 1$$

$$\frac{17}{16}y^2 = 1$$

$$y = \pm\frac{4}{\sqrt{17}}$$

Meaning two solutions obtained when $x = \frac{1}{4}y$ are

$$(\frac{1}{\sqrt{17}}, \frac{4}{\sqrt{17}}), (-\frac{1}{\sqrt{17}}, -\frac{4}{\sqrt{17}})$$

When $x = \frac{5}{4}y$,

$$\frac{25}{16}y^2 + y^2 = 1$$

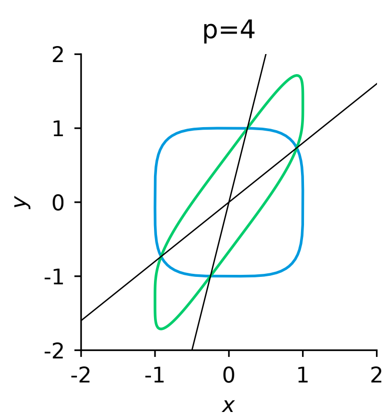
$$\frac{41}{16}y^2 = 1$$

$$y = \pm\frac{4}{\sqrt{41}}$$

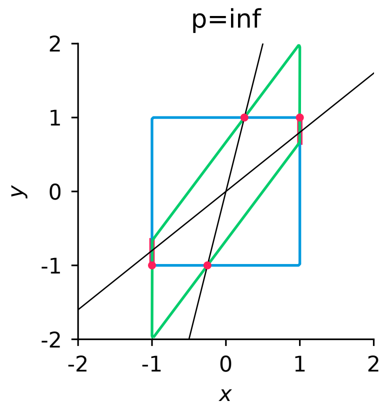
And the solutions are

$$(\frac{5}{\sqrt{41}}, \frac{4}{\sqrt{41}}), (-\frac{5}{\sqrt{41}}, -\frac{4}{\sqrt{41}})$$

(b)



(c)



Writing b as $b = (x, y)$. $\|b\|_\infty = 1$ implies either $|x| = 1$ and $|y| \leq 1$ or $|x| \leq 1$ and $|y| = 1$. If we consider first case,

$$2 = \|Ab\|_\infty = \max\{|4x - 3y|, |2x|\} = \max\{|4x - 3y|, 2\}$$

This would imply it is necessary for $|4x - 3y| \leq 2$.

When x and y have opposite sign, solution won't exist because $|4x - 3y| \geq 4 > 2$. If $x = 1$, then $-2 \leq 4 - 3y \leq 2$. Solving this would require: $\frac{2}{3} \leq y \leq 2$, but recall that the absolute value of y can't be greater than 1. This gives us $x = 1, \frac{2}{3} \leq y \leq 1$ as solution. Similary when we consider $x = -1$, we get $x = -1, -1 \leq y \leq -\frac{2}{3}$. If we consider the second case, we get

$$2 = \|Ab\|_\infty = \max\{|4x - 3y|, |2x|\}$$

When x and y have opposite sign, solution won't exist because $|4x - 3y| \geq 3 > 2$. So, let's consider case when x and y have same sign. If $|4x - 3y| = 2$ this would require either $|4x| = 5$ or $|4x| = 1$. But x can't be greater than 1 so, first two possible solution is $(0.25, 1)$ and $(-0.25, -1)$. Another possible case is when $|2x| = 2$ and solutions satisfying that conditions are $(1, 1)$ and $(-1, -1)$

(d) For arbitrary p, it would require

$$|x|^p + |y|^p = 1$$

$$|4x - 3y|^p + |2x|^p = 2^p$$

The first equation can be multiplied by 2^p and then be subtracted from the second equation. This would result in

$$|4x - 3y|^p = |2y|^p$$

This implies either $4x - 3y = 2y$ or $4x - 3y = -2y$ Which means it satisfies either $x = \frac{y}{4}$ or $x = \frac{5}{4}y$ Using this, let's plug it back into $|x|^p + |y|^p = 1$ For first case,

$$(\frac{1}{4}^p + 1)|y|^p = 1$$

as p goes toward ∞, this would become

$$y^{\infty} = 1,$$

in this case $y = \pm 1$ so $(0.25, 1)$ and $(-0.25, -1)$ are on this line. For second case,

$$(\frac{5}{4}^{p} + 1)|y|^{p} = 1$$

$$|y| = (\frac{1}{\frac{5}{4}^{p} + 1})^{\frac{1}{p}}$$

and as p goes toward ∞,

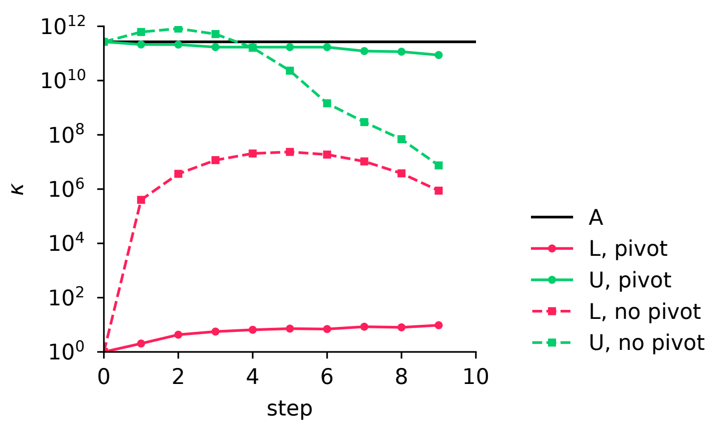
$$y = \pm\frac{4}{5}$$

so this passes through $(1, \frac{4}{5})$ and $(-1, -\frac{4}{5})$ on the solution. So, in this case limit as p goes toward ∞, solution differs from solution at ∞ norm.

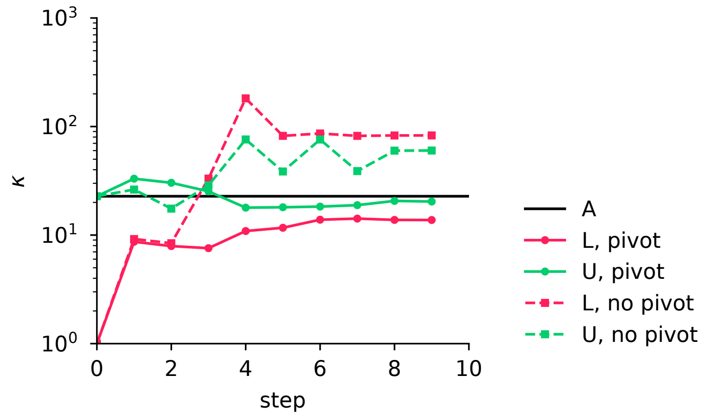
P2. Condition number of LU factorization

See solution code in [p2_lu.py].

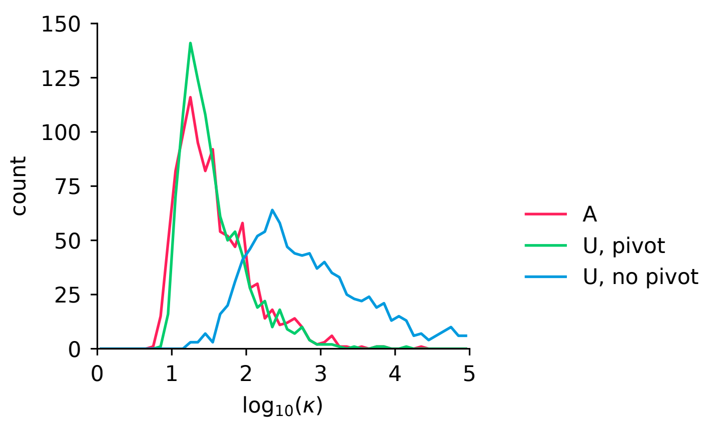
(b) Condition numbers $\kappa(A)$ and $\kappa(U)$ after each step of the factorization for the Vandermonde matrix. With pivoting, $\kappa(U)$ remains small and $\kappa(L)$ does not change from its initial value $\kappa(A)$. Without pivoting, $\kappa(U)$ rapidly increases after the first step and does not change much afterwards, while $\kappa(L)$ steadily decreases.



(c) Condition numbers $\kappa(A)$ and $\kappa(U)$ after each step for the pseudorandom matrix. With pivoting, both $\kappa(U)$ and $\kappa(L)$ do not exceed κA. Without pivoting, both grow above $\kappa(A)$. This example confirms the trend observed in part **(c)**, where the algorithm without pivoting tends to result in larger condition numbers.



(d) Histogram of the condition number. The most probable values are: $\kappa(A) = 17.783$, $\kappa(U) = 17.783$ with pivoting, and $\kappa(U) = 223.87$ without pivoting. The algorithm with pivoting is expected to provide a smaller condition number, which is consistent with the observations.



P3. Sparse linear algebra

See solution code in [p3_sparse.py].

(a) See function `mul_dense()` for dense matrix-vector multiplication, function `mul_csr()` for matrix-vector multiplication stored in CSR format, function `mul_csc()` for CSC format.

(b) See function `mul_csr_csc()`.

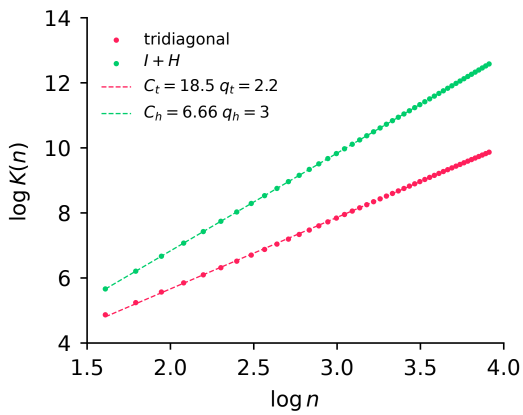
(c) See function `elim_csr()`.

(d) See function `sparse_lu()`.

(e) In functions `elim_csr()` and `mul_csr_csc()` we add a global variable to count the flops when implementing `sparse_lu()` for both the tridiagonal matrices and $I + H$ matrices. Below is the log-log plot of the number of operations $K(n)$ as a function of n (dots), and the fitted straight line $\log C + q \log n \approx \log K(n)$ (solid lines), with fitted parameters

$$\text{Tridiagonal matrix: } C_t = 18.5, \quad q_t = 2.2$$
$$I + H : C_h = 6.66, \quad q_h = 3$$

which means the total number of flops is slightly larger than $O(n^2)$ for the $n \times n$ tridiagonal matrix , and amounts to $O(n^3)$ for the $n \times n$ matrix $I + H$.



P4. Unstable LU factorization

(a) See symbolic computations in [p4_lu_sympy.py].

$$G(4,c) = \begin{bmatrix} c & 0 & 0 & c \\ -1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

Case $0 < c < 1$. Partial pivoting at step j selects row $j + 1$.

$$U_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & c & 0 & c \\ 0 & -2 & 1 & 0 \\ 0 & -2 & -1 & 0 \end{bmatrix} \quad U_2 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & \frac{c}{2} & c \\ 0 & 0 & -2 & 0 \end{bmatrix} \quad U_3 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -2 & 1 & 0 \\ 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & c \end{bmatrix}$$

Case $c > 1$. Partial pivoting at step j selects row j.

$$U_1 = \begin{bmatrix} c & 0 & 0 & c \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & -1 & -1 & 1 \end{bmatrix} \quad U_2 = \begin{bmatrix} c & 0 & 0 & c \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & -1 & 2 \end{bmatrix} \quad U_3 = \begin{bmatrix} c & 0 & 0 & c \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

(b)

$$G(n,c) = \begin{bmatrix} c & 0 & 0 & \dots & 0 & c \\ -1 & 1 & 0 & \dots & 0 & 0 \\ -1 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & & \ddots & \vdots & \vdots \\ -1 & -1 & -1 & \dots & 1 & 0 \\ -1 & -1 & -1 & \dots & -1 & 0 \end{bmatrix}$$

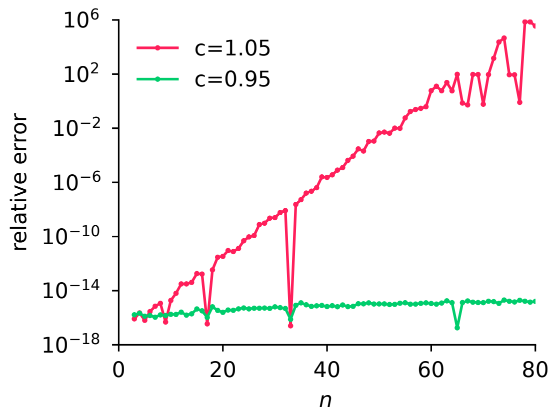
Case $0 < c < 1$. Elements in the last column do not grow.

$$U = \begin{bmatrix} -1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & 0 & 0 & \dots & -2 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & c \end{bmatrix}$$

Case $c > 1$. Elements in the last column grow exponentially, and the matrix becomes ill-conditioned. For example, the last two rows approach “linear dependency”, i.e. the cosine of the angle between them $2^{2n-5}/\sqrt{(1+4^{n-3})4^{n-2}} \approx 1$ for large n.

$$U = \begin{bmatrix} c & 0 & 0 & \dots & 0 & c \\ 0 & 1 & 0 & \dots & 0 & 1 \\ 0 & 0 & 1 & \dots & 0 & 2 \\ \vdots & & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 2^{n-3} \\ 0 & 0 & 0 & \dots & 0 & 2^{n-2} \end{bmatrix}$$

(c, d) See solution code in [p4_lu_unstable.py]. In the case $c = 1.05$, the relative error rapidly increases with n and reaches 100% for $n \approx 60$. This corresponds to $c > 1$ where the elements in the last column of U increase exponentially while the diagonal element is 1, which leads to loss of precision.



P5. QR factorization applied to a bouncing ball

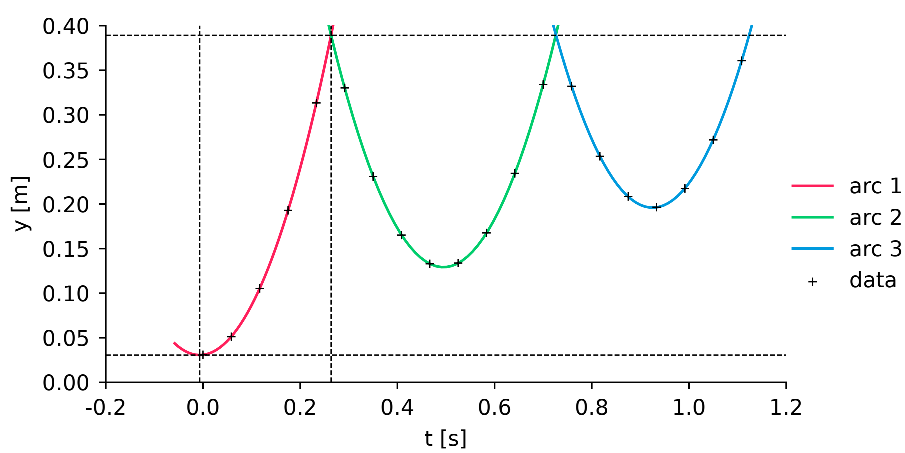
See solution code in [p5_ball.py].

(a) The function givens_QR() implements the QR factorization using Givens rotations. Note that the function does not explicitly construct a full Givens rotation matrix, but rather implements the required multiplications by modifying specific rows of R or columns of Q.

(b) The obtained coefficients for each parabolic arc are:

	α [m/s^2]	β [m/s]	γ [m]
arc 1	4.90863	0.06586	0.03083
arc 2	4.86767	-4.81635	1.32054
arc 3	4.89995	-9.06585	4.38929

The fits plotted together with the data points. The dashes lines show the estimate time and position of the ball's release and the first bounce, required in part **(c)**.



(c) Estimates of the gravitational acceleration, given as 2α:

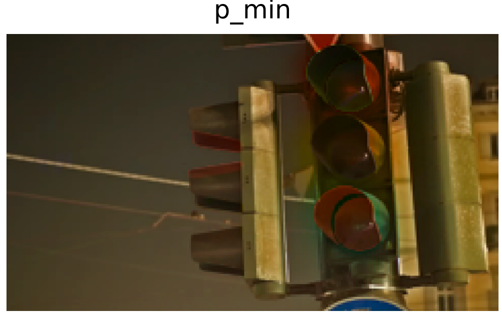
	g [m/s^2]
arc 1	9.81726
arc 2	9.73535
arc 3	9.79990

The minimum of arc 1 provides the time and position of the ball's release: $t = -0.00671$ s, $y = 0.03061$ m. The intersection of arcs 1 and 2 provides the time and position of the first bounce: $t = 0.26358$ s, $y = 0.38922$ m. The difference between the two positions gives the distance from the table top to the ball bottom: 0.35861 m. No need to subtract the radius: the difference would be zero if the ball was released exactly from the table.

P6. Traffic light images from PCA

The program [p6_pca.py] loads the 64 images and calculates $p_{\min}$, and performs PCA on those images.

(a) The following image corresponds to $p_{\min}$.



(b) The second row in Figure 1 shows the first three principal components.

(c) To obtain F, we need to solve the normal equations,

$$V(q_k - V^T F^T g_k) = 0, \quad k = 1, 2, 3.$$

where V consists of orthonormal rows, i.e. $VV^T = I$. They can be written in a more compact form

$$VQ^T = F^T G,$$

where $Q \in \mathbb{R}^{3\times n}$ has rows q_1, q_2, q_3 and $G \in \mathbb{R}^{3\times 3}$ has rows g_1, g_2, g_3. Therefore F can be expressed as

$$F = (G^{-1})^T (Q^T V),$$

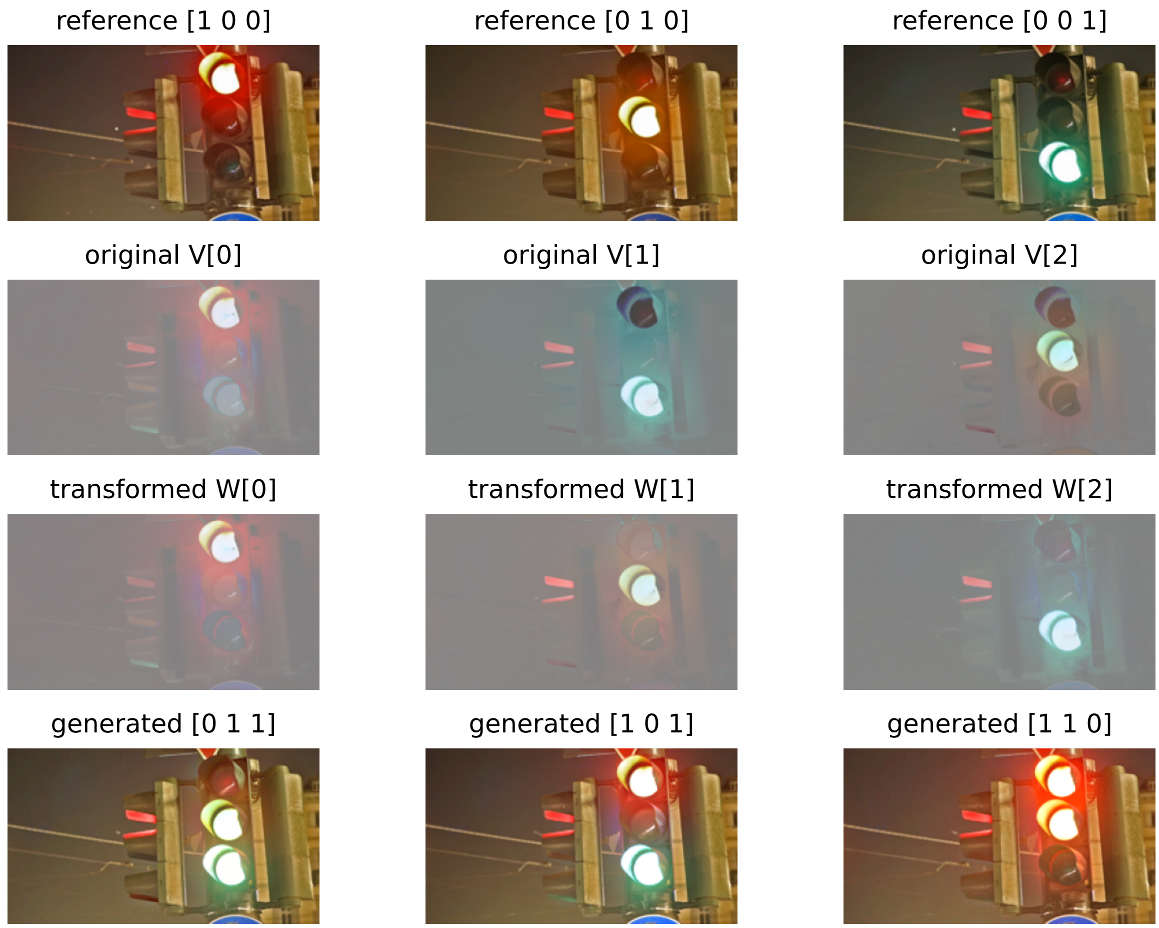


Figure 1: First row: Image 0, 37, and 5 used for calibration. Second row: the first three principal component after normalization; Third row: the first three principal component after transformation (after normalization). Fourth row: three new generated states $g = (0, 1, 1)$, $(1, 0, 1)$, and $(1, 1, 0)$.

which evaluates to

$$F = \begin{bmatrix} -47.459 & -11.554 & -1.366 \\ -29.877 & 2.796 & -37.817 \\ -24.368 & 38.137 & -0.094 \end{bmatrix}.$$

Then W is computed as $W = FV$. The third row in Figure 1 shows the first three transformed principal components w_k ($k = 1, 2, 3$) after normalization.

(d) The fourth row in Figure 1 shows the images generated from the states $g = (0, 1, 1)$, $(1, 0, 1)$, and $(1, 1, 0)$.