Announcements

1) Job candidates arriving next week, please try to attend their talks, especially if you are in the master's program in applied math!

Recall: flops

One flop corresponds to one of the operations $+, -, \times, \div$, or $\sqrt{\ }$.

Definition: (cost) The cost of an algorithm is the number of flops used in its execution.

Example 1: Lets calculate the number of flops used in the classical Gram-Schmidt algorithm for $A \in \mathbb{C}^{m \times n}$,

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}.$$

We have $v_j = a_j$ at zero flops.

The cost of

$$r_{jj} = \|v_j\|_2$$

is $2m$ flops since $v \in \mathbb{C}^m$ (we figured this out last class)

The cost of

$$q_i = v_j / r_{j,j}$$

is m flops. Together this is $3m$ flops, and run ($j=1$ to n) times gives

$3mn$ flops.

Now for the "inner loop".

We have

$$r_{ij} = q_i^* a_j$$

at a cost of $2m-1$ flops since $q_i, a_j \in \mathbb{C}^m$ and the inner product involves m mutiplications and $(m-1)$ additions.

Then

$$v_j = v_j - r_{i,j} q_i$$

is a cost of m for $r_{i,j} q_i$ since $r_{i,j} \in \mathbb{C}$ and m for subtracting the m-vectors v_j and $r_{i,j} q_i$, so $2m$ total.

Then for each i and j,

we have

$2m + (2m-1) = 4m-1$

flops. Finally,

running from $j=1$ to n

and $i=1$ to $j-1$

gives

$$\sum_{j=1}^{n} \sum_{i=1}^{j-1} (4m-1)$$

We can rewrite this as

$$(4m-1)\sum_{j=1}^{n}\sum_{i=1}^{j-1} 1$$

$$=(4m-1)\sum_{j=1}^{n}(j-1)$$

$$=(4m-1)\frac{n(n-1)}{2}$$

Distributing, we get

$$2mn^2 - 2mn - \frac{n^2}{2} + \frac{n}{2} \text{ flops}$$

Finally, adding all the costs together, we get

$$\left(2mn^2 - 2mn - \frac{n^2}{2} + \frac{n}{2}\right) + 3mn$$

$$= 2mn^2 - \frac{n^2}{2} + \frac{n}{2} + mn$$

flops!

Asymptotics

We figured out the cost of CIgs algorithm to be

$$2mn^2 + nm - \frac{n^2}{2} + \frac{n}{2}.$$

If we take the limit as $n, m \to \infty$, which term wins?

Dividing the cost by $2mn^2$, we get

$$1 + \frac{1}{2n} - \frac{1}{4m} + \frac{1}{4mn}$$

$$\longrightarrow 1 \quad \text{as } m, n \to \infty.$$

The effective cost for this algorithm is

$$\boxed{2mn^2}$$

Stability

Using Matlab and the example on page 67 in the text, we see that Matlab's QR factorization is 5 orders of magnitude better than either our cgs or mgs algorithms. What is Matlab doing?

Householder Triangulation

— or —

How Matlab Computes a QR Decomposition

HW Q: If $P \in \mathbb{C}^{n \times n}$ is an orthogonal projection, then $I_n - 2P$ is a (self-adjoint) unitary, as is $2P - I_n$.

Proposition: (outer product projections)

If $v \in \mathbb{C}^n$ is a unit vector, then the matrix $P \in \mathbb{C}^n$ given by

$$P = v \cdot v^*$$

(the outer product of v with itself) is the orthogonal projection onto $\text{span}(v)$.

proof:

1) P self-adjoint

$$P^* = (V \cdot V^*)^*$$

$$= (V^*)^* \cdot V^* = V \cdot V^*$$

$$= P$$

2) $P^2 = P$ ⟶

$$P^2 = (v \cdot v^*) \cdot (v \cdot v^*)$$

$$= v\,(v^* \cdot v)\,v^*$$

| Since v is a unit vector

$$= v \cdot v^* = P$$

3) range (P) = span (v).

$$Pv = (v \cdot v^*)v$$

$$= v \cdot (\underbrace{v^* \cdot v}_{=1})$$

$$= v$$

Moreover, if $w \perp v$, then

$$Pw = (v v^*)w$$

$$= v(\underbrace{v^* w}_{=0})$$

$$= 0$$

This shows

$$\operatorname{ran}(P) = \operatorname{span}(v)$$

Corollary: If $v \in \mathbb{C}^n$ is a unit vector, then $I_n - vv^*$ is the orthogonal projection onto $(\text{span}(v))^\perp$, which is an $(n-1)$-dimensional subspace of $\mathbb{C}^n$, called a hyperplane.