- Gauss-Seidel method for approximating solutions to $Ax=b$..

But first a little more on matrix norms:

$$\|A\| = \max\{\|Ax\| : \|x\| \leq 1\}$$

($\|A\|$: matrix norm; $\|Ax\|$: vector norms; $\|x\| \leq 1$)

So far we've only considered the usual Euclead vector norm

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \sqrt{x \cdot x}$$

to tell the norms apart we'll use subscripts..

$$\|x\|_3 = \sqrt[3]{x_1^3 + x_2^3 + \cdots + x_n^3}$$

a whole family of vector norms:

$$\|x\|_p = \sqrt[p]{x_1^p + x_2^p + \cdots + x_n^p} \quad \text{for } p \in [1, \infty)$$

$$\|A\|_p = \max\{\|Ax\|_p : \|x\|_p \leq 1\}$$

define a p matrix norm.

Computing this for arbitrary $p \geq 1$ is difficult.

$$\|x\|_\infty = \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$

Note that $\|x\|_\infty = \lim_{p \to \infty} \|x\|_p$

but this is not important today.

Define an ∞-matrix ...

$$\|A\|_\infty = \max\{\|Ax\|_\infty : \|x\|_\infty \leq 1\}$$

Norms of interest are $\|x\|_1$, $\|x\|_2$ and $\|x\|_\infty$

limiting cases

geometric meaning of usual distance.

limiting case.

There is one more advantage of these: The matrix is surprisingly simple to compute corresponding to these matrix norms...

For next time (maybe) we simplify

$$\|A\|_1 = \max\{\|Ax\|_1 : \|x\|_1 \le 1\}$$

and

$$\|A\|_\infty = \max\{\|Ax\|_\infty : \|x\|_\infty \le 1\}$$ Step 16

Note we've already simplified the 2-norm to get

$$\|A\|_2 = \max\{\|Ax\|_2 : \|x\|_2 \le 1\}$$

$$= \max\{\lambda_i^{1/2} : i = 1, \dots, n\}$$

where λ_i's are the eigenvalues of A^TA.

$$= \max\{\sigma_i : i = 1, \dots, n\}$$

where $\sigma_i = \lambda_i^{1/2}$ are the singular values of A.

SYSTEMS OF LINEAR EQUATIONS 3

The Gauss-Seidel iterative method

① Gaussian-Elimination doesn't give the exact answer because of rounding error..

② Do something simpler that's an approximation in the first place.

③ Iterative methods exhibit some robustness against rounding errors, because at each step the previous approximation is improved, thereby correcting previously made rounding errors.

Solve using iteration

$$\begin{aligned} 10x_1 + 2x_2 + x_3 &= 13 \\ 2x_1 + 10x_2 + x_3 &= 13 \\ 2x_1 + x_2 + 10x_3 &= 13 \end{aligned}$$

$$\text{new } x_1 = \frac{13 - 2x_2 - x_3}{10}$$

$$\text{new } x_2 = \frac{13 - 2x_1 - x_3}{10}$$

$$\text{new } x_3 = \frac{13 - 2x_1 - x_2}{10}$$

Let (x_1^0, x_2^0, x_3^0) be an initial approximation then define

$$x_1^{n+1} = \frac{13 - 2x_2^n - x_3^n}{10}$$

$$x_2^{n+1} = \frac{13 - 2x_1^{n+1} - x_3^n}{10}$$

$$x_3^{n+1} = \frac{13 - 2x_1^{n+1} - x_2^{n+1}}{10}$$

these large denominator help to make $\|g'\| < 1$.

when does this sequence converge to the solution?

Let $x_{n+1} = g(x_n)$ be the above iteration. If g is a contraction then this iteration converges...

Need $\|g'(x)\| < 1$ and then it works...

matrix norm...

$g: \mathbb{R}^3 \to \mathbb{R}^3$ so g' is a 3×3 matrix of partials

$$g'(x) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \frac{\partial g_1}{\partial x_3} \\ \vdots & & \\ \frac{\partial g_3}{\partial x_1} & \cdots & \frac{\partial g_3}{\partial x_3} \end{bmatrix}$$

Theorem if the denominators are bigger (in absolute value) than the sums of the (abs values of the) coefficients in the numerator the Gauss-Seidel method converges.

Note if instead we tries

$$\begin{aligned} 10x_1 + 2x_2 + x_3 &= 13 \\ 2x_1 + 10x_2 + x_3 &= 13 \\ 2x_1 + x_2 + 10x_3 &= 13 \end{aligned}$$

$$x_2 = \frac{13 - 10x_1 - x_3}{2}$$

$$x_3 = \frac{13 - 2x_1 - 10x_2}{1}$$

$$x_1 = \frac{13 - x_2 - 10x_2}{2}$$

This iteration is not a contraction and won't converge...

Now write code to perform the Gauss-Seidel iteration.

Call the function gs rather than g

```
function gs(A,b,x)
    y=copy(x)
    for i=1:size(A)[1]
        t=b[i]
        for j=1:size(A)[2]
            if i!=j
                t-=A[i,j]*y[j]
            end
        end
        t/=A[i,i]
        y[i]=t
    end
    return y
end
```

number of rows

number of columns

skip term we're solving for

make the numerator

divide by the denominator

update the new value

Code is simpler than Gaussian elimination with partial pivoting.

```
julia> A=[10 2 1; 2 10 1; 2 1 10]
3×3 Matrix{Int64}:
 10   2   1
  2  10   1
  2   1  10

julia> b=[13,13,13]
3-element Vector{Int64}:
 13
 13
 13
```

Solve the same problem $Ax=b$ as before

```
julia> xn=zeros(3)
3-element Vector{Float64}:
 0.0
 0.0
 0.0
```

initial guess

The initial guess could also be random or come from a different algorithm that used lower precision arithmetic.

```
julia> xn=gs(A,b,xn)
3-element Vector{Float64}:
 1.3
 1.04
 0.9359999999999999

julia> xn=gs(A,b,xn)
3-element Vector{Float64}:
 0.9984
 1.00672
 0.999648

julia> xn=gs(A,b,xn)
3-element Vector{Float64}:
 0.9986912
 1.00029696
 1.000232064
```

```
julia> xn=gs(A,b,xn)
3-element Vector{Float64}:
 0.9999174016
 0.9999933132799999
 1.000017188352

julia> xn=gs(A,b,xn)
3-element Vector{Float64}:
 0.9999996185087999
 0.99999835746304
 1.0000002405519361
```

converging to $x=\begin{bmatrix}1\\1\\1\end{bmatrix}$