Lecture 17: Problems and Conditioning

In this lecture we introduce the notion of a *problem*. Before that, we mention how `Bool`'s can be used to do logic.

Bool

A `Bool` is a type that represents either `true` or `false`.

In [30]:

```
x=5

b=  x==5
println(b)
println(typeof(b))
```

```
true
Bool
```

While a `Bool` has only two possible values, therefore plays the same role as a single bit, it requires 8 bits to store as every memory address stores 8 bits:

In [32]:

```
bits(b)
```

Out[32]:

```
"00000001"
```

Logical and is done with `&&` while logical or is done with `||`:

In [34]:

```
println(x==5 && x==6)    # && means and
x==5 || x==6   # || means or
```

```
false
```

Out[34]:

```
true
```

In [10]:

```
true && true
true && false
false && false
```

Out[10]:

```
false
```

We can do boolean logic using for loops. The following checks if any entry of `x` is 5, which is true:

In [35]:

```
x=collect(1:10)


# check if any entry of x is 5
b = false
for k=1:10
    b = b || x[k]==5
end

b
```

Out[35]:

```
true
```

While this checks if any entry of `x` is 12, which is false:

In [36]:

```
b = false
for k=1:10
    b = b || x[k]==12
end

b
```

Out[36]:

```
false
```

Problems

A *normed vector space* is a vector space, like $\mathbb{R}^n$, which has a norm attached, such as the 2-norm.

A *problem* is a function from one normed space X to another Y:

$$f : X \to Y$$

The norm attached to X describes the error we expect in the input, while the norm attached to Y describes the error we are trying to measure in the output.

Examples

We have some simple examples:

Example 1: For a given matrix $A \in \mathbb{R}^{n \times m}$, define the problem of matrix-vector multiplication $f : \mathbb{R}^m \to \mathbb{R}^n$, with the 2-norms attached:

$$f(\mathbf{x}) = A\mathbf{x}$$

This problem encodes the sensitivity of matrix multiplication to perturbations in the *vector*.

Example 2: For a given vector $\mathbf{x} \in \mathbb{R}^m$, define the problem of matrix-vector multiplication $f : \mathbb{R}^{n \times m} \to \mathbb{R}^n$, with the 2-norms attached:

$$f(A) = A\mathbf{x}$$

This problem encodes the sensitivity of matrix multiplication to perturbations in the *matrix*.

Example 3: For a given matrix $A \in \mathbb{R}^{n \times m}$, define the problem of solving a linear system $f : \mathbb{R}^m \to \mathbb{R}^n$, with the 2-norms attached:

$$f(\mathbf{x}) = A^{-1}\mathbf{x}$$

This problem encodes the sensitivity of matrix inversion to perturbations in the *vector*.

Example 4: Define the problem of matrix-vector multiplication $f : \mathbb{R}^{n \times m} \times \mathbb{R}^m \to \mathbb{R}^n$:

$$f(A, \mathbf{x}) = A\mathbf{x}$$

We can attach the 2-norm to the output. For the input, we attach the norm

$$\|(A, \mathbf{x})\| = \max\{\|A\|_2, \|\mathbf{x}\|_2\}$$

This problem encodes the sensitivity of matrix multiplication to perturbations in both the *matrix* and *vector*.

Example 5: Define the problem of squaring a number $f : \mathbb{R} \to \mathbb{R}$, with the absolute value as the norm:

$$f(x) = x^2$$

Relative vs Absolute error

We can measure the error using either absolute or relative error. The *absolute error* for the data x perturbed by Δx is

$$\|f(x + \Delta x) - f(x)\|$$

But in practice, we usually care more about relative error:

$$\frac{\|f(x + \Delta x) - f(x)\|}{\|f(x)\|}$$

For example, consider the problem of calculating the exponential

$$f(x) = e^x$$

with the absolute value attached.

When `x` is small, the absolute error is fairly small:

In [43]:

```
x=3
Δx=0.000001
abs(exp(x)-exp(x+Δx))
```

Out[43]:

```
2.008554696786291e-5
```

But when `x` is large, the absolute error is very large:

In [45]:

```
x=25
Δx=0.000001
abs(exp(x)-exp(x+Δx))
```

Out[45]:

```
72004.9354095459
```

But the actual digits are accurate:

In [46]:

```
exp(x),exp(x+Δx)
```

Out[46]:

```
(7.200489933738588e10,7.200497134232129e10)
```

Thus it is more reliable to look at relative error, which remains small even when `x` is large:

In [49]:

```
x=3
Δx=0.000001
abs(exp(x)-exp(x+Δx))/abs(exp(x))
```

Out[49]:

```
1.000000500094933e-6
```

In [50]:

```
x=25
Δx=0.000001
abs(exp(x)-exp(x+Δx))/abs(exp(x))
```

Out[50]:

```
1.0000005009681335e-6
```

Absolute condition number

The *absolute condition number* of a problem is a measure of how much the absolute error in the input is magnified to cause absolute error in the output. The mathematical definition of the absolute condition number is

$$\hat{\kappa_f}(\mathbf{x}, \epsilon) \triangleq \sup_{\|\Delta\mathbf{x}\|_X \leq \epsilon} \frac{\|f(\mathbf{x} + \Delta\mathbf{x}) - f(\mathbf{x})\|_Y}{\|\Delta\mathbf{x}\|_X}$$

This gives us a bound on absolute errors: if $\|\Delta\mathbf{x}\|_X \leq \epsilon$ we have

$$\|f(\mathbf{x} + \Delta\mathbf{x}) - f(\mathbf{x})\|_Y \leq \hat{\kappa_f}(\mathbf{x}, \epsilon)\|\Delta\mathbf{x}\|_X$$

Example 1

For the problem $f(\mathbf{x}) = A\mathbf{x}$, the absolute condition number is:

$$\hat{\kappa_f}(\mathbf{x}, \epsilon) = \sup_{\|\Delta\mathbf{x}\|_X \leq \epsilon} \frac{\|A(\mathbf{x} + \Delta\mathbf{x}) - A\mathbf{x}\|_Y}{\|\Delta\mathbf{x}\|_X} = \sup_{\|\Delta\mathbf{x}\|_X \leq \epsilon} \frac{\|A(\Delta\mathbf{x})\|_Y}{\|\Delta\mathbf{x}\|_X} = \sup_{\|\mathbf{v}\|=1} \frac{\|A\mathbf{v}\|_Y}{\|\Delta\mathbf{v}\|_X} = \|A\|_{X\to Y}$$