Mathematical fundamentals for ISS

Honza Černocký, ÚPGM FIT VUT

Please open Python notebooks **`matika`** and **`cos_jexp`**.

Functions and sequences

- Signals with **continuous time** (analog, real world) are defined for all times *t*. Signály se spojitým časem, spojité signály.
  - From math point of view, they are **functions** $x(t)$ ← round brackets
- Signals with **discrete time** (computer, digital world), sampled signals are defined only for integer times (sample numbers) *n*. Signály s diskrétním časem, diskrétní signály, vzorkované signály.
  - From math point of view, they are **sequences** $x[n]$ ← square brackets.

How is it done in Python ? Just 4 steps …

1. Generate the independent variable - 1 line of Python code
2. Generate or read the signal - 1 line of Python code
3. Do something with the signal - 1 line of Python code
4. Show the result - 1 line of Python code (or more if you want to have it nice)

Operations with functions

- As you know them from primary school, just do it for all times:
- Addition $y(t) = x_1(t) + x_2(t)$
- Subtraction $y(t) = x_1(t) - x_2(t)$
- Multiplication $y(t) = x_1(t)x_2(t)$
    - Mind the “masking effect” of zeros

#basicops

Operations with sequences

- As you know them from primary school, just do it for all times.
- Addition $y[n] = x_1[n] + x_2[n]$
- Subtraction $y[n] = x_1[n] - x_2[n]$
- Multiplication $y[n] = x_1[n]x_2[n]$
  - Mind the "masking effect" of zeros
- When plotting in Python, use `stem` rather than `plot` to show that we have discrete values.
- But remember this is a lecture example, in normal life, we'll be happily (and all the time) using `plot` also with discrete signals ☺

#basicops_seq

Linear function $x(t) = at + b$

- Parameter *a* – slope (směrnice, sklon)
- Parameter *b* – offset, bias (posun).

#linear

Derivations and integrals

- Example on a simple quadratic function:

$$x(t) = t^2 + 2t + 3$$

- Derivation done analytically

$$\frac{dx(t)}{dt} = 2t + 2$$

- Primitive function (antiderivative, primitivní funkce) done analytically

$$x_p(t) = \frac{t^3}{3} + t^2 + 3t$$

#deriv_int_anal

Derivation done numerically

- Analytical form of derivation might not exist, or we are given an input signal, or just too lazy …
- Make use of the definition of derivation

$$\left.\frac{dx(t)}{dt}\right|_{t_1} \approx \frac{x(t_2) - x(t_1)}{t_2 - t_1}$$

- and just estimate the derivative from 2 points close to each other …

#deriv_num

Finite integral from t_1 to t_2

- Analytically using the primitive function:

$$\int_{t_1}^{t_2} x(t)dt = [x_p(t)]_{t_1}^{t_2} = x_p(t_2) - x_p(t_1)$$

- But we will rather meet the numerical approximation of integral:
  - Finite integral is the surface under the function (plocha pod křivkou)
  - Define a time step *Δ*, filling the interval t_1 to t_2 N times.
  - Approximate the integral as the sum of surfaces of "noodles", and simplify as the width of all noodles is the same:

$$\int_{t_1}^{t_2} x(t)dt \approx \sum_{n=0}^{N-1} x(t_1 + n\Delta)\Delta$$

$$\int_{t_1}^{t_2} x(t)dt \approx \Delta \sum_{n=0}^{N-1} x(t_1 + n\Delta)$$

#int_anal_num

Cosine function (cosínusovka) – continuous time

- Basic cosine

$$x(t) = \cos(t)$$

- has
  - Period (perioda) 2π rad – we are measuring angles in radians.
  - Magnitude (amplitude) 1
  - Phase shift (fázové posunutí, fáze) 0

#cosine_basic

Cosine with more interesting parameters

- Setting the number of periods per second: **frequency** (frekvence, kmitočet) f_1 in Hertz.
- The period (in seconds) then becomes

$$T_1 = \frac{1}{f_1}$$

- Making the cosine do f_1 periods in second: multiply its argument by

$$\omega_1 = \frac{2\pi}{T_1} = 2\pi f_1$$

- This value is called **angular or circular frequency or velocity** (kruhová / úhlová frekvence / rychost) and is measured in rad/s

Cosine with more interesting parameters II

- Changing the magnitude - just multiply the cosine by constant C_1.
- Changing the phase – add angle Φ_1 to the argument
  - Positive – shift to the left - advancing the signal (předběhnutí)
  - Negative – shift to the right – delaying the signal (zpoždění)
  - After adding / subtracting 2π, we obtain the same, as the periodicity of cosine is 2π.
- Full cosine:

$$x(t) = C_1 \cos(\omega_1 t + \phi_1)$$

#cosine_full

Cosine sequence – discrete time

- Sequence $x[n] = \cos(n)$
  - Not very interesting - something like 6.28 samples per period ???
- Enforcing the number of samples in one period to be *N:*

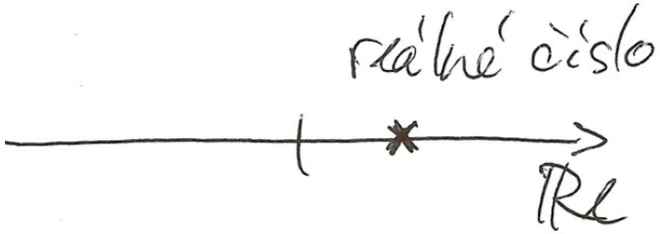
$$x[n] = \cos\left(\frac{2\pi}{N}n\right)$$

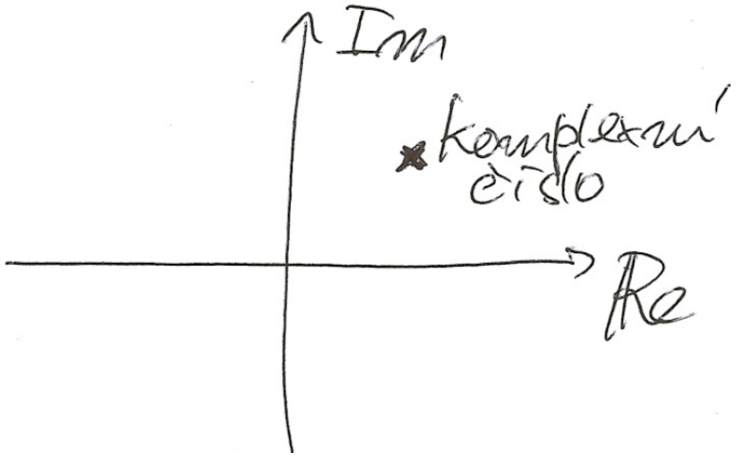
- *2π/N* is called **normalized angular frequency** (normovaná kruhová frekvence) – also denoted as ω_1.
- Similarly as for the continuous one, we can change the **magnitude** and **phase**

#cosine_disc

Complex numbers – basics

- The real numbers are 1D – real axis $\Re$
- Complex numbers are 2D – lay in the **complex plane** (komplexní rovina) with two dimensions – the real one $\Re$ and the imaginary one $\Im$
- Mathematicians use *i*, electrical (and IT) engineers use *j*, Python uses `1j`

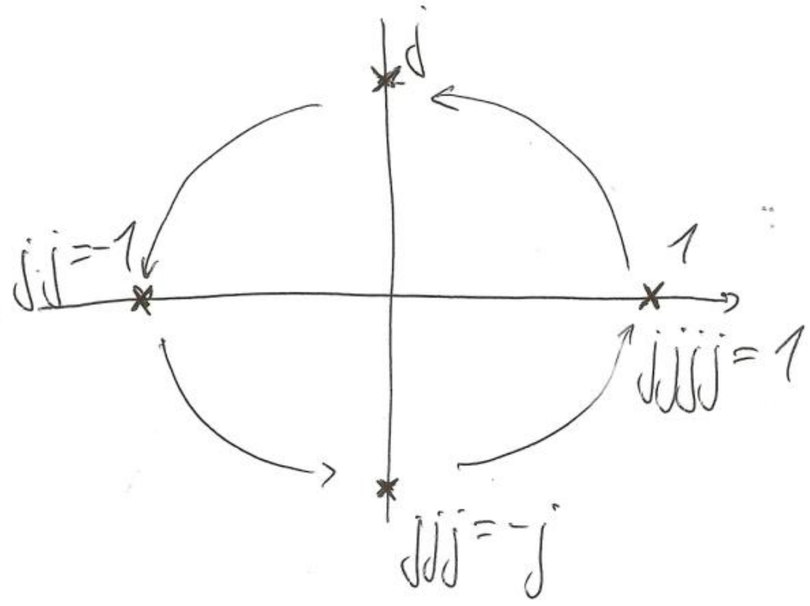




Complex unit

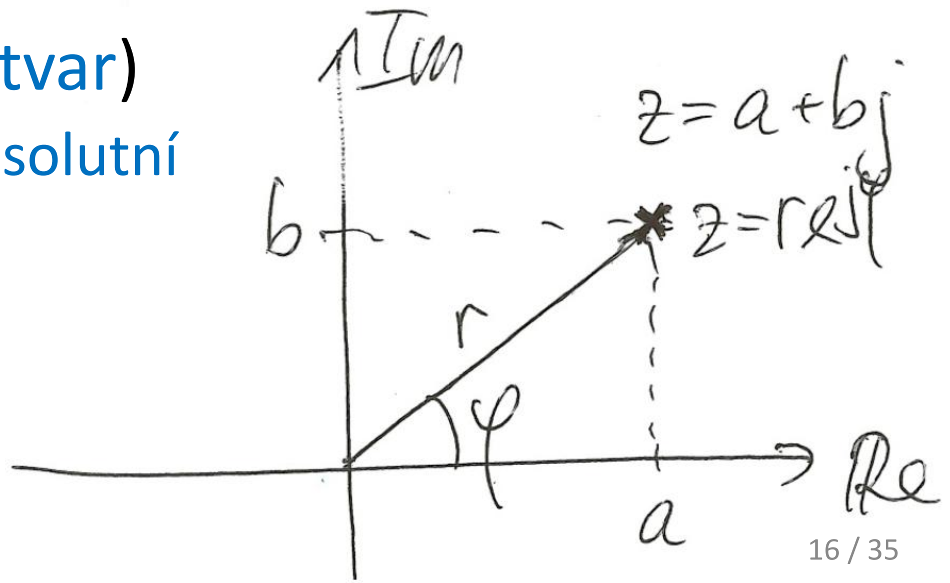
- Is defined as $j = \sqrt{-1}$
- Has "circular property"

$$1, \quad 1j = j, \quad 1jj = -1, \quad 1jjj = -j, \quad 1jjjj = 1,$$



Complex number as a vector and its two forms

- It is convenient to imagine complex numbers as vectors starting in *0+0j* and going to the given complex number.
- **Component form** (složkový tvar) of a complex number $z = a + jb$
  - a is real component
  - *b* is imaginary component
- **Polar form** (polární nebo goniometrický tvar)
  - r is magnitude or absolute value (modul, absolutní hodnota, magnituda)
  - ϕ is phase (fáze, úhel, argument)



Conversion of forms using basic geometry

- From polar to component $a = r\cos\phi \qquad b = r\sin\phi$

  … so the component form can be written as $z = r\cos\phi + jr\sin\phi.$
- From component to polar $r = \sqrt{a^2 + b^2} \qquad \phi = \tan^{-1}\frac{b}{a}$
  - Be careful about the inverse tangens and always check that you have got the result correctly.
  - See the 1st numerical exercise.
- Python (and all other languages supporting math) have appropriate functions
  - `np.real()`, `np.imag()`
  - `np.abs()`, `np.angle()`

Exponential form of complex number

- $z = re^{j\phi}$ or (if lazy to type in LaTeX) $z = r \exp j\phi,$

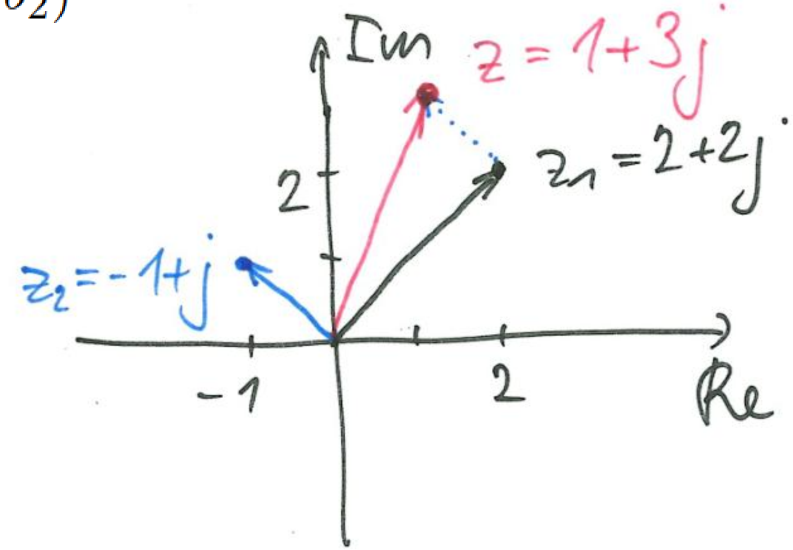
  or (el. engineers) $z = r\measuredangle\phi$

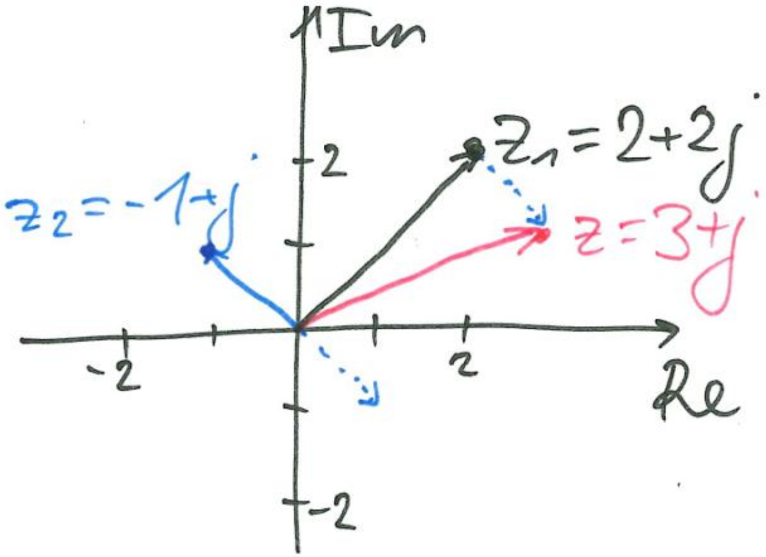
Operations with complex numbers

- **Addition** and **subtraction** in composite form
- Easy to imagine as vectors in the complex plane !

$$z_1 + z_2 = a_1 + a_2 + j(b_1 + b_2)$$

$$z_1 - z_2 = a_1 - a_2 + j(b_1 - b_2)$$





Operations with complex numbers II

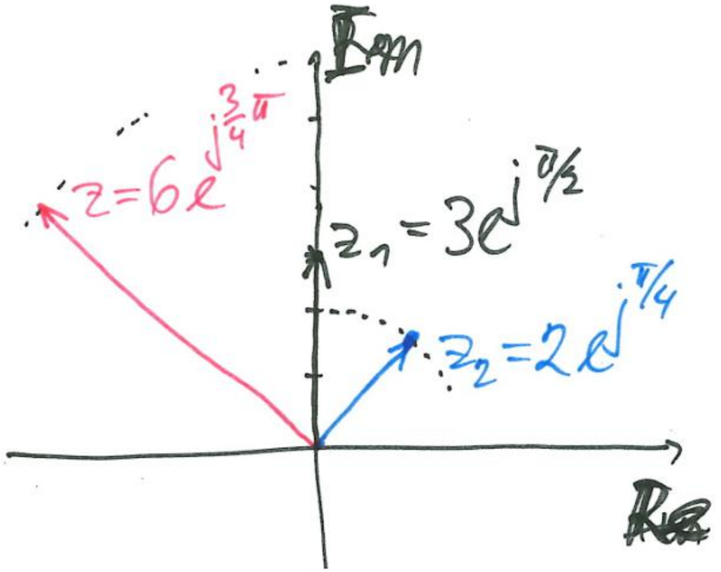
- **Multiplication**
  - Possible in component form, but too complicated …

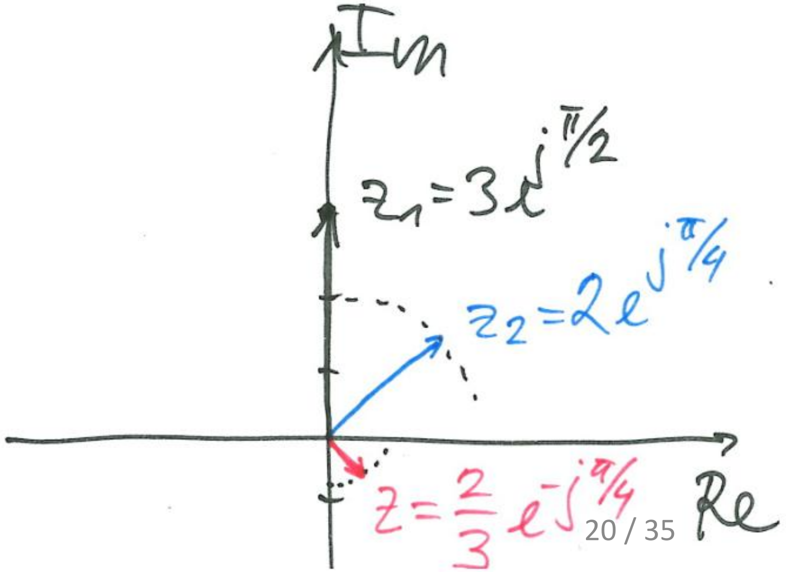
$$z_1 z_2 = (a_1 + jb_1)(a_2 + jb_2) = a_1 a_2 + j(b_1 a_2 + a_1 b_2) - b_1 b_2,$$

  - Much more convenient in the exponential form: **multiply the magnitudes and sum the phases** ! $z_1 z_2 = r_1 r_2 e^{j(\phi_1 + \phi_2)}$

- **Division**: similar

$$\frac{z_1}{z_2} = \frac{r_1}{r_2} e^{j(\phi_1 - \phi_2)}$$





Operations with complex numbers III

- **Complex conjugation** (komplexní sdružení)
  - Component form: imaginary part swapped
  - Exponential form: phase swapped

$$z^\star = a - jb = re^{-j\phi}.$$

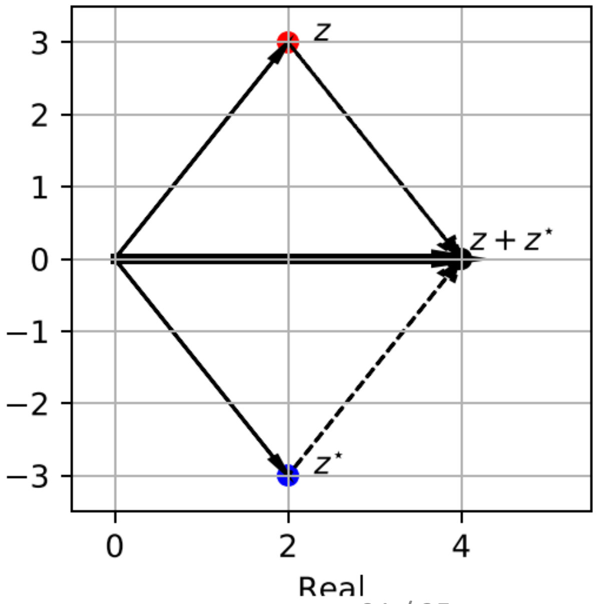
- Sum of complex-conjugated complex numbers is a real number

$$z + z^\star = a + jb + (a - jb) = 2a + j(b - b) = 2a$$

- Product of complex-conjugated complex numbers is the square of their magnitude
  - Will be handy in determing powers and energies

$$zz^\star = re^{j\phi}re^{-j\phi} = r^2e^{j(\phi-\phi)} = r^2e^0 = r^2$$

  - `np.conj(x) * x` is faster to write and to compute than `np.power(np.abs(x), 2.0)`

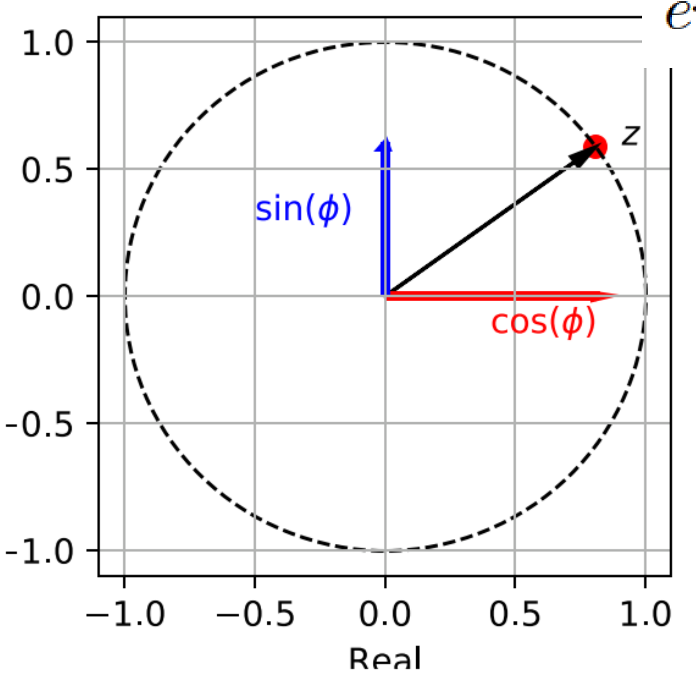


Unit circle (jednotková kružnice) $r = 1$

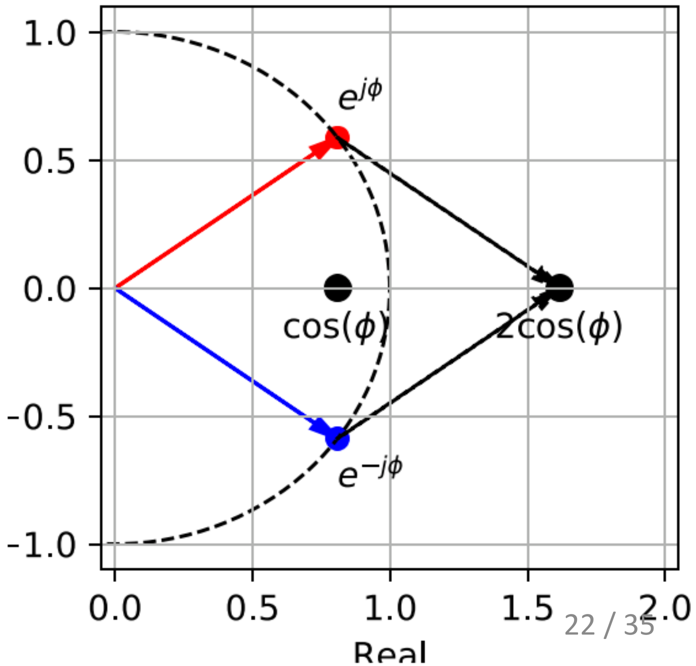
$$e^{j\phi} = \cos\phi + j\sin\phi$$

- Useful to derive some formulas in case you forget them …

$$e^{j\phi} + e^{-j\phi} = \cos\phi + j\sin\phi + \cos\phi - j\sin\phi = 2\cos\phi$$

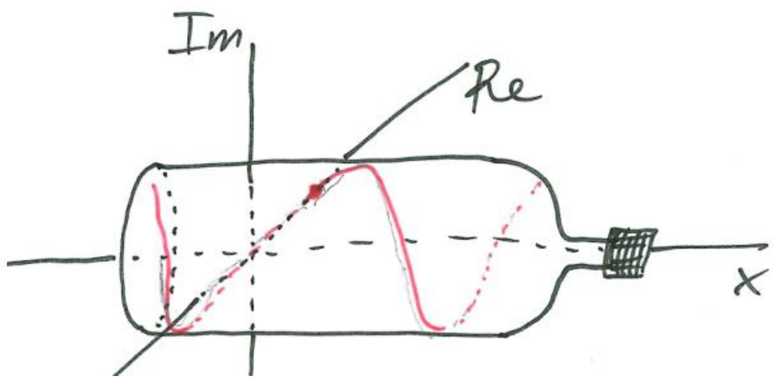


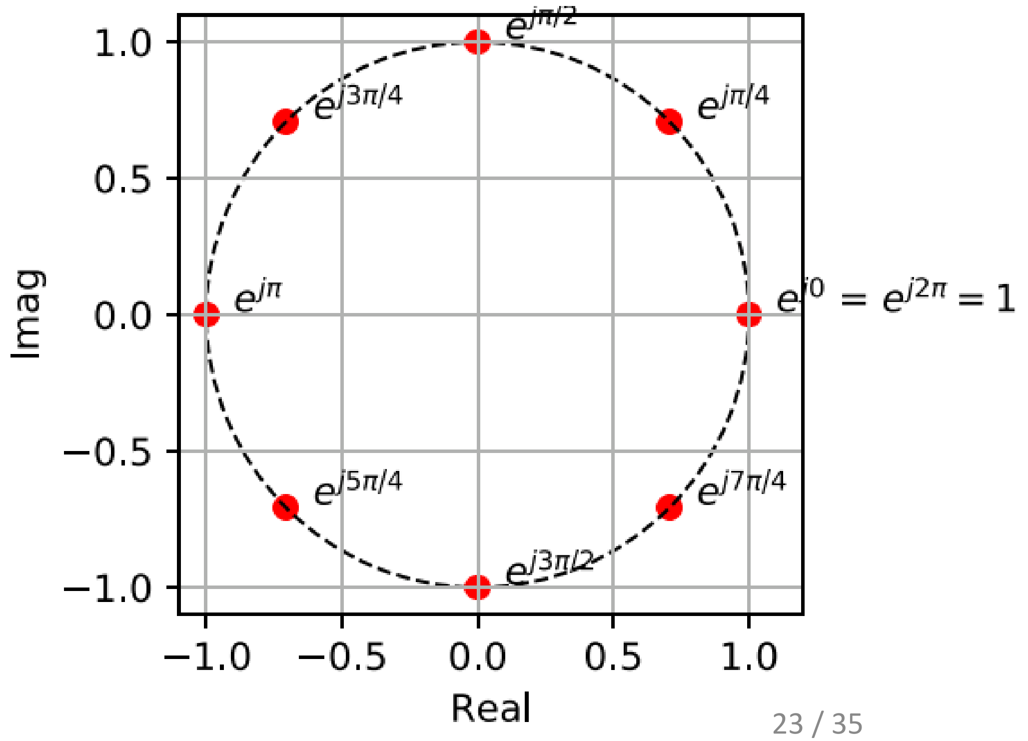
$$\cos\phi = \frac{e^{j\phi} + e^{-j\phi}}{2}$$



Complex exponential e^{jx}

- For zero phase: $e^{j0} = 1$
- The point then starts moving counter-clock-wise (proti směru hodinových ručiček)
- Much better to see it on a physical model the „complex bottle“





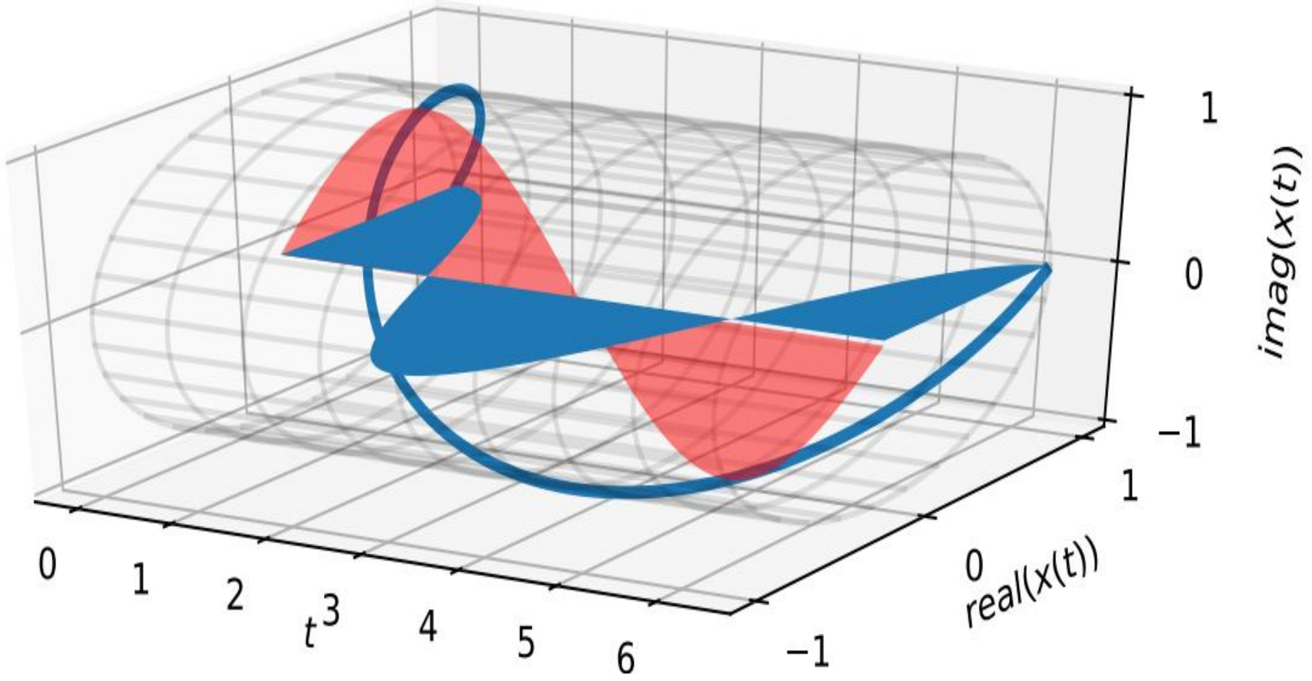
- and visualize it in Python

#complex_exp

Operations with complex exponential

Showing that it can be really decomposed into cos and sin …

$\Re(e^{jx}) = \cos x \qquad \Im(e^{jx}) = \sin x$



turning in opposite direction:

- e^{-jx} instead of e^{jx}

#complex_exp_minus

Operations with complex exponential II

Changing the magnitude – just multiply with a constant

#complex_exp_magnitude

Changing the phase – just multiply with a complex number laying on the unit circle with the desired phase

- the complex exponential will acquire the desired phase

$$e^{j\frac{\pi}{4}}e^{j0} = e^{j\frac{\pi}{4}}1 = e^{j\frac{\pi}{4}}$$

#complex_exp_phase

Changing both – multiply with a complex number, magnitude -> desired magnitude change, phase -> desired phase change

#complex_exp_magnitude_phase

$$ce^{jx} = |c|e^{j\arg(c)}e^{jx} = |c|e^{j(x+\arg(c))}$$

Complex exponential with period different from *2π*

- Exactly the same as for the cosine: multiply time by angular frequency $\omega_1 = \frac{2\pi}{T_1}$ so that

$$x(t) = e^{j\omega_1 t},$$

- the angular frequency can be even **negative**, in this case, the complex exponential will "turn" in the opposite direction

$$x(t) = e^{-j\omega_1 t}.$$

#complex_exponential_ang_freq

Complex exponential – the whole thing

- the **complex** coefficient c_1
  - Its **magnitude** determines the magnitude of resulting complex exponential.
  - Its **phase** determines the phase of resulting complex exponential.

$$x(t) = c_1 e^{j\omega_1 t} = |c_1| e^{\arg c_1} e^{j\omega_1 t} = |c_1| e^{j\omega_1 t + \arg c_1}$$

- the **angular frequency** ω_1 determines the number of revolutions of the complex exponential per second.

Breaking a cosine into two complex exponentials

- Using the well known formula $$\cos x = \frac{e^{jx} + e^{-jx}}{2}$$
- The cosine without initial phase can be decomposed as

$$C_1 \cos(\omega_1 t) = \frac{C_1}{2} e^{j\omega_1 t} + \frac{C_1}{2} e^{-j\omega_1 t}.$$

  - Don't be scared by the negative angular frequency !

#cos_decomposition

- And the full one similarly making use of $e^{a+b} = e^a e^b$

$$C_1 \cos(\omega_1 t + \phi_1) = \frac{C_1}{2} e^{j(\omega_1 t + \phi_1)} + \frac{C_1}{2} e^{-j(\omega_1 t + \phi_1)} = \frac{C_1}{2} e^{j\phi_1} e^{j\omega_1 t} + \frac{C_1}{2} e^{-j\phi_1} e^{-j\omega_1 t}$$

  - $c_1 = \frac{C_1}{2} e^{j\phi_1}$ and $c_{-1} = \frac{C_1}{2} e^{-j\phi_1}$ are complex constants.

#cos_decomposition_full

Complex exponential - discrete-time

- Very similar to discrete cosine … for 1 period in *N* samples $x[n] = e^{j\frac{2\pi}{N}n}$
- *k* periods in *N* samples: $x[n] = e^{j\frac{2\pi k}{N}n}.$

#complex_exp_discrete

#complex_exp_discrete_2

General complex exponential with discrete time

- Similarly as for the continuous-time one,
  - magnitude of *c* determines the magnitude of resulting complex exponential.
  - phase of *c* determines the phase of resulting complex exponential.
  - *k* determines the number of periods ("turns") in *N* samples

$$ce^{j\frac{2\pi k}{N}n} = |c|e^{j\arg(c)}e^{j\frac{2\pi k}{N}n} = |c|e^{j(\frac{2\pi k}{N}n+\arg(c))},$$

#complex_exp_discrete_general

Decomposing a discrete cosine into two complex exponentials

- Again using the well known formula $\cos x = \frac{e^{jx} + e^{-jx}}{2}$

$$C_1 \cos(\frac{2\pi k}{N} n + \phi_1) = \frac{C_1}{2} e^{j(\frac{2\pi k}{N} n + \phi_1)} + \frac{C_1}{2} e^{-j(\frac{2\pi k}{N} n + \phi_1)} = \frac{C_1}{2} e^{j\phi_1} e^{j\frac{2\pi k}{N} n} + \frac{C_1}{2} e^{-j\phi_1} e^{-j\frac{2\pi k}{N} n}$$

- Complex numbers $c_1 = \frac{C_1}{2} e^{j\phi_1}$ and $c_{-1} = \frac{C_1}{2} e^{-j\phi_1}$ are again complex coefficients.
- The example is for $x[n] = 5\cos(\frac{2\pi}{N} n - \frac{\pi}{3})$

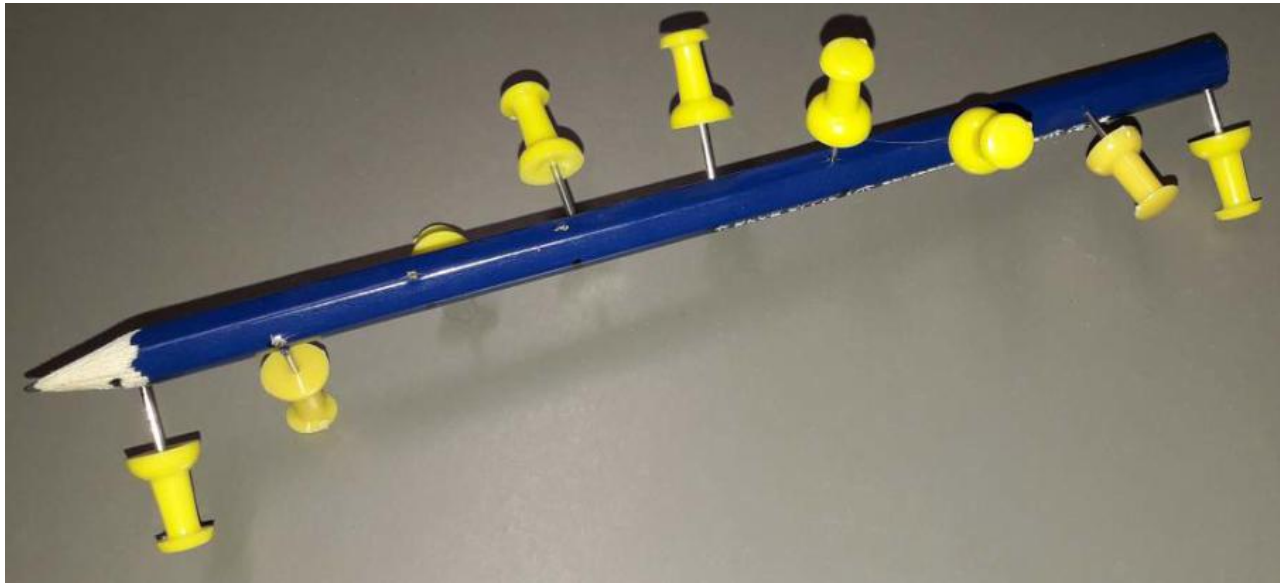
$$x[n] = \frac{5}{2} e^{-j\frac{\pi}{3}} e^{j\frac{2\pi k}{32} n} + \frac{5}{2} e^{+j\frac{\pi}{3}} e^{-j\frac{2\pi k}{32} n}.$$

$$c_1 = \frac{5}{2} e^{-j\frac{\pi}{3}}, \quad c_{-1} = \frac{5}{2} e^{j\frac{\pi}{3}}.$$

#discrete_cos_decomposition

Physical model for discrete complex exponential

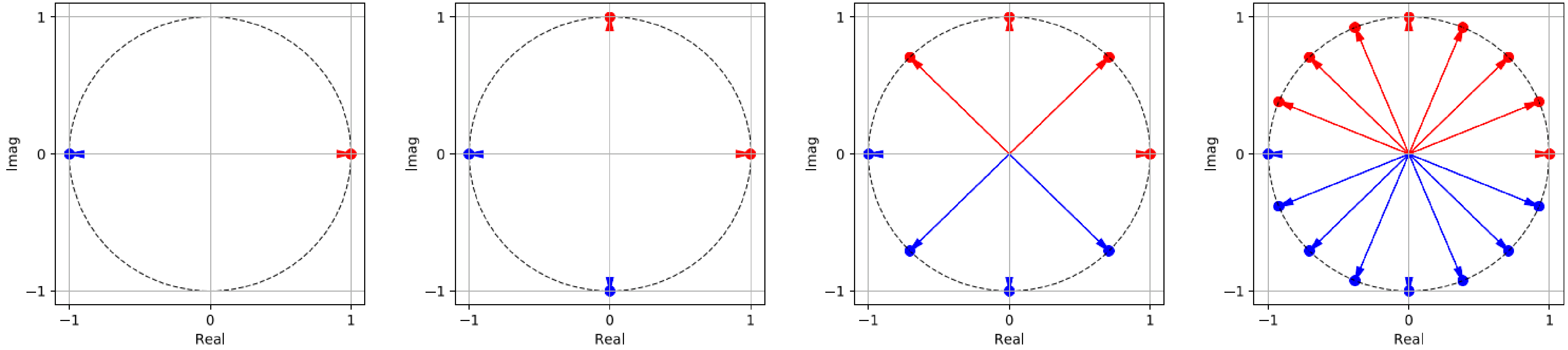
- Use whatever long object and put something into it …



Sum of **discrete** complex exponential over one period

$$\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n}$$

- Examples for N = 2, 4, 8, and 16.
- Obvious that the blue parts will cancel out with the red ones => **zero**



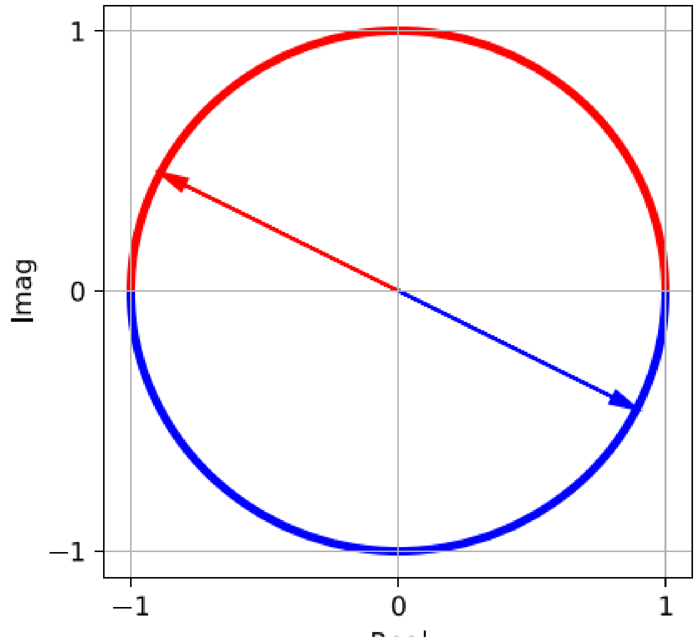
- It generalizes also for other *N*'s. In case more math needed, here it is !

$$\sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}n} = \sum_{n=0}^{\frac{N}{2}-1} \left[e^{j\frac{2\pi}{N}n} + e^{j\frac{2\pi}{N}(n+\frac{N}{2})} \right] \xrightarrow{e^{a+b} = e^a e^b} \sum_{n=0}^{\frac{N}{2}-1} e^{j\frac{2\pi}{N}n} \left[1 + e^{j\frac{2\pi}{N}\frac{N}{2}} \right] \longrightarrow \sum_{n=0}^{\frac{N}{2}-1} e^{j\frac{2\pi}{N}n} [1 + (-1)] = 0.$$

Sum of a **continuous** complex exponential over one period $\int_{T_1} e^{j\omega_1 t} dt.$

- Red and blue parts cancel out (integral is nothing but a sum …)

$$\int_0^{T_1} e^{j\omega_1 t} dt = \int_0^{\frac{T_1}{2}} \left[e^{j\omega_1 t} + e^{j\omega_1 t + \pi}\right] dt.$$



- If you really want a proof, here it is:

$$\int_0^{T_1} e^{j\omega_1 t} dt = \int_0^{\frac{T_1}{2}} e^{j\omega_1 t} \left[1 + e^{j\pi}\right] dt = \int_0^{\frac{T_1}{2}} e^{j\omega_1 t} \left[1 + (-1)\right] dt = 0.$$

Summary

- ISS is actually an **applied math course** (also dubbed “Fourier hell” …)
- Cosines, sines and complex exponentials are the very basis of **spectral analysis**.
- It is good to write equations but writing 2 lines of **Python code**, and visualizing the result, helps **understanding** the math!
- Funny TODO: Impress your friends in a pub by creating a **physical model** of a complex exponential (please send me photos!)
- Serious TODO: think about the **relations of regular and normalized frequencies** when one samples a continuous signal with sampling frequency F_s.
  - Consider both standard and angular ones.
  - What are their units ?
  - How to convert one to another ?