Neuromorphic engineering I

Lab 6: Integrator Circuits

Group number: 18

Team member 1: Quillan Favey

Team member 2: Hooman Javaheri

Date: 11.1.2022

Objectives of this lab: In this lab we will begin to explore the time domain using the follower-integrator and the follower-differentiator circuit.

Both circuits simply contain a transconductance amplifier and a capacitor to implement a low-pass or high-pass filter.

We will use the follower-integrator (FOI) and follower-differentiator circuit on the CoACH chip. The capacitance for both capacitors is $1pF$.

The objectives of the lab are:

1. Understand the behavior of the first-order low-pass follower-integrator and high-pass follower-differentiator circuit in the time and frequency domain in small signal operation.

First-order means that the transfer function amplitude decreases as 1/frequency.

Low-pass (*High-pass*) means that the circuit passes low (high) frequencies and blocks high (low) frequencies.

Follower-integrator (*Follower-differentiator*) means that the output follows the input at low (high) frequencies, and integrates (differentiates) at higher (lower) frequencies.

2. Understand the large signal behavior and other limitations of using a transconductance amplifier to model a linear resistor.

1 Reading

See Chapters 8 and 9 of the Carver Mead book ("Analog VLSI and Neural Systems"), paying particular attention to the time and frequency domain treatments of the RC circuit, pages

240-241 and 246-249 in Chapter 8, and the follower-integrator circuit, pages 252-256 in Chapter 9. Slides are also available introducing linear systems analysis.

2 Prelab

1. How are capacitors constructed in CMOS chip technology? There are several different possible implementations. How are they constructed in neurons? What is the capacitance per square micron of a SiO_2 capacitor with oxide thickness of 10nm? What is the capacitance per square micron area of a lipid-bilayer capacitance with thickness of 5nm? (You will need to look up the dielectric constants for SiO_2 and lipid bilayers; remember to provide your sources in your writeup. One standard source for lipid bilayers is Ohki, Shinpei. "Dielectric constant and refractive index of lipid bilayers." Journal of Theoretical Biology 19.1 (1968): 97-115\footnote{\url{https://www.sciencedirect.com/science/article/pii/0022519368900088%7D

- MOS capacitors are made out of a semi-conductor substrate (n or p type), an insulator (silicon dioxide layer), and a metal (heavily doped polycrystalline silicon) electrode (the gate). Drain and source of a classic FET can be grounded, in order to mimic the previous MOS capacitor architecture.
- In neurons, the membrane (or phospholipid-bilayer) acts as a capacitor. The charge carriers would be the ions.
- $C = \epsilon \frac{A}{d}$ with $\epsilon = \epsilon_0 \epsilon_{SiO_2}$ all constants, transistor Width and Length (useless as we want capacitance per square micron) from practical 3.

$$\epsilon_0 = 8.86 \times 10^{-12} [F \cdot m - 1] \tag{1}$$
$$\epsilon_{SiO_2} = 3.9 \tag{2}$$
$$d = 10 \times 10^{-9} [m] \tag{3}$$
$$A = 1[m^2] = 1 \times 10^{-12} [\mu m^2] \tag{4}$$

Plugging everything in we get:

$$C = 8.86 \times 10^{-12} \times 3.9 \times \frac{1 \times 10^{-12}}{10 \times 10^{-9}} = 0.0034[F/m^2] = 3.4[fF/\mu m^2] \tag{5}$$

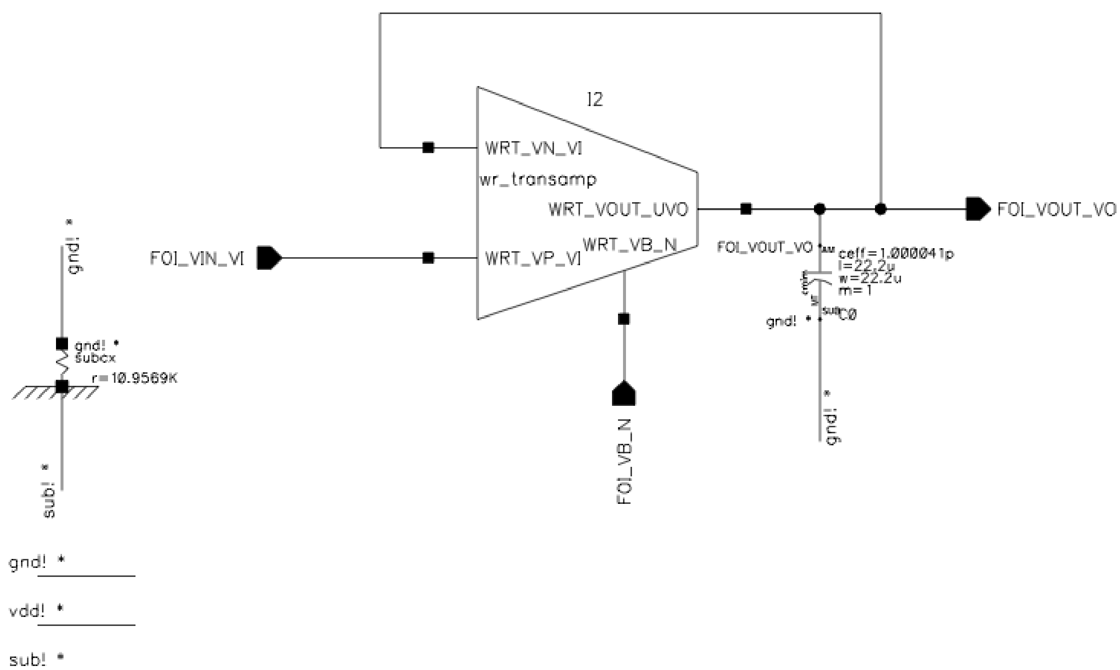
- $C = \epsilon \frac{A}{d}$ with $\epsilon = \epsilon_0 \epsilon_{Bilayer}$ (from the "Dielectric constant and refractive index of lipid bilayers"), transistor Width and Length from practical 3.

$$\epsilon_0 = 8.86 \times 10^{-12}[F \cdot m - 1] \tag{6}$$
$$\epsilon_{Bilayer} = 2.2 \tag{7}$$
$$d = 5 \times 10^{-9}[m] \tag{8}$$
$$A = 1[m^2] = 1 \times 10^{-12}[\mu m^2] \tag{9}$$

Plugging everything in we get:

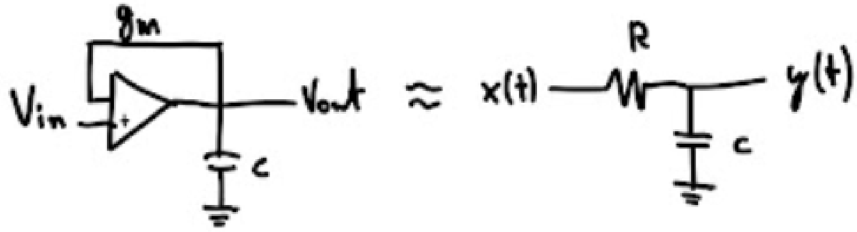
$$C = 8.854 \times 10^{-12} \times 2.2 \times \frac{1 \times 10^{-12}}{10 \times 10^{-9}} = 1.9[fF/\mu m^2] \tag{10}$$

2. Derive the transfer function $H(s) = \frac{V_{out}}{V_{in}}$ for the follower-integrator, using the s-plane notation, expressed in terms of complex frequency s and the time constant τ.



Assuming small signal regime:

Follower integrator Transfer Function



1) Apply KCL

current through capacitor $i = C\frac{dV}{dt}$

(1) $i_R = i_C$

$G = \frac{k}{2U_T}$

(2) $I_b \tanh\left(\frac{\kappa(V_{in} - V_{out})}{2U_T}\right) = C\frac{dV_{out}}{dt}$

(2.1) $G(V_{in} - V_{out}) = C\frac{dV_{out}}{dt}$

(3) $\frac{(V_{in} - V_{out})}{R} = C\frac{dV_{out}}{dt}$

(4) $V_{in} - V_{out} = \underbrace{RC}_{\tau}\frac{dV_{out}}{dt}$

(5) $V_{in} = \tau\frac{dV_{out}}{dt} + V_{out}$

(6) $x(t) = \tau\dot{y}(t) + y(t)$

$y(t)$ is of the form: $y(t) = Ye^{-\frac{t}{\tau}}$
$= Ye^{st}$

2) To Laplace domain

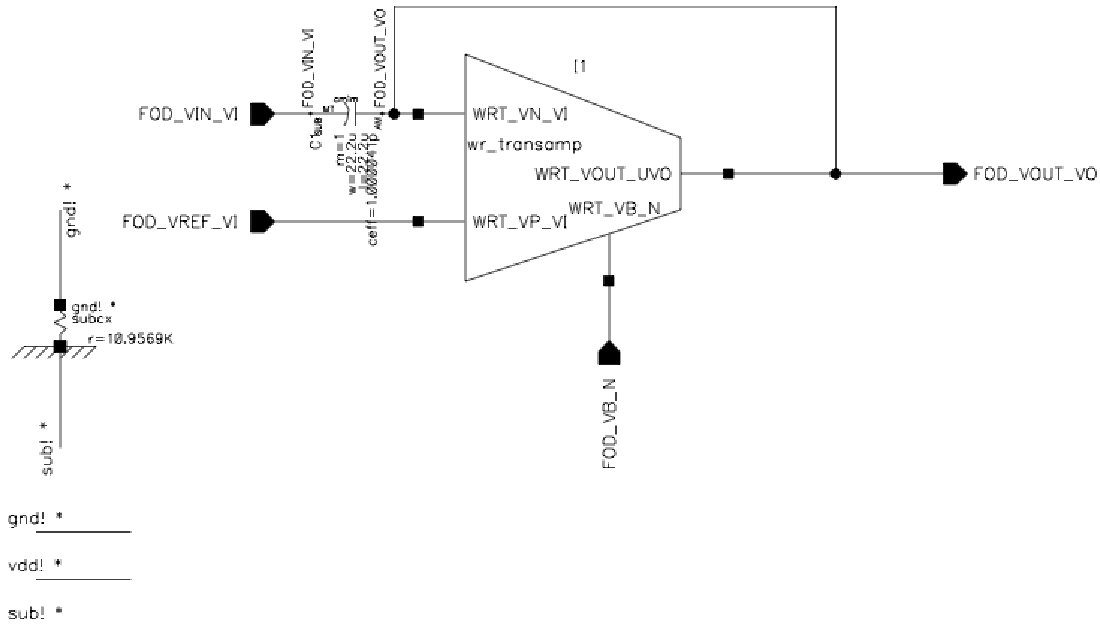
s operator $\simeq \frac{d(\dots)}{dt}$ complex freq. domain

(1) $X = \tau \cdot s \cdot Y + Y$

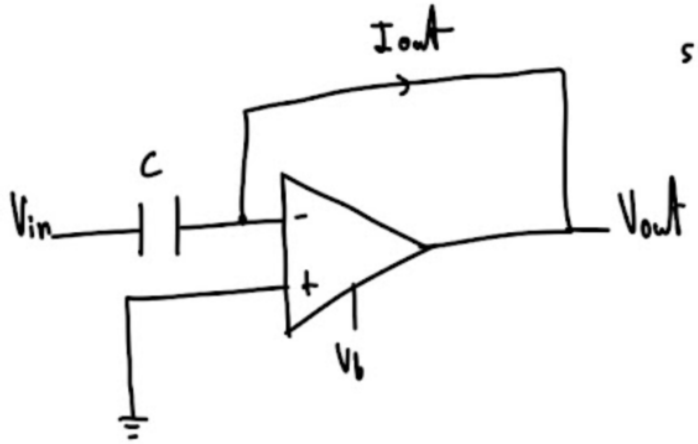
(2) $X = Y(\tau \cdot s + 1)$

(3) $\boxed{\frac{Y}{X} = \frac{1}{(\tau \cdot s + 1)} = H(s) = \frac{V_{out}}{V_{in}}}$

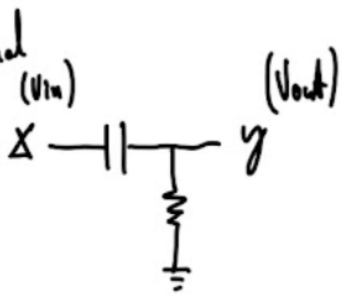
3. Compute the transfer function $H(s) = \frac{V_{out}}{V_{in}}$ for the follower-differentiator, using the s-plane notation, expressed in terms of complex frequency s and the time constant τ.



Follower diff. transfer function



≈



1)

(1) $i_C = i_R$

(2) $C\frac{d(V_{in}-V_{out})}{dt} = G V_{out}$

2) $\tau \frac{d(V_{in}-V_{out})}{dt} = V_{out}$

3) $\tau \dot{x} - \dot{y} = y$

4) $\tau sX - \tau sY = Y$

5) $\tau sX = Y + \tau sY$

6) $\tau sX = Y(\tau s + 1)$

7) $\boxed{\frac{Y}{X} = \frac{\tau s}{(\tau s + 1)} = H(s)}$

4. Compute the magnitude $|H(s)|$ for the follower integrator for input angular frequency ω. At what frequency f in Hz does the power drop to half its low frequency value (amplitude drops to $1/\sqrt{2}$)?

$$s = \omega j$$

$$2)\ H(j\omega) = \frac{1}{1 + j\omega\tau} \qquad j^2 = -1$$

$$|H(j\omega)| = \frac{1}{\sqrt{1 + (\omega\tau)^2}} \qquad (8.9.4)$$

$$= \frac{\sqrt{1+(\omega\tau)^2}}{(1+(\omega\tau)^2)} = \frac{1}{\sqrt{2}}$$

$\rightarrow$ solve for ω

$$\sqrt{2+(\omega\tau)^2 \cdot 2} = 1 + (\omega\tau)^2$$

$$2+(\omega\tau)^2 \cdot 2 = (1+(\omega\tau)^2)^2$$

$$2 + \cancel{2\omega^2\tau^2} = 1 + \omega^4\tau^4 + \cancel{2\omega^2\tau^2}$$

$$\omega^4\tau^4 = 1$$

$$\omega\tau = 1$$

$$\omega = \frac{1}{\tau}$$

$$F = \underbrace{\frac{\omega}{2\pi}}_{\text{basics}} = \frac{1}{2\pi\tau} \quad //$$

(Where F is in Hz)

5. Compare the simple *RC* integrator, constructed from a resistor and a capacitor, and the follower-integrator to show how the transfer function falls short in describing the follower integrator. In particular, how does the follower integrator respond to large signal inputs? This question is related to the next one, which is

When the input to the follower-integrator is large, the output current (from the transconductance amplifier) will saturate at $\pm I_b$ and act as a constant current source (and not as a linear conductance). Also, while $|V_{out} - V_{in}| > 4U_T$, V_{out} is linear (vs time). As small signal regime is entered, V_{out} increases or decreases exponentially (and the amp acts like a linear conductance).

6. What does ''small-signal'' mean? In other words, what voltage range will this regime correspond to? For the follower-integrator circuit is it the *amplitude* of the input or the output or the *difference* between the two that matters? Why?

Small signal means : $|V_{out} - V_{in}| < 4U_T$ It's the difference between the two that matters, if it is greater than 4UT, the response will be linear instead of exponential

3 Setup

3.1 Connect the device

In [ ]:
```
# import the necessary library to communicate with the hardware
import pyplane
import time
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
# create a Plane object and open the communication
if 'p' not in locals():
    p = pyplane.Plane()
    try:
        p.open('/dev/ttyACM0')
    except RuntimeError as e:
        del p
        print(e)
```

In [ ]:
```
p.get_firmware_version()
```

Out[ ]:
```
(1, 8, 4)
```

In [ ]:
```
# Send a reset signal to the board, check if the LED blinks
p.reset(pyplane.ResetType.Soft)
time.sleep(0.5)
# NOTE: You must send this request events every time you do a reset operetion, othe
# Because the class chip need to do handshake to get the communication correct.
p.request_events(1)
```

In [ ]:
```
# Try to read something, make sure the chip responses
p.read_current(pyplane.AdcChannel.GO0_N)
```

Out[ ]:
```
2.4169921175598574e-07
```

3.1 Chip configuration

Both circuits we use today uses the same configuration, so we just need to do it once at the beginning.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_aerc_event(
    pyplane.Coach.CurrentOutputSelect.SelectLine0,
    pyplane.Coach.VoltageOutputSelect.SelectLine1,
    pyplane.Coach.VoltageInputSelect.SelectLine2,
    pyplane.Coach.SynapseSelect.NoneSelected,0)])
```

3.2 Bias Generator (BiasGen or BG)

In a simplified form, the output of a branch of the BiasGen will be the gate voltage V_b for the bias current I_b, and if the current mirror has a ratio of w and the bias transistor operates in subthreshold-saturation:

$$I_b = w\frac{BG_{fine}}{256}I_{BG_{master}} \tag{11}$$

Where $I_{BG_{master}}$ is the `BiasGenMasterCurrent` $\in \{60\ \mathrm{pA}, 460\ \mathrm{pA}, 3.8\ \mathrm{nA}, 30\ \mathrm{nA}, 240\ \mathrm{nA}\}$, BG_{fine} is the integer fine value $\in [0, 256)$

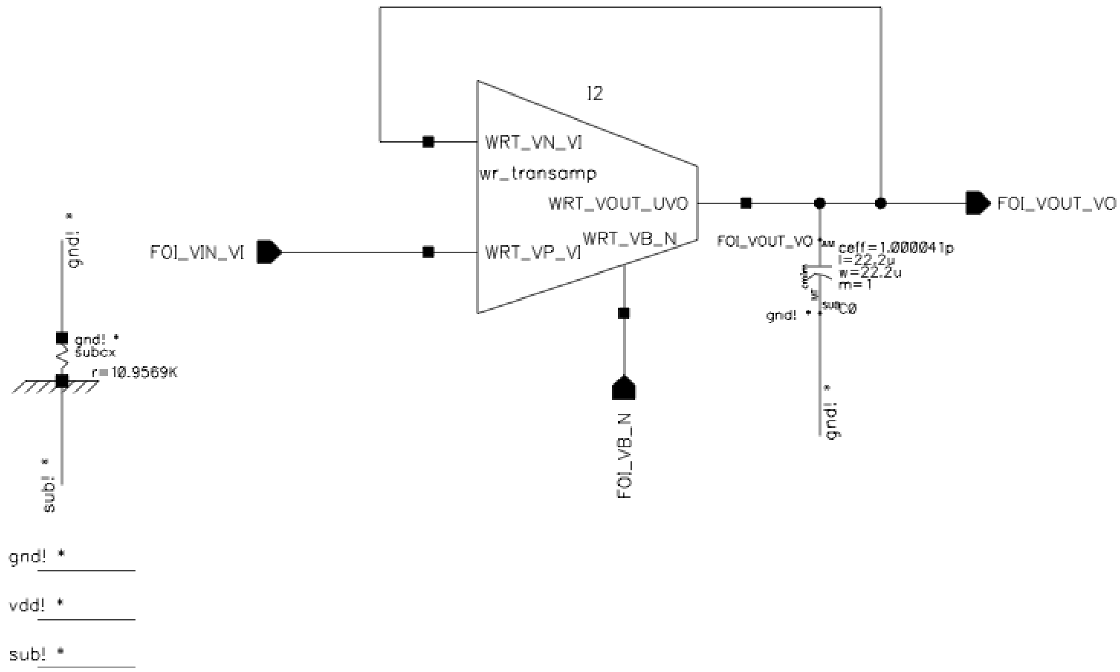
To set a bias, use the function similar to the following:

```
p.send_coach_event(pyplane.Coach.generate_biasgen_event(\

pyplane.Coach.BiasAddress.BIAS_NAME_STARTS_WITH_THREE_LETTER_CIRCUIT_NAME,
\
    pyplane.Coach.BiasType.MATCH_LAST_CHAR_OF_BIAS_NAME, \
    pyplane.Coach.BiasGenMasterCurrent.MASTER_CURRENT, FINE_VALUE))
```

4 Follower-integrator (FOI)

4.1 Schematic and pin map



V_{in} **= FOI_VIN_VI = AIN9**

V_{out} **= FOI_VOUT_VO = ADC[10]**

C **= ceff = 1 pF**

- The W/L of the output transistor of the BiasGen is 4u/12u, and the bias transistor M_b of wide-range-transamp is 1u/1u, which give a ratio of w **= 3**. This means if you set the bias current to (3.8 nA, 225), you will get $I_b = 3.8 \times \frac{67}{256} \times 3 = 3$ nA instead of 1 nA.

4.2 Time-domain response of small signal

4.2.1 Set parameters

- The maximum sampling frequency of the ADC is 100 kHz (10 μ s), and the maximum number of samples is 250. Let's say if we want to cover the time period of 10τ with these 250 samples, what should the value of τ be (assume $\kappa = 1$)?

In order to cover a period of 10τ with n samples, each sample should have the duration

$$\Delta t = \frac{10\tau}{n}.$$

As

$$f = \frac{1}{\Delta t},$$

it can be inferred that

$$\tau = \frac{n}{10f}.$$

Setting $f = 100 \cdot 10^3 \mathrm{Hz}$ and $n = 250$ yields

$$\tau = 250 \cdot 10^{-6} \mathrm{s} = 250 \mu \mathrm{s}.$$

- To get this τ, what is the value of I_b?

From the prelab, it is known that

$$\tau = \frac{2U_T}{I_b \kappa} C.$$

With $U_T \approx 25 \mathrm{mV}$, $\kappa = 1$, $\tau = 250 \mu \mathrm{s}$ and $C = 1 \mathrm{pF}$ it follows that

$$I_b = \frac{2U_T}{\tau} C \approx 200 \mathrm{pA}.$$

- What `MasterCurrent` and `fine` value should we use for I_b? Please set below.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(
    pyplane.Coach.BiasAddress.FOI_VB_N,
    pyplane.Coach.BiasType.N,
    pyplane.Coach.BiasGenMasterCurrent.I460pA, 37)])
```

For a given value of I_b, it is sensible to choose the next largest value for $I_{BG_{master}}$. In the case of $I_b = 200 \mathrm{pA}$, this is

$$I_{BG_{master}} = 460 \mathrm{pA}.$$

With $I_b = 200 \mathrm{pA}$ and $w = 3$, it can be calculated that

$$BG_{fine} = \frac{256 I_b}{w I_{BG_{master}}} \approx 37.10,$$

which has to be rounded down to

$$BG_{fine} \approx 37$$

as BG_{fine} must be an integer. This yields the bias current

$$I_b = w \frac{BG_{fine}}{256} I_{BG_{master}} \approx 199.5 \mathrm{pA}.$$

- What should the input before the step $(V_{in}(t < 0))$ be (Hint: common-mode voltage of the (wide-range) transamp)? What is the corresponding V_{out} in steady state

$\left(V_{out}(t=0^{-})\right)$?

In [ ]:
```
Vi_pre = 0.9
```

In stead state, the value of V_{in} should be high enough that the transamp is capable of operating, e.g.

$$V_{in}(t<0)=0.9\mathrm{V}.$$

The value of the output voltage will then also be

$$V_{out}(t<0)=0.9\mathrm{V}.$$

- What does "small signal" mean? What should the magnitude of the step input (ΔV_{in}) be so that it can be treated as "small"?

In [ ]:
```
dVi = 0.09
```

In the prelab, it was determined that small signals can be assumed if the change between the input and output voltage is limited to approximately $^{(1)}$ $|V_{out}-V_{in}|=|\Delta V_{in}|<4U_T$ over one time interval τ

$$\frac{d\left|\Delta V_{in}\right|}{d\tau}<4U_T\approx 100\mathrm{mV}.$$

As a step response is considered, the change $|\Delta V_{in}|$ occurs instantaneously, and the above condition can simply be written as

$$|\Delta V_{in}|<4U_T\approx 100\mathrm{mV}.$$

Thus, it is sensible to set e.g. $|\Delta V_{in}|\approx 90\mathrm{mV}$.

$^{(1)}$ For $\kappa\approx 1$.

4.2.2 Data aquisition

- Verify the steady state

In [ ]:
```
p.set_voltage(pyplane.DacChannel.AIN9, Vi_pre)
time.sleep(0.5) # wait for the circuit to reach steady state
Vo_pre = p.read_voltage(pyplane.AdcChannel.AOUT10)
print(Vo_pre)
```

```
0.895092785358429
```

- There is a small offset of the DAC of around -10 mV, but the ADC is correct, so we want to correct `Vi_pre` accordingly.

Ignore offset as setting resolution of the DAC isn't good enough to compensate the offset.

In [ ]:
```
Vi_off = Vo_pre - Vi_pre
Vi_pre_corr = Vi_pre - Vi_off
print(Vi_pre_corr)
```

```
0.9049072146415711
```

- Acquire the step response.

In [ ]:
```
p.set_bit_depth(pyplane.BitDepth(10))
```

In [ ]:
```
Vout_ex_4_2_2 = np.zeros(511)
for _ in range(10):
    p.set_voltage(pyplane.DacChannel.AIN9, Vi_pre_corr)
    time.sleep(0.5) # wait for the circuit to reach steady state
    Vout_ex_4_2_2 = Vout_ex_4_2_2 + p.acquire_transient_response(pyplane.DacChannel
    time.sleep(0.5) # wait to receive the measured data from the microcontroller

Vout_ex_4_2_2 /= 10
print(Vout_ex_4_2_2)
```

```
[0.90194092 0.90298828 0.90838623 0.91313964 0.91773192 0.92248535
 0.92675537 0.93110596 0.93642334 0.93924316 0.94375488 0.94754151
 0.95044189 0.95406739 0.9571289  0.95922363 0.96301026 0.96478271
 0.9671997  0.97001954 0.97138917 0.97300049 0.97541748 0.97630371
 0.9775122  0.97968749 0.98041259 0.98121827 0.98307129 0.98371583
 0.98460206 0.98572998 0.98572998 0.98677735 0.98750244 0.98782471
 0.9885498  0.98895262 0.98919434 0.9897583  0.99032226 0.99064453
 0.99128905 0.99161132 0.99201416 0.99249755 0.99281982 0.99298095
 0.99257811 0.99298095 0.99298096 0.99314209 0.99418945 0.99386719
 0.99354492 0.99410889 0.99418945 0.99410889 0.99523683 0.99475341
 0.99499512 0.99483399 0.99451172 0.99515625 0.99507568 0.99507568
 0.99580078 0.99547852 0.99491455 0.99572023 0.99507568 0.99531738
 0.99563965 0.99588134 0.99563965 0.99555908 0.99604249 0.99515625
 0.99604249 0.99555908 0.99547851 0.99588135 0.99547852 0.99547852
 0.9964453  0.99523681 0.99555908 0.99620361 0.99555908 0.99572022
 0.99612304 0.99547852 0.99612305 0.99620362 0.99563966 0.99580078
 0.99596192 0.99588135 0.99555907 0.99572021 0.99547852 0.99604249
 0.99604248 0.99539795 0.99555908 0.99652588 0.99531738 0.99515625
 0.99547851 0.99539794 0.99580078 0.99604248 0.99547852 0.99588135
 0.99612306 0.99580078 0.99580078 0.99596192 0.99580078 0.99644531
 0.99612306 0.99604248 0.99588135 0.99612306 0.99572021 0.99547852
 0.99555909 0.99555908 0.99628418 0.99604248 0.99499511 0.99628418
 0.99580079 0.99555908 0.99628417 0.99555909 0.99572021 0.99604248
 0.99523682 0.99572021 0.99604248 0.99531739 0.99539796 0.99580078
 0.99555908 0.99555908 0.99604248 0.99539795 0.99604248 0.99580078
 0.99572022 0.99555909 0.99612305 0.99563965 0.99555908 0.99628418
 0.99604248 0.99612303 0.99588135 0.99539795 0.99459229 0.99684814
 0.99547852 0.99580079 0.99588135 0.99547852 0.99604248 0.99563965
 0.99563965 0.99596191 0.99612305 0.99588135 0.99636474 0.99547852
 0.99588135 0.99620361 0.99612306 0.99563965 0.99604248 0.99612306
 0.99612305 0.99620361 0.99620363 0.99572022 0.99604248 0.99636475
 0.99612305 0.99596192 0.99588135 0.99547852 0.99604248 0.99563965
 0.99555908 0.99644532 0.99555909 0.99531739 0.99588135 0.99499512
 0.99588134 0.99580078 0.99539796 0.99539794 0.99604248 0.99563965
 0.99636474 0.99612306 0.99644532 0.99531738 0.99580079 0.99572022
 0.99563965 0.99588135 0.99604248 0.99572021 0.99620362 0.99563966
 0.99539795 0.99588135 0.99563965 0.99475343 0.99580079 0.99572022
 0.99572022 0.99580079 0.99539794 0.99628417 0.99612306 0.99539795
 0.99588134 0.99580079 0.99539795 0.99596191 0.99580079 0.99555909
 0.99604248 0.99588135 0.99580078 0.99596191 0.99596192 0.99547852
 0.99596191 0.99515625 0.99563965 0.99563965 0.99507569 0.99515625
 0.99580079 0.99588135 0.99523682 0.99563966 0.99572023 0.99515626
 0.99612306 0.99580079 0.99531739 0.99596192 0.99588135 0.99572022
 0.99604248 0.99580079 0.99507568 0.99596192 0.99572021 0.99555908
 0.99580078 0.99531738 0.99523682 0.99620361 0.99523681 0.99451172
 0.99612305 0.99604248 0.99636475 0.99596192 0.99547852 0.99604248
 0.99620361 0.99523681 0.99588135 0.99604248 0.99539795 0.99580078
 0.99596192 0.99580079 0.99604248 0.99588135 0.99604248 0.99596192
 0.99580078 0.99628417 0.99572021 0.99572023 0.99547852 0.99612305
 0.99555909 0.99547852 0.99555908 0.99563966 0.99555908 0.99563965
 0.99572022 0.99515625 0.99539796 0.99636475 0.99539796 0.99612306
 0.99604249 0.99499512 0.99636474 0.99563965 0.99539795 0.99620362
 0.99572023 0.99531739 0.99628418 0.99523682 0.99539796 0.99620362
 0.99547852 0.99523681 0.99499512 0.99531739 0.99523681 0.99580079
 0.99515625 0.99555908 0.99580079 0.99547852 0.99596191 0.99588135
 0.99499511 0.99539794 0.99620362 0.99563965 0.99652588 0.99572023
```

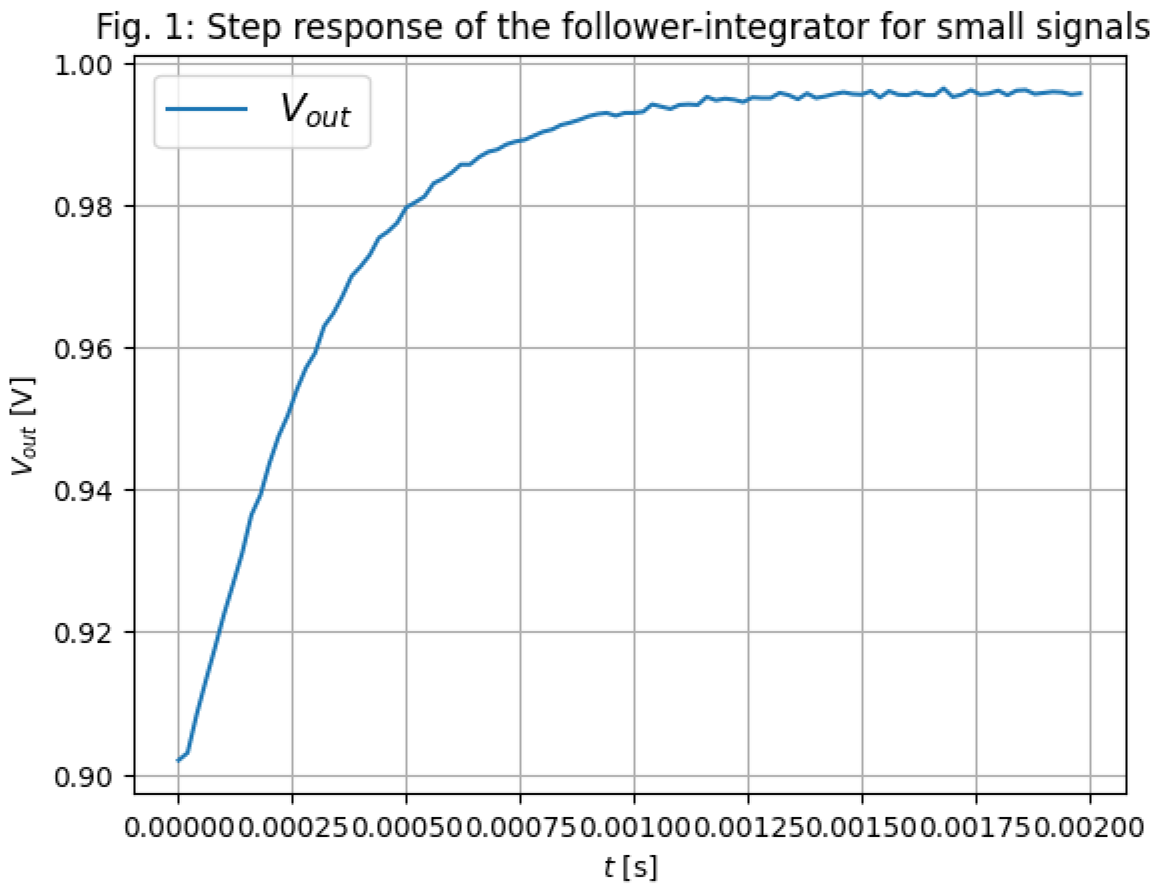
```
 0.99628417 0.99596191 0.99588135 0.99580078 0.99604247 0.99572022
 0.99555908 0.99612305 0.99572022 0.99563965 0.99588135 0.99604248
 0.99523681 0.99652588 0.99580079 0.99539795 0.99668702 0.99588135
 0.99507568 0.99620363 0.99547852 0.99523681 0.99636475 0.99580079
 0.99539796 0.99604248 0.99572022 0.99507568 0.99604249 0.99515625
 0.99515625 0.99620362 0.99547852 0.99539795 0.99547851 0.99523681
 0.99539795 0.99596192 0.99547852 0.99531739 0.99555908 0.99563965
 0.99572021 0.99580079 0.99563965 0.99555908 0.99596192 0.99668702
 0.99604248 0.99628419 0.99596192 0.99563965 0.99572022 0.99668702
 0.99588136 0.99596191 0.996687   0.99612305 0.99588135 0.99483398
 0.99588135 0.99515626 0.99555908 0.99596192 0.99539796 0.99483398
 0.99660645 0.99531738 0.99523681 0.99628419 0.99555908 0.99539794
 0.99628418 0.99515625 0.99523681 0.99588135 0.99580079 0.99539795
 0.99604248 0.99555908 0.99475341 0.99588136 0.99523681 0.99515625
 0.99580079 0.99547852 0.99596192 0.99604249 0.99531739 0.99563965
 0.99700928 0.99539796 0.99467285 0.99596192 0.99547852 0.99555908
 0.99604248 0.99563965 0.99531738 0.99604249 0.99547852 0.99628416
 0.99588135 0.99580079 0.99572022 0.99580079 0.99588135 0.99612304
 0.99580079 0.99555908 0.99604247 0.99572021 0.99547851 0.99588135
 0.99588135 0.99572022 0.99612305 0.99547853 0.99523681 0.99620362
 0.99604249 0.99507568 0.99660644 0.99660646 0.99588135 0.99620361
 0.99555909 0.99563965 0.99604249 0.99563965 0.99499511 0.99604248
 0.99539796 0.99499512 0.99596192 0.99539796 0.99539796 0.99596192
 0.99580078 0.99539795 0.99660645 0.99572022 0.99636475 0.99604249
 0.99507568 0.99620361 0.99620362 0.99563965 0.99604248 0.99604249
 0.99531739 0.99572022 0.99620361 0.99563965 0.99539794 0.99612306
 0.99596192 0.99523681 0.99572023 0.99523681 0.99572021 0.99547852
 0.99539796 0.99596192 0.99531739 0.99555908 0.99644532 0.99596192
 0.99547852 0.99652588 0.99531739 0.99539795 0.99628418 0.99547852
 0.99539795]
```

- plot V_{out}

In [ ]:

```python
import matplotlib.pyplot as plt
import numpy as np

Vout_ex_4_2_2 = np.loadtxt('data_ex_4_2_2.csv', delimiter=',')
t=np.arange(0,len(Vout_ex_4_2_2))*0.00002
plt.plot(t[:100], Vout_ex_4_2_2[:100])
plt.xlabel('$t$ [s]')
plt.ylabel('$V_{out}$ [V]')
plt.legend(['$V_{out}$'],prop={'size': 14})
plt.title('Fig. 1: Step response of the follower-integrator for small signals.')
plt.grid()
plt.show()
```



- save data

In [ ]:
```
# if the data looks nice, save it!
data_ex_4_2_2= [Vout_ex_4_2_2]
# save to csv file
np.savetxt('data_ex_4_2_2.csv', data_ex_4_2_2, delimiter=',')
```

4.2.3 Compute the time constant τ by reading from the decay in the curve

- Assume the input step happens at exactly $t = 0$, what is the expression of V_{out} at $t = \tau$?

In the prelab, the transfer function of the follower-integrator was derived to be

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{1}{\tau s + 1}.$$

It is known that the unit step can be described in the s-plane as

$$U(s) = \frac{\Delta V_{in}}{s}.$$

The unit step response can thus be evaluated to

$V(s) = H(s)U(s) = \dfrac{\Delta V_{in}}{s(\tau s + 1)}$. Transformed back into the time domain, the output voltage can now be described by

$$V_{out}(t) = \Delta V_{in}\left(1 - e^{-\dfrac{t}{\tau}}\right) + V_{out}(t=0).$$

Thus

$V_{out}(t = \tau) = ?$

- Compute τ by "reading" this point from the curve:

In [ ]:
```
import scipy.interpolate as interpolate

f = interpolate.interp1d(Vout_ex_4_2_2,t) # Interpolate t vs. Vout
v_1tau = dVi*(1-np.exp(-1))+Vout_ex_4_2_2[0]
tau1 = f(v_1tau) # Get tau = t(Vout=1-e^(-1))
print(tau1)
```

```
0.00029625856271414445
```

- Compute the actual κ by comparing the measured τ and the estimated value in 4.2.1 with $\kappa = 1$.

In [ ]:
```
Ibe = 200
Ibm = 199.5
taue = 250
taum = tau1*10**6
kappa1 = (taue*Ibe)/(taum*Ibm)
print(kappa1)
```

```
0.8459723969493024
```

4.2.5 Analysis

- Is the "small signal" assumption validated by the measurement? Why or why not?

> Yes, because with the small signal we only see the exponential response and there is no (or very little) linear portion.

- Is the assumption "input step happens at exactly $t = 0$" validated by the measurement? How can you get the actual time it takes place?

> It almost happens at time 0, we can see a small delay in the impulse response that is measured (on the plot).

4.3 Time-domain response of large signal

4.3.1 Set parameters

- What does "large signal" mean? What should the magnitude of the step input (ΔV_{in}) be so that it can be treated as "large"?

In [ ]:
```
dVi = 0.3
```

Large signal behavior can be assumed when $|V_{in} - V_{out}| > 4U_T$, which translates to the step input

$|\Delta V_{in}| > 4U_T \approx 100\mathrm{mV}.$

Thus, it is reasonable to assumed that e.g. $\Delta V_{in} = 300\mathrm{mV}.$

- What should the input before the step $(V_{in}(t < 0))$ be (Hint: common-mode voltage of the (wide-range) transamp)? What is the corresponding V_{out} in steady state $\left(V_{out}(t = 0^-)\right)$?

In [ ]:
```
Vi_pre = 0.9
```

In stead state, the value of V_{in} should be high enough that the transamp is capable of operating, e.g.

$V_{in}(t < 0) = 0.9\mathrm{V}.$

The value of the output voltage will then also be

$V_{out}(t < 0) = 0.9\mathrm{V}.$

- Let's still set ADC sampling rate as 100 kHz (10 μs), and the maximum number of samples as 250. If we want the linear part to be about 40% of the whole sampled period, what should the slew rate be (Hint: in [V/s])?

Assuming that $f = 100 \cdot 10^3\mathrm{Hz}$ and that $n = 250$, the total sample period will be

$t_{tot} = \frac{n}{f} = 2.5 \cdot 10^{-3}\mathrm{s} = 2.5\mathrm{ms}.$

Therefore, the linear part should be approximately

$$t_{lin} = 0.4t_{tot} = 1\text{ms}$$

long.

The slew rate limits the circuit as long as the $\tanh$ in I_{out} is more or less saturated. In terms of $V_{in} - V_{out}$, this is given when the extrapolated linear section of I_{out} intercepts I_b. Since the linear section of I_{out} is given by

$$I_{out} = g_m \left(V_{in} - V_{out}\right) = I_b \frac{\kappa}{2U_T}(V_{in} - V_{out}).$$

Thus (assuming that $\kappa = 1$)

$$V_{in} - V_{out} = \frac{2U_T}{\kappa} = 50\text{mV}$$

when the circuit is no longer limited by the slew rate. Since $\Delta V_{in} = 300\text{mV}$, the corresponding change in output voltag thus has to reach

$$\Delta V_{out} = 250\text{mV}.$$

The corresponding slew rate would thus be

$$sr = \frac{\Delta V_{out}}{t_{lin}} = \frac{250\text{mV}}{1\text{ms}} = 250\frac{V}{s}.$$

- To get this slew rate, what is the value of I_b (Hint: C = 1 pF)?

For large signals, the output current of the transamp is saturated and the transamp can thus be treated as a constant current source with $I_{out} = I_b$. This current now charges the capacitor linearily. With $t_{lin} = 1\text{ms}$, $\Delta V_{out,lin} = 250\text{mV}$ and $C = 1\text{pF}$, it can be determined what value I_b must have in order to achieve the slew rate determined in the previous exercise

$$I_{out} = I_b = \frac{C}{t_{lin}} \Delta V_{out,lin} = C \cdot sr = 250\text{pA}.$$

- What `MasterCurrent` and `fine` value should we use for I_b? Please set below.

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(
    pyplane.Coach.BiasAddress.FOI_VB_N,
    pyplane.Coach.BiasType.N,
    pyplane.Coach.BiasGenMasterCurrent.I460pA, 46)])
```

For a given value of I_b, it is sensible to choose the next largest value for $I_{BG_{master}}$. In the case of $I_b = 250\text{pA}$, this is

$I_{BG_{master}} = 460\text{pA}.$

With $I_b = 250\text{pA}$ and $w = 3$, it can be calculated that

$$BG_{fine} = \frac{256 I_b}{w I_{BG_{master}}} \approx 46.38,$$

which has to be rounded down to

$$BG_{fine} \approx 46$$

as BG_{fine} must be an integer. This yields the bias current

$$I_b = w \frac{BG_{fine}}{256} I_{BG_{master}} \approx 248.0\text{pA}.$$

4.3.2 Data aquisition

- Verify the steady state

In [ ]:
```
p.set_voltage(pyplane.DacChannel.AIN9, Vi_pre)
time.sleep(0.5) # wait for the circuit to reach steady state
Vo_pre = p.read_voltage(pyplane.AdcChannel.AOUT10)
print(Vo_pre)
```

```
0.897509753704071
```

- There is a small offset of the DAC of around -10 mV, but the ADC is correct, so we want to correct the DAC accordingly.

In [ ]:
```
Vi_off = Vo_pre - Vi_pre
Vi_pre_corr = Vi_pre - Vi_off
print(Vi_pre_corr)
```

```
0.902490246295929
```

- Acquire the step response

In [ ]:
```
p.set_bit_depth(pyplane.BitDepth(10))
```

In [ ]:
```
Vout_ex_5_2_2 = np.zeros(252)
for _ in range(10):
    p.set_voltage(pyplane.DacChannel.AIN9, Vi_pre_corr)
    time.sleep(0.5) # wait for the circuit to reach steady state
    Vout_ex_5_2_2 = Vout_ex_5_2_2 + p.acquire_transient_response(pyplane.DacChannel
    time.sleep(0.5) # wait to receive the measured data from the microcontroller

Vout_ex_5_2_2 /= 10
print(Vout_ex_5_2_2)
```

```
[0.89670411 0.89799317 0.9005713  0.90250487 0.90516357 0.90733888
 0.91015869 0.91185058 0.91475098 0.91724855 0.91934326 0.92127686
 0.92409668 0.92619141 0.92917237 0.93175048 0.93408691 0.93618164
 0.9389209  0.94190185 0.94375489 0.94625244 0.94866943 0.95052246
 0.95374513 0.95632325 0.95825683 0.96002929 0.96373535 0.96510499
 0.96728027 0.97138917 0.97283936 0.97517579 0.97791504 0.98065431
 0.98307128 0.98581055 0.98677735 0.99016114 0.99394775 0.99515625
 0.99781494 1.00039307 1.0017627  1.00482422 1.00796629 1.00941651
 1.01296145 1.01513673 1.01666747 1.02013184 1.02254883 1.02456295
 1.02706054 1.03004149 1.03213623 1.03616456 1.03801757 1.04027344
 1.04389893 1.04542968 1.04792726 1.05098876 1.05227783 1.05541993
 1.05799804 1.05985107 1.06250976 1.06476563 1.06661867 1.06943849
 1.0724194  1.07386963 1.07781738 1.07950928 1.08055663 1.08345703
 1.08700196 1.08821043 1.09086914 1.09296386 1.09473633 1.09787842
 1.1002954  1.10158448 1.10464602 1.10698243 1.10802979 1.11149415
 1.11342773 1.11471682 1.11769775 1.11938968 1.12051758 1.12438476
 1.12575437 1.127688   1.12978271 1.13139406 1.13244141 1.13485838
 1.13687254 1.13775876 1.13969238 1.14138429 1.14235107 1.1446875
 1.14589598 1.14758788 1.14968264 1.15024657 1.1518579  1.15451659
 1.15395263 1.15564452 1.15790039 1.15733643 1.15878661 1.16120361
 1.1614453  1.1624121  1.16402344 1.16394287 1.16507081 1.166521
 1.16635987 1.16700441 1.16901854 1.16861572 1.16982423 1.17119385
 1.17071046 1.17264403 1.17377198 1.17256348 1.17522216 1.17522219
 1.17570556 1.17667236 1.17763915 1.17659179 1.17828369 1.17908934
 1.17828369 1.17957274 1.18021729 1.17925049 1.18134521 1.18134521
 1.18062012 1.18263428 1.18327881 1.18215088 1.18303711 1.18343993
 1.18384279 1.18424561 1.1842456  1.18513186 1.18561524 1.1852124
 1.18617921 1.18593751 1.18593751 1.18650147 1.18706541 1.18642088
 1.18674316 1.18843507 1.18714601 1.18746825 1.18827393 1.18770995
 1.1883545  1.18907958 1.18795167 1.1885962  1.18964356 1.18803222
 1.18916017 1.18956298 1.18891845 1.1898047  1.18988525 1.18891845
 1.19020753 1.19020753 1.18916017 1.1905298  1.19149656 1.18899903
 1.19061037 1.1898047  1.18980467 1.19061035 1.19101319 1.19012696
 1.19109374 1.19093263 1.19069092 1.19181886 1.19125488 1.19165773
 1.19133546 1.19109377 1.19173828 1.19133546 1.19149657 1.19189941
 1.19189942 1.1917383  1.19157716 1.19230224 1.19189943 1.19222169
 1.19254396 1.19197998 1.19214112 1.19302734 1.19189942 1.19230226
 1.19286621 1.19197998 1.1924634  1.19286621 1.19157715 1.19270509
 1.19262451 1.19197999 1.19270507 1.19286621 1.19197997 1.19318849
 1.19238281 1.19181885 1.19326903 1.19286621 1.19141603 1.19326905
 1.1933496  1.19181887 1.19286622 1.19286622 1.19189942 1.19310791
 1.19302734 1.19230225 1.19278564 1.19310788 1.19286622 1.19286621]
```

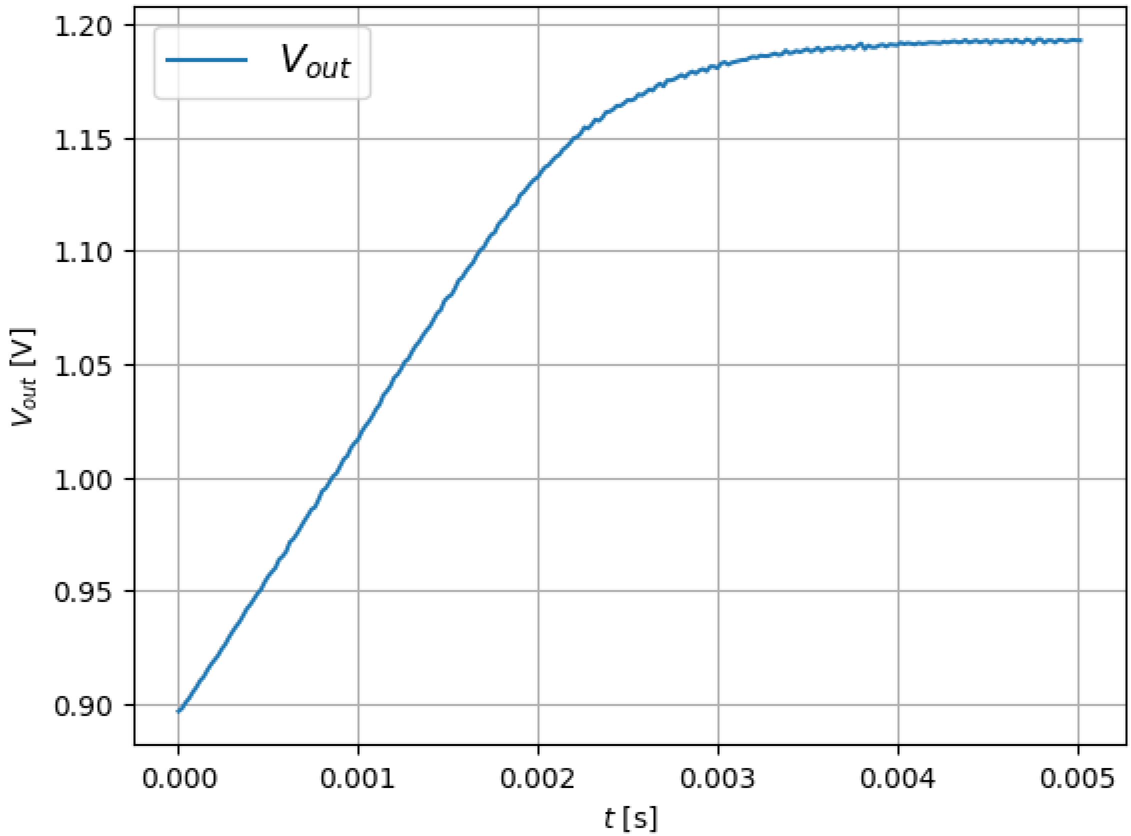
- plot V_{out}

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

Vout_ex_5_2_2 = np.loadtxt('data_ex_5_2_2.csv', delimiter=',')
t=np.arange(0,len(Vout_ex_5_2_2))*0.00002
plt.plot(t, Vout_ex_5_2_2)
plt.xlabel('$t$ [s]')
plt.ylabel('$V_{out}$ [V]')
plt.legend(['$V_{out}$'],prop={'size': 14})
plt.title('Fig. 2: Step response of the follower-integrator for large signals.')
```

```
plt.grid()
plt.show()
```

Fig. 2: Step response of the follower-integrator for large signals.




```
The Kernel crashed while executing code in the the current cell or a previous cel
l. Please review the code in the cell(s) to identify a possible cause of the failu
re. Click <a href='https://aka.ms/vscodeJupyterKernelCrash'>here</a> for more inf
o. View Jupyter <a href='command:jupyter.viewOutput'>log</a> for further details.
```

- save data

In [ ]:

```
# if the data looks nice, save it!
data_ex_5_2_2= [Vout_ex_5_2_2]
# save to csv file
np.savetxt('data_ex_5_2_2.csv', data_ex_5_2_2, delimiter=',')
```

4.3.3 Data processing

- Compute the slew rate by fitting the linear part of the curve

In [ ]:

```
import numpy as np

p0, p1 = np.polyfit(t[:20],Vout_ex_5_2_2[:20],1)
print(p0,p1)
```

```
215.73625045611496 0.9024427293028146
```

$sr = \frac{dV_{out}}{dt} = 215.73[V/\mu sec]$

- Calculate the bias current I_b from the slew rate and compare it with the set value. Comment on possible reason of any discrepancy.

> Calculated: $I_b = C \cdot sr = 215.7\text{pA}$.

> Set: $I_{bset} = 248pA$

> Discrepancy may be due to digitalization, measurement errors, transistor mismatch,...

- Compute the time constant τ and the κ by fitting the exponential part of the curve. Do they make sense?

In [ ]:
```
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

def func(x, a, b, c):
    return a * np.exp(-(1/b) * x) + c

popt, pcov = curve_fit(func, t[20:100], Vout_ex_5_2_2[20:100])

print(popt)


import scipy.interpolate as interpolate
dvi = Vout_ex_5_2_2[100]-Vout_ex_5_2_2[20]

f = interpolate.interp1d(Vout_ex_5_2_2[20:100],t[20:100]) # Interpolate t vs. Vout
v_1tau = dvi*(1-np.exp(-1))+Vout_ex_5_2_2[20]
tau1 = f(v_1tau) # Get tau = t(Vout=1-e^(-1))
tau1 = tau1 - t[20]
print(tau1)
```

```
[-705.5950271   104.29235567  706.58066005]
0.00021253705853055 68
/home/quillan/environments/neuromorphic/lib/python3.8/site-packages/scipy/optimiz
e/_minpack_py.py:881: OptimizeWarning: Covariance of the parameters could not be e
stimated
  warnings.warn('Covariance of the parameters could not be estimated',
```

$\tau = 2.12 \times 10^{-4}$

In [ ]:
```
#compute kappa
tau1 = 0.00021253705853055 68
Ibe = 200
Ibm = 199.5
taue = 250
```

```
taum = tau1*10**6
kappa1 = (taue*Ibe)/(taum*Ibm)
print('kappa: ',kappa1)
```

```
kappa:  1.1792134893972248
```

> Given the fact that they are determined based on data that is not really exponential they do not really make sense

4.3.4 Analysis

- Is the assumption "input step happens at exactly $t = 0$" validated by the measurement? How can you get the actual time it takes place?

> We don't see a delay in the response and the step happens at time 0.

- Could you give a marginal value of ΔV_{in} between "small" and large signal"? (Hint: assume for the transamp, $I_{out} \propto (V_1 - V_2)$ if $g_m|V_1 - V_2| < I_b$)

> The marginal ΔV_{in} region for large signals is very small (small exponential "component"). But for small signals it's larger (the exponential part is bigger).

4.4 Frequency-domain response

4.4.1 Methodology

In the prelab we have computed the magnitude of the transfer function $H(s)$. In this exercise we are going to measure this curve.

- What will the output signal V_{out} look like if the input V_{in} is a sine wave in steady state (after several cycles)?

The output signal will be phase shifted and scaled depending on the frequency of the input sine wave.

- In order to make the measurement more accurate, we set I_b to minimum (about 0.7 pA):

In [ ]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(
    pyplane.Coach.BiasAddress.FOI_VB_N,
    pyplane.Coach.BiasType.N,
    pyplane.Coach.BiasGenMasterCurrent.I60pA,100)])
```

- With this I_b, what is the cutoff frequency approximately?

It is known from the prelab that at the cutoff frequency,

$$\omega = 2\pi f = \frac{1}{\tau}.$$

As

$$\tau = \frac{C}{g_m} = \frac{\kappa}{2I_bU_T}C$$

the cut of frequency can thus be calculated (using $\kappa = 0.9616$ as determined in exercise 4.2.3)

$$f_c = I_b\frac{\kappa}{4\pi U_T C} \approx 2.152\text{Hz}.$$

4.4.2 Observe the input and output waveforms

- set a sine wave

In [ ]:
```
import numpy as np
wf = np.sin(np.arange(0, 8*3.14, 0.1))
```

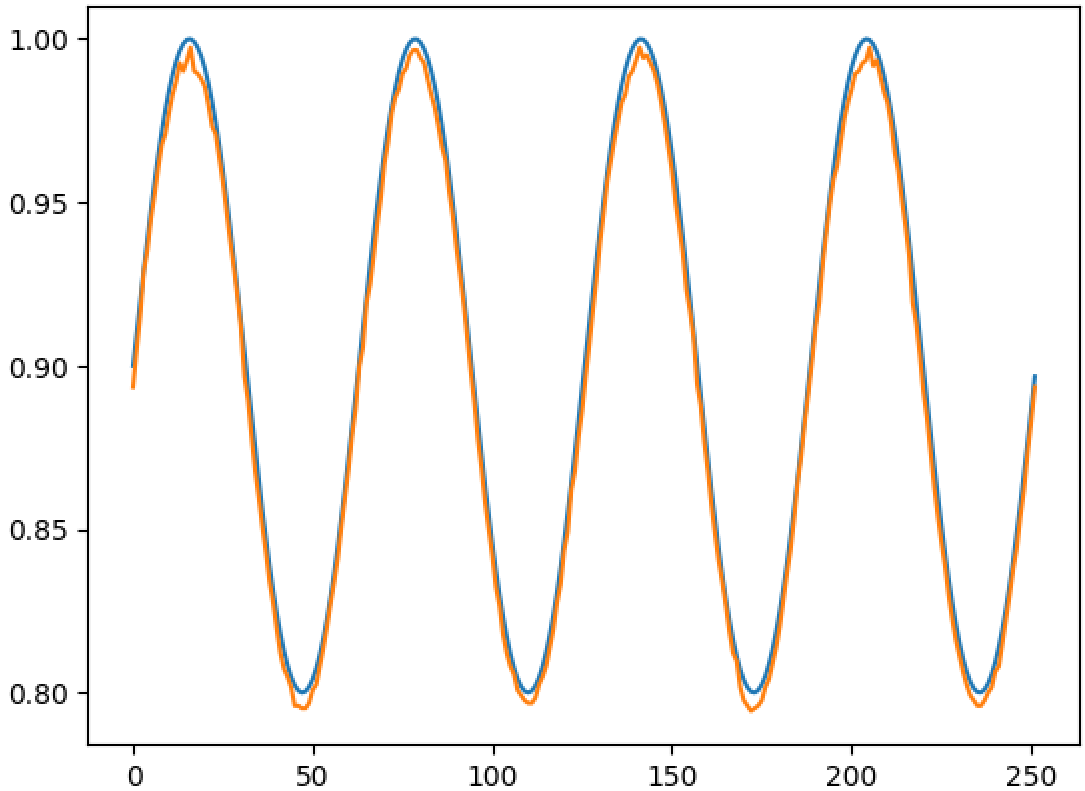
- set the input voltage (offset of the input sine wave is 0.9, amplitude of the input sine wave is 0.1)

In [ ]:
```
p.set_voltage_waveform(0.9 + 0.1 * wf)
```

- Plot V_{in} and V_{out} in the same figure

In [ ]:
```
plt.plot(0.9 + 0.1 * wf)
Vout_ex_4_4_2 = p.acquire_waveform(pyplane.DacChannel.AIN9, pyplane.AdcChannel.AOUT
plt.plot(Vout_ex_4_4_2)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7efc538deac0>]
```

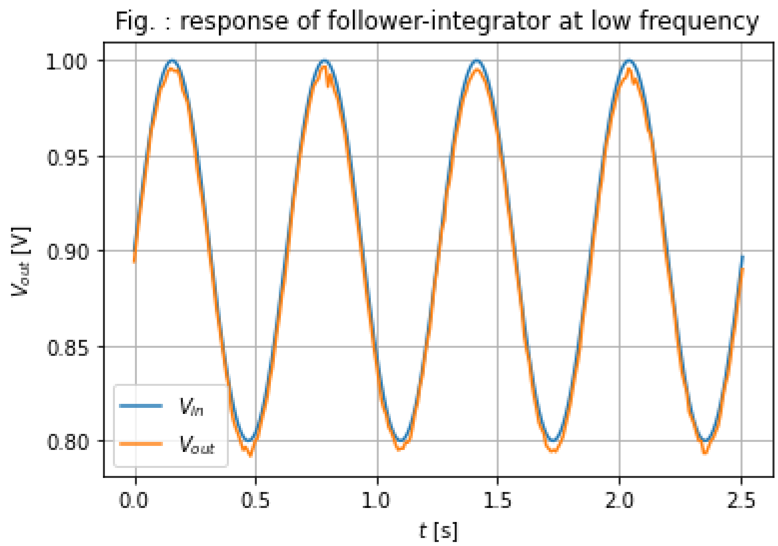


- Save data

In [ ]:
```
# if the data looks nice, save it!
data_ex_4_4_2= [Vout_ex_4_4_2]
# save to csv file
np.savetxt('data_ex_4_4_2.csv', data_ex_4_4_2, delimiter=',')
```

In [ ]:
```
import matplotlib.pyplot as plt

x = np.array(range(len(wf))) * 0.01
plt.plot(x, 0.9 + 0.1 * wf, label="$V_{in}$")
Vout_ex_4_4_2 = np.loadtxt('data_ex_4_4_2.csv', delimiter=',')
plt.plot(x, Vout_ex_4_4_2, label='$V_{out}$')
plt.xlabel('$t$ [s]')
plt.ylabel('$V_{out}$ [V]')
plt.legend()
plt.grid()
plt.title('Fig. : response of follower-integrator at low frequency')
plt.show()
```



- Is the circuit "following" or "integrating"?

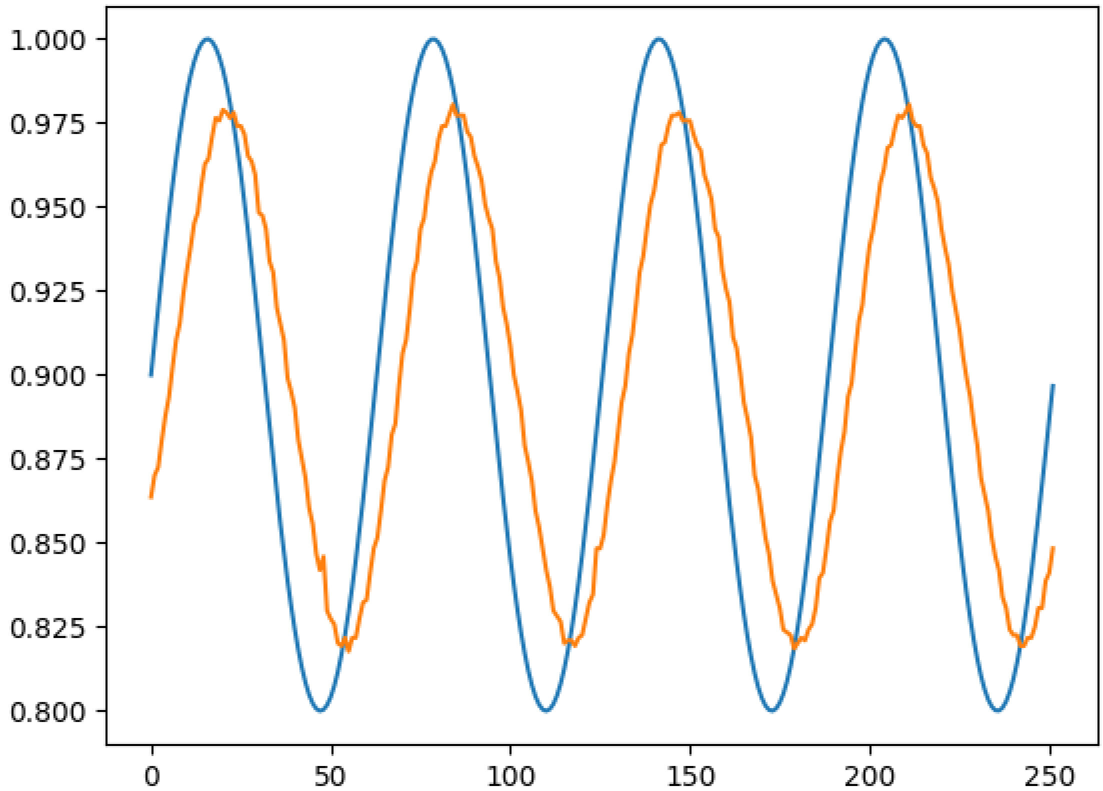
> The circuit is following the input voltage

- Can you change the frequency to make it operate in the other regime and plot V_{in} and V_{out} in the same figure

In [ ]:
```
wf = np.sin(np.arange(0, 8*3.14, 0.1))
p.set_voltage_waveform(0.9 + 0.1 * wf)
```

In [ ]:
```
plt.plot(0.9 + 0.1 * wf)
Vout_ex_4_4_3 = p.acquire_waveform(pyplane.DacChannel.AIN9, pyplane.AdcChannel.AOUT
plt.plot(Vout_ex_4_4_3)
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7efc51c17c10>]
```

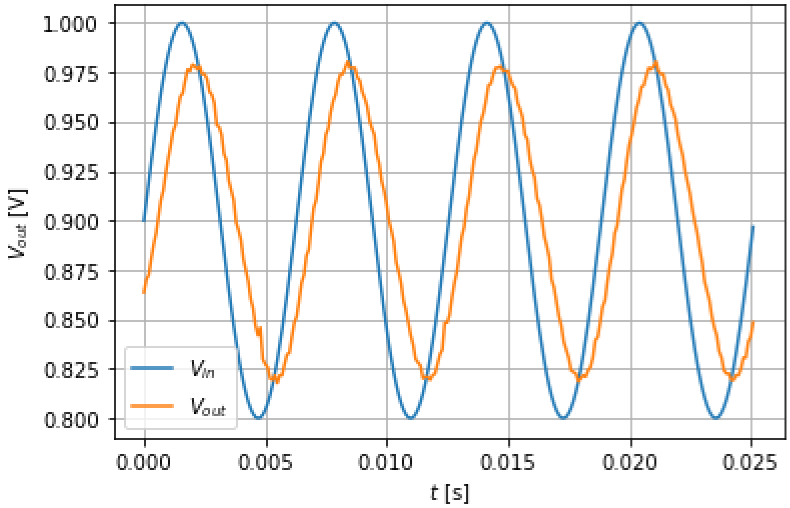


- Save data

In [ ]:
```
# if the data looks nice, save it!
# save to csv file
data_ex_4_4_3= [Vout_ex_4_4_3]
# save to csv file
np.savetxt('data_ex_4_4_3.csv', data_ex_4_4_3, delimiter=',')
```

In [ ]:
```
wf = np.sin(np.arange(0, 8*3.14, 0.1))
x = np.array(range(len(wf))) * 0.0001
plt.plot(x, 0.9 + 0.1 * wf, label="$V_{in}$")
Vout_ex_4_4_2 = np.loadtxt('data_ex_4_4_3.csv', delimiter=',')
plt.plot(x, Vout_ex_4_4_2, label='$V_{out}$')
plt.xlabel('$t$ [s]')
plt.ylabel('$V_{out}$ [V]')
plt.legend()
plt.grid()
plt.title('Fig. 4: response of follower-integrator at high frequency')
plt.show()
```

Fig. 4: response of follower-integrator at high frequency



- How to compute the transfer function at one frequency and extract the cutoff frequenc? Can you briefly give the basic method? (Optional)