Lab 6: Integrator Circuits

Group number: 4.5

Team member 1: Jan Hohenheim

Team member 2: Maxim Gärtner

Date:

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.

1. 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.

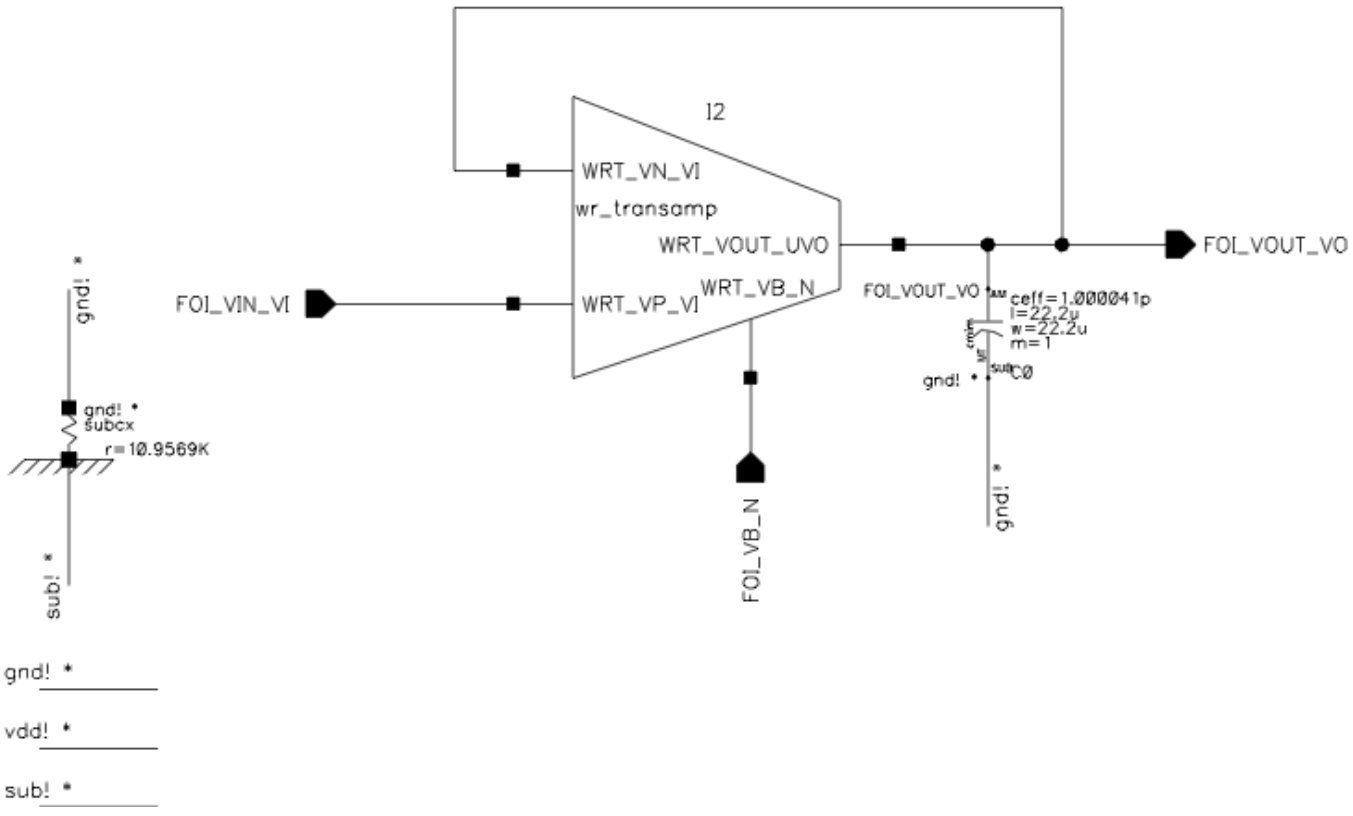
2 Prelab

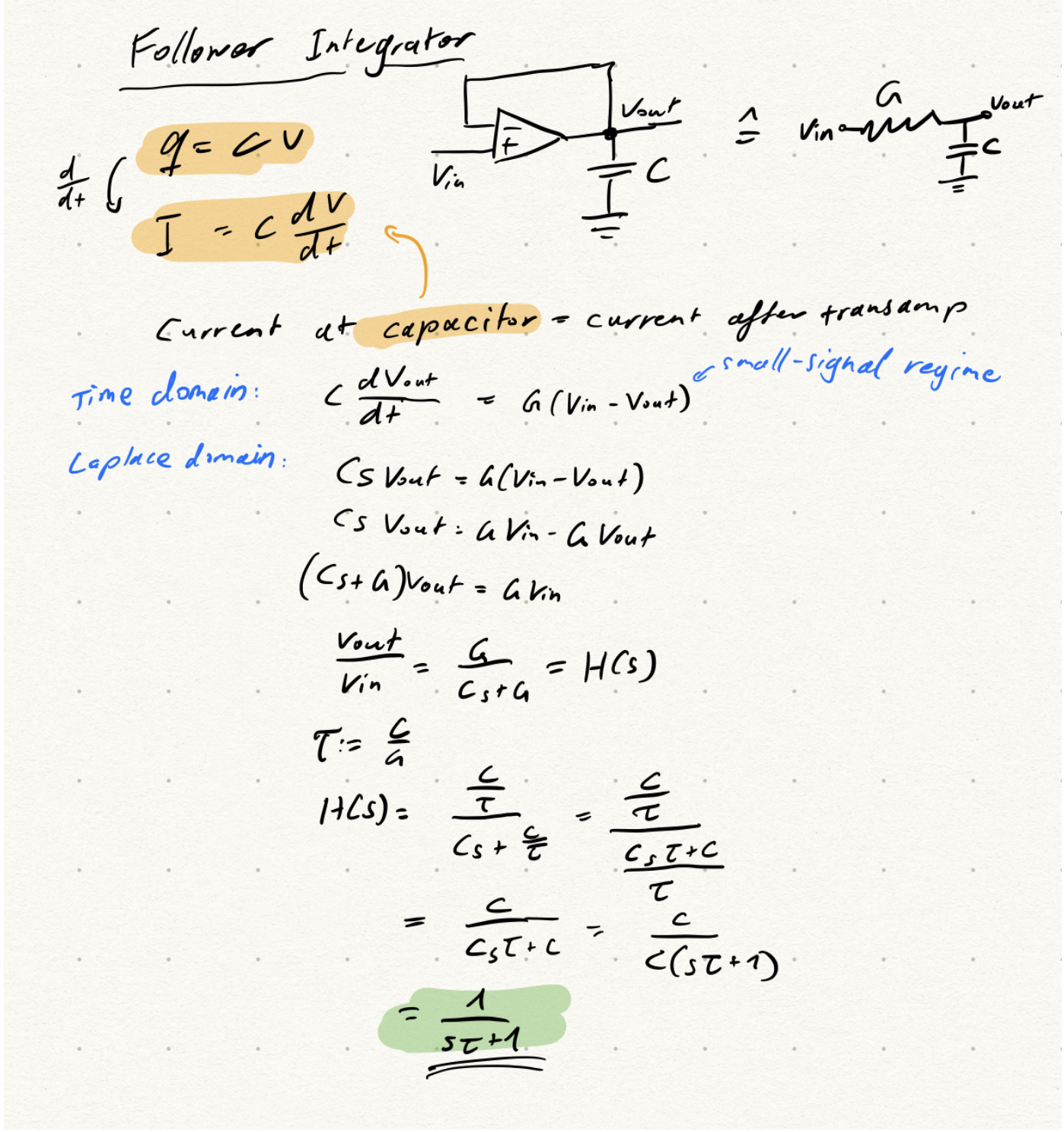
1. How are capacitors constructed in CMOS chip technology? There are several different possible implementations. How are they constructed in neurons? What is the capacitance per square micron of a SiO_2 capacitor with oxide thickness of 10nm? What is the capacitance per square micron area of a lipid-bilayer 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}}.)

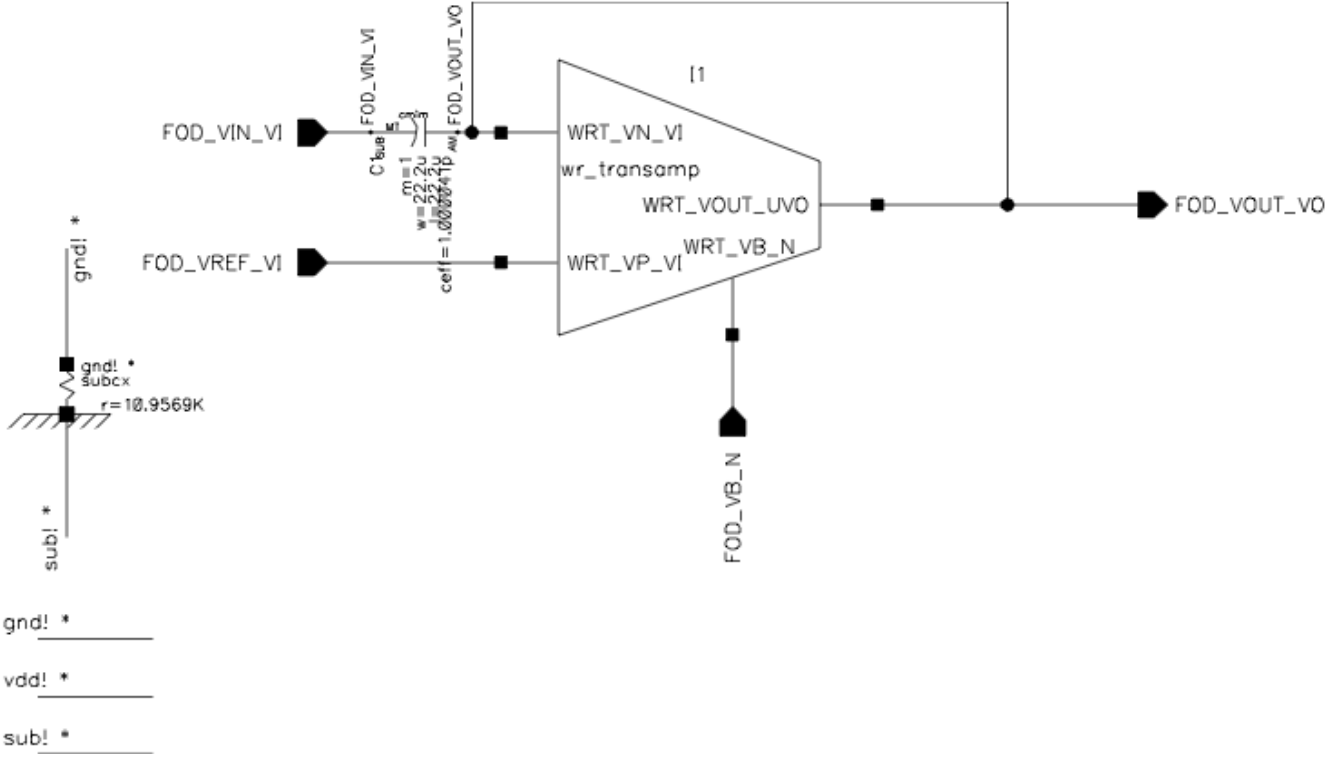
> - CMOS capacitors are mostly either implemented as metal-oxide-metal layers in a pattern that resembles intersecting forks or by using the transistor itself as a capacitor, with the gate and the drain + source acting as plates and the thin layer oxide as insulator. The latter's capacitance is dependent on the gate voltage.
> - A neuron uses it's phospholipid-bylayer as an insulator, where K^+, Ca^{2+}, Na^+ and Cl^- ions and charged proteins are the charge carriers. This behaves like a capacitor, since to change the membrane potential, the membrane capacitance needs to be charged.
> - Capacitance: $C = K\varepsilon_0\frac{A}{d}$ where $\varepsilon_0 \approx 8.86\mathrm{e}{-12}\frac{F}{m}$
> - For the class chip, $K \approx 3.9$ per the data we received in lab 2. So the capacitance is $3.9 \times 8.86\mathrm{e}{-12}\frac{F}{m}\frac{1m^{-2}}{10\mathrm{e}{-9}\mathrm{m}} = 3.46\mathrm{e}{-3}\frac{F}{m^2} = 3.46\frac{fF}{\mu m^2}$
> - For a phospholipid bilayer of thickness 5nm, $K \approx 5$ (Source: *J. C. Weaver and K. H. Schoenbach, "Biodielectrics," in IEEE Transactions on Dielectrics and Electrical Insulation, vol. 10, no. 5, pp. 715-716, Oct. 2003, doi: 10.1109/TDEI.2003.1237322.*). So the capacitance is $5 \times 8.86\mathrm{e}{-12}\frac{F}{m}\frac{1m^{-2}}{5\mathrm{e}{-9}\mathrm{m}} = 8.86\mathrm{e}{-3}\frac{F}{m^2} = 8.86\frac{fF}{\mu m^2}$

1. 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.

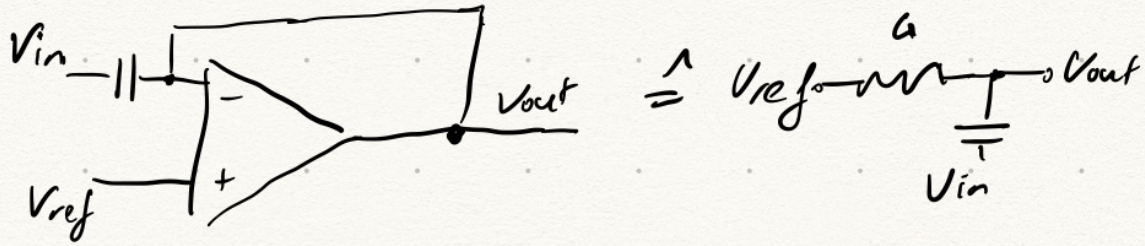




1. Compute the transfer function $H(s) = \frac{V_{out}}{V_{in}}$ for the follower-differentiator, using the s-plane notation, expressed in terms of complex frequency s and the time constant τ.


FOD_VIN_VI
FOD_VIN_VI
C1
m=1
w=22.2u
FOD_VOUT_VO
FOD_VOUT_VO
I1
WRT_VN_VI
wr_transamp
WRT_VOUT_UVO
WRT_VB_N
FOD_VREF_VI
WRT_VP_VI
FOD_VB_N
gnd! *
gnd! *
subcx
r=10.9569K
sub! *
gnd! *
vdd! *
sub! *

Follower differentiator



current after capacitor = current after transamp

time domain: $C\frac{d(V_{out}-V_{in})}{dt} = G(V_{ref}-V_{out})$

Laplace domain: $Cs(V_{out}-V_{in}) = G(V_{ref}-V_{out})$

$CsV_{out} - CsV_{in} = GV_{ref} - GV_{out}$

$V_{out}(Cs+G) = GV_{ref} + CsV_{in}$

$\frac{C}{G} = J$

$$H(s) = \frac{V_{out}}{V_{in}} = \frac{G\frac{V_{ref}}{V_{in}} + Cs}{Cs+G}$$

$$= \frac{G\frac{V_{ref}}{V_{in}} + GJs}{JGs+G} = \frac{\frac{V_{ref}}{V_{in}} + Js}{Js+1}$$

Assuming $V_{in} \gg V_{ref} \Rightarrow$ $H(s) = \frac{Js}{Js+1}$

1. 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 $1/\sqrt{2}$)?

$$H(s) = \frac{1}{s\tau + 1} = \frac{1}{1 + i\omega\tau}$$

$$s = \mu e^{i\theta} = i\omega$$

$$\mu = \sqrt{\sigma^2 + \omega^2}$$

$$|H(s)| = \left| \frac{1}{1 + i\omega\tau} \right|$$

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

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

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

$$\Rightarrow (\tau\omega)^4 + 2(\tau\omega)^2 + 1 = 2 + 2(\tau\omega)^2$$

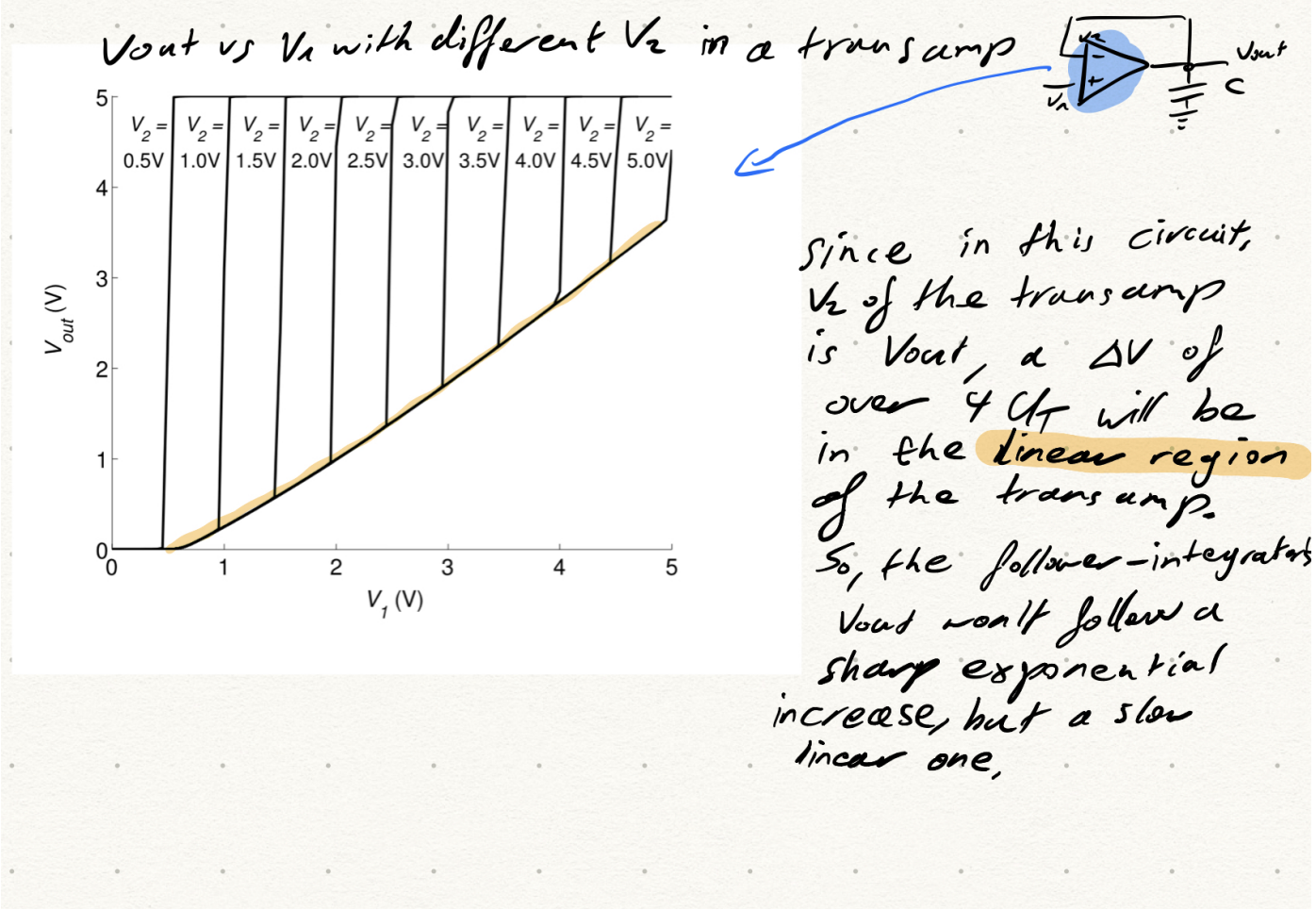
$$\Rightarrow \tau\omega = 1$$

$$\Rightarrow \omega = \tau^{-1}$$

$$\omega = 2\pi f$$

$$\Rightarrow f = (2\pi\tau)^{-1}$$

1. 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



1. 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?

> The difference: $|V_{in} - V_{out}| < 4U_T$ (assuming $\kappa \approx 1$).
> The reason is stated above.

3 Setup

3.1 Connect the device

In [1]:
```
# import the necessary library to communicate with the hardware
import pyplane
import time
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
# 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 [3]:
```
p.get_firmware_version()
```

Out[3]:
```
(1, 8, 4)
```

In [4]:
```
# 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, otherwise t
# Because the class chip need to do handshake to get the communication correct.
p.request_events(1)
```

In [5]:
```
# Try to read something, make sure the chip responses
p.read_current(pyplane.AdcChannel.GO0_N)
```

Out[5]:
```
1.7724609335800778e-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 [6]:
```
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:

$$I_b = w\frac{BG_{fine}}{256}I_{BG_{master}} \tag{1}$$

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

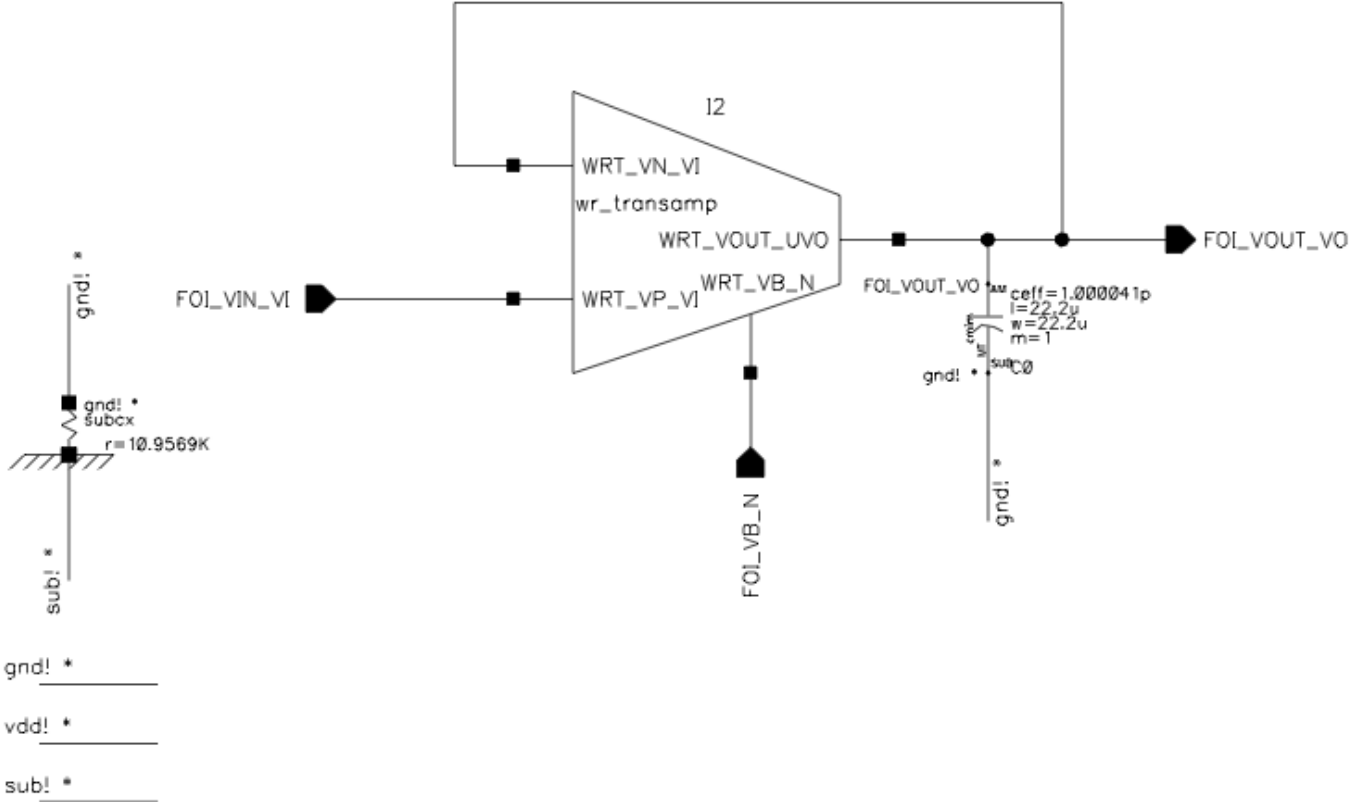
To set a bias, use the function similar to the following:

```
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 μs), 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 $f = 100 \cdot 10^3\text{Hz}$ and $n = 250$ yields

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

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

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

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

In [7]:
```
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 $I_b = 200\text{pA}$, this is

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

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

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

which has to be rounded down to

$BG_{fine} \approx 37$

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

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

- 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 [8]:
```
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.

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

The value of the output voltage will then also be

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

- 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 [9]:
```
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

$$\frac{d\,|\Delta V_{in}|}{d\tau} < 4U_T \approx 100\text{mV}.$$

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

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

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

[1] For $\kappa \approx 1$.

4.2.2 Data aquisition

- Verify the steady state

In [10]:
```
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)
```

```
0.905566394329071
```

- 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 [11]:
```
Vi_off = Vo_pre - Vi_pre
Vi_pre_corr = Vi_pre - Vi_off
print(Vi_pre_corr)
```

```
0.894433605670929
```

- Acquire the step response.

In [12]:
```
p.set_bit_depth(pyplane.BitDepth(10))
```

In [13]:
```
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.AIN9,
    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)
```

```
[0.90113526 0.90468018 0.9072583  0.91120605 0.91338134 0.91902099
 0.92425781 0.9273999  0.93142822 0.93392578 0.93908203 0.94383546
 0.94657471 0.94987794 0.95132813 0.9564038  0.96067383 0.96244628
 0.96558838 0.96615235 0.97098632 0.97372559 0.97557862 0.97662598
 0.97735108 0.98186279 0.98339356 0.98548828 0.98508545 0.98581054
 0.99032227 0.99040284 0.9916919  0.99080566 0.99217529 0.99491454
 0.99555908 0.99676757 0.99451172 0.99700929 0.999104   0.99918457
 0.9995874  0.99845948 0.99999024 1.00192383 1.00176268 1.00240722
 1.00039307 1.00264893 1.00409914 1.00353516 1.00409914 1.00176269
 1.00482423 1.00522707 1.00482423 1.00490479 1.0032129  1.00571047
 1.00611327 1.00571045 1.00587157 1.0029712  1.00691894 1.00651611
 1.00611329 1.00450195 1.00466307 1.00748291 1.00635497 1.00635499
 1.00490478 1.00571043 1.00909427 1.00643554 1.00675781 1.00530764
 1.00635498 1.00877198 1.00716064 1.00708008 1.00498534 1.00691894
 1.0079663  1.0070801  1.00748289 1.00530761 1.00732179 1.00788577
 1.00756348 1.00740234 1.00522704 1.00780518 1.00828856 1.00748291
 1.00699952 1.00546874 1.00812744 1.00828857 1.00708007 1.00659668
 1.00587158 1.00836914 1.00836914 1.00724121 1.00603268 1.00595216
 1.00853027 1.0079663  1.00716064 1.00571046 1.00611329 1.00828856
 1.00780517 1.00764405 1.00530761 1.00699952 1.00877197 1.00724121
 1.00716064 1.00554931 1.00756347 1.00917479 1.00772462 1.00716064
 1.00562989 1.0079663  1.00828855 1.00772462 1.00716063 1.0049048
 1.00877197 1.00901368 1.00772458 1.00675781 1.00546875 1.00828857
 1.00869141 1.00836914 1.0062744  1.00514648 1.0084497  1.00828859
 1.00756346 1.00571045 1.00579101 1.00820801 1.00804689 1.00764406
 1.00619386 1.00643555 1.00861083 1.00780517 1.00764407 1.00522707
 1.00635496 1.00869141 1.00812744 1.00836915 1.00562987 1.00708007
 1.00885254 1.00740236 1.00716064 1.00530763 1.00780518 1.00732177
 1.00748291 1.0072412  1.00683837 1.00812743 1.00877196 1.00740236
 1.00635498 1.00562989 1.00836916 1.00861086 1.00740234 1.00627441
 1.00635499 1.00861084 1.00788575 1.00708007 1.00587157 1.00659668
 1.00877197 1.008208   1.00804689 1.00651612 1.00627441 1.00869142
 1.00844972 1.00587158 1.00571046 1.00708007 1.00861083 1.00772462
 1.00788573 1.00562989 1.00683838 1.00877197 1.00780516 1.00683837
 1.00579102 1.00780519 1.00853026 1.00796632 1.00708007 1.00595216
 1.00836915 1.00828859 1.00780516 1.00675781 1.00579102 1.00861084
 1.00885255 1.00764405 1.00595214 1.00683839 1.00869141 1.00812745
 1.0079663  1.00571046 1.00627441 1.00885253 1.00804688 1.00724121
 1.00546875 1.0065161  1.00885253 1.00748291 1.00748291 1.0055493
 1.00699949 1.00861082 1.00788577 1.00740234 1.00571043 1.00748291
 1.00877197 1.00756348 1.00716066 1.00546876 1.00869139 1.00893312
 1.00772462 1.00716065 1.00571045 1.00844971 1.00836915 1.00716064
 1.00595213 1.00546876 1.00820802 1.00861087 1.00780519 1.00538816
 1.00522705 1.00853027 1.00836915 1.00756347 1.00571046 1.00716064
 1.00885255 1.00691894 1.00788574 1.00546875 1.00643556 1.0077246
 1.00788573 1.00740232 1.00619385 1.0072412  1.00869138 1.00772462
 1.00732177 1.00562986 1.00764403 1.00828857 1.0074829  1.00812743
 1.00562987 1.00796632 1.00965819 1.00828857 1.00691897 1.00554931
 1.00804688 1.00844971 1.00772461 1.00659667 1.00579102 1.00740235
 1.00836915 1.00756347 1.00635498 1.00603273 1.00869142 1.00812742
 1.00748291 1.00546875 1.00595214 1.00836912 1.00788575 1.00724121
 1.00603272 1.00683837 1.00877197 1.00756348 1.00812743 1.00579101
 1.0072412  1.00861086 1.00748292 1.00732177 1.00571046 1.0072412
 1.00869141 1.0079663  1.00611326 1.00571046 1.00780517 1.00861084
 1.00820802 1.00708008 1.00514649 1.00861084 1.0079663  1.00772462
 1.00603272 1.00619384 1.00869142 1.00877196 1.00772462 1.00530763
 1.00627441 1.00861083 1.00788573 1.00724119 1.00571046 1.00627441
 1.00828856 1.00756348 1.0072412  1.00554932 1.00691893 1.00877196
 1.0073218  1.00724123 1.00579102 1.00683838 1.00885253 1.00756347
 1.00667725 1.00595216 1.00732177 1.00853027 1.00788574 1.0072412
 1.00603272 1.008208   1.00836914 1.00764403 1.00683838 1.00587158
 1.00828859 1.00828856 1.00764402 1.00643554 1.00619385 1.00804687
 1.00836914 1.00732179 1.00595214 1.00619386 1.00877196 1.00764403
 1.00716064 1.00530761 1.00716064 1.00861083 1.00740234 1.00748291
 1.00554932 1.00683837 1.00869139 1.00772462 1.00756348 1.00587158
```

```
1.0067578  1.00869142 1.0072412  1.0072412  1.00587157 1.00756348
1.00869141 1.00772461 1.00772461 1.00635498 1.00812744 1.00820801
1.00828856 1.00740234 1.00627441 1.00853028 1.00788573 1.00812746
1.00603271 1.00587159 1.00869142 1.00764405 1.00659667 1.00579102
1.00651611 1.00901368 1.0074829  1.00716064 1.00595217 1.00603273
1.00877197 1.00748292 1.00724123 1.00587158 1.00691893 1.00901368
1.00740234 1.00708008 1.00554929 1.00748291 1.00853027 1.00788575
1.00764403 1.00595217 1.00764402 1.00828857 1.00764405 1.00691894
1.005791   1.00804685 1.00804689 1.00861084 1.00619385 1.00595214
1.00869142 1.00812744 1.00740231 1.00627441 1.00603272 1.00917482
1.00764405 1.0072412  1.00579103 1.00538819 1.0098999  1.00772461
1.00740235 1.00579101 1.00675781 1.0089331  1.00732177 1.00740235
1.00530763 1.00732176 1.00853027 1.00788575 1.00788575 1.00554931
1.0074829  1.00893309 1.00788575 1.00667725 1.00579102 1.0075635
1.008208   1.0079663  1.00708005 1.00571046 1.00804689 1.00836915
1.00812743 1.00716065 1.00562989 1.00812743 1.0079663  1.00691895
1.00611329 1.00683838 1.00885255 1.00780516 1.00756347 1.00571045
1.00651611 1.0089331  1.00683837 1.00732179 1.00627441 1.00683838
1.00877196 1.00716064 1.00740235 1.00522707 1.00732177 1.00853029
1.00772462 1.00716064 1.00603271 1.00716062 1.00853027 1.00788575
1.00732177]
```

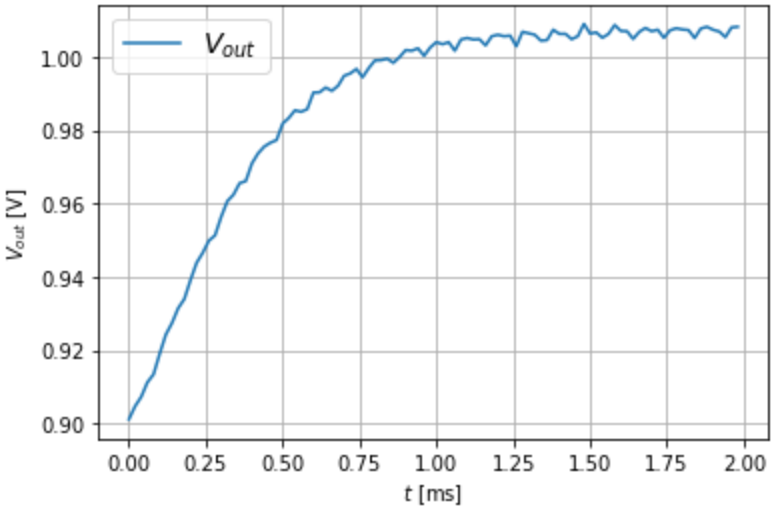
- plot V_{out}

In [1]:

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

Vout_ex_4_2_2 = np.loadtxt('data/data_ex_4_2_2.csv', delimiter=',')
t=np.arange(0,len(Vout_ex_4_2_2))*0.00002
plt.plot(t[:100]*1000, Vout_ex_4_2_2[:100])
plt.xlabel('$t$ [ms]')
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()
```

Fig. 1: Step response of the follower-integrator for small signals.



- save data

In [15]:

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

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

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

Thus

$V_{out}(t = \tau) = \Delta V_{in}(1 - \frac{1}{e}) + V_{out}(t = 0)$

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

In [18]:
```
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)
```

```
0.0003075985834036632
```

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

In [19]:
```
Ibe = 200
Ibm = 199.5
taue = 250
taum = tau1*10**6
kappa1 = (taue*Ibe)/(taum*Ibm)
print(kappa1)
```

```
0.8147845274278835
```

4.2.5 Analysis

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

> Yes, because the output rises quickly exponentially.

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

> V_{out} at $t = 0$ is already 0.9 V. We can extrapolate the input step time by following the slope of V_{out} to the left until $V_{out} = 0$.

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 [19]:
```
dVi = 0.3
```

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

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

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

- 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 [18]:
```
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.

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

The value of the output voltage will then also be

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

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

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

Therefore, the linear part should be approximately

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

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 $\kappa = 1$)

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

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

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

The corresponding slew rate would thus be

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

- 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 $t_{lin} = 1\mathrm{ms}$, $\Delta V_{out,lin} = 250\mathrm{mV}$ and $C = 1\mathrm{pF}$, it can be determined what value I_b must have in order to achieve the slew rate determined in the previous exercise

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

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

In [53]:
```
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 $I_b = 250\mathrm{pA}$, this is

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

With $I_b = 250\mathrm{pA}$ 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

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

4.3.2 Data aquisition

- Verify the steady state

In [54]:
```
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)
```

```
0.909594714641571
```

- 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 [55]:
```
Vi_off = Vo_pre - Vi_pre
Vi_pre_corr = Vi_pre - Vi_off
print(Vi_pre_corr)
```

```
0.890405285358429
```

- Acquire the step response

In [56]:
```
p.set_bit_depth(pyplane.BitDepth(10))
```

In [60]:
```
Vout_ex_4_3_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
    timestep = 0.00002
    Vout_ex_4_3_2 = Vout_ex_4_3_2 + p.acquire_transient_response(pyplane.DacChannel.AIN9,
    time.sleep(0.5) # wait to receive the measured data from the microcontroller

Vout_ex_4_3_2 /= 10
print(Vout_ex_4_3_2)
```

```
[0.90032959 0.90250489 0.90717773 0.91072266 0.9204712  0.92828614
 0.93150879 0.93650391 0.93924317 0.9489917  0.95793457 0.96139893
 0.96639404 0.97026123 0.9792041  0.98790527 0.99104736 0.99733154
 0.99950685 1.00949708 1.01779541 1.02085693 1.02722168 1.0297998
 1.03850099 1.0478467  1.05090821 1.05606445 1.0600122  1.06806885
 1.07660888 1.08063722 1.08635744 1.08861328 1.09650879 1.10512937
 1.11028564 1.11455568 1.11681154 1.12365967 1.13163574 1.13646971
 1.13961182 1.1415454  1.1463794  1.15250244 1.15798095 1.15918945
 1.16055908 1.16273438 1.16942137 1.17353029 1.1734497  1.17449706
 1.17457763 1.17916992 1.18352051 1.18327879 1.18335938 1.18231202
 1.18625977 1.19036865 1.18899904 1.1883545  1.18650147 1.19044923
 1.19391357 1.19197998 1.19270507 1.19004639 1.19286619 1.19608889
 1.1947998  1.1950415  1.19181885 1.19391357 1.19689454 1.19665283
 1.19633058 1.1933496  1.19471924 1.19834472 1.19882812 1.19681396
 1.19512209 1.19479982 1.19850587 1.20035889 1.19745851 1.19633057
 1.19463867 1.19810301 1.20052001 1.19858642 1.19697511 1.19479979
 1.19761962 1.20092286 1.19810305 1.19778076 1.19471923 1.19778076
 1.20068115 1.19810301 1.19818362 1.19479979 1.19705566 1.20060058
 1.19810305 1.19786131 1.19488037 1.19705567 1.19995605 1.19906982
 1.19834473 1.19512206 1.19689454 1.19955323 1.20019777 1.19721678
 1.19584714 1.19633056 1.19923096 1.20035889 1.19842529 1.19552491
 1.19633054 1.19890871 1.20132568 1.19866699 1.19705567 1.19488037
 1.19834472 1.20084229 1.19753907 1.19753907 1.19576659 1.19802246
 1.20132568 1.19761963 1.19786134 1.19512206 1.19737793 1.20076171
 1.19850587 1.19786134 1.19488038 1.19721681 1.20035887 1.19858643
 1.19915038 1.19471924 1.19705566 1.19987549 1.19971436 1.19874755
 1.19536377 1.19665283 1.19939209 1.20011721 1.19858643 1.19592773
```

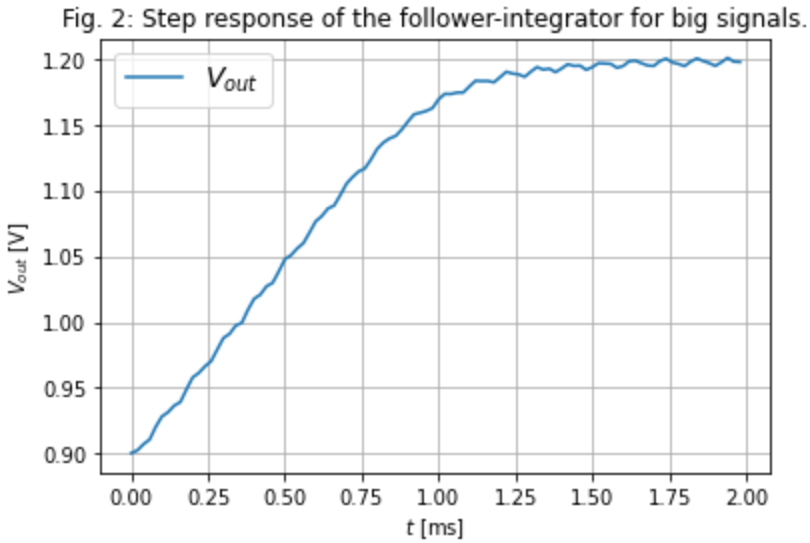
```
1.19633056 1.19955323 1.20076171 1.19858642 1.19544433 1.19568604
1.19882812 1.20132569 1.19874755 1.19713624 1.1952832  1.19842528
1.20084231 1.19826416 1.19770019 1.19520264 1.19810301 1.20060059
1.19850585 1.19786133 1.19488035 1.19778078 1.20116457 1.19890867
1.198667   1.19512206 1.19705566 1.20035888 1.19906981 1.19810301
1.19512205 1.19681398 1.19979492 1.20011718 1.1985864  1.19560546
1.19641116 1.19931153 1.20043947 1.19842529 1.19681396 1.19625001
1.19898925 1.20084229 1.19915038 1.19697509 1.19633056 1.19979494
1.20108398 1.19826416 1.19729738 1.19520264 1.19826415 1.20076171
1.19802245 1.19753907 1.19504149 1.19786131 1.20100342 1.1985864
1.19761963 1.19528319 1.19761964 1.20060058 1.19906983 1.19850588
1.19463867 1.19713625 1.20011717 1.1993921  1.19890869 1.19576659
1.1967334  1.19987549 1.19987549 1.19874755 1.19649169 1.19625
1.1993921  1.20052003 1.19882812 1.19608886 1.19568602 1.1989087
1.20092285 1.19866698 1.19713622 1.19520265 1.1995532  1.20100342]
```

- plot V_{out}

In [2]:

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

Vout_ex_4_3_2= np.loadtxt('data/data_ex_4_3_2.csv', delimiter=',')
t=np.arange(0,len(Vout_ex_4_3_2))*0.00002
plt.plot(t[:100]*1000, Vout_ex_4_3_2[:100])
plt.xlabel('$t$ [ms]')
plt.ylabel('$V_{out}$ [V]')
plt.legend(['$V_{out}$'],prop={'size': 14})
plt.title('Fig. 2: Step response of the follower-integrator for big signals.')
plt.grid()
plt.show()
```



- save data

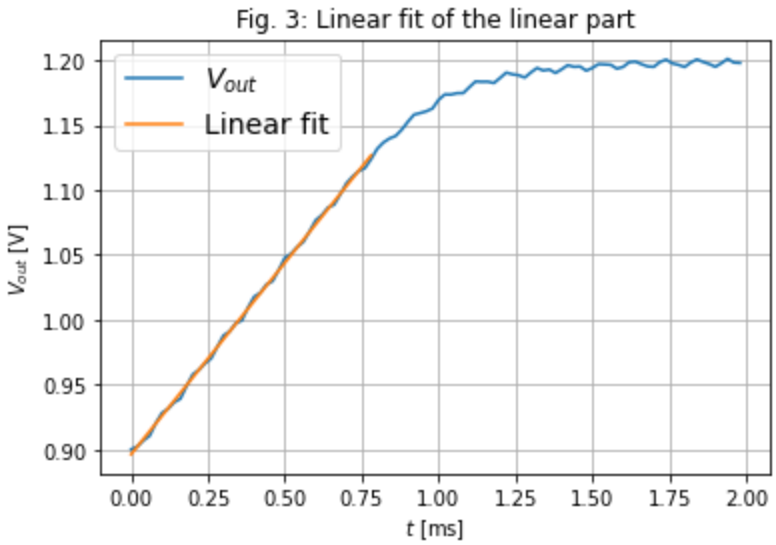
In [61]:

```
# if the data looks nice, save it!
data_ex_4_3_2= [Vout_ex_4_3_2]
# save to csv file
np.savetxt('data/data_ex_4_3_2.csv', data_ex_4_3_2, delimiter=',')
```

4.3.3 Data processing

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

In [6]:
```
Vout_ex_4_3_2= np.loadtxt('data/data_ex_4_3_2.csv', delimiter=',')
t=np.arange(0,len(Vout_ex_4_3_2))*0.00002
linear_part = Vout_ex_4_3_2[:40]
a0, a1 = np.polyfit(t[:40], Vout_ex_4_3_2[:40], 1)
plt.plot(t[:100]*1000, Vout_ex_4_3_2[:100], label='$V_{out}$')
plt.plot(t[:40]*1000, t[:40]*a0 + a1, label='Linear fit')
plt.xlabel('$t$ [ms]')
plt.ylabel('$V_{out}$ [V]')
plt.legend(prop={'size': 14})
plt.title('Fig. 3: Linear fit of the linear part')
plt.grid()
plt.show()
slew_rate = a0
print("The slew rate is: ", slew_rate)
```




```
The slew rate is:  294.89685751595636
```

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

In [7]:
```
capacitance = 1e-12
I_b = capacitance*slew_rate
print("The calculated bias current from the measurement is: ", I_b * 1e12, "pA")
set_I_b = 250e-12
print("The bias current calculated when setting BGfine is: ", set_I_b * 1e12, "pA")
print("So, the difference is ", (I_b - set_I_b)*1e12, "pA")
```

```
The calculated bias current from the measurement is:  294.89685751595636 pA
The bias current calculated when setting BGfine is:  250.00000000000003 pA
So, the difference is  44.89685751595635 pA
```

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

In [29]:
```
import scipy.interpolate as interpolate

f = interpolate.interp1d(Vout_ex_4_3_2,t) # Interpolate t vs. Vout
v_1tau = dVi*(1-np.exp(-1))+Vout_ex_4_3_2[0]
tau1 = f(v_1tau) # Get tau = t(Vout=1-e^(-1))
print(tau1)
```

```
0.0006634259335780621
```

In [30]:

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

```
0.3777762576514505
```

> The measured τ and κ are too low because they are in the part of the graph that does not follow the exponential formula. So, this calculation doesn't make much 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?

> V_{out} at $t = 0$ is already 0.9 V. We can extrapolate the input step time by following the slope of V_{out} to the left until $V_{out} = 0$.

- 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 margin is at $g_m \Delta V_{in} = I_b = 294pA$.
> Using $C = 1pF$ and $g_m = \frac{C}{\tau}$ with $\tau = 0.3ms$, we get
> $\Delta V_{in} = \frac{I_b \tau}{C} = 0.0882$, which is also $4U_T\kappa$

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 [63]:
```
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

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

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 [36]:
```
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)

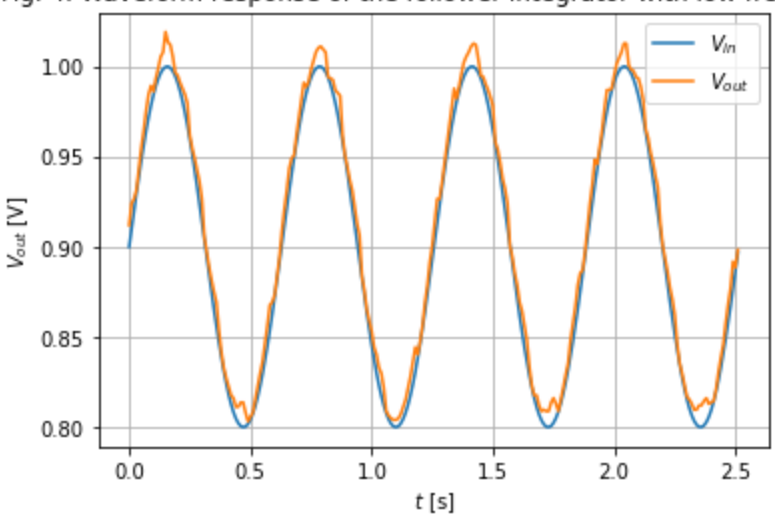
In [105…
```
p.set_voltage_waveform(0.9 + 0.1 * wf)
```

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

In [111…
```
interval = 0.01
Vout_ex_4_4_2 = p.acquire_waveform(pyplane.DacChannel.AIN9, pyplane.AdcChannel.AOUT10, int
```

In [39]:
```
x = np.array(range(len(wf))) * interval
plt.plot(x, 0.9 + 0.1 * wf, label="$V_{in}$")
Vout_ex_4_4_2 = np.loadtxt('data/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. 4: Waveform response of the follower-integrator with low frequency')
plt.show()
```

Fig. 4: Waveform response of the follower-integrator with low frequency



- Save data

In [112…
```
# if the data looks nice, save it!
data_ex_4_4_2= [Vout_ex_4_4_2]
# save to csv file
np.savetxt('data/data_ex_4_4_2.csv', data_ex_4_4_2, delimiter=',')
```

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

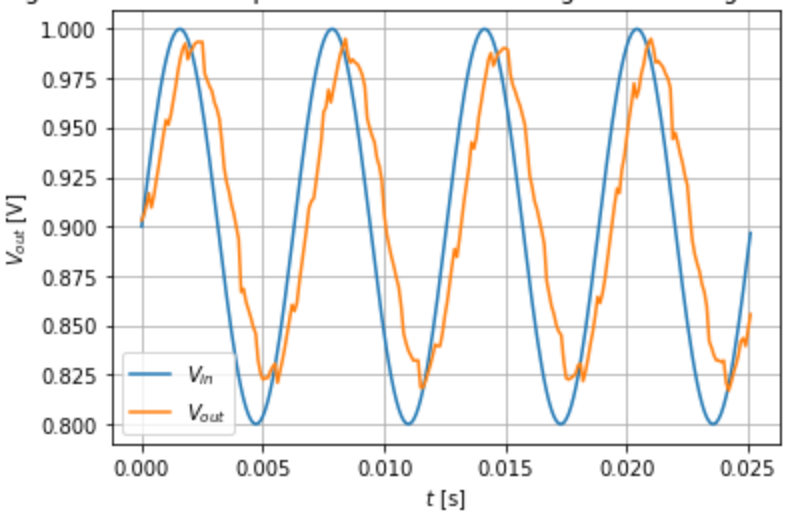
> It's following the input current

- 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 [120…
```
interval = 0.0001
Vout_ex_4_4_3 = p.acquire_waveform(pyplane.DacChannel.AIN9, pyplane.AdcChannel.AOUT10, int
```

In [41]:
```
x = np.array(range(len(wf))) * interval
plt.plot(x, 0.9 + 0.1 * wf, label="$V_{in}$")
Vout_ex_4_4_3 = np.loadtxt('data/data_ex_4_4_3.csv', delimiter=',')
plt.plot(x, Vout_ex_4_4_3, label='$V_{out}$')
plt.xlabel('$t$ [s]')
plt.ylabel('$V_{out}$ [V]')
plt.legend()
plt.grid()
plt.title('Fig. 5: Waveform response of the follower-integrator with high frequency.')
plt.show()
```

Fig. 5: Waveform response of the follower-integrator with high frequency.



- Save data

In [121…
```
# if the data looks nice, save it!
data_ex_4_4_3= [Vout_ex_4_4_3]
# save to csv file
np.savetxt('data/data_ex_4_4_3.csv', data_ex_4_4_3, delimiter=',')
```

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