Neuromorphic engineering I

Lab 5: Static Circuits: Transconductance Amplifier

Team member 1: Jan Hohenheim

Team member 2: Maxim Gärtner

Group number: 4.5

Date:

Lab objectives

The objectives of this lab are to understand and characterize one of the most important circuit in analog IC design.

The experimental objectives are as follows:

1. To characterize a simple differential transconductance amplifier and understand its operation in terms of the behavior of the differential pair and the current mirror. Specifically, to understand the dependence of the output current on the differential input voltages.
2. To characterize single-stage 2-transistor "common-source" amplifier gain, and how it arises from transconductance and output impedance.

1 Prelab

1.1 Transconductance amplifier

- Now consider a simple differential transconductance amplifier which is built from a differential pair and a current mirror. The output current should be equal to the the difference of the two differential pair currents, i. e. $I_{out} = I_1 - I_2$. Is this statement true? Justify your answer by stating your assumptions about transistor saturation and drain conductance.

> To stay in saturation, we need to fulfill $\kappa(max(V_1, V_2) - V_b) + 4U_T < V_{out} < V_{dd} - 4U_T$.
> So the relationship $I_{out} = I_1 - I_2$ is true only for the currents that can be reached under the above condition. If V{out} *is too low, $I*{out}$willbewaytoohigh. If$V{out}$istoohigh,$I{out}$ will be way too low.

- Now consider the transconductance amplifier with the output open-circuited (i.e. no current flows into or out of the output node). Say V_2 is fixed at some voltage in the middle of the rails, e.g., $\frac{Vdd}{2}$. Explain what happens to the output voltage as V_1 is swept from below V_2 to above V_2 for a subthreshold bias. Discuss the current through the differential pair transistors and the current mirror, and the voltage on the internal node common to the differential pair transistors. Try to keep the discussion concise.

> Assuming we start with V_{out} in a valid range: V_{out} will stay small enough, since V_{dd} is not changed, but as soon as V_1 is bigger than V_2, the minimum allowed value for V_{out} will grow,

> so we will have to adjust it to at least $\kappa(V_1 - V_b) + 4U_T$. As soon as $V_1 = \kappa^{-1}(V_{dd} - 8U_T) + V_b$, there will be no valid V_{out} anymore.

- What is the transconductance $g_m = \dfrac{\mathrm{d}I_{out}}{\mathrm{d}V_{in}}$, where $V_{in} \equiv V_1 - V_2$, in sub-threshold? How does it change if the circuit is operated super-threshold?

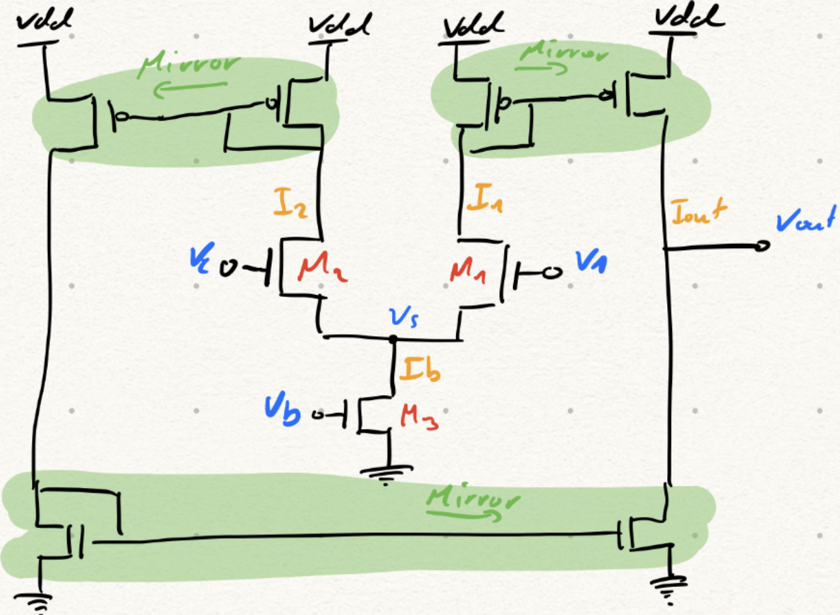
> In words, the transconductance is how much change in output current we get when we change the input voltage.
> In equations:
>
> - sub-threshold: $g_m = \frac{I_b \kappa}{2U_T}$, grows linearly with I_b.
> - super-threshold: $g_m = \sqrt{\beta I_B}$, so now it only grows sublinearly with I_B.

- Quantitatively, what is the relationship between transconductance, output resistance r_o, and voltage gain A of a transconductance amplifier?

> Assuming sub-threshold and all relevant transistors in saturation:
>
> - $g_m = \frac{\partial I_{out}}{\partial V_{in}} \approx \frac{I_b \kappa}{2U_T}$
> - $g_d = \frac{\partial I_{out}}{\partial V_{out}} = \frac{\partial V_{in}}{\partial V_{out}} g_m \approx \frac{I_b}{V_E} = \frac{2U_T}{\kappa V_E} g_m$
> - $r_o = g_d^{-1} = \frac{\partial V_{out}}{\partial I_{out}} = \frac{\partial V_{out}}{\partial V_{in}} g_m^{-1} \approx \frac{V_E}{I_b} = \frac{\kappa V_E}{2U_T} g_m^{-1}$
> - $A = \frac{g_m}{g_d} = g_m r_o$

1.2 Wide-Range Transamp

- Draw the schematic of a wide-range transconductance amplifier and explain why it does not have the simple 5-transistor transamp restriction on allowable output voltage. You can either draw the schematic directly on the Jupyter notebook using the *schemdraw* , or sketch it with pen and paper and paste a picture in a Markdown cell.



M_1 & M_2 are downstream of diode connected current mirrors, so to keep them in saturation, $V_{s_{max}}$ is $V_{dd} - 4U_T$. To keep M_3 in saturation, $V_{s\,min}$ is $4U_T$. So the range is pretty broad, but more importantly, it doesn't jump around with different settings of V_1 or V_2. By extend, V_{out} has the same range, since it is sandwiched between more current mirrors. There are barely restrictions at all compared to the original transductance amplifier circuit.

2 Setup

2.1 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]:

```
1.8530273848682555e-07
```

In [6]:

```
# If any of the above steps fail, delete the object, and restart the kernel

# del p
```

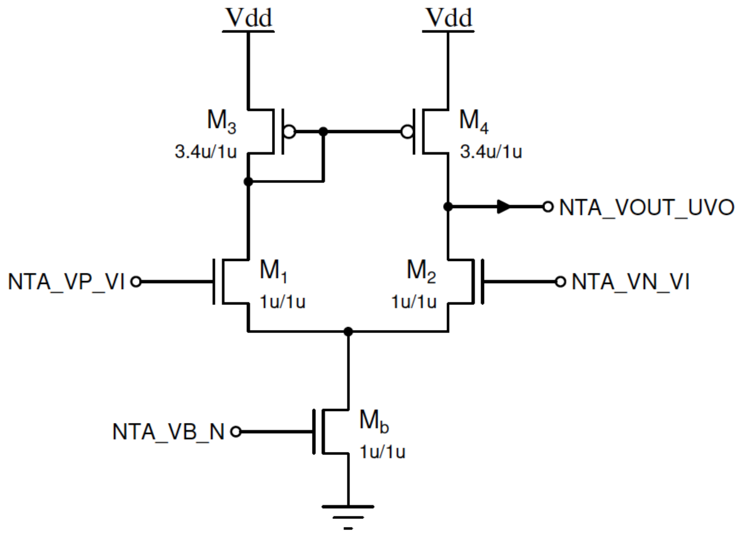
2.2 Setup C2F and voltage output buffer

In [7]:

```
# setup C2F
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_HYS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I60pA, 100)])

time.sleep(0.2)
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_BIAS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])

time.sleep(0.2)
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_PWLK_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])

time.sleep(0.2)
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_REF_L, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 255)])

time.sleep(0.2)
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.C2F_REF_H, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 255)])

time.sleep(0.2)
# setup output rail-to-rail buffer
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.RR_BIAS_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I240nA, 255)])
```

3 N-Type 5T Transamp

3.0 Schematic and pin map



V_1 = V_p = NTA_VP_VI = AIN3

V_2 = V_n = NTA_VN_VI = AIN4

V_{out} = NTA_VOUT_UVO = ADC[13]

I_{out} = $I_+ - I_-$ = NTA_IOUT_UO - NTA_IOUT_UBO = C2F[11] - C2F[12]

Note: There are three identical NTA circuits with the same bias and input voltages, one with the output open-circuited and routed out at NTA_VOUT*UVO, the other two with $V*{out}* $fixed to 1V but$ I_{out}$ routed out through N- and P- type current mirror at NTA_IOUT_UO and NTA_IOUT_UBO.

3.1 Chip configuration

In [9]:

```
p.send_coach_events([pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.SelectLine1, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)])
```

3.2 Calibration of C2F channels

Here you need to calibrate NTA_IOUT_UO and NTA_IOUT_UBO in the same way as the last lab

3.2.1 NTA_IOUT_UO

- Set fixed voltages for V_1 and V_2

In [10]:

```
p.set_voltage(pyplane.DacChannel.AIN3,0.8)  # V1 = 0.8
time.sleep(0.2)  # settle time
p.set_voltage(pyplane.DacChannel.AIN4,0.2)  # V2 = 0.2
```

Out[10]:

```
0.19882699847221375
```

Set voltages such that $V_1 \gg V_2$.

- Data aquisition (Hint: use master current for I_b = 30 nA)

In [11]:

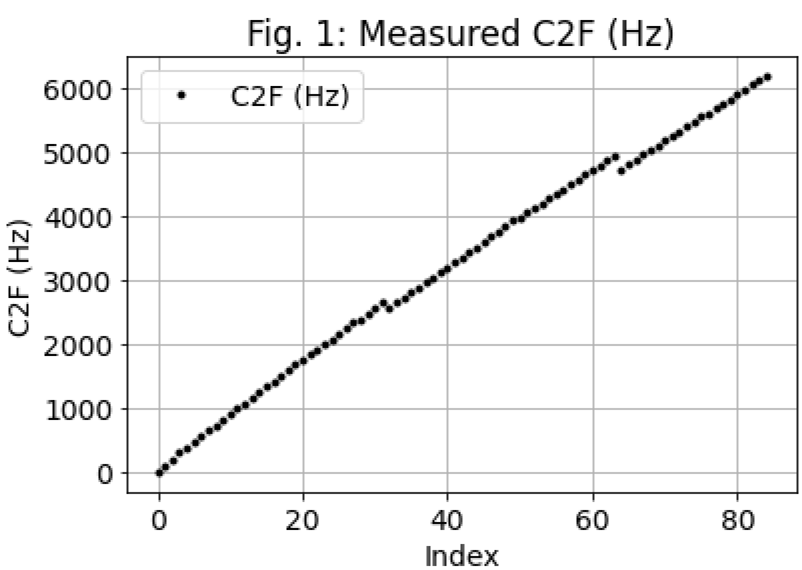
```
import numpy as np
```

```
import time

calIout_UO_ex3 = np.arange(0,85,1) # bias current sweep range, fine value

c2f_Iout_UO_ex3 = []   # what you get is frequency

for n in range(len(calIout_UO_ex3)):

    # set bias
    p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, calIout_UO_ex3[n])])

    time.sleep(0.2) # settle time

    # read c2f values
    c2f_Iout_UO_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_ex3.append(c2f_Iout_UO_ex3_temp[11])

print(c2f_Iout_UO_ex3)
```

```
[1, 105, 198, 298, 365, 464, 552, 650, 724, 820, 907, 1002, 1064, 1157, 1242, 1332, 1410,
1503, 1585, 1678, 1737, 1827, 1911, 1998, 2068, 2157, 2236, 2325, 2383, 2475, 2552, 2639,
2553, 2641, 2720, 2804, 2866, 2955, 3029, 3124, 3187, 3276, 3349, 3443, 3497, 3587, 3668,
3757, 3824, 3917, 3977, 4060, 4119, 4182, 4266, 4346, 4411, 4496, 4571, 4652, 4705, 4789,
4863, 4941, 4726, 4805, 4881, 4963, 5016, 5096, 5171, 5246, 5303, 5392, 5464, 5548, 5595,
5679, 5748, 5819, 5885, 5971, 6039, 6113, 6171]
```

- Plot

In [32]:
```
plt.plot(c2f_Iout_UO_ex3, '.k', label="C2F (Hz)")
plt.xlabel("Index")
plt.ylabel("C2F (Hz)")
plt.title("Fig. 1: Measured C2F (Hz)")
plt.legend()
plt.grid()
plt.show()
```



- Save data

In [14]:
```
# if the data looks nice, save it!
data_Iout_UO_ex3_cal= [c2f_Iout_UO_ex3, calIout_UO_ex3]
```

```
# save to csv file
np.savetxt('./data/c2f_Iout_UO_ex3_cal_I_plus.csv', data_Iout_UO_ex3_cal, delimiter=',')
```

- Load data you saved

In [22]:
```
# load the saved data
c2f_Iout_UO_ex3_save,calIout_UO_ex3_save = np.loadtxt('./data/c2f_Iout_UO_ex3_cal_I_plus.c
```

- C2f plot

In [23]:
```
# C2f plot
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 14})

Iout_UO_ex3 = calIout_UO_ex3_save/256*30

plt.plot(Iout_UO_ex3,c2f_Iout_UO_ex3_save,'k+')

plt.xlabel('$I_b$ (nA)')
plt.ylabel('C2F (Hz)')
# plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 2: C2F values vs. $I_1$ for $V_1 \gg V_2$.')
plt.grid()
plt.show()
```

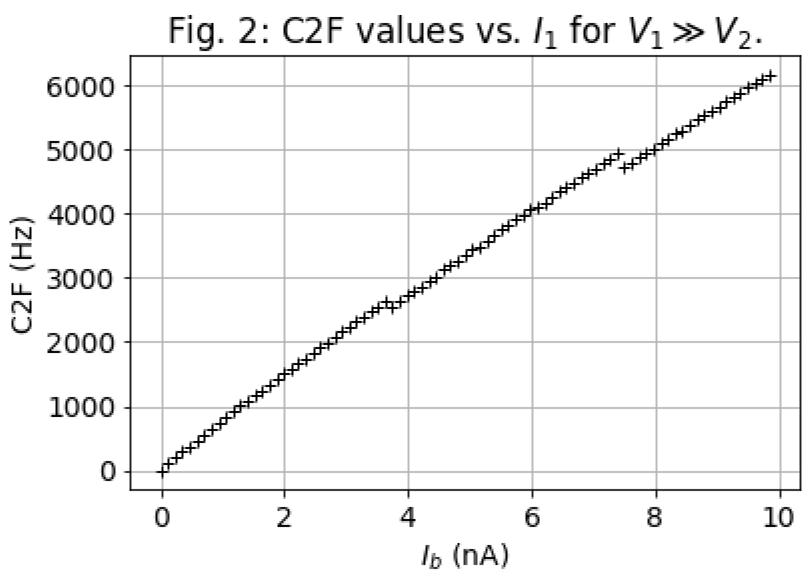


Fig. 1 shows the C2F values obtained by sweeping the bias current over the range $I_b \in [0\mathrm{nA}, 10\mathrm{nA}]$, whereas $V_1 = 0.8\mathrm{V}$ and $V_2 = 0.2\mathrm{V}$.

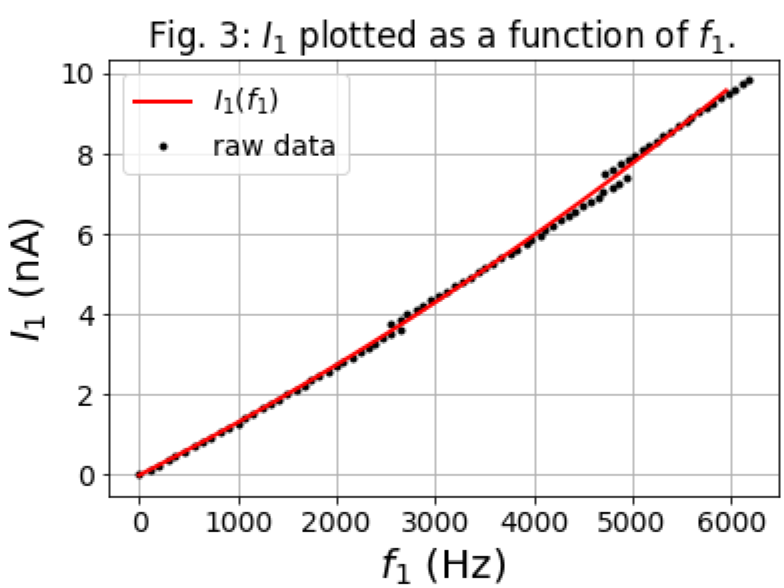
The values for V_1 and V_2 were chosen such that $V_1 \gg V_2$. For these values, the corresponding currents becomes $I_1 \approx I_b$ and $I_2 \approx 0$. The measured data can therefore be utilized to determine the mapping between $I_1 \approx I_b$ and the C2F measurements for the transconductance amplifier.

- Extract the function $I_+(f_+)$ (Hint: use higher order polynomial to increase accuracy)

In [24]:
```
# plot the raw data
raw_UO, = plt.plot(c2f_Iout_UO_ex3_save, Iout_UO_ex3, '.k')

# data range you want to fit
```

```
low_bound = 2
high_bound = 80
# print(c2f_Iout_UO_ex3[low_bound:high_bound])

# fit polynomial to C2F (frequency) vs I data
a2_1, a1_1, a0_1 = np.polyfit(c2f_Iout_UO_ex3_save[low_bound:high_bound], Iout_UO_ex3[low_
# print(a0)
# print(a1)
# print(a2)

# Print out the function I(f) you got
I_freq = np.polyfit(c2f_Iout_UO_ex3_save[low_bound:high_bound], Iout_UO_ex3[low_bound:high
print ('The I1(f1) function of NTA_IOUT_UO is :')
print (np.poly1d(I_freq))


# select frequency range that you want to plot
freq = np.arange(0, 6000, 50)
# print(freq)

I1 = a2_1*freq**2 + a1_1*freq + a0_1    # function I(f)
fit, = plt.plot(freq, I1, 'r-', linewidth=2)


plt.xlabel('$f_1$ (Hz)', {'size':20})
plt.ylabel('$I_1$ (nA)', {'size':20})
plt.legend([fit, raw_UO], ['$I_1(f_1)$', 'raw data'],prop={'size': 14})
plt.title('Fig. 3: $I_1$ plotted as a function of $f_1$. ')
plt.grid()
plt.show()
```

```
The I1(f1) function of NTA_IOUT_UO is :
           2
5.967e-08 x + 0.001255 x - 0.01069
```



3.2.2 NTA_IOUT_UBO

- Set fixed voltages for V_1 and V_2

In [19]:

```
p.set_voltage(pyplane.DacChannel.AIN3,0.2) # V1 = 0.2
time.sleep(0.2) # settle time
p.set_voltage(pyplane.DacChannel.AIN4,0.8) # V2 = 0.8
```

Out[19]: 0.798827052116394

Set voltages such that $V_1 \ll V_2$.

- Data aquisition (Hint: use master current for I_b 30 nA)

In [25]:

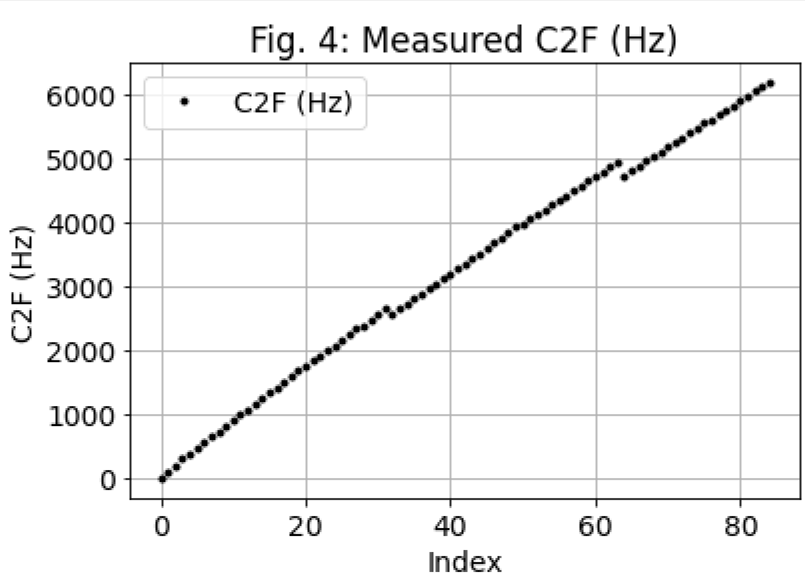
```
import numpy as np
import time

calIout_UBO_ex3 = np.arange(0,85,1) # bias current sweep range

c2f_Iout_UBO_ex3 = []

for n in range(len(calIout_UBO_ex3)):

    # set bias
    p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, calIout_UBO_ex3[n])])

    time.sleep(0.2) # settle time

    # read c2f values
    c2f_Iout_UBO_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UBO_ex3.append(c2f_Iout_UBO_ex3_temp[12])

print(c2f_Iout_UBO_ex3)
```

```
[1, 85, 159, 240, 293, 370, 441, 518, 579, 655, 723, 798, 848, 923, 989, 1065, 1121, 1199,
1266, 1340, 1383, 1461, 1527, 1602, 1652, 1727, 1789, 1860, 1908, 1980, 2035, 2109, 2043,
2116, 2171, 2240, 2287, 2367, 2419, 2492, 2545, 2623, 2676, 2752, 2801, 2863, 2936, 3002,
3060, 3125, 3202, 3280, 3310, 3376, 3446, 3507, 3564, 3624, 3694, 3741, 3781, 3864, 3920,
3986, 3818, 3880, 3934, 3993, 4037, 4109, 4167, 4226, 4281, 4347, 4394, 4459, 4502, 4568,
4636, 4689, 4747, 4824, 4869, 4934, 4973]
```

- Plot

In [36]:

```
plt.plot(c2f_Iout_UO_ex3, '.k', label="C2F (Hz)")
plt.xlabel("Index")
plt.ylabel("C2F (Hz)")
plt.title("Fig. 4: Measured C2F (Hz)")
plt.legend()
plt.grid()
plt.show()
```



- Save data

In [27]:

```
# if the data looks nice, save it!
data_Iout_UBO_ex3_cal= [c2f_Iout_UBO_ex3, calIout_UBO_ex3]
# save to csv file
np.savetxt('./data/c2f_Iout_UBO_ex3_cal_I_minus.csv', data_Iout_UBO_ex3_cal, delimiter=','
```

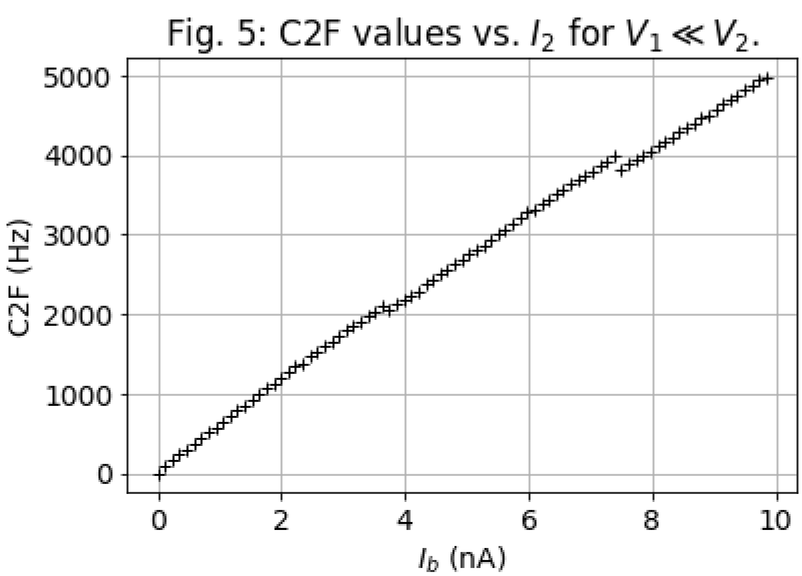
- Load data you saved

In [25]:

```
[c2f_Iout_UO_ex3_save,calIout_UO_ex3_save] = np.loadtxt('./data/c2f_Iout_UBO_ex3_cal_I_mir
```

In [26]:

```
# C2f plot
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 14})

Iout_UO_ex3 = calIout_UO_ex3_save/256*30

plt.plot(Iout_UO_ex3,c2f_Iout_UO_ex3_save,'k+')

plt.xlabel('$I_b$ (nA)')
plt.ylabel('C2F (Hz)')
# plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 5: C2F values vs. $I_2$ for $V_1 \ll V_2$.')
plt.grid()
plt.show()
```



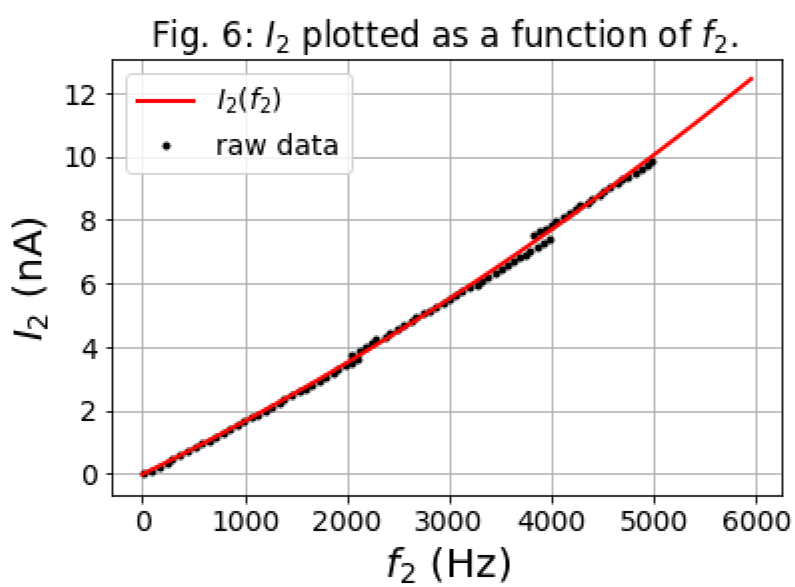
- Extract the function $I_-(f_-)$ (Hint: use higher order polynomial to increase accuracy)

In [27]:

```
# plot the raw data
raw_UO, = plt.plot(c2f_Iout_UO_ex3_save, Iout_UO_ex3, '.k')

# data range you want to fit
low_bound = 2
high_bound = 80
# print(c2f_Iout_UO_ex3[low_bound:high_bound])

# fit polynomial to C2F (frequency) vs I data
a2_2, a1_2, a0_2 = np.polyfit(c2f_Iout_UO_ex3_save[low_bound:high_bound], Iout_UO_ex3[low_
```

```
# print(a0)
# print(a1)
# print(a2)

# Print out the function I(f) you got
I_freq = np.polyfit(c2f_Iout_UO_ex3_save[low_bound:high_bound], Iout_UO_ex3[low_bound:high
print ('The I1(f1) function of NTA_IOUT_UO is :')
print (np.poly1d(I_freq))


# select frequency range that you want to plot
freq = np.arange(0, 6000, 50)
# print(freq)

I1 = a2_2*freq**2 + a1_2*freq + a0_2    # function I(f)
fit, = plt.plot(freq, I1, 'r-', linewidth=2)


plt.xlabel('$f_2$ (Hz)', {'size':20})
plt.ylabel('$I_2$ (nA)', {'size':20})
plt.legend([fit, raw_UO], ['$I_2(f_2)$', 'raw data'],prop={'size': 14})
plt.title('Fig. 6: $I_2$ plotted as a function of $f_2$. ')
plt.grid()
plt.show()
```

```
The I1(f1) function of NTA_IOUT_UO is :
           2
8.479e-08 x + 0.001588 x - 0.01483
```



3.3 Output voltage vs. input voltage

3.3.1 Basic measurement

- Set bias current I_b

In [53]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 42)])
```

The bias current is set to

$$I_b = w\frac{BG_{\text{fine}}}{256} I_{BG_{\text{master}}} = \frac{42}{256} \cdot 30\text{nA} \approx 5\text{nA}.$$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [54]:
```
p.set_voltage(pyplane.DacChannel.AIN4, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN4)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

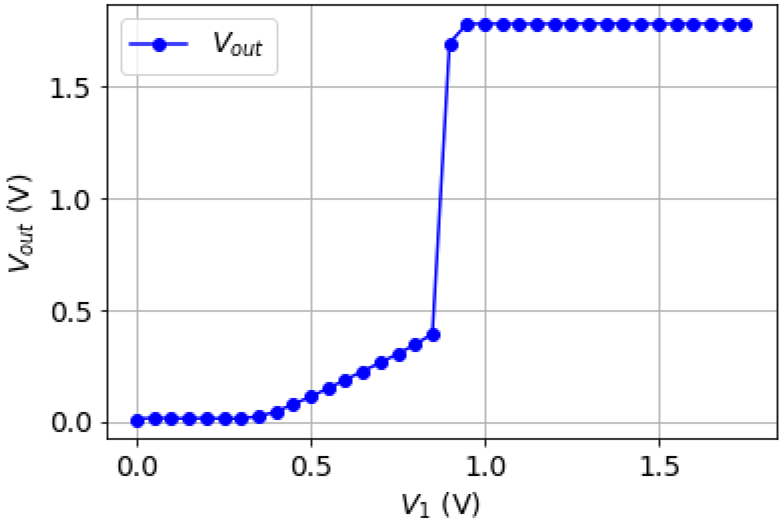
- Sweep V_1 and measure V_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [55]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex3_getset = p.get_set_voltage(pyplane.DacChannel.AIN4)

Vout_V1_sweep_ex3 = []
V1_sweep_ex3_getset = []

for n in range(len(V1_sweep_ex3)):

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex3.append(p.read_voltage(pyplane.AdcChannel.AOUT13))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [56]:
```
plt.plot(V1_sweep_ex3_getset, Vout_V1_sweep_ex3, 'bo-', label="$V_{out}$")
plt.title("Fig. 7: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 7: V_{out} vs. V_1 with $V_2 = 0.9V$ and $I_b = 5nA$



- Save raw data

In [50]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex3 = [V1_sweep_ex3_getset, Vout_V1_sweep_ex3]
# save to csv file
np.savetxt('./data/Vout_sweep_ex3_1.csv', data_Vout_V1_sweep_ex3, delimiter=',')
```

In [ ]:
```
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save] = np.loadtxt('./data/Vout_sweep_ex3_1.c
```

3.3.2 Different bias currents

- Repeat 3.3.1 with another two bias currents and compare the three curves

The bias current was switched from $I_b \approx 5\text{nA}$ to $I_b \approx 10\text{nA}$.

In [57]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [ ]:
```
p.set_voltage(pyplane.DacChannel.AIN4, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN4)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

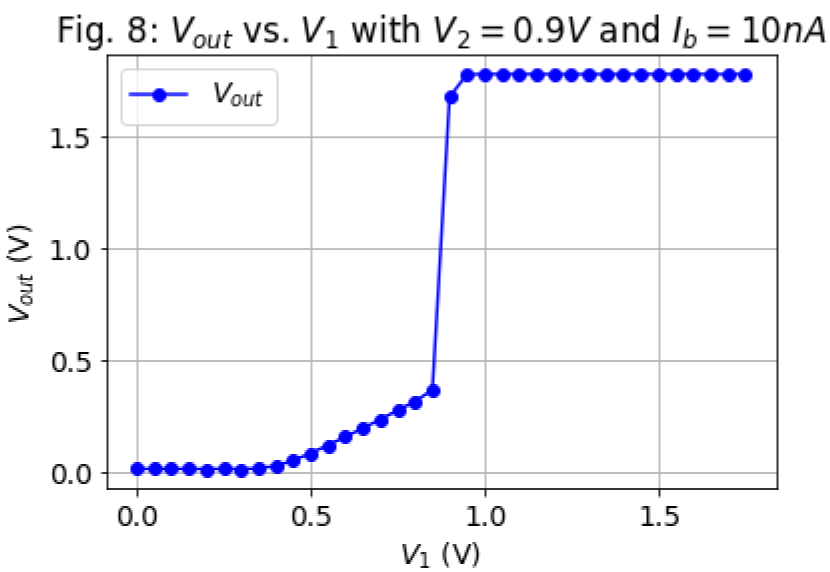
In [58]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex3_getset = p.get_set_voltage(pyplane.DacChannel.AIN4)

Vout_V1_sweep_ex3 = []
V1_sweep_ex3_getset = []
```

```
for n in range(len(V1_sweep_ex3)):

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex3.append(p.read_voltage(pyplane.AdcChannel.AOUT13))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [67]:

```
plt.plot(V1_sweep_ex3_getset, Vout_V1_sweep_ex3, 'bo-', label="$V_{out}$")
plt.title("Fig. 8: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [60]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex3 = [V1_sweep_ex3_getset, Vout_V1_sweep_ex3]
# save to csv file
np.savetxt('./data/Vout_sweep_ex3_2.csv', data_Vout_V1_sweep_ex3, delimiter=',')
```

In [66]:

```
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save] = np.loadtxt('./data/Vout_sweep_ex3_2.c
```

The bias current was switched from $I_b \approx 5\mathrm{nA}$ to $I_b \approx 15\mathrm{nA}$.

In [62]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 122)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

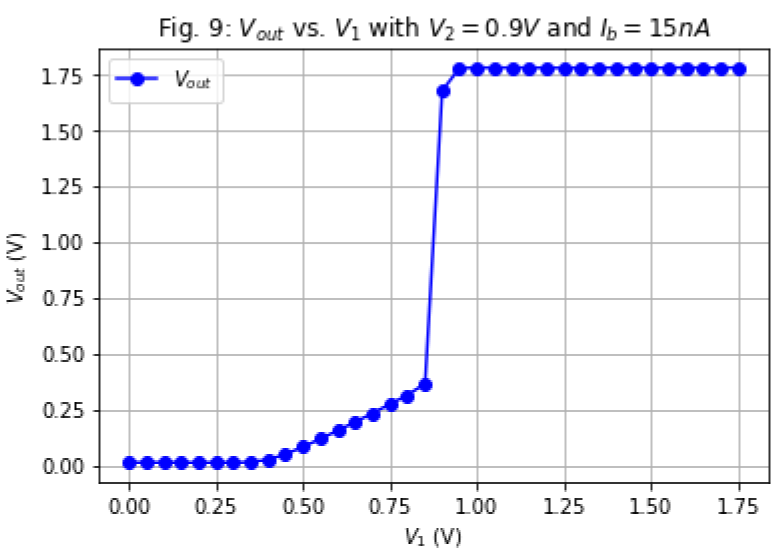
In [10]:
```
p.set_voltage(pyplane.DacChannel.AIN4, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN4)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

In [63]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex3_getset = p.get_set_voltage(pyplane.DacChannel.AIN4)

Vout_V1_sweep_ex3 = []
V1_sweep_ex3_getset = []

for n in range(len(V1_sweep_ex3)):

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex3.append(p.read_voltage(pyplane.AdcChannel.AOUT13))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [5]:
```
plt.plot(V1_sweep_ex3_getset, Vout_V1_sweep_ex3, 'bo-', label="$V_{out}$")
plt.title("Fig. 9: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 15 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

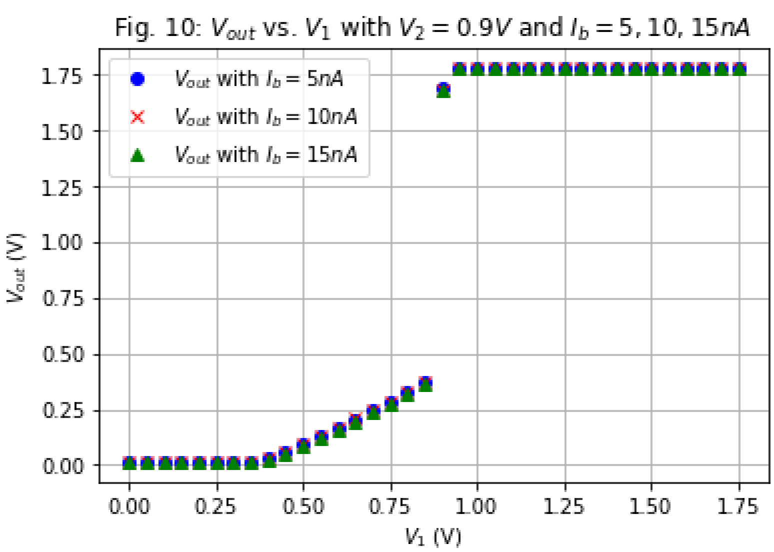


- Save raw data

In [68]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex3 = [V1_sweep_ex3_getset, Vout_V1_sweep_ex3]
# save to csv file
np.savetxt('./data/Vout_sweep_ex3_3.csv', data_Vout_V1_sweep_ex3, delimiter=',')
```

In [4]:
```
[V1_sweep_ex3_getset, Vout_V1_sweep_ex3] = np.loadtxt('./data/Vout_sweep_ex3_3.csv',  deli
```

In [8]:
```
# your codes

[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1] = np.loadtxt('./data/Vout_sweep_ex3_1
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_2] = np.loadtxt('./data/Vout_sweep_ex3_2
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_3] = np.loadtxt('./data/Vout_sweep_ex3_3
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1, 'bo', label="$V_{out}$ with $
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_2, 'rx', label="$V_{out}$ with $
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_3, 'g^', label="$V_{out}$ with $
plt.title("Fig. 10: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5, 10, 15 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



To conclude your observations:

> The bias current does not affect the output voltage when one input voltage is fixed.
> $\frac{dV_{out}}{dI_b} = 0$

3.3.3 Different fixed voltages V_n

- Repeat 3.3.1 with another two fixed voltages V_2 and compare the three curves

Switch voltage from $V_2 = 0.9\text{V}$ to $V_2 = 0.4\text{V}$. The bias current was $I_b = 10\text{nA}$

In [10]:
```
# Set V2 = 04
p.set_voltage(pyplane.DacChannel.AIN4, 0.4) # V2 = 0.4?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN4)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.399413526058197 V
```

The bias current is set to $I_b \approx 10\mathrm{nA}$.

In [11]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [12]:

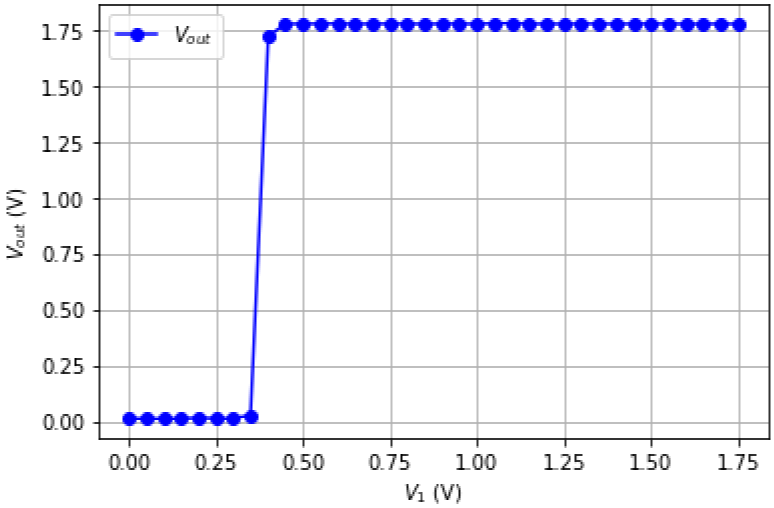
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex3_getset = p.get_set_voltage(pyplane.DacChannel.AIN4)

Vout_V1_sweep_ex3 = []
V1_sweep_ex3_getset = []

for n in range(len(V1_sweep_ex3)):

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex3.append(p.read_voltage(pyplane.AdcChannel.AOUT13))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [13]:

```
plt.plot(V1_sweep_ex3_getset, Vout_V1_sweep_ex3, 'bo-', label="$V_{out}$")
plt.title("Fig. 11: $V_{out}$ vs. $V_1$ with $V_2 = 0.4 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 11: V_{out} vs. V_1 with $V_2 = 0.4V$ and $I_b = 10nA$



- Save raw data

In [14]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex3 = [V1_sweep_ex3_getset, Vout_V1_sweep_ex3]
# save to csv file
np.savetxt('./data/Vout_sweep_ex3_4.csv', data_Vout_V1_sweep_ex3, delimiter=',')
```

In [15]:

```
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save] = np.loadtxt('./data/Vout_sweep_ex3_4.c
```

Switch voltage from $V_2 = 0.9\mathrm{V}$ to $V_2 = 1.2\mathrm{V}$. The bias current was $I_b = 10\mathrm{nA}$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [27]:

```
p.set_voltage(pyplane.DacChannel.AIN4, 1.2) # V2 = 0.2
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN4)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 1.1982406377792358 V
```

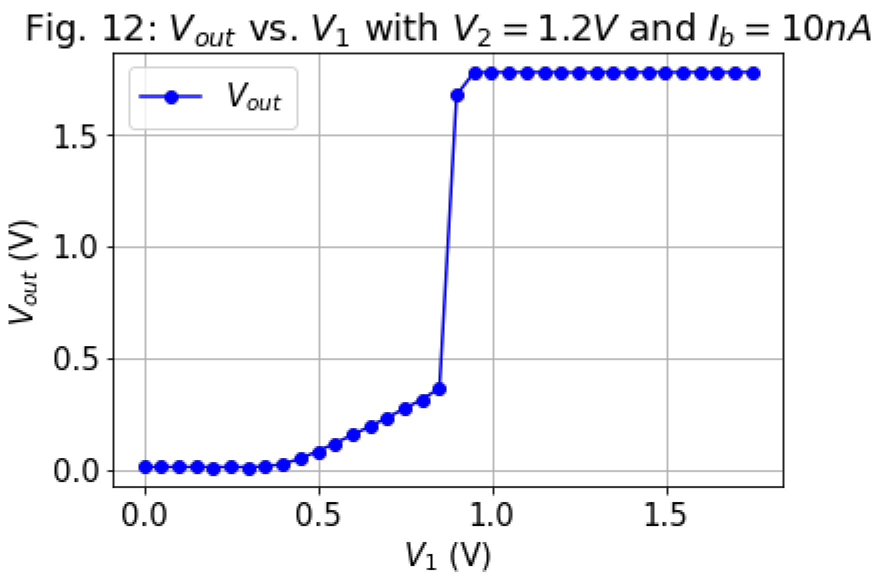
In [28]:

```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex3_getset = p.get_set_voltage(pyplane.DacChannel.AIN4)

Vout_V1_sweep_ex3 = []
V1_sweep_ex3_getset = []

for n in range(len(V1_sweep_ex3)):

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex3.append(p.read_voltage(pyplane.AdcChannel.AOUT13))

# print(V2_ex3_getset)
```

```
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [66]:
```
plt.plot(V1_sweep_ex3_getset, Vout_V1_sweep_ex3, 'bo-', label="$V_{out}$")
plt.title("Fig. 12: $V_{out}$ vs. $V_1$ with $V_2 = 1.2 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

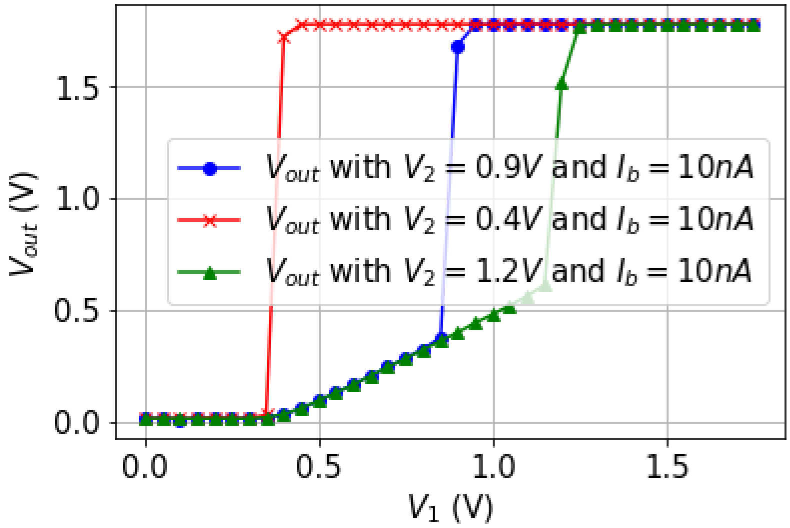


In [30]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex3 = [V1_sweep_ex3_getset, Vout_V1_sweep_ex3]
# save to csv file
np.savetxt('./data/Vout_sweep_ex3_5.csv', data_Vout_V1_sweep_ex3, delimiter=',')
```

In [65]:
```
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save] = np.loadtxt('./data/Vout_sweep_ex3_5.c
```

In [67]:
```
# your codes


[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1] = np.loadtxt('./data/Vout_sweep_ex3_1
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_4] = np.loadtxt('./data/Vout_sweep_ex3_4
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_5] = np.loadtxt('./data/Vout_sweep_ex3_5
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1, 'bo-', label="$V_{out}$ with
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_4, 'rx-', label="$V_{out}$ with
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_5, 'g^-', label="$V_{out}$ with
plt.title("Fig. 13: $V_{out}$ vs. $V_1$ with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 13: V_{out} vs. V_1 with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10nA$



To conclude your observations:

> V_{out} has a linear increase with a slope of κ until V_1 surpasses V_2, at which point V_{out} increases drastically to V_{dd}. So, shifting V_2 means shifting the graph. The most ideal amplifier behavior is reached with a low V_2.

3.4 Output current vs. input voltage

3.4.1 Basic measurement

- Set bias current I_b

In [47]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

Switch bias current back to $I_b = 10\text{nA}$.

- Assign common mode voltage V_{cm}

In [48]:
```
Vcm_ex3 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [49]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN4)

V2_ex3 = []
V1_sweep_ex3_getset = []
V2_ex3_getset = []
c2f_Iout_UO_Vcm_ex3 = []
```

```
c2f_Iout_UBO_Vcm_ex3 = []

for n in range(len(V1_sweep_ex3)):

    # calculate V2 via Vcm and V1
    V2_ex3.append(2*Vcm_ex3-V1_sweep_ex3[n])

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #
    p.set_voltage(pyplane.DacChannel.AIN4,V2_ex3[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
    V2_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN4))

    # read c2f values
    c2f_Iout_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex3.append(c2f_Iout_ex3_temp[11])
    c2f_Iout_UBO_Vcm_ex3.append(c2f_Iout_ex3_temp[12])

# print(V1_sweep_ex3_getset)
# print(V2_ex3_getset)
# print(c2f_Iout_UO_Vcm_ex3)
# print(c2f_Iout_UBO_Vcm_ex3)
```

- Save raw data

In [50]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex3 = [V1_sweep_ex3_getset,V2_ex3_getset,c2f_Iout_UO_Vcm_ex3,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex3.csv', data_Iout_Vcm09_ex3, delimiter=',')
```

- Plot raw data (C2F rate vs. $Vdiff$)

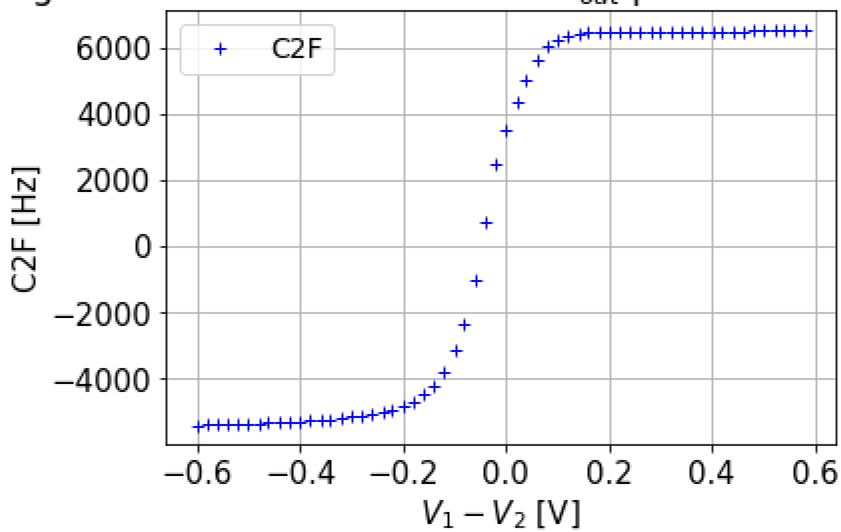
In [51]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UBO

Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
c2f_Iout_Vcm09 = c2f_Iout_UO_Vcm09_ex3 - c2f_Iout_UBO_Vcm09_ex3
print(c2f_Iout_Vcm09)

plt.plot(Vdiff_Vcm09,c2f_Iout_Vcm09,'b+')

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 12: Measured C2F data for $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
```

```
 0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
 0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
 0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-5399. -5392. -5379. -5371. -5364. -5352. -5339. -5327. -5313. -5296.
 -5280. -5263. -5242. -5220. -5189. -5150. -5116. -5066. -5000. -4932.
 -4825. -4673. -4460. -4212. -3775. -3114. -2332.  -997.   745.  2480.
  3512.  4363.  5012.  5623.  6033.  6243.  6384.  6450.  6477.  6484.
  6485.  6483.  6482.  6481.  6475.  6470.  6475.  6483.  6493.  6495.
  6497.  6500.  6506.  6512.  6522.  6529.  6535.  6540.  6546.  6555.]
```

Fig. 12: Measured C2F data for I_{out} plotted over $V_1 - V_2$.



- Convert rate to current and plot

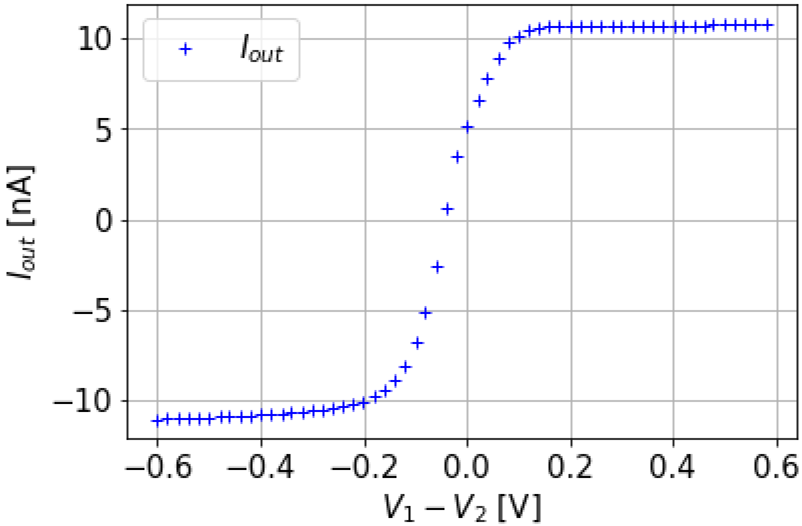
In [65]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 13: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-11.04927323 -11.03424004 -11.00419504 -10.98542851 -10.97041363
```

```
-10.94415038 -10.91664264 -10.89288028 -10.86289847 -10.8291549
-10.79791324 -10.76910425 -10.73400987 -10.696413   -10.64374661
-10.57846772 -10.52675294 -10.44850799 -10.34361371 -10.23471808
-10.05740268  -9.79077914  -9.40038558  -8.92843683  -8.06934214
 -6.74319185  -5.18283258  -2.59702606   0.56295362   3.4724337
  5.1476028    6.61557127   7.79318415   8.94744904   9.74754114
 10.16494104  10.44814861  10.58109276  10.63580413  10.65044318
 10.65247215  10.64841433  10.64594541  10.64435698  10.6321878
 10.62205009  10.6321878   10.64797403  10.66870823  10.67276845
 10.67682914  10.68247874  10.69466511  10.70729953  10.72762805
 10.74186511  10.75362646  10.76380235  10.77646519  10.79479683]
```

Fig. 13: Measured I_{out} plotted over $V_1 - V_2$.



- Compute transconductance

In [66]:
```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

- Explain any asymmetries in the amplifier's I-V curve and the offset voltage in terms of mismatch between devices in the mirror and differential pair, and the Early effect. How can you distinguish the effects of mismatch in the mirror and in the differential pair? The main point here is to recognize that there will be non-idealities, to understand where they arise, and to quantify them in the simplest manner possible.

> A mismatch in the diff-pair moves the curve away from a symmetry at $V_1 - V_2 = 0$, as we can see in the curve above, which is moved a bit to left left. The reason is that through minuscule imperfections, the two input branches won't have the exact same behavior; one branch will tend to conduct a bit more electricity.
> A mismatch in the mirror will cause $|I_{\text{out, min}}| \neq I_{\text{out, max}}$. The reason is that I_1 will not be mirrored correctly to I_2. This seems to not be a problem in the graph above.
> The curve is not completely flat at the edge of the sigmoid because of the Early effect.

3.4.2 Different bias currents

- Repeat 3.4.1 with another two bias currents and compare the three curves

The bias current was switched from $I_b \approx 10\text{nA}$ to $I_b \approx 5\text{nA}$.

In [44]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 42)])
```

- Assign common mode voltage V_{cm}

In [45]:
```
Vcm_ex3 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [46]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN4)

V2_ex3 = []
V1_sweep_ex3_getset = []
V2_ex3_getset = []
c2f_Iout_UO_Vcm_ex3 = []
c2f_Iout_UBO_Vcm_ex3 = []

for n in range(len(V1_sweep_ex3)):

    # calculate V2 via Vcm and V1
    V2_ex3.append(2*Vcm_ex3-V1_sweep_ex3[n])

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #
    p.set_voltage(pyplane.DacChannel.AIN4,V2_ex3[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
    V2_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN4))

    # read c2f values
    c2f_Iout_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex3.append(c2f_Iout_ex3_temp[11])
    c2f_Iout_UBO_Vcm_ex3.append(c2f_Iout_ex3_temp[12])

# print(V1_sweep_ex3_getset)
# print(V2_ex3_getset)
# print(c2f_Iout_UO_Vcm_ex3)
# print(c2f_Iout_UBO_Vcm_ex3)
```

- Save raw data

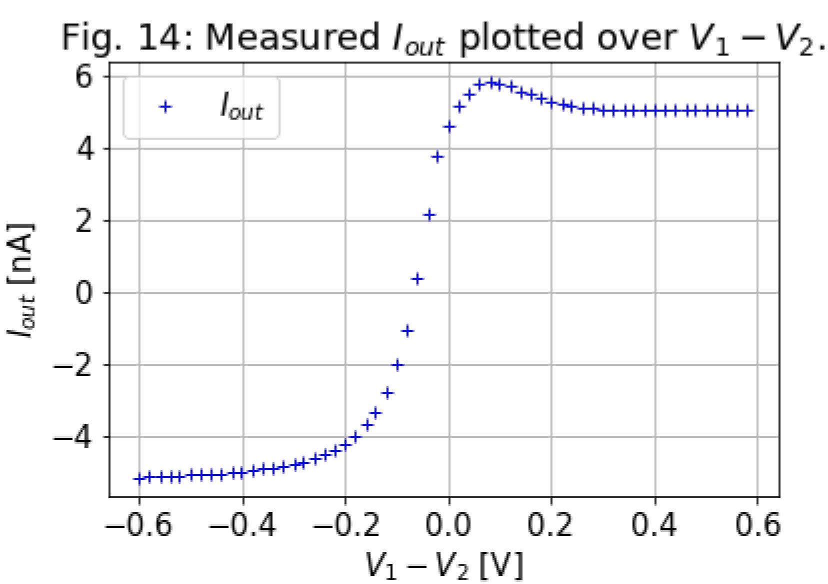
In [47]:
```
# if the data looks nice, save it!
data_Iout_Vcm09_ex3 = [V1_sweep_ex3_getset,V2_ex3_getset,c2f_Iout_UO_Vcm_ex3,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex3_2.csv', data_Iout_Vcm09_ex3, delimiter=',')
```

- Convert rate to current and plot

In [48]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 14: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-5.14960903 -5.13550576 -5.12391864 -5.12221481 -5.10066511 -5.08612073
 -5.07444436 -5.06769542 -5.04936331 -5.01899392 -5.00304727 -4.95762888
 -4.92073978 -4.89694184 -4.83153657 -4.77225272 -4.71278063 -4.63067512
 -4.51408955 -4.39561913 -4.22148833 -3.99689282 -3.69193545 -3.35689283
 -2.80019302 -1.99387406 -1.08140569  0.36199626  2.1354708   3.75131163
  4.58036953  5.16602632  5.50521722  5.7563349   5.81981468  5.78197381
  5.68774816  5.55795943  5.45937392  5.34599043  5.27340952  5.19947123
  5.14592865  5.10077172  5.07071543  5.04736505  5.0289575   5.02229598
  5.01737751  5.01404795  5.0090545   5.01071886  5.01064294  5.02396118
  5.02736906  5.02237275  5.03736491  5.04069781  5.0472865   5.05562264]
```



- Compute transconductance

In [78]:
```
kappa = 0.78
U_T = 0.025
I_b = 5e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 7.8e-08
```

The bias current was switched from $I_b \approx 10\mathrm{nA}$ to $I_b \approx 15\mathrm{nA}$.

In [39]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 122)])
```

- Assign common mode voltage V_{cm}

In [40]:
```
Vcm_ex3 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [41]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN4)

V2_ex3 = []
V1_sweep_ex3_getset = []
V2_ex3_getset = []
c2f_Iout_UO_Vcm_ex3 = []
c2f_Iout_UBO_Vcm_ex3 = []

for n in range(len(V1_sweep_ex3)):

    # calculate V2 via Vcm and V1
    V2_ex3.append(2*Vcm_ex3-V1_sweep_ex3[n])

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #
    p.set_voltage(pyplane.DacChannel.AIN4,V2_ex3[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
    V2_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN4))

    # read c2f values
    c2f_Iout_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex3.append(c2f_Iout_ex3_temp[11])
    c2f_Iout_UBO_Vcm_ex3.append(c2f_Iout_ex3_temp[12])

# print(V1_sweep_ex3_getset)
# print(V2_ex3_getset)
# print(c2f_Iout_UO_Vcm_ex3)
# print(c2f_Iout_UBO_Vcm_ex3)
```

- Save raw data

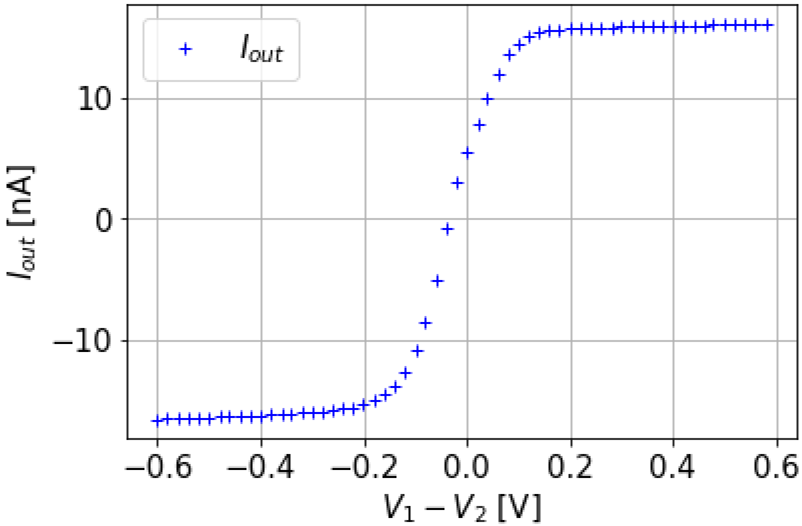
In [42]:
```
# if the data looks nice, save it!
data_Iout_Vcm09_ex3 = [V1_sweep_ex3_getset,V2_ex3_getset,c2f_Iout_UO_Vcm_ex3,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex3_3.csv', data_Iout_Vcm09_ex3, delimiter=',')
```

- Convert rate to current and plot

In [43]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UBO

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 15: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-16.4818515  -16.46065792 -16.44231914 -16.4040708  -16.37596552
 -16.32514748 -16.28856604 -16.26651394 -16.23028813 -16.18558182
 -16.16103849 -16.12265639 -16.04783401 -16.0182652  -15.95131612
 -15.89780095 -15.84298639 -15.76360594 -15.64329713 -15.49920411
 -15.29407736 -14.97369599 -14.47133252 -13.83461945 -12.64802882
 -10.78258246  -8.57341858  -5.12331126  -0.82324655   3.07048062
   5.47973798   7.82443958   9.86784962  11.96917982  13.48728357
  14.33235635  14.95830599  15.30604057  15.48014768  15.59347935
  15.65335572  15.69946936  15.71792819  15.74029193  15.75876698
  15.78259422  15.78721681  15.81033689  15.82884154  15.86124307
  15.87672503  15.8774526   15.90293654  15.92611629  15.94698827
  15.96481715  15.97874219  16.01663136  16.04219407  16.03986959]
```

Fig. 15: Measured I_{out} plotted over $V_1 - V_2$.



- Compute transconductance

In [82]:

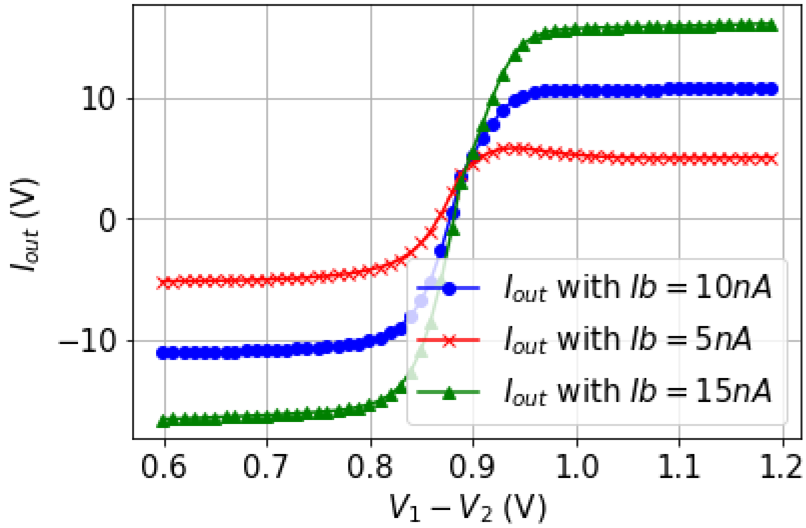
```
kappa = 0.78
U_T = 0.025
I_b = 15e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 2.3399999999999998e-07
```

In [9]:

```
# your codes

import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset_1,V2_Iout_Vcm09_ex3_getset_1,c2f_Iout_UO_Vcm09_ex3_1,c2f_Io
V1_sweep_Iout_Vcm09_ex3_getset_2,V2_Iout_Vcm09_ex3_getset_2,c2f_Iout_UO_Vcm09_ex3_2,c2f_Io
V1_sweep_Iout_Vcm09_ex3_getset_3,V2_Iout_Vcm09_ex3_getset_3,c2f_Iout_UO_Vcm09_ex3_3,c2f_Io

I_plus_1 = a2_1*c2f_Iout_UO_Vcm09_ex3_1**2 + a1_1*c2f_Iout_UO_Vcm09_ex3_1 + a0_1
I_minus_1 = a2_2*c2f_Iout_UBO_Vcm09_ex3_1**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3_1 + a0_2

I_plus_2 = a2_1*c2f_Iout_UO_Vcm09_ex3_2**2 + a1_1*c2f_Iout_UO_Vcm09_ex3_2 + a0_1
I_minus_2 = a2_2*c2f_Iout_UBO_Vcm09_ex3_2**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3_2 + a0_2

I_plus_3 = a2_1*c2f_Iout_UO_Vcm09_ex3_3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3_3 + a0_1
I_minus_3 = a2_2*c2f_Iout_UBO_Vcm09_ex3_3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3_3 + a0_2

plt.plot(V1_sweep_Iout_Vcm09_ex3_getset_1, I_plus_1 - I_minus_1, 'bo-', label="$I_{out}$ w
plt.plot(V1_sweep_Iout_Vcm09_ex3_getset_2, I_plus_2 - I_minus_2, 'rx-', label="$I_{out}$ w
plt.plot(V1_sweep_Iout_Vcm09_ex3_getset_3, I_plus_3 - I_minus_3, 'g^-', label="$I_{out}$ w
plt.title("Fig. 13: $I_{out}$ vs. $V_1-V_2$ with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10 n
plt.xlabel("$V_1-V_2$ (V)")
plt.ylabel("$I_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 13: I_{out} vs. $V_1 - V_2$ with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10nA$



To conclude your observations:

> The range of the sigmoid is proportional to the bias current. The slope is just as proportional, resolting in $I_{\text{out, min}}$ and asI_{\text{out, max}} $always being reached at approximately the same$\varDelta V$

3.4.3 Different common mode voltages

- Repeat 3.4.1 with another two common mode voltages and compare the three curves

In [24]:
```
# Assign common mode voltage Vcm
Vcm_ex3 = 0.45
```

The bias current was switched back to $I_b \approx 10\text{nA}$.

In [25]:
```
 p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [28]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0.2, 0.7, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN4)

V2_ex3 = []
V1_sweep_ex3_getset = []
V2_ex3_getset = []
c2f_Iout_UO_Vcm_ex3 = []
c2f_Iout_UBO_Vcm_ex3 = []

for n in range(len(V1_sweep_ex3)):
```

```
    # calculate V2 via Vcm and V1
    V2_ex3.append(2*Vcm_ex3-V1_sweep_ex3[n])

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n]) #
    p.set_voltage(pyplane.DacChannel.AIN4,V2_ex3[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
    V2_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN4))

    # read c2f values
    c2f_Iout_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex3.append(c2f_Iout_ex3_temp[11])
    c2f_Iout_UBO_Vcm_ex3.append(c2f_Iout_ex3_temp[12])

# print(V1_sweep_ex3_getset)
# print(V2_ex3_getset)
# print(c2f_Iout_UO_Vcm_ex3)
# print(c2f_Iout_UBO_Vcm_ex3)
```

- Save raw data

In [29]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex3 = [V1_sweep_ex3_getset,V2_ex3_getset,c2f_Iout_UO_Vcm_ex3,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex3_5.csv', data_Iout_Vcm09_ex3, delimiter=',')
```

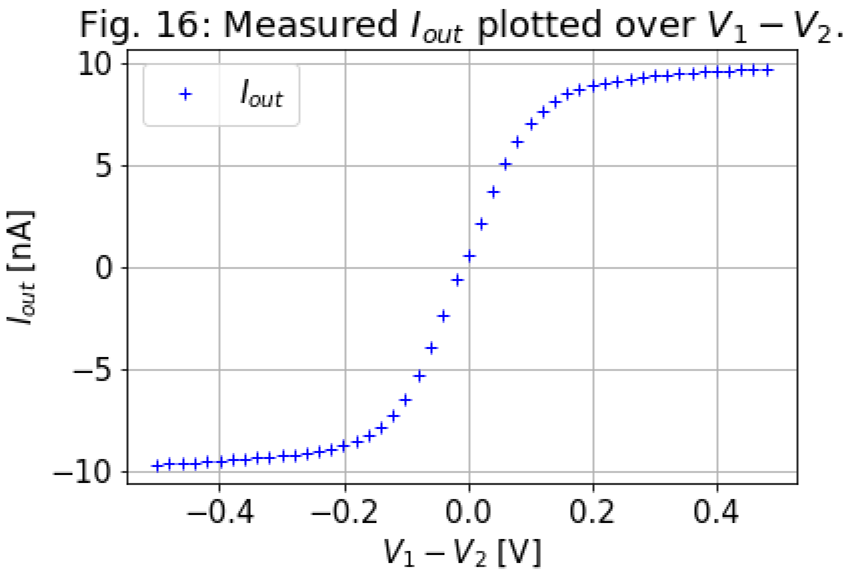
- Convert rate to current and plot

In [31]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 16: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.49970683 -0.48035194 -0.45923758 -0.43988276 -0.42052792 -0.39941356
 -0.38005868 -0.36070383 -0.33958948 -0.32023466 -0.29912028 -0.27976546
 -0.26041058 -0.2392962  -0.21994138 -0.20058653 -0.17947218 -0.16011736
 -0.14076245 -0.1196481  -0.10029328 -0.0791789  -0.05982405 -0.04046923
 -0.01935485  0.          0.01935485  0.04046923  0.05982405  0.0791789
  0.10029328  0.1196481   0.14076245  0.16011736  0.17947218  0.20058653
  0.21994138  0.2392962   0.26041058  0.27976546  0.29912028  0.32023466
  0.33958948  0.36070383  0.38005868  0.39941356  0.42052792  0.43988276
  0.45923758  0.48035194]
```

```
[-9.67819741 -9.65169952 -9.62041029 -9.58559795 -9.54831376 -9.51951789
 -9.47043886 -9.42849205 -9.37829336 -9.33180962 -9.26859566 -9.20437501
 -9.12967673 -9.02190625 -8.91808983 -8.753604   -8.5270472  -8.25254093
 -7.84100843 -7.2614521  -6.51411103 -5.29729611 -3.91310765 -2.37536086
 -0.57574807  0.57412705  2.12581461  3.75822132  5.08239939  6.18758162
  7.09034229  7.67851314  8.15466922  8.49433063  8.70794984  8.91121131
  9.06163752  9.1545684   9.25519719  9.3311434   9.37247649  9.42914313
  9.47807424  9.52707994  9.5659626   9.59740168  9.62728288  9.65878306
  9.69228549  9.71595502]
```



- Compute transconductance

In [32]:
```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

In [33]:
```
 p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Assign common mode voltage V_{cm}

In [34]:
```
Vcm_ex3 = 0.2 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [35]:
```
import numpy as np
import time

V1_sweep_ex3 = np.arange(0.1, 0.3, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN4)

V2_ex3 = []
```

```
V1_sweep_ex3_getset = []
V2_ex3_getset = []
c2f_Iout_UO_Vcm_ex3 = []
c2f_Iout_UBO_Vcm_ex3 = []

for n in range(len(V1_sweep_ex3)):

    # calculate V2 via Vcm and V1
    V2_ex3.append(2*Vcm_ex3-V1_sweep_ex3[n])

    p.set_voltage(pyplane.DacChannel.AIN3,V1_sweep_ex3[n])  #
    p.set_voltage(pyplane.DacChannel.AIN4,V2_ex3[n])  #

    time.sleep(0.2)  # settle time

    V1_sweep_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN3))
    V2_ex3_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN4))

    # read c2f values
    c2f_Iout_ex3_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex3.append(c2f_Iout_ex3_temp[11])
    c2f_Iout_UBO_Vcm_ex3.append(c2f_Iout_ex3_temp[12])

# print(V1_sweep_ex3_getset)
# print(V2_ex3_getset)
# print(c2f_Iout_UO_Vcm_ex3)
# print(c2f_Iout_UBO_Vcm_ex3)
```

- Save raw data

In [36]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex3 = [V1_sweep_ex3_getset,V2_ex3_getset,c2f_Iout_UO_Vcm_ex3,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex3_6.csv', data_Iout_Vcm09_ex3, delimiter=',')
```

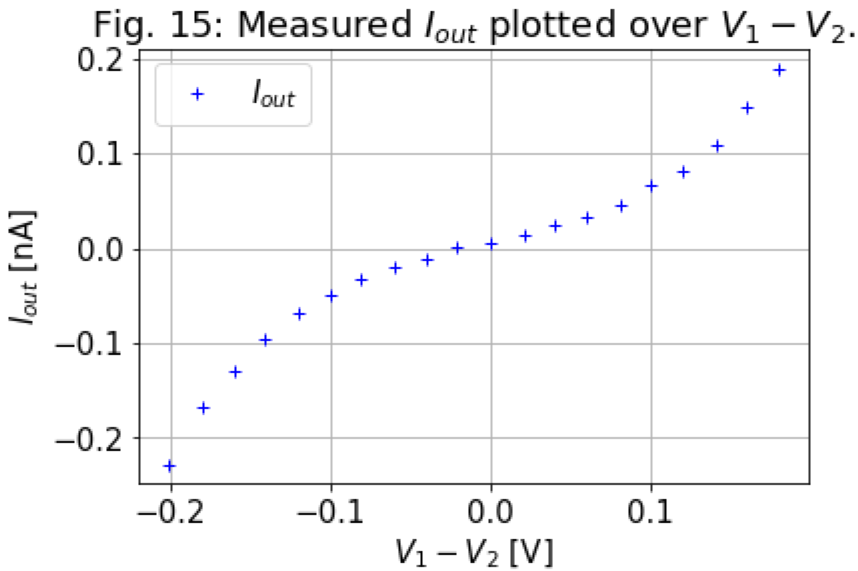
- Convert rate to current and plot

In [37]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 15: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.20058654 -0.17947215 -0.16011732 -0.14076249 -0.1196481  -0.10029326
 -0.08093843 -0.05982405 -0.04046921 -0.02111436  0.          0.02111436
```

```
  0.04046921  0.05982405  0.08093843  0.10029326  0.1196481   0.14076249
  0.16011732  0.17947215]
[-0.22795071 -0.16873977 -0.13022458 -0.09625979 -0.06784572 -0.05029833
 -0.03310884 -0.02003752 -0.01174919  0.00063197  0.00506447  0.01385363
  0.02390555  0.03270725  0.0465503   0.06577805  0.08185232  0.10965527
  0.14893019  0.18959235]
```



- Compute transconductance

In [38]:

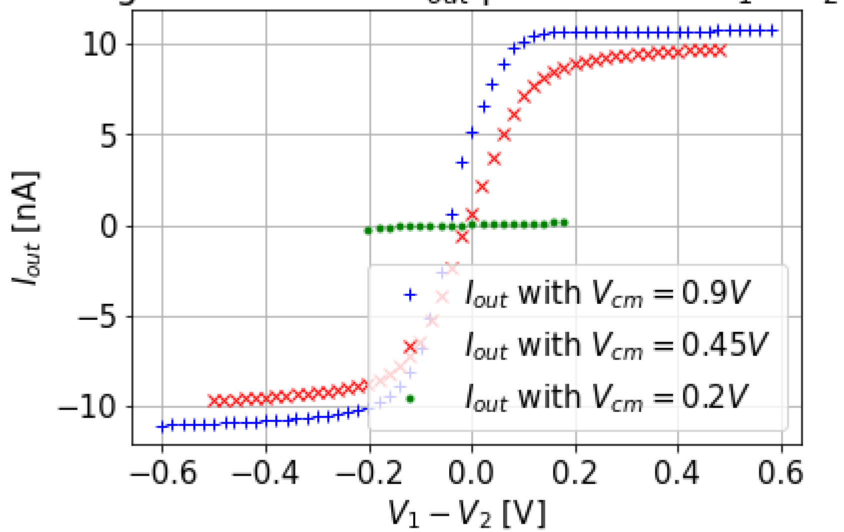
```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

In [28]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

files = ['./data/V1_sweep_Iout_Vcm09_ex3.csv', './data/V1_sweep_Iout_Vcm09_ex3_5.csv', './
styles = ['b+','rx', 'g.']
v_cms = [0.9, 0.45, 0.2]
for (file, style), v_cm in zip(zip(files, styles), v_cms):
    V1_sweep_Iout_Vcm09_ex3_getset,V2_Iout_Vcm09_ex3_getset,c2f_Iout_UO_Vcm09_ex3,c2f_Iout

    I_plus = a2_1*c2f_Iout_UO_Vcm09_ex3**2 + a1_1*c2f_Iout_UO_Vcm09_ex3 + a0_1
    I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex3 + a0_2
    Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex3_getset-V2_Iout_Vcm09_ex3_getset
    I_out = I_plus - I_minus

    plt.plot(Vdiff_Vcm09,I_out, style, label=f"$I_{{out}}$ with $V_{{cm}} = {v_cm}V$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')

plt.legend()
plt.title('Fig. 15: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

Fig. 15: Measured I_{out} plotted over $V_1 - V_2$.



To conclude your observations:

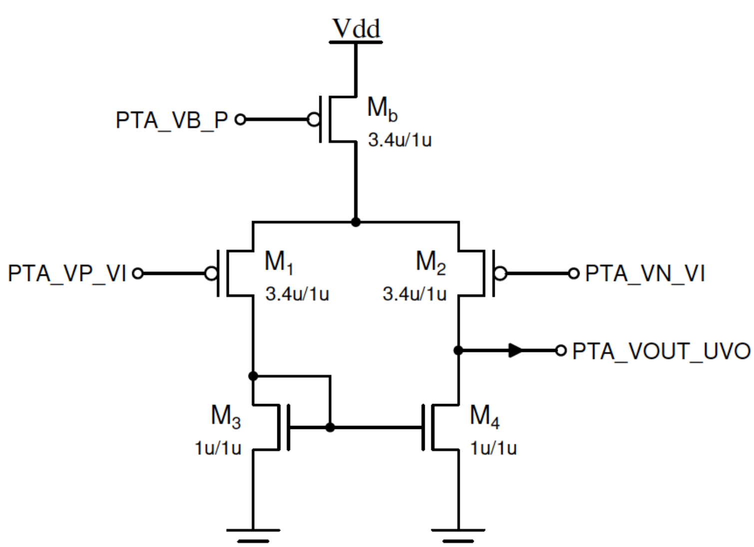
> Increasing V_{cm} has the same effect as increasing I_b, but a lower V_{cm} results in a lower range of available ΔV

What do you observe when the common mode voltage V_{cm} is too small (e.g. 0.2V or 0.3V)? Does it have a sigmoid shape? If not, try to explain why.

> The sigmoid is flipped: The borders are steep and the center is flat. M_4 of the schematic above is no longer in saturation, its requirement is $\max(V_1, V_2) > V_b + \frac{4U_T}{\kappa}$. With $V_{cm} = 0.2V$, $\max(V_1, V_2) \leq 0.3V$ is too low.

4 P-Type 5T Transamp

4.0 Schematic and pin map



V_1 = V_p = **PTA_VP_VI = AIN7**

V_2 = V_n = **PTA_VN_VI = AIN8**

V_{out} **= PTA_VOUT_UVO = ADC[12]**

I_{out} **=** $I_+ - I_-$ **= PTA_IOUT_UO - PTA_IOUT_UBO = C2F[13] - C2F[14]**

Note: There are three identical PTA circuits with the same bias and input voltages, one with the output open-circuited and routed out at PTA_VOUT*UVO, the other two with $V{out}*$fixedto1Vbut$ I_{out}$ routed out through N- and P- type current mirror at PTA_IOUT_UO and PTA_IOUT_UBO.

4.1 Chip configuration

In [8]:
```
p.send_coach_events([pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.SelectLine1, \
    pyplane.Coach.VoltageInputSelect.SelectLine1, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)])
```

4.2 Calibration of C2F channels

Here you need to calibrate PTA_IOUT_UO and PTA_IOUT_UBO in the same way as the last lab. Notice the W/L ratio of 3.4 of Mb.

4.2.1 PTA_IOUT_UO

- Set fixed voltages for V_1 and V_2

In [9]:
```
p.set_voltage(pyplane.DacChannel.AIN7, 0.8) # V1 = ???
p.set_voltage(pyplane.DacChannel.AIN8, 0.2) # V2 = ???
```

Out[9]:
```
0.19882699847221375
```

Set $V_1 \gg V_2$.

- Data aquisition (Hint: use master current 30 nA)

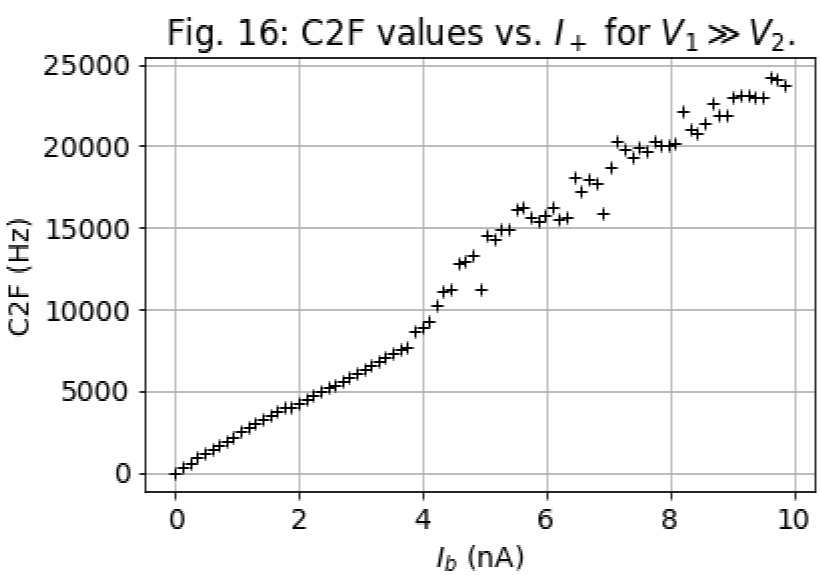
In [10]:
```
import numpy as np
import time

calIout_UO_ex4 = np.arange(0,85,1) # bias current sweep range, fine value

c2f_Iout_UO_ex4 = []  # what you get is frequency

for n in range(len(calIout_UO_ex4)):

    # set bias
    p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, calIout_UO_ex4[n])])

    time.sleep(0.2) # settle time

    # read c2f values
    c2f_Iout_UO_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_ex4.append(c2f_Iout_UO_ex4_temp[13])

print(c2f_Iout_UO_ex4)
```

```
[1, 313, 570, 873, 1112, 1406, 1649, 1934, 2192, 2470, 2706, 2989, 3211, 3487, 3710, 3966, 3977, 4244, 4456, 4711, 4919, 5170, 5386, 5637, 5858, 6108, 6312, 6564, 6772, 7029, 7258, 7596, 7660, 8605, 8932, 9276, 10236, 11148, 11209, 12837, 12942, 13271, 11197, 14479, 14315, 14893, 14860, 16069, 16280, 15688, 15433, 15768, 16227, 15541, 15647, 18150, 17282, 17946, 17724, 15863, 18737, 20275, 19781, 19275, 19928, 19653, 20330, 20082, 20110, 20195, 22121, 21014, 20835, 21447, 22632, 21859, 21838, 23026, 23065, 23071, 23018, 22991, 24242, 24060, 23714]
```

- Plot

In [30]:

```
# C2f plot
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 14})

Iout_UO_ex4 = calIout_UO_ex4_save/256*30

plt.plot(Iout_UO_ex4,c2f_Iout_UO_ex4_save, 'k+')

plt.xlabel('$I_b$ (nA)')
plt.ylabel('C2F (Hz)')
# plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 16: C2F values vs. $I_+$ for $V_1 \gg V_2$.')
plt.grid()
plt.show()
```



- Save raw data

In [11]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [c2f_Iout_UO_ex4, calIout_UO_ex4]
# save to csv file
np.savetxt('./data/c2f_Iout_UO_ex4_cal_I_plus.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

- Load data

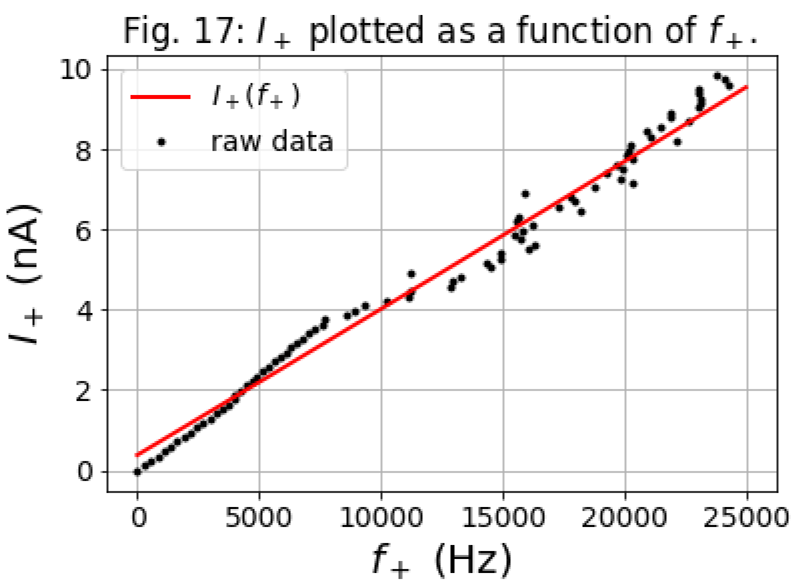
In [29]:

```
[c2f_Iout_UO_ex4_save, calIout_UO_ex4_save] = np.loadtxt('./data/c2f_Iout_UO_ex4_cal_I_plu
```

- Extract the function $I_+(f_+)$ (Hint: use higher order polynomial to increase accuracy)

In [31]:
```python
# plot the raw data
raw_UO, = plt.plot(c2f_Iout_UO_ex4_save, Iout_UO_ex4, '.k')

# data range you want to fit
low_bound = 2
high_bound = 80
# print(c2f_Iout_UO_ex5[low_bound:high_bound])

# fit polynomial to C2F (frequency) vs I data
a2_1, a1_1, a0_1 = np.polyfit(c2f_Iout_UO_ex4_save[low_bound:high_bound], Iout_UO_ex4[low_
# print(a0)
# print(a1)
# print(a2)

# Print out the function I(f) you got
I_freq = np.polyfit(c2f_Iout_UO_ex4_save[low_bound:high_bound], Iout_UO_ex4[low_bound:high
print ('The I+(f+) function of NTA_IOUT_UO is :')
print (np.poly1d(I_freq))


# select frequency range that you want to plot
freq = np.arange(0, 25000, 50)
# print(freq)

I1 = a2_1*freq**2 + a1_1*freq + a0_1    # function I(f)
fit, = plt.plot(freq, I1, 'r-', linewidth=2)


plt.xlabel('$f_+$ (Hz)', {'size':20})
plt.ylabel('$I_+$ (nA)', {'size':20})
plt.legend([fit, raw_UO], ['$I_+(f_+)$', 'raw data'],prop={'size': 14})
plt.title('Fig. 17: $I_+$ plotted as a function of $f_+$. ')
plt.grid()
plt.show()
```

```
The I+(f+) function of NTA_IOUT_UO is :
          2
2.05e-10 x + 0.0003619 x + 0.377
```



4.2.2 PTA_IOUT_UBO

- Set fixed voltages for V_1 and V_2

In [17]:
```python
p.set_voltage(pyplane.DacChannel.AIN7, 0.2) # V1 = ??
p.set_voltage(pyplane.DacChannel.AIN8, 0.8) # V2 = ??
```

Out[17]: 0.798827052116394

Set $V_1 \ll V_2$.

- Data aquisition (Hint: use master current 30 nA)

In [18]:

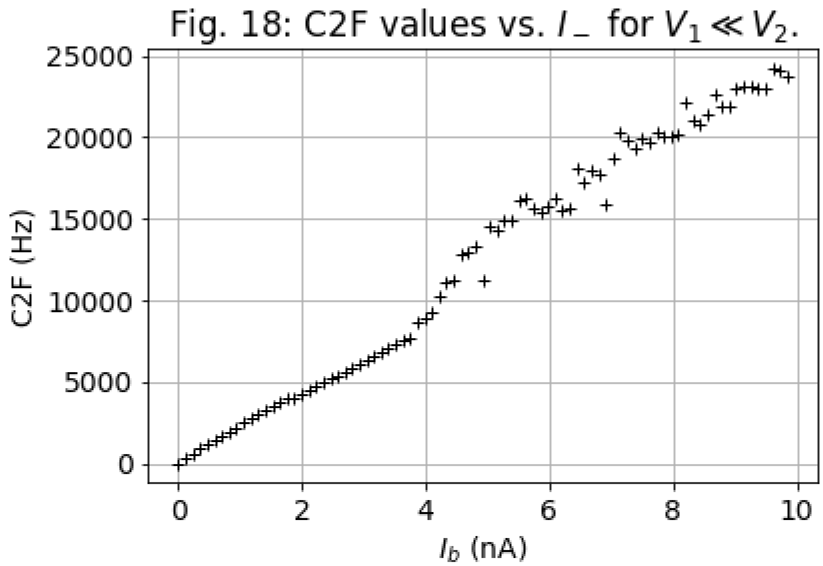
```
import numpy as np
import time

calIout_UO_ex4 = np.arange(0,85,1) # bias current sweep range, fine value

c2f_Iout_UO_ex4 = []  # what you get is frequency

for n in range(len(calIout_UO_ex4)):

    # set bias
    p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, calIout_UO_ex4[n])])

    time.sleep(0.2) # settle time

    # read c2f values
    c2f_Iout_UO_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_ex4.append(c2f_Iout_UO_ex4_temp[14])

print(c2f_Iout_UO_ex4)
```

```
[207, 494, 809, 1174, 1430, 1894, 2112, 2540, 2959, 3366, 3698, 3908, 4271, 4664, 4931, 59
11, 5791, 5501, 6002, 6141, 6345, 6953, 7431, 8182, 7755, 6657, 8384, 8423, 9071, 9111, 10
553, 10001, 9963, 9992, 9972, 10746, 11344, 9359, 11552, 12242, 12049, 12277, 12081, 1033
7, 13877, 13619, 14343, 12548, 12626, 15869, 15178, 14079, 15986, 15385, 15216, 16272, 157
73, 16951, 16599, 16160, 12898, 17900, 18548, 17471, 18769, 17960, 18455, 18833, 17848, 13
672, 17916, 21499, 20687, 20089, 20973, 20646, 20588, 20119, 21471, 21922, 21887, 21641, 2
1723, 21869, 23074]
```

- Plot

In [33]:

```
# C2f plot
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 14})

Iout_UO_ex4 = calIout_UO_ex4_save/256*30

plt.plot(Iout_UO_ex4,c2f_Iout_UO_ex4_save,'k+')

plt.xlabel('$I_b$ (nA)')
plt.ylabel('C2F (Hz)')
# plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 18: C2F values vs. $I_-$ for $V_1 \ll V_2$.')
plt.grid()
plt.show()
```



- Save raw data

In [19]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [c2f_Iout_UO_ex4, calIout_UO_ex4]
# save to csv file
np.savetxt('./data/c2f_Iout_UBO_ex4_cal_I_minus.csv', data_Vout_V1_sweep_ex4, delimiter=',
```

- Load data

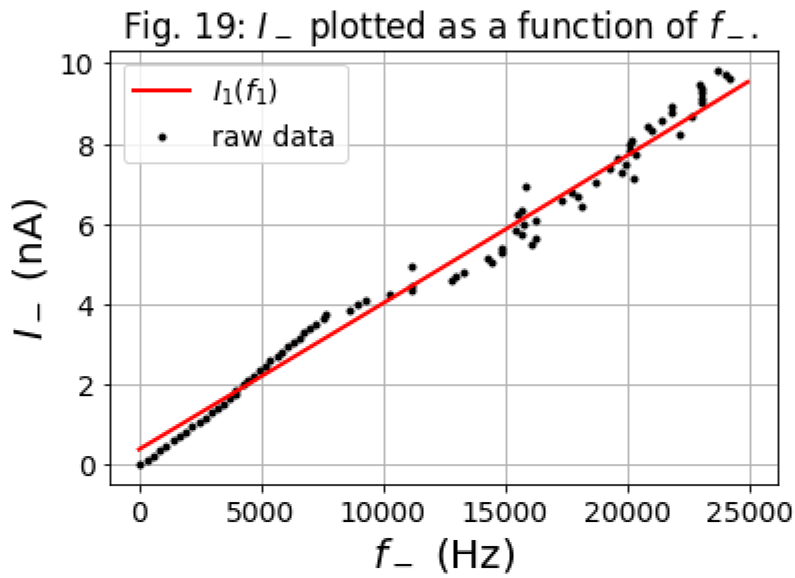
In [32]:
```
# if the data looks nice, save it!
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save] = np.loadtxt('./data/c2f_Iout_UBO_ex4_c
```

- Extract the function $I_-(f_-)$ (Hint: use higher order polynomial to increase accuracy)

In [35]:
```
# plot the raw data
raw_UO, = plt.plot(c2f_Iout_UO_ex4_save, Iout_UO_ex4, '.k')

# data range you want to fit
low_bound = 2
high_bound = 80
# print(c2f_Iout_UO_ex5[low_bound:high_bound])

# fit polynomial to C2F (frequency) vs I data
a2_2, a1_2, a0_2 = np.polyfit(c2f_Iout_UO_ex4_save[low_bound:high_bound], Iout_UO_ex4[low_
# print(a0)
# print(a1)
# print(a2)

# Print out the function I(f) you got
I_freq = np.polyfit(c2f_Iout_UO_ex4_save[low_bound:high_bound], Iout_UO_ex4[low_bound:high
print ('The I1(f1) function of NTA_IOUT_UO is :')
print (np.poly1d(I_freq))


# select frequency range that you want to plot
freq = np.arange(0, 25000, 50)
# print(freq)

I1 = a2_2*freq**2 + a1_2*freq + a0_2   # function I(f)
fit, = plt.plot(freq, I1, 'r-', linewidth=2)
```

```
plt.xlabel('$f_-$ (Hz)', {'size':20})
plt.ylabel('$I_-$ (nA)', {'size':20})
plt.legend([fit, raw_UO], ['$I_1(f_1)$', 'raw data'],prop={'size': 14})
plt.title('Fig. 19: $I_-$ plotted as a function of $f_-$. ')
plt.grid()
plt.show()
```

```
The I1(f1) function of NTA_IOUT_UO is :
          2
2.05e-10 x + 0.0003619 x + 0.377
```



4.3 Output voltage vs. input voltage

4.3.1 Basic measurement

- Set bias current I_b

In [33]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
   pyplane.Coach.BiasAddress.PTA_VB_P, \
   pyplane.Coach.BiasType.P, \
   pyplane.Coach.BiasGenMasterCurrent.I30nA, 42)])
```

The bias current is set to

$$I_b = w\frac{BG_{\text{fine}}}{256}I_{BG_{\text{master}}} = \frac{42}{256} \cdot 30\text{nA} = 5\text{nA}.$$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [34]:

```
p.set_voltage(pyplane.DacChannel.AIN8, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

- Sweep V_1 and measure V_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

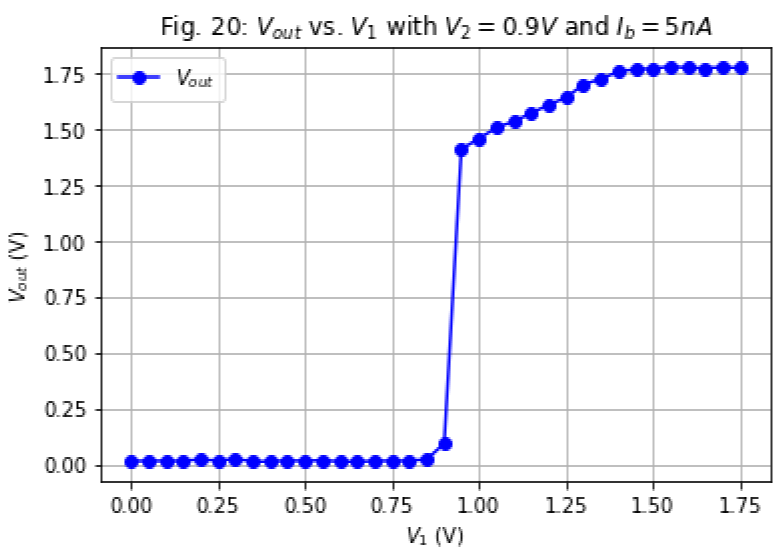
In [35]:

```
import numpy as np
```

```
import time

V1_sweep_ex4 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex4_getset = p.get_set_voltage(pyplane.DacChannel.AIN7)

Vout_V1_sweep_ex4 = []
V1_sweep_ex4_getset = []

for n in range(len(V1_sweep_ex4)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex5.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex4.append(p.read_voltage(pyplane.AdcChannel.AOUT12))

# print(V2_ex5_getset)
# print(V1_sweep_ex5_getset)
# print(Vout_V1_sweep_ex5)
```

- Plot raw data

In [36]:
```
plt.plot(V1_sweep_ex4_getset, Vout_V1_sweep_ex4, 'bo-', label="$V_{out}$")
plt.title("Fig. 20: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [37]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [V1_sweep_ex4_getset, Vout_V1_sweep_ex4]
# save to csv file
np.savetxt('./data/Vout_sweep_ex4_1.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

In [19]:
```
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save] = np.loadtxt('./data/Vout_sweep_ex4_1.c
```

4.3.2 Different bias currents

- Repeat 4.3.1 with another two bias currents and compare the three curves

Switch bias current from $I_b = 5\text{nA}$ in the basic measurement to $I_b = 10\text{nA}$.

In [20]:
```
 p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [21]:
```
p.set_voltage(pyplane.DacChannel.AIN8, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

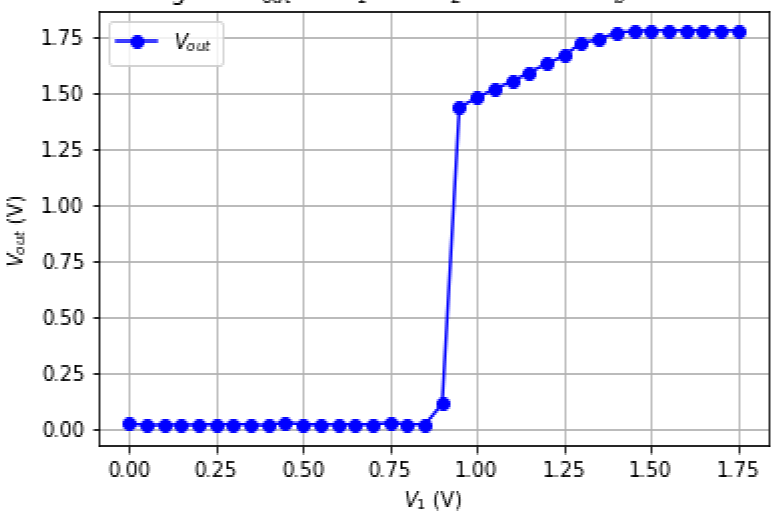
- Sweep V_1 and measure V_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [22]:
```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex4_getset = p.get_set_voltage(pyplane.DacChannel.AIN7)

Vout_V1_sweep_ex4 = []
V1_sweep_ex4_getset = []

for n in range(len(V1_sweep_ex4)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex4.append(p.read_voltage(pyplane.AdcChannel.AOUT12))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [23]:
```
plt.plot(V1_sweep_ex4_getset, Vout_V1_sweep_ex4, 'bo-', label="$V_{out}$")
plt.title("Fig. 21: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 21: V_{out} vs. V_1 with $V_2 = 0.9V$ and $I_b = 10nA$



- Save raw data

In [24]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [V1_sweep_ex4_getset, Vout_V1_sweep_ex4]
# save to csv file
np.savetxt('./data/Vout_sweep_ex4_2.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

In [25]:
```
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save] = np.loadtxt('./data/Vout_sweep_ex4_2.c
```

Switch bias current from $I_b = 5\mathrm{nA}$ in the basic measurement to $I_b = 15\mathrm{nA}$.

In [26]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 128)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [27]:
```
p.set_voltage(pyplane.DacChannel.AIN8, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

- Sweep V_1 and measure V_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

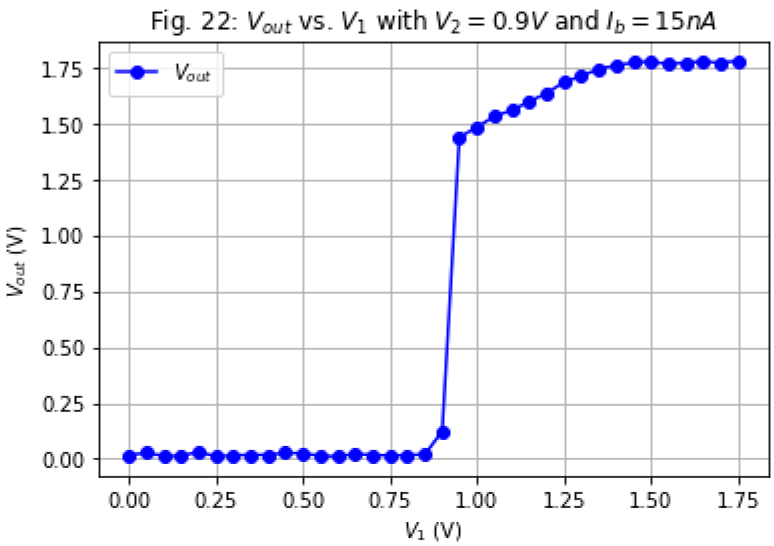
In [28]:
```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex4_getset = p.get_set_voltage(pyplane.DacChannel.AIN7)

Vout_V1_sweep_ex4 = []
V1_sweep_ex4_getset = []

for n in range(len(V1_sweep_ex4)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
```

```
    time.sleep(0.3) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex4.append(p.read_voltage(pyplane.AdcChannel.AOUT12))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [29]:

```
plt.plot(V1_sweep_ex4_getset, Vout_V1_sweep_ex4, 'bo-', label="$V_{out}$")
plt.title("Fig. 22: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 15 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [30]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [V1_sweep_ex4_getset, Vout_V1_sweep_ex4]
# save to csv file
np.savetxt('./data/Vout_sweep_ex4_3.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

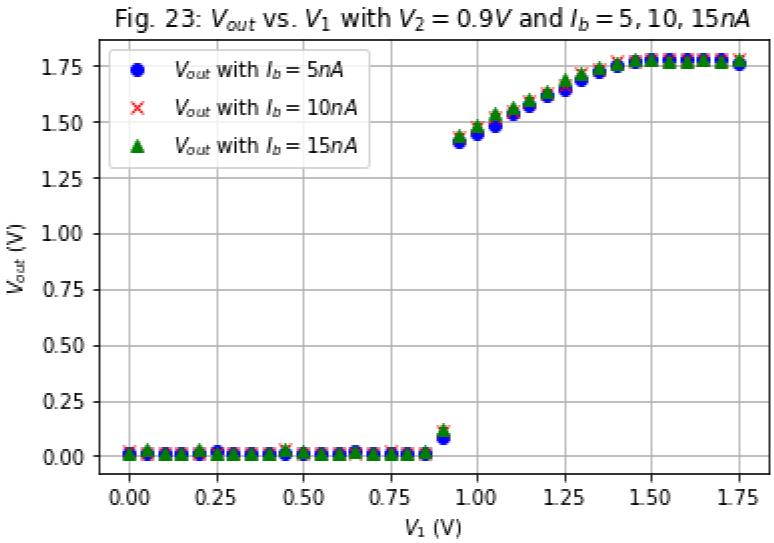
In [31]:

```
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save] = np.loadtxt('./data/Vout_sweep_ex4_3.c
```

In [32]:

```
# your codes

[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1] = np.loadtxt('./data/Vout_sweep_ex4_1
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_2] = np.loadtxt('./data/Vout_sweep_ex4_2
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_3] = np.loadtxt('./data/Vout_sweep_ex4_3
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1, 'bo', label="$V_{out}$ with $
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_2, 'rx', label="$V_{out}$ with $
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_3, 'g^', label="$V_{out}$ with $
plt.title("Fig. 23: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5, 10, 15 nA$")
plt.xlabel("$V_1$ (V)")
```

```
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



To conclude your observations:

> The bias current does not affect the output voltage when one input voltage is fixed.
> $\frac{dV_{out}}{dI_b} = 0$

4.3.3 Different fixed voltages V_n

- Repeat 4.3.1 with another two fixed voltages V_2 and compare the three curves

Switch voltage from $V_2 = 0.9\text{V}$ to $V_2 = 0.4\text{V}$. The bias current was $I_b = 10\text{nA}$

In [39]:
```
# Set V2 = 04
p.set_voltage(pyplane.DacChannel.AIN8, 0.4) # V2 = 0.4
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.399413526058197 V
```

The bias current is set to $I_b \approx 10\text{nA}$.

In [40]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

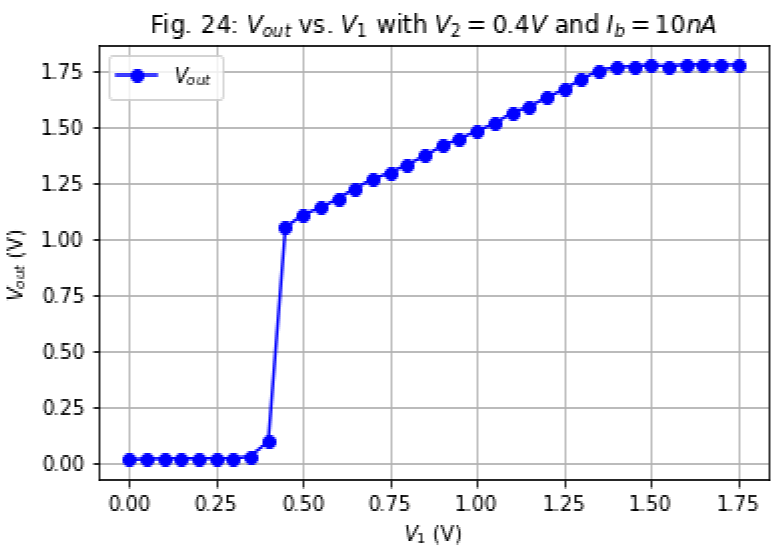
In [41]:
```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex4_getset = p.get_set_voltage(pyplane.DacChannel.AIN8)

Vout_V1_sweep_ex4 = []
V1_sweep_ex4_getset = []
```

```
for n in range(len(V1_sweep_ex4)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex4.append(p.read_voltage(pyplane.AdcChannel.AOUT12))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [42]:

```
plt.plot(V1_sweep_ex4_getset, Vout_V1_sweep_ex4, 'bo-', label="$V_{out}$")
plt.title("Fig. 24: $V_{out}$ vs. $V_1$ with $V_2 = 0.4 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [43]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [V1_sweep_ex4_getset, Vout_V1_sweep_ex4]
# save to csv file
np.savetxt('./data/Vout_sweep_ex4_3.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

In [44]:

```
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save] = np.loadtxt('./data/Vout_sweep_ex4_3.c
```

Switch voltage from $V_2 = 0.9\mathrm{V}$ to $V_2 = 1.2\mathrm{V}$. The bias current was $I_b = 10\mathrm{nA}$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [45]:

```
p.set_voltage(pyplane.DacChannel.AIN8, 1.2) # V2 = 0.2
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
```

```python
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 1.1982406377792358 V
```

In [47]:

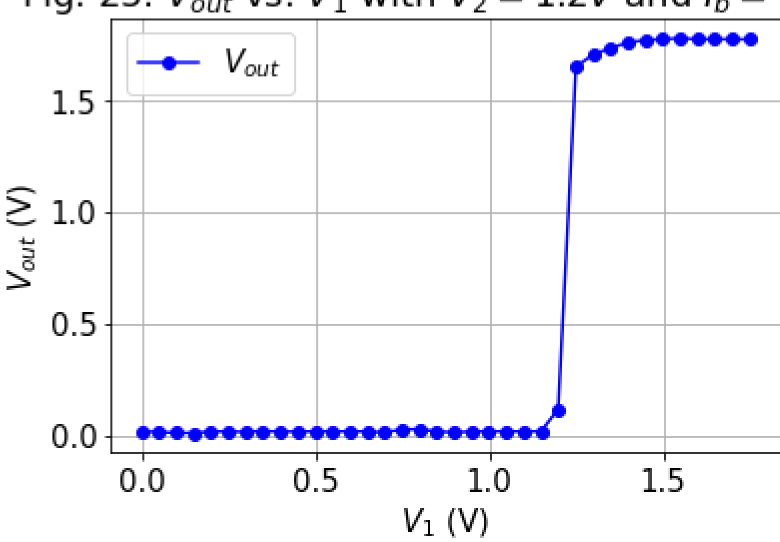
```python
import numpy as np
import time

V1_sweep_ex4 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex4_getset = p.get_set_voltage(pyplane.DacChannel.AIN7)

Vout_V1_sweep_ex4 = []
V1_sweep_ex4_getset = []

for n in range(len(V1_sweep_ex4)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex4.append(p.read_voltage(pyplane.AdcChannel.AOUT12))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [71]:

```python
plt.plot(V1_sweep_ex4_getset, Vout_V1_sweep_ex4, 'bo-', label="$V_{out}$")
plt.title("Fig. 25: $V_{out}$ vs. $V_1$ with $V_2 = 1.2 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 25: V_{out} vs. V_1 with $V_2 = 1.2V$ and $I_b = 10nA$



In [49]:

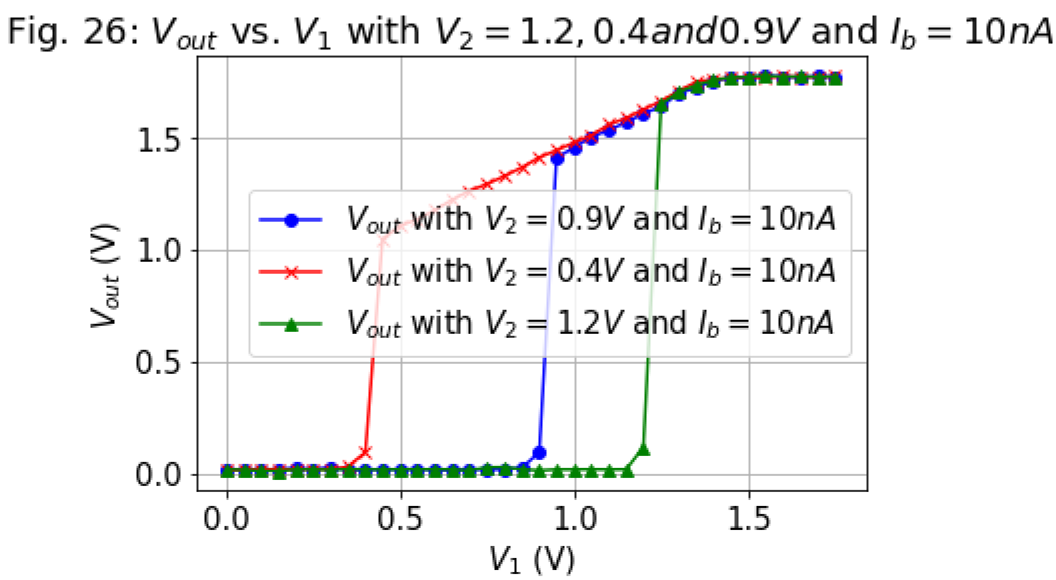
```python
# if the data looks nice, save it!
data_Vout_V1_sweep_ex4 = [V1_sweep_ex4_getset, Vout_V1_sweep_ex4]
# save to csv file
np.savetxt('./data/Vout_sweep_ex4_4.csv', data_Vout_V1_sweep_ex4, delimiter=',')
```

In [70]:

```
[V1_sweep_ex4_getset, Vout_V1_sweep_ex4] = np.loadtxt('./data/Vout_sweep_ex4_4.csv',  deli
```

In [73]:

```
# your codes


[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1] = np.loadtxt('./data/Vout_sweep_ex4_1
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_4] = np.loadtxt('./data/Vout_sweep_ex4_3
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_5] = np.loadtxt('./data/Vout_sweep_ex4_4
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1, 'bo-', label="$V_{out}$ with
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_4, 'rx-', label="$V_{out}$ with
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_5, 'g^-', label="$V_{out}$ with
plt.title("Fig. 26: $V_{out}$ vs. $V_1$ with $V_2 = 1.2, 0.4 and 0.9V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



To conclude your observations:

> In contrast to nFET, this graph stays at nearly GND while $V_1 < V_2$. At $V_1 = V_2$, it rises nearly instantly but takes a while to reach V_{dd}, since at some point it starts rising linearly with a slope of κ. Shifting V_2 translates to shifting the graph. The most ideal amplifier is reached with a high V_2.

4.4 Output current vs. input voltage

4.4.1 Basic measurement

- Set bias current I_b

In [53]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

Switch bias current back to $I_b = 10\text{nA}$.

- Assign common mode voltage V_{cm}

In [54]:
```
Vcm_ex4 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [55]:
```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex3_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN8)

V2_ex4 = []
V1_sweep_ex4_getset = []
V2_ex4_getset = []
c2f_Iout_UO_Vcm_ex4 = []
c2f_Iout_UBO_Vcm_ex4 = []

for n in range(len(V1_sweep_ex4)):

    # calculate V2 via Vcm and V1
    V2_ex4.append(2*Vcm_ex4-V1_sweep_ex4[n])

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
    p.set_voltage(pyplane.DacChannel.AIN8,V2_ex4[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    V2_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN8))

    # read c2f values
    c2f_Iout_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex4.append(c2f_Iout_ex4_temp[13])
    c2f_Iout_UBO_Vcm_ex4.append(c2f_Iout_ex4_temp[14])

# print(V1_sweep_ex3_getset)
# print(V2_ex4_getset)
# print(c2f_Iout_UO_Vcm_ex4)
# print(c2f_Iout_UBO_Vcm_ex4)
```

- Save raw data

In [56]:
```
# if the data looks nice, save it!
data_Iout_Vcm09_ex4 = [V1_sweep_ex4_getset,V2_ex4_getset,c2f_Iout_UO_Vcm_ex4,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex4_1.csv', data_Iout_Vcm09_ex4, delimiter=',')
```

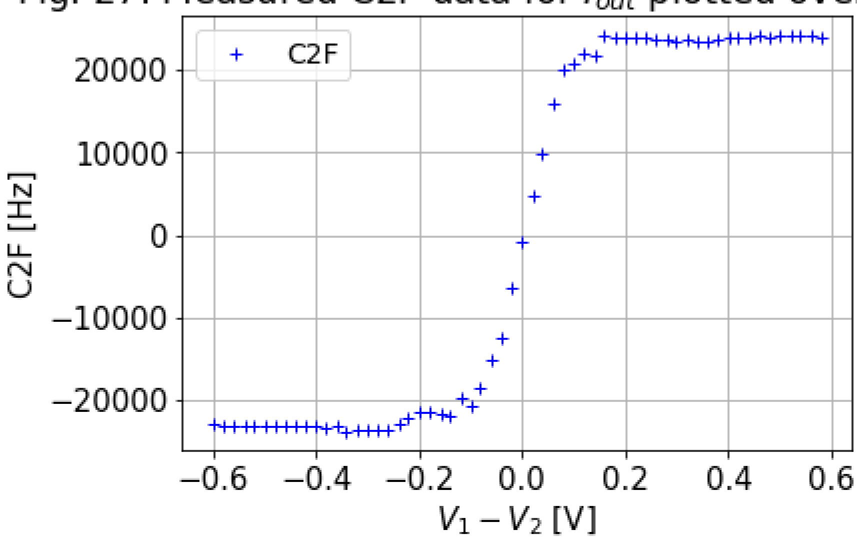
- Plot raw data (C2F rate vs. $Vdiff$)

In [57]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UBC
```

```
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
c2f_Iout_Vcm09 = c2f_Iout_UO_Vcm09_ex4 - c2f_Iout_UBO_Vcm09_ex4
print(c2f_Iout_Vcm09)

plt.plot(Vdiff_Vcm09,c2f_Iout_Vcm09,'b+')

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('C2F [Hz]')
plt.legend(['C2F'],prop={'size': 14})
plt.title('Fig. 27: Measured C2F data for $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-22954. -23044. -23115. -23115. -23087. -23135. -23141. -23109. -23026.
 -23049. -22985. -23318. -23164. -23714. -23676. -23666. -23622. -23520.
 -22825. -22247. -21330. -21294. -21542. -21905. -19719. -20721. -18437.
 -15103. -12343.  -6335.   -709.   4803.   9772.  15884.  20077.  20844.
  21930.  21728.  24156.  23954.  23924.  23808.  23800.  23583.  23564.
  23459.  23531.  23406.  23471.  23664.  23786.  23934.  23870.  24056.
  23794.  24033.  24132.  24080.  24036.  23975.]
```

Fig. 27: Measured C2F data for I_{out} plotted over $V_1 - V_2$.



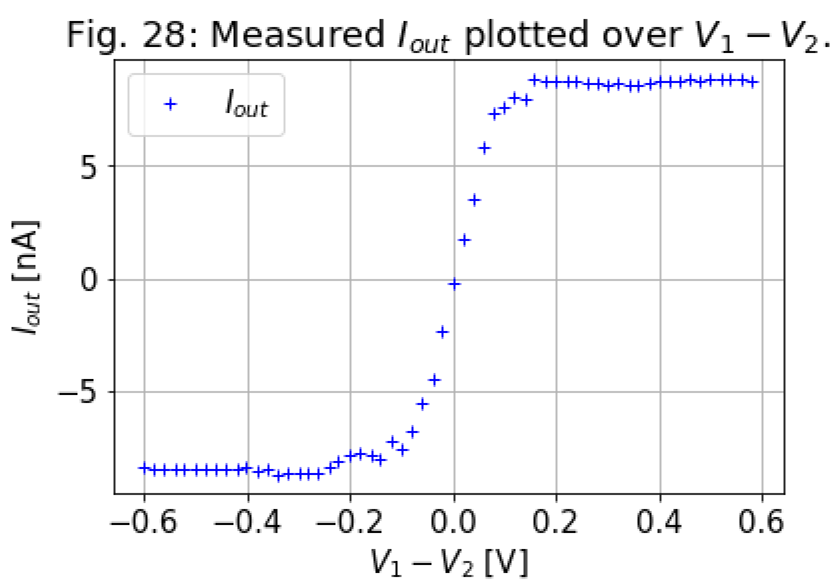
- Convert rate to current and plot

In [24]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
```

```
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 28: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-8.41565393 -8.44908532 -8.47544441 -8.47545389 -8.4650453  -8.48288203
 -8.48510102 -8.47322548 -8.44239122 -8.45094219 -8.42717498 -8.55085713
 -8.49365312 -8.69799778 -8.68387543 -8.68015912 -8.66380785 -8.62590569
 -8.36776429 -8.15323061 -7.81315284 -7.79980894 -7.89174432 -8.0263565
 -7.21653368 -7.58749016 -6.74251694 -5.51293226 -4.49848746 -2.30103051
 -0.25671092  1.74306145  3.55631961  5.80055763  7.34902377  7.63305509
  8.03562929  7.96071241  8.86230645  8.78720522  8.776053    8.73293455
  8.72996106  8.64931535  8.64225512  8.60323113  8.62999297  8.58354949
  8.6076993   8.67940617  8.72475753  8.77977036  8.75598972  8.82512553
  8.72773097  8.81658435  8.85338266  8.83404858  8.81768984  8.79501199]
```



- Compute transconductance

In [ ]:

```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

- Explain any asymmetries in the amplifier's I-V curve and the offset voltage in terms of mismatch between devices in the mirror and differential pair, and the Early effect. How can you distinguish the effects of mismatch in the mirror and in the differential pair? The main point here is to recognize that there will be non-idealities, to understand where they arise, and to quantify them in the simplest manner possible.

> A mismatch in the diff-pair moves the curve away from a symmetry at $V_1 - V_2 = 0$. The reason is that through minuscule imperfections, the two input branches won't have the exact same behavior; one branch will tend to conduct a bit more electricity. This seems to not be a problem in the graph above.
>
> A mismatch in the mirror will cause $|I_{\text{out, min}}| \neq I_{\text{out, max}}$. The reason is that I_1 will not be mirrored correctly to I_2. We can see this a bit in the graph.
>
> The curve is not completely flat at the edge of the sigmoid because of the Early effect.

4.4.2 Different bias currents

- Repeat 4.4.1 with another two bias currents and compare the three curves

The bias current was switched from $I_b \approx 10\text{nA}$ to $I_b \approx 5\text{nA}$.

In [25]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 42)])
```

- Assign common mode voltage V_{cm}

In [26]:

```
Vcm_ex4 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [30]:

```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex4_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN8)

V2_ex4 = []
V1_sweep_ex4_getset = []
V2_ex4_getset = []
c2f_Iout_UO_Vcm_ex4 = []
c2f_Iout_UBO_Vcm_ex4 = []

for n in range(len(V1_sweep_ex4)):

    # calculate V2 via Vcm and V1
    V2_ex4.append(2*Vcm_ex4-V1_sweep_ex4[n])

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
    p.set_voltage(pyplane.DacChannel.AIN8,V2_ex4[n]) #

    time.sleep(0.2) # settle time
```

```
    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    V2_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN8))

    # read c2f values
    c2f_Iout_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex4.append(c2f_Iout_ex4_temp[13])
    c2f_Iout_UBO_Vcm_ex4.append(c2f_Iout_ex4_temp[14])

# print(V1_sweep_ex4_getset)
# print(V2_ex4_getset)
# print(c2f_Iout_UO_Vcm_ex4)
# print(c2f_Iout_UBO_Vcm_ex4)
```

- Save raw data

In [31]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex4 = [V1_sweep_ex4_getset,V2_ex4_getset,c2f_Iout_UO_Vcm_ex4,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex4_2.csv', data_Iout_Vcm09_ex4, delimiter=',')
```

- Convert rate to current and plot

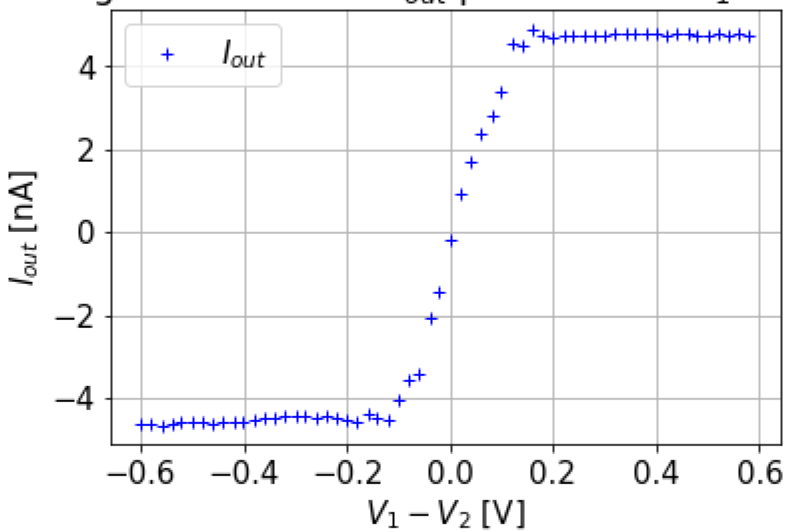
In [32]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 29: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
  0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
  0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
  0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
  0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-4.60529795 -4.62402089 -4.65008794 -4.6111717  -4.60052561 -4.59244949
 -4.57226007 -4.61007037 -4.59281659 -4.57923446 -4.59869011 -4.54326289
 -4.47390028 -4.47206548 -4.43940792 -4.43463798 -4.43537182 -4.46178582
 -4.45261733 -4.47022567 -4.5161033  -4.58730509 -4.39244543 -4.49774843
 -4.54436401 -4.06719459 -3.5819359  -3.39644618 -2.0674551  -1.43135354
 -0.18173935  0.94856537  1.69794156  2.35716934  2.8249091   3.41253973
  4.55721058  4.50619428  4.89396991  4.76502785  4.70075937  4.74960227
```

```
4.7231601   4.7569477   4.76135503  4.76172767  4.77568464  4.79661582
4.78559642  4.79661043  4.79992172  4.74666415  4.78192337  4.79771778
4.75217315  4.74629689  4.77237899  4.75291304  4.78082146  4.76062582]
```

Fig. 29: Measured I_{out} plotted over $V_1 - V_2$.



- Compute transconductance

In [ ]:
```python
kappa = 0.78
U_T = 0.025
I_b = 5e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 7.8e-08
```

The bias current was switched from $I_b \approx 10\mathrm{nA}$ to $I_b \approx 15\mathrm{nA}$.

In [33]:
```python
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 122)])
```

- Assign common mode voltage V_{cm}

In [34]:
```python
Vcm_ex4 = 0.9 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [35]:
```python
import numpy as np
import time

V1_sweep_ex4 = np.arange(0.6, 1.2, 0.01) # voltage V1 sweep range

#V2_ex4_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN8)

V2_ex4 = []
V1_sweep_ex4_getset = []
V2_ex4_getset = []
c2f_Iout_UO_Vcm_ex4 = []
c2f_Iout_UBO_Vcm_ex4 = []
```

```
for n in range(len(V1_sweep_ex4)):

    # calculate V2 via Vcm and V1
    V2_ex4.append(2*Vcm_ex4-V1_sweep_ex4[n])

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
    p.set_voltage(pyplane.DacChannel.AIN8,V2_ex4[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    V2_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN8))

    # read c2f values
    c2f_Iout_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex4.append(c2f_Iout_ex4_temp[13])
    c2f_Iout_UBO_Vcm_ex4.append(c2f_Iout_ex4_temp[14])

# print(V1_sweep_ex4_getset)
# print(V2_ex4_getset)
# print(c2f_Iout_UO_Vcm_ex4)
# print(c2f_Iout_UBO_Vcm_ex4)
```

- Save raw data

In [36]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex4 = [V1_sweep_ex4_getset,V2_ex4_getset,c2f_Iout_UO_Vcm_ex4,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex4_3.csv', data_Iout_Vcm09_ex4, delimiter=',')
```

- Convert rate to current and plot

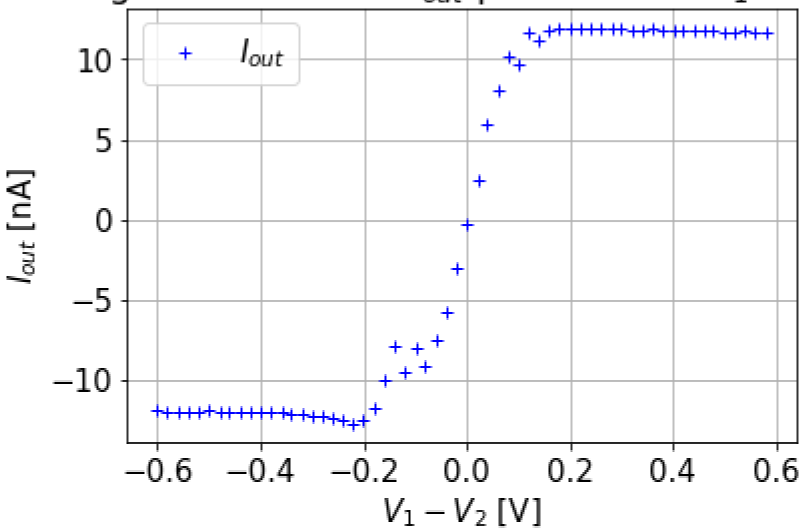
In [37]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 30: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.60000008 -0.58064526 -0.55953091 -0.53841656 -0.52082115 -0.49970675
 -0.4785924  -0.4609971  -0.43988276 -0.41876841 -0.40117306 -0.38005871
 -0.35894436 -0.34134907 -0.32023472 -0.29912025 -0.2815249  -0.26041055
 -0.2392962  -0.22170091 -0.20058656 -0.17947215 -0.1583578  -0.14076245
 -0.1196481  -0.09853375 -0.08093846 -0.05982405 -0.0387097  -0.02111435
  0.          0.02111435  0.0387097   0.05982405  0.08093846  0.09853375
```

```
 0.1196481   0.14076245  0.1583578   0.17947215  0.20058656  0.22170091
 0.2392962   0.26041055  0.2815249   0.29912025  0.32023472  0.34134907
 0.35894436  0.38005871  0.40117306  0.41876841  0.43988276  0.4609971
 0.4785924   0.49970675  0.52082115  0.53841656  0.55953091  0.58064526]
[-11.93522337 -11.98888152 -11.98212689 -11.98512893 -12.00276647
 -11.97237043 -12.00689453 -12.02941212 -12.05868719 -12.06319126
 -12.09209384 -12.06206524 -12.04630128 -12.1240021  -12.17806521
 -12.24715821 -12.33166581 -12.424836   -12.58869635 -12.72856542
 -12.51352138 -11.728924   -10.07813115  -7.89137357  -9.55796151
  -7.99149306  -9.1230496   -7.55822814  -5.80239982  -3.07729612
  -0.28641888   2.39910219   5.90263079   8.01077877  10.17443333
   9.70777803  11.64718875  11.18713583  11.76192357  11.95435923
  11.90370781  11.91121125  11.87519622  11.87669677  11.89132749
  11.89958098  11.87219514  11.85981595  11.87896082  11.81292914
  11.74879836  11.7802997   11.74804836  11.7652987   11.78181295
  11.72517425  11.71842481  11.75779845  11.71542509  11.7446734 ]
```

Fig. 30: Measured I_{out} plotted over $V_1 - V_2$.



- Compute transconductance

In [38]:

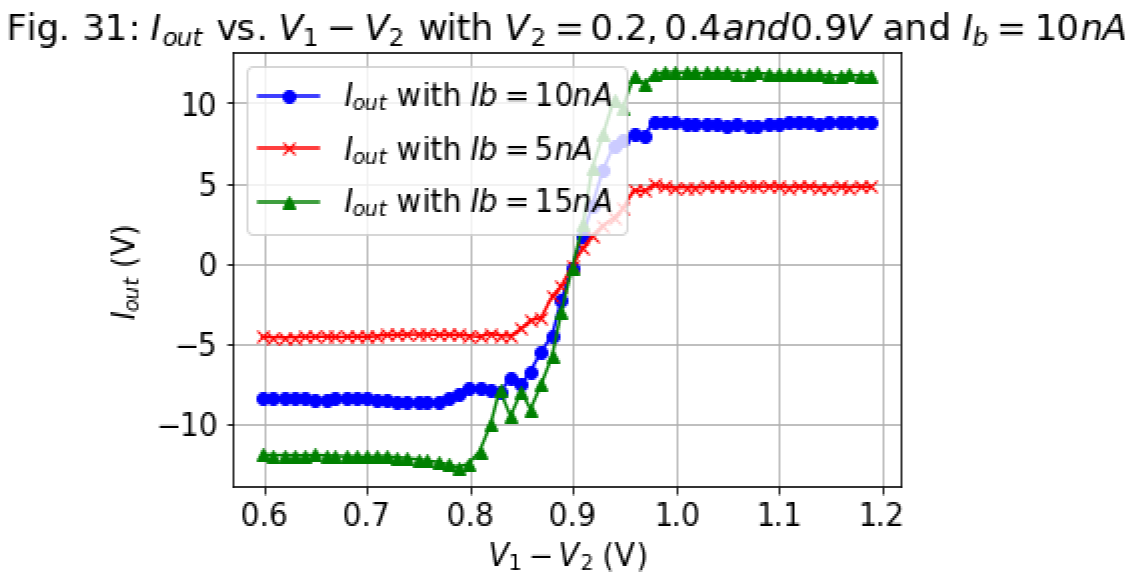
```
kappa = 0.78
U_T = 0.025
I_b = 15e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 2.3399999999999998e-07
```

In [21]:

```
# your codes

import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset_1,V2_Iout_Vcm09_ex4_getset_1,c2f_Iout_UO_Vcm09_ex4_1,c2f_Io
V1_sweep_Iout_Vcm09_ex4_getset_2,V2_Iout_Vcm09_ex4_getset_2,c2f_Iout_UO_Vcm09_ex4_2,c2f_Io
V1_sweep_Iout_Vcm09_ex4_getset_3,V2_Iout_Vcm09_ex4_getset_3,c2f_Iout_UO_Vcm09_ex4_3,c2f_Io

I_plus_1 = a2_1*c2f_Iout_UO_Vcm09_ex4_1**2 + a1_1*c2f_Iout_UO_Vcm09_ex4_1 + a0_1
I_minus_1 = a2_2*c2f_Iout_UBO_Vcm09_ex4_1**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4_1 + a0_2

I_plus_2 = a2_1*c2f_Iout_UO_Vcm09_ex4_2**2 + a1_1*c2f_Iout_UO_Vcm09_ex4_2 + a0_1
I_minus_2 = a2_2*c2f_Iout_UBO_Vcm09_ex4_2**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4_2 + a0_2

I_plus_3 = a2_1*c2f_Iout_UO_Vcm09_ex4_3**2 + a1_1*c2f_Iout_UO_Vcm09_ex4_3 + a0_1
I_minus_3 = a2_2*c2f_Iout_UBO_Vcm09_ex4_3**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4_3 + a0_2
```

```
plt.plot(V1_sweep_Iout_Vcm09_ex4_getset_1, I_plus_1 - I_minus_1, 'bo-', label="$I_{out}$ w
plt.plot(V1_sweep_Iout_Vcm09_ex4_getset_2, I_plus_2 - I_minus_2, 'rx-', label="$I_{out}$ w
plt.plot(V1_sweep_Iout_Vcm09_ex4_getset_3, I_plus_3 - I_minus_3, 'g^-', label="$I_{out}$ w
plt.title("Fig. 31: $I_{out}$ vs. $V_1-V_2$ with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10 n
plt.xlabel("$V_1-V_2$ (V)")
plt.ylabel("$I_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



To conclude your observations:

> The range of the sigmoid is proportional to the bias current. The slope is just as proportional, resolting in $I_{\text{out, min}}$ and asI_{\text{out, max}} $alwaysbeingreachedatapproximatelythesame$\varDelta V. $Interestingly,$I_b = 15nA$ generated a lot of noise, as well as rising earlier than the other graphs.

4.4.3 Different common mode voltages

- Repeat 4.4.1 with another two common mode voltages and compare the three curves

In [13]:

```
# Assign common mode voltage Vcm
Vcm_ex4 = 0.45
```

The bias current was switched back to $I_b \approx 10\text{nA}$.

In [14]:

```
 p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [15]:

```
import numpy as np
import time
```

```
V1_sweep_ex4 = np.arange(0.2, 0.7, 0.01) # voltage V1 sweep range

#V2_ex4_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN8)

V2_ex4 = []
V1_sweep_ex4_getset = []
V2_ex4_getset = []
c2f_Iout_UO_Vcm_ex4 = []
c2f_Iout_UBO_Vcm_ex4 = []

for n in range(len(V1_sweep_ex4)):


    # calculate V2 via Vcm and V1
    V2_ex4.append(2*Vcm_ex4-V1_sweep_ex4[n])

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
    p.set_voltage(pyplane.DacChannel.AIN8,V2_ex4[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    V2_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN8))

    # read c2f values
    c2f_Iout_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex4.append(c2f_Iout_ex4_temp[13])
    c2f_Iout_UBO_Vcm_ex4.append(c2f_Iout_ex4_temp[14])

# print(V1_sweep_ex4_getset)
# print(V2_ex4_getset)
# print(c2f_Iout_UO_Vcm_ex4)
# print(c2f_Iout_UBO_Vcm_ex4)
```

- Save raw data

In [16]:

```
# if the data looks nice, save it!
data_Iout_Vcm09_ex4 = [V1_sweep_ex4_getset,V2_ex4_getset,c2f_Iout_UO_Vcm_ex4,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex4_4.csv', data_Iout_Vcm09_ex4, delimiter=',')
```

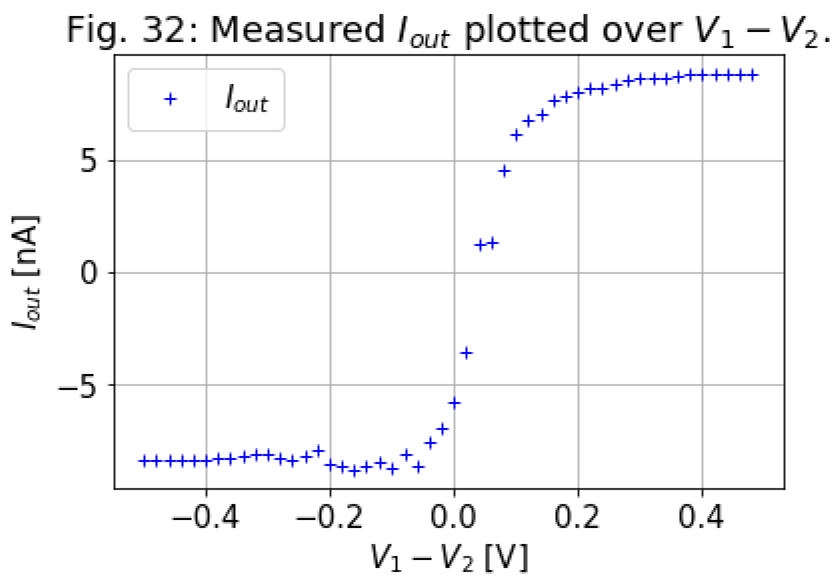
- Convert rate to current and plot

In [29]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)

plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
```

```
plt.title('Fig. 32: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.49970683 -0.48035194 -0.45923758 -0.43988276 -0.42052792 -0.39941356
 -0.38005868 -0.36070383 -0.33958948 -0.32023466 -0.29912028 -0.27976546
 -0.26041058 -0.2392962  -0.21994138 -0.20058653 -0.17947218 -0.16011736
 -0.14076245 -0.1196481  -0.10029328 -0.0791789  -0.05982405 -0.04046923
 -0.01935485  0.          0.01935485  0.04046923  0.05982405  0.0791789
  0.10029328  0.1196481   0.14076245  0.16011736  0.17947218  0.20058653
  0.21994138  0.2392962   0.26041058  0.27976546  0.29912028  0.32023466
  0.33958948  0.36070383  0.38005868  0.39941356  0.42052792  0.43988276
  0.45923758  0.48035194]
[-8.34029039 -8.36478467 -8.35848232 -8.36218572 -8.31801591 -8.30167294
 -8.27978149 -8.23374826 -8.15879636 -8.07308498 -8.08941658 -8.27866793
 -8.31987206 -8.156199   -7.883217   -8.52039664 -8.56757442 -8.73925329
 -8.64671419 -8.40452341 -8.68276054 -8.05899788 -8.64708578 -7.5297094
 -6.87740629 -5.76264299 -3.51027571  1.22095147  1.30365296  4.55585821
  6.17738368  6.76694557  7.0316531   7.61842853  7.83329619  8.02348293
  8.1532262   8.21987769  8.36535717  8.54584152  8.59556111  8.60471959
  8.65268855  8.73519986  8.7600111   8.7708705   8.77868668  8.81495456
  8.79839901  8.80351537]
```



- Compute transconductance

In [ ]:

```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

In [18]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.PTA_VB_P, \
    pyplane.Coach.BiasType.P, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Assign common mode voltage V_{cm}

In [19]:

```
Vcm_ex4 = 0.2 # Vcm = 0.9V
```

- Sweep differential voltage V_{diff} and measure I_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [20]:
```
import numpy as np
import time

V1_sweep_ex4 = np.arange(0.1, 0.3, 0.01) # voltage V1 sweep range

#V2_ex4_getset = p.get_set_voltage(pyplane.VoltageChannel.AIN8)

V2_ex4 = []
V1_sweep_ex4_getset = []
V2_ex4_getset = []
c2f_Iout_UO_Vcm_ex4 = []
c2f_Iout_UBO_Vcm_ex4 = []

for n in range(len(V1_sweep_ex4)):

    # calculate V2 via Vcm and V1
    V2_ex4.append(2*Vcm_ex4-V1_sweep_ex4[n])

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex4[n]) #
    p.set_voltage(pyplane.DacChannel.AIN8,V2_ex4[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    V2_ex4_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN8))

    # read c2f values
    c2f_Iout_ex4_temp = p.read_c2f_output(0.1)
    c2f_Iout_UO_Vcm_ex4.append(c2f_Iout_ex4_temp[13])
    c2f_Iout_UBO_Vcm_ex4.append(c2f_Iout_ex4_temp[14])

# print(V1_sweep_ex4_getset)
# print(V2_ex4_getset)
# print(c2f_Iout_UO_Vcm_ex4)
# print(c2f_Iout_UBO_Vcm_ex4)
```

- Save raw data

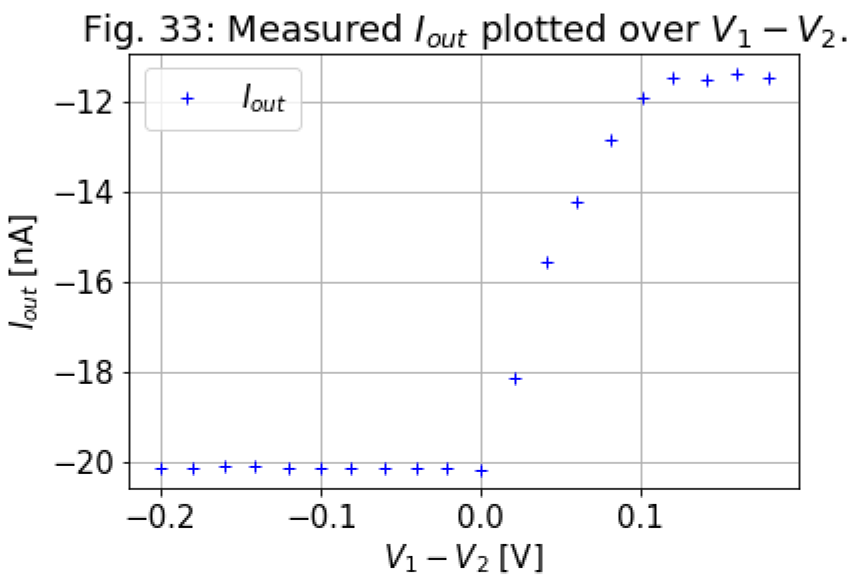
In [21]:
```
# if the data looks nice, save it!
data_Iout_Vcm09_ex4 = [V1_sweep_ex4_getset,V2_ex4_getset,c2f_Iout_UO_Vcm_ex4,c2f_Iout_UBO_
# save to csv file
np.savetxt('./data/V1_sweep_Iout_Vcm09_ex4_5.csv', data_Iout_Vcm09_ex4, delimiter=',')
```

- Convert rate to current and plot

In [30]:
```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout_UB(

I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
print(Vdiff_Vcm09)
I_out = I_plus - I_minus
print(I_out)
```

```
plt.plot(Vdiff_Vcm09,I_out,'b+', label="$I_{out}$")

plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')
plt.legend()
plt.title('Fig. 33: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```

```
[-0.20058654 -0.17947215 -0.16011732 -0.14076249 -0.1196481  -0.10029326
 -0.08093843 -0.05982405 -0.04046921 -0.02111436  0.          0.02111436
  0.04046921  0.05982405  0.08093843  0.10029326  0.1196481   0.14076249
  0.16011732  0.17947215]
[-20.12717032 -20.14368482 -20.09491163 -20.09222353 -20.11987346
 -20.12870652 -20.13869198 -20.11257674 -20.14980779 -20.11104061
 -20.15070882 -18.12195847 -15.56102576 -14.22136788 -12.85304043
 -11.93654203 -11.49079446 -11.5241492  -11.37596959 -11.45904337]
```



- Compute transconductance

In [ ]:

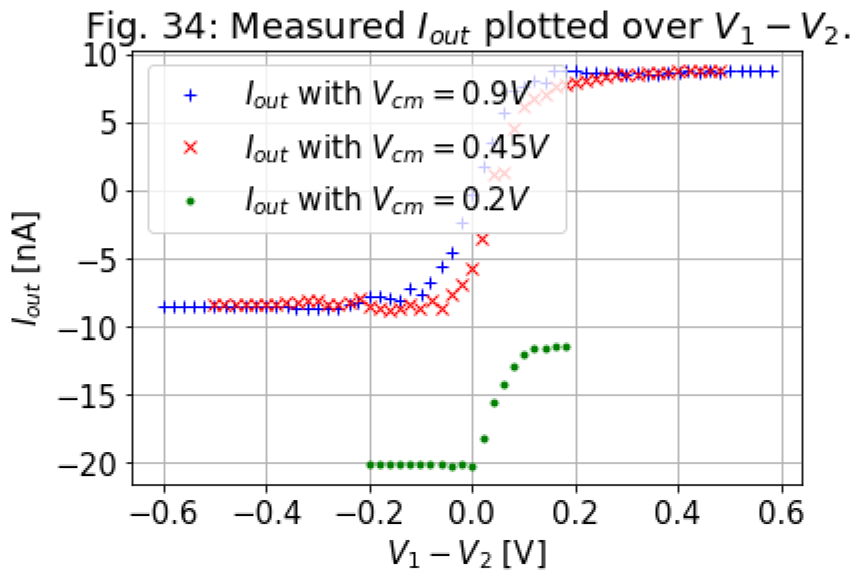
```
kappa = 0.78
U_T = 0.025
I_b = 10e-9
g_m = I_b*kappa/(2*U_T)
print("The transconductance g_m is:", g_m)
```

```
The transconductance g_m is: 1.56e-07
```

In [36]:

```
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams.update({'font.size': 15})

files = ['./data/V1_sweep_Iout_Vcm09_ex4_1.csv', './data/V1_sweep_Iout_Vcm09_ex4_4.csv', '
styles = ['b+','rx', 'g.']
v_cms = [0.9, 0.45, 0.2]
for (file, style), v_cm in zip(zip(files, styles), v_cms):
    V1_sweep_Iout_Vcm09_ex4_getset,V2_Iout_Vcm09_ex4_getset,c2f_Iout_UO_Vcm09_ex4,c2f_Iout

    I_plus = a2_1*c2f_Iout_UO_Vcm09_ex4**2 + a1_1*c2f_Iout_UO_Vcm09_ex4 + a0_1
    I_minus = a2_2*c2f_Iout_UBO_Vcm09_ex4**2 + a1_2*c2f_Iout_UBO_Vcm09_ex4 + a0_2
    Vdiff_Vcm09 = V1_sweep_Iout_Vcm09_ex4_getset-V2_Iout_Vcm09_ex4_getset
    I_out = I_plus - I_minus

    plt.plot(Vdiff_Vcm09,I_out, style, label=f"$I_{{out}}$ with $V_{{cm}} = {v_cm}V$")
```

```
plt.xlabel('$V_1-V_2$ [V]')
plt.ylabel('$I_{out}$ [nA]')

plt.legend()
plt.title('Fig. 34: Measured $I_{out}$ plotted over $V_1-V_2$.')
plt.grid()
plt.show()
```



To conclude your observations:

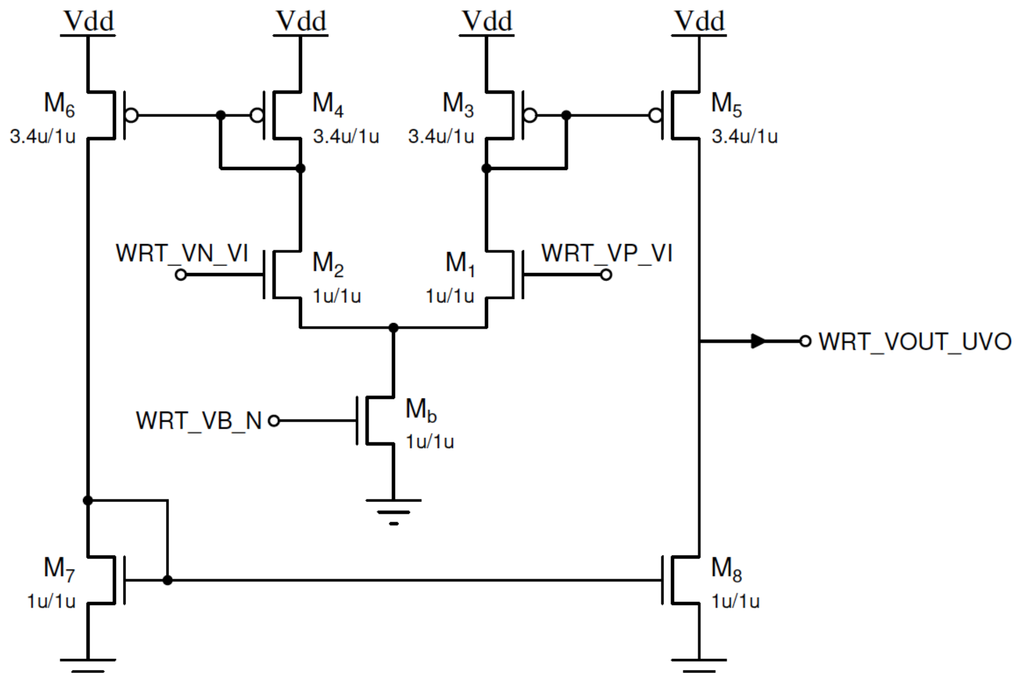
> Increasing V_{cm} has the same effect as increasing I_b, but a lower V_{cm} results in a lower range of available ΔV

What do you observe when the common mode voltage V_{cm} is too small (e.g. 0.2V or 0.3V)? Does it have a sigmoid shape? If not, try to explain why.

> The sigmoid is cut off when $V_2 > V_1$. M_2 of the schematic above is no longer in saturation, its requirement is $V_{out} < \max(V_1, V_2) - 4U_T$. With $V_{cm} = 0.2V$, $\max(V_1, V_2) \leq 0.3V$ is too low. This explains why the right side of the graph looks normal: When all current flows through M_1 and none through M_2, it doesn't matter that M_2 is not in saturation.

5 Wide-range Transamp

5.0 Schematic and pin map



V_1 = V_p = **WRT_VP_VI = AIN7**

V_2 = V_n = **WRT_VN_VI = AIN8**

V_{out} = **WRT_VOUT_UVO = ADC[11]**

5.1 Chip configuration

In [32]:
```
p.send_coach_events([pyplane.Coach.generate_aerc_event( \
    pyplane.Coach.CurrentOutputSelect.SelectLine5, \
    pyplane.Coach.VoltageOutputSelect.SelectLine1, \
    pyplane.Coach.VoltageInputSelect.SelectLine2, \
    pyplane.Coach.SynapseSelect.NoneSelected, 0)])
```

5.2 Output voltage vs. input voltage

5.2.1 Basic measurement

- Set bias current I_b

In [33]:
```
 p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.WRT_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

The bias current is set to

$$I_b = w\frac{BG_{\text{fine}}}{256} I_{BG_{\text{master}}} = \frac{85}{256} \cdot 30\text{nA} \approx 9.961\text{nA}.$$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [34]:
```
p.set_voltage(pyplane.DacChannel.AIN8,0.9) # V2 = 0.9
```

Out[34]: 0.8991203308105469

The input voltage is set to $V_2 = 0.9\mathrm{V}$.

- Sweep V_1 and measure V_{out} (Hint: use get_set_voltage to get the real value set on the DAC)

In [35]:

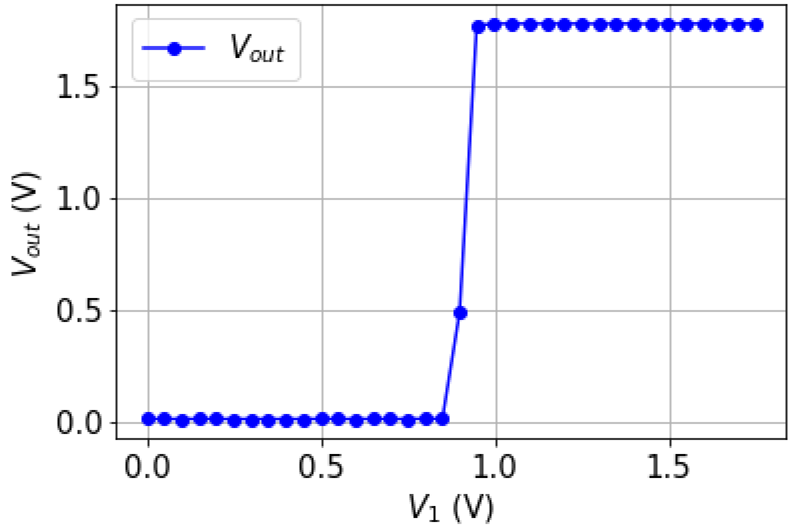
```
import numpy as np
import time

V1_sweep_ex5 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex5_getset = p.get_set_voltage(pyplane.DacChannel.AIN8)

Vout_V1_sweep_ex5 = []
V1_sweep_ex5_getset = []

for n in range(len(V1_sweep_ex5)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex5[n]) #

    time.sleep(0.2) # settle time

    V1_sweep_ex5_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
    Vout_V1_sweep_ex5.append(p.read_voltage(pyplane.AdcChannel.AOUT11))

print(V2_ex5_getset)
print(V1_sweep_ex5_getset)
print(Vout_V1_sweep_ex5)
```

```
0.8991203308105469
[0.0, 0.04926686733961105, 0.0985337346792221, 0.1495601385831833, 0.19882699847221375, 0.
24985340237617493, 0.2991202771663666, 0.3483871519565582, 0.399413526058197, 0.4486804008
4838867, 0.49970680475234985, 0.5489736795425415, 0.5982405543327332, 0.6492669582366943,
0.698533833026886, 0.7495601773262024, 0.798827052116394, 0.8498534560203552, 0.8991203308
105469, 0.9483872056007385, 0.9994136095046997, 1.0486804246902466, 1.0997068881988525, 1.
1489737033843994, 1.1982406377792358, 1.2492669820785522, 1.2985339164733887, 1.3495602607
72705, 1.3988271951675415, 1.449853539466858, 1.4991203546524048, 1.5483872890472412, 1.59
94136333465576, 1.648680567741394, 1.6997069120407104, 1.7489738464355469]
[0.01369628868997097, 0.01369628868997097, 0.012890624813735485, 0.01369628868997097, 0.01
369628868997097, 0.012890624813735485, 0.012890624813735485, 0.012890624813735485, 0.01289
0624813735485, 0.012890624813735485, 0.01369628868997097, 0.01369628868997097, 0.012890624
813735485, 0.01369628868997097, 0.01369628868997097, 0.012890624813735485, 0.0145019534975
2903, 0.01369628868997097, 0.4890380799770355, 1.7724609375, 1.780517578125, 1.78051757812
5, 1.780517578125, 1.780517578125, 1.780517578125, 1.780517578125, 1.7797119617462158, 1.7
80517578125, 1.780517578125, 1.778906226158142, 1.780517578125, 1.780517578125, 1.78051757
8125, 1.780517578125, 1.780517578125, 1.780517578125]
```

- Plot raw data

In [50]:

```
plt.plot(V1_sweep_ex5_getset, Vout_V1_sweep_ex5, 'bo-', label="$V_{out}$")
plt.title("Fig. 35: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [39]:
```
# if the data looks nice, save it!
np.savetxt('./data/Vout_sweep_ex5_1.csv',[V1_sweep_ex5,V1_sweep_ex5_getset,Vout_V1_sweep_e
```

In [49]:
```
[V1_sweep_ex5,V1_sweep_ex5_getset,Vout_V1_sweep_ex5] = np.loadtxt('./data/Vout_sweep_ex5_1
```

5.2.2 Different bias currents

- Repeat 5.2.1 with another two bias currents and compare the three curves

The bias current was switched from $I_b \approx 5\text{nA}$ to $I_b \approx 10\text{nA}$.

In [41]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.WRT_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [42]:
```
p.set_voltage(pyplane.DacChannel.AIN8, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

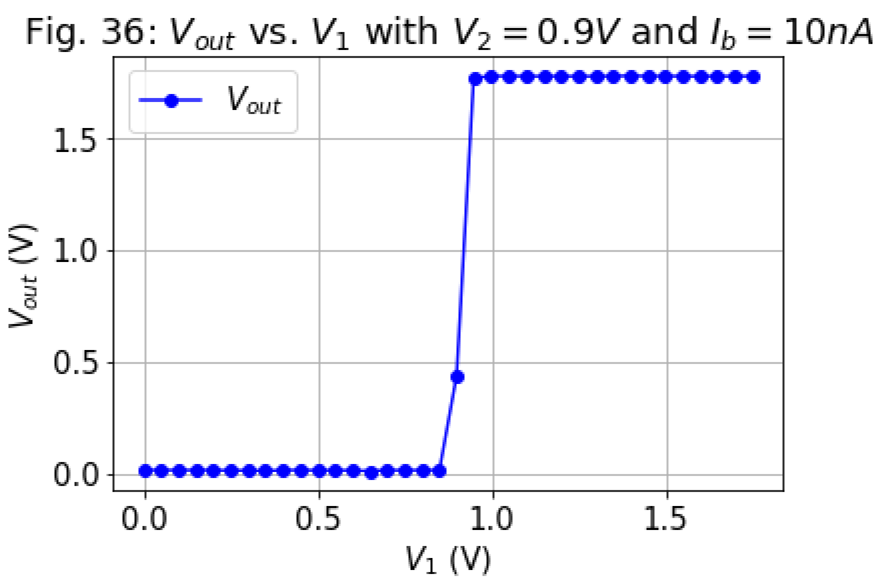
In [43]:
```
import numpy as np
import time

V1_sweep_ex5 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex5_getset = p.get_set_voltage(pyplane.DacChannel.AIN8)

Vout_V1_sweep_ex5 = []
V1_sweep_ex5_getset = []

for n in range(len(V1_sweep_ex5)):
```

```
    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex5[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex5_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex5.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex5.append(p.read_voltage(pyplane.AdcChannel.AOUT11))

# print(V2_ex5_getset)
# print(V1_sweep_ex5_getset)
# print(Vout_V1_sweep_ex5)
```

- Plot raw data

In [44]:

```
plt.plot(V1_sweep_ex5_getset, Vout_V1_sweep_ex5, 'bo-', label="$V_{out}$")
plt.title("Fig. 36: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



- Save raw data

In [45]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex5 = [V1_sweep_ex5_getset, Vout_V1_sweep_ex5]
# save to csv file
np.savetxt('./data/Vout_sweep_ex5_2.csv', data_Vout_V1_sweep_ex5, delimiter=',')
```

In [46]:

```
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save] = np.loadtxt('./data/Vout_sweep_ex5_2.c
```

The bias current was switched from $I_b \approx 5\mathrm{nA}$ to $I_b \approx 15\mathrm{nA}$.

In [47]:

```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.NTA_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 122)])
```

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

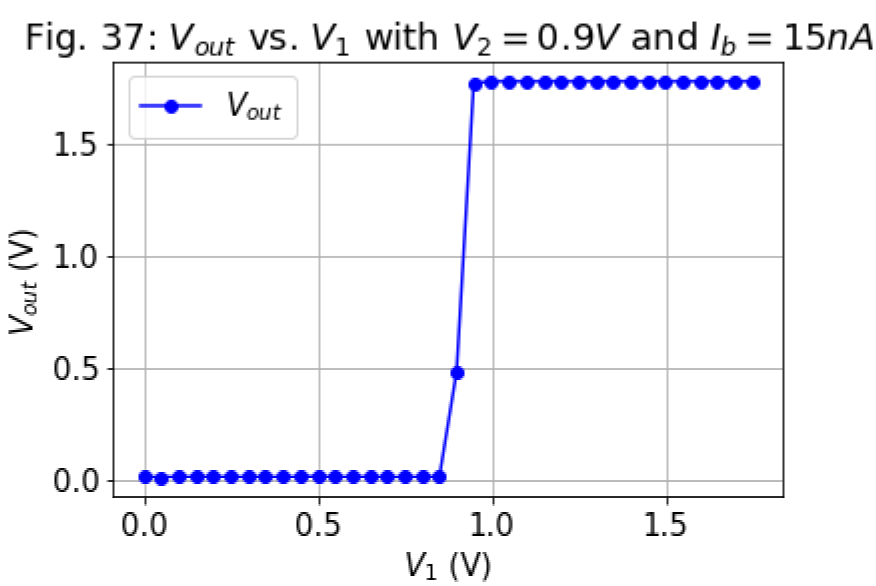
In [48]:
```
p.set_voltage(pyplane.DacChannel.AIN8, 0.9) # V2 = 0.9?
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.8991203308105469 V
```

In [49]:
```
import numpy as np
import time

V1_sweep_ex5 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex5_getset = p.get_set_voltage(pyplane.DacChannel.AIN8)

Vout_V1_sweep_ex5 = []
V1_sweep_ex5_getset = []

for n in range(len(V1_sweep_ex5)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex5[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex5_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex5.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex5.append(p.read_voltage(pyplane.AdcChannel.AOUT11))

# print(V2_ex5_getset)
# print(V1_sweep_ex5_getset)
# print(Vout_V1_sweep_ex5)
```

- Plot raw data

In [50]:
```
plt.plot(V1_sweep_ex5_getset, Vout_V1_sweep_ex5, 'bo-', label="$V_{out}$")
plt.title("Fig. 37: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 15 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

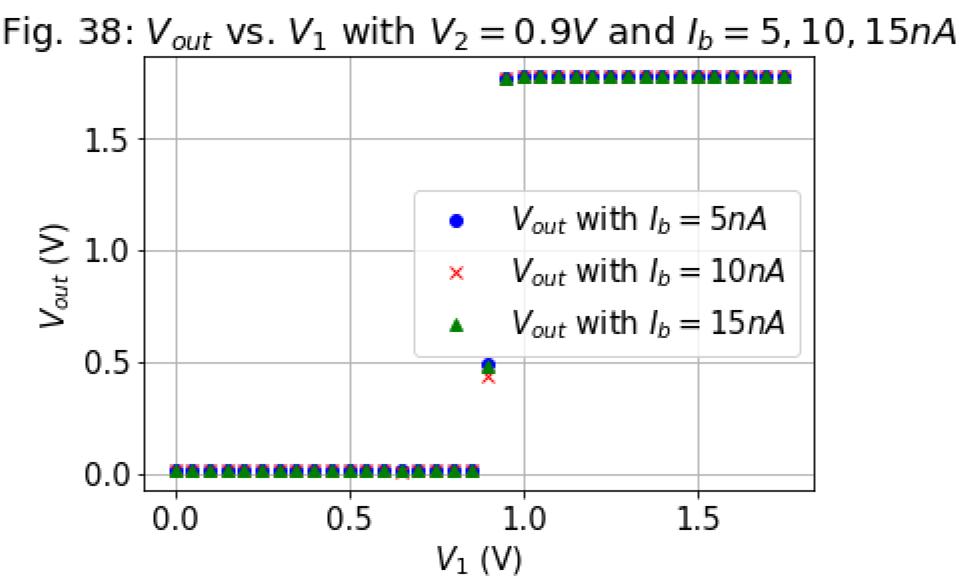


- Save raw data

In [51]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex5 = [V1_sweep_ex5_getset, Vout_V1_sweep_ex5]
# save to csv file
np.savetxt('./data/Vout_sweep_ex5_3.csv', data_Vout_V1_sweep_ex5, delimiter=',')
```

In [52]:
```
[V1_sweep_ex5_getset, Vout_V1_sweep_ex5] = np.loadtxt('./data/Vout_sweep_ex5_3.csv',  deli
```

In [55]:
```
# your codes

[V1_sweep_ex5_save,V1_sweep_ex5_getset,Vout_V1_sweep_ex5_save_1] = np.loadtxt('./data/Vout
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_2] = np.loadtxt('./data/Vout_sweep_ex5_2
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_3] = np.loadtxt('./data/Vout_sweep_ex5_3
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_1, 'bo', label="$V_{out}$ with $
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_2, 'rx', label="$V_{out}$ with $
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_3, 'g^', label="$V_{out}$ with $
plt.title("Fig. 38: $V_{out}$ vs. $V_1$ with $V_2 = 0.9 V$ and $I_b = 5, 10, 15 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



To conclude your observations:

> The bias current does not affect the output voltage when one input voltage is fixed. $\frac{dV_{out}}{dI_b} = 0$ This is the same behavior as in the regular transamp.

5.2.3 Different fixed voltages V_n

- Repeat 5.2.1 with another two fixed voltages V_2 and compare the three curves

Switch voltage from $V_2 = 0.9\text{V}$ to $V_2 = 0.4\text{V}$. The bias current was $I_b = 10\text{nA}$

In [56]:
```
# Set V2 = 04
p.set_voltage(pyplane.DacChannel.AIN8, 0.4) # V2 = 0.4
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 0.399413526058197 V
```

The bias current is set to $I_b \approx 10\mathrm{nA}$.

In [57]:
```
p.send_coach_events([pyplane.Coach.generate_biasgen_event(\
    pyplane.Coach.BiasAddress.WRT_VB_N, \
    pyplane.Coach.BiasType.N, \
    pyplane.Coach.BiasGenMasterCurrent.I30nA, 85)])
```

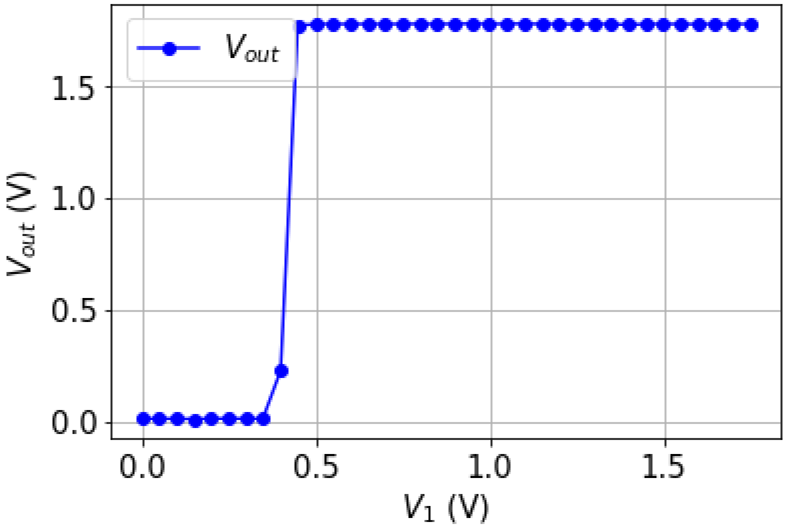
- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [58]:
```
import numpy as np
import time

V1_sweep_ex5 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex5_getset = p.get_set_voltage(pyplane.DacChannel.AIN8)

Vout_V1_sweep_ex5 = []
V1_sweep_ex5_getset = []

for n in range(len(V1_sweep_ex5)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex5[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex5_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#     Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex5.append(p.read_voltage(pyplane.AdcChannel.AOUT11))

# print(V2_ex3_getset)
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [59]:
```
plt.plot(V1_sweep_ex5_getset, Vout_V1_sweep_ex5, 'bo-', label="$V_{out}$")
plt.title("Fig. 39: $V_{out}$ vs. $V_1$ with $V_2 = 0.4 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 39: V_{out} vs. V_1 with $V_2 = 0.4V$ and $I_b = 10nA$



- Save raw data

In [60]:
```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex5 = [V1_sweep_ex5_getset, Vout_V1_sweep_ex5]
# save to csv file
np.savetxt('./data/Vout_sweep_ex5_3.csv', data_Vout_V1_sweep_ex5, delimiter=',')
```

In [61]:
```
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save] = np.loadtxt('./data/Vout_sweep_ex5_3.c
```

Switch voltage from $V_2 = 0.9\mathrm{V}$ to $V_2 = 1.2\mathrm{V}$. The bias current was $I_b = 10\mathrm{nA}$

- Set fixed value of V_2 (Hint: use get_set_voltage to get the real value set on the DAC)

In [62]:
```
p.set_voltage(pyplane.DacChannel.AIN8, 1.2) # V2 = 0.2
v2_real = p.get_set_voltage(pyplane.DacChannel.AIN8)
print("V2 is set to {} V".format(v2_real))
```

```
V2 is set to 1.1982406377792358 V
```

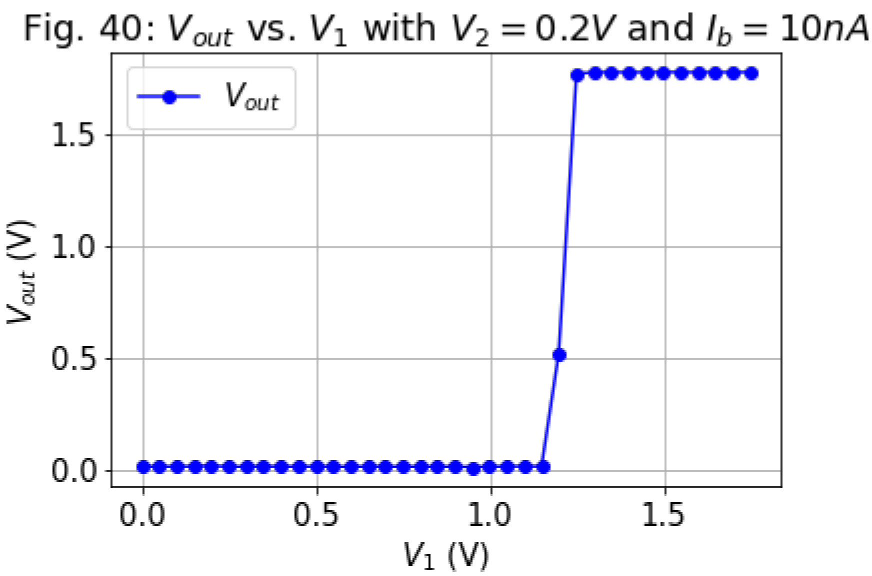
In [63]:
```
import numpy as np
import time

V1_sweep_ex5 = np.arange(0,1.8,0.05) # voltage V1 sweep range

V2_ex5_getset = p.get_set_voltage(pyplane.DacChannel.AIN7)

Vout_V1_sweep_ex5 = []
V1_sweep_ex5_getset = []

for n in range(len(V1_sweep_ex5)):

    p.set_voltage(pyplane.DacChannel.AIN7,V1_sweep_ex5[n]) #

    time.sleep(0.3) # settle time

    V1_sweep_ex5_getset.append(p.get_set_voltage(pyplane.DacChannel.AIN7))
#    Vout_V1_sweep_ex3.append(p.read_adc_instantaneous(13))
    Vout_V1_sweep_ex5.append(p.read_voltage(pyplane.AdcChannel.AOUT11))

# print(V2_ex3_getset)
```

```
# print(V1_sweep_ex3_getset)
# print(Vout_V1_sweep_ex3)
```

- Plot raw data

In [64]:

```
plt.plot(V1_sweep_ex5_getset, Vout_V1_sweep_ex5, 'bo-', label="$V_{out}$")
plt.title("Fig. 40: $V_{out}$ vs. $V_1$ with $V_2 = 0.2 V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



In [65]:

```
# if the data looks nice, save it!
data_Vout_V1_sweep_ex5 = [V1_sweep_ex5_getset, Vout_V1_sweep_ex5]
# save to csv file
np.savetxt('./data/Vout_sweep_ex5_4.csv', data_Vout_V1_sweep_ex5, delimiter=',')
```

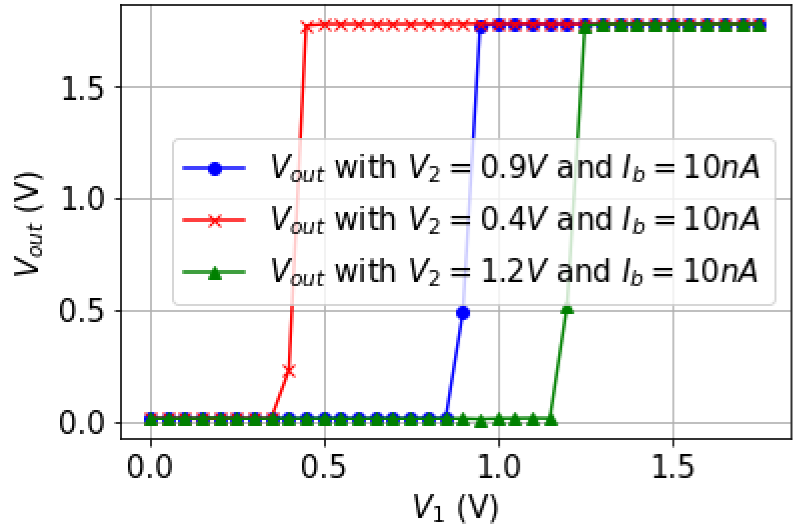
In [66]:

```
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save] = np.loadtxt('./data/Vout_sweep_ex5_4.c
```

In [74]:

```
# your codes


[V1_sweep_ex5_save,V1_sweep_ex5_getset_save,Vout_V1_sweep_ex5_save_1] = np.loadtxt('./data
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_4] = np.loadtxt('./data/Vout_sweep_ex5_3
[V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_5] = np.loadtxt('./data/Vout_sweep_ex5_4
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_1, 'bo-', label="$V_{out}$ with
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_4, 'rx-', label="$V_{out}$ with
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_5, 'g^-', label="$V_{out}$ with
plt.title("Fig. 41: $V_{out}$ vs. $V_1$ with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10 nA$")
plt.xlabel("$V_1$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 41: V_{out} vs. V_1 with $V_2 = 0.2, 0.4 and 0.9V$ and $I_b = 10nA$



To conclude your observations:

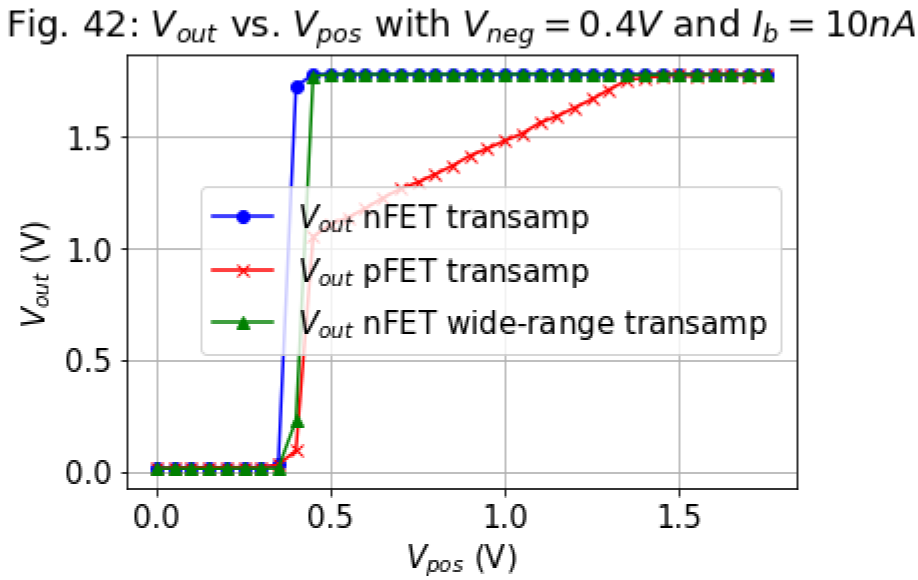
> In contrast to the regular transamp, the wide range transamp behaves like a regular amplifier: its $V_{out} isnearlyatGNDwhen$V_1 > V2$andriseslearlyinstantlytonearly$V{dd}$when$V_1 < V_2$. Thismeansthatthegraphisnotdependenton$\kappa
> $. Asintheregulartransamp, shiftingV_2 results in shifting the graph.

5.3 Comparison with 5T transamps

Compare the V_{out} vs V_{pos} (V_1) curves of the three transamps with different V_{neg} (V_2)

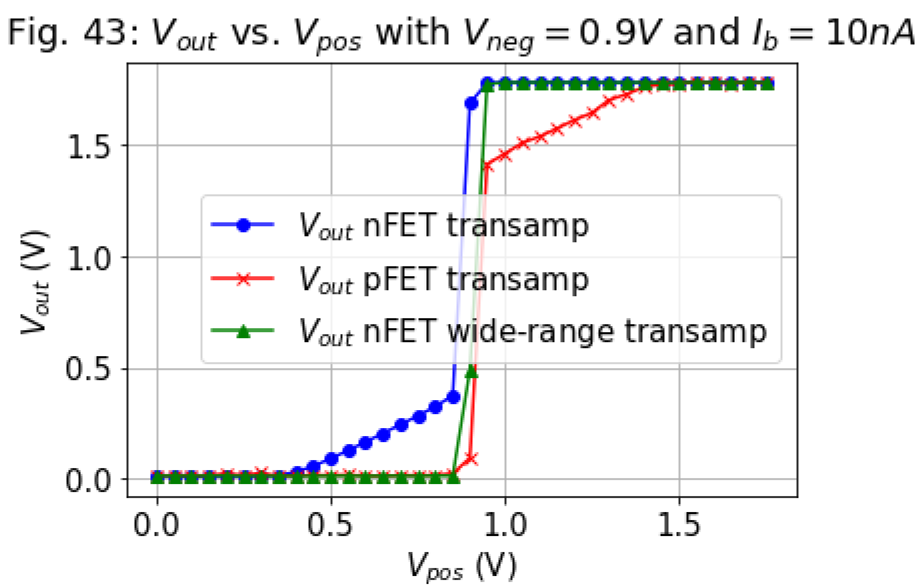
In [64]:

```
# fix Vn < 0.9V, Compare Vout vs Vpos

[V1_sweep_ex5_getset_save,Vout_V1_sweep_ex5_save_1] = np.loadtxt('./data/Vout_sweep_ex5_3.
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1] = np.loadtxt('./data/Vout_sweep_ex3_4
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1] = np.loadtxt('./data/Vout_sweep_ex4_3
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1, 'bo-', label="$V_{out}$ nFET
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1, 'rx-', label="$V_{out}$ pFET
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_1, 'g^-', label="$V_{out}$ nFET
plt.title("Fig. 42: $V_{out}$ vs. $V_{pos}$ with $V_{neg} = 0.4V$ and $I_b = 10 nA$")
plt.xlabel("$V_{pos}$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```



In [56]:

```
# fix Vn = 0.9V, Compare Vout vs Vpos

[V1_sweep_ex5_save,V1_sweep_ex5_getset_save,Vout_V1_sweep_ex5_save_1] = np.loadtxt('./data
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1] = np.loadtxt('./data/Vout_sweep_ex3_1
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1] = np.loadtxt('./data/Vout_sweep_ex4_1
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1, 'bo-', label="$V_{out}$ nFET
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1, 'rx-', label="$V_{out}$ pFET
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_1, 'g^-', label="$V_{out}$ nFET
plt.title("Fig. 43: $V_{out}$ vs. $V_{pos}$ with $V_{neg} = 0.9V$ and $I_b = 10 nA$")
plt.xlabel("$V_{pos}$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

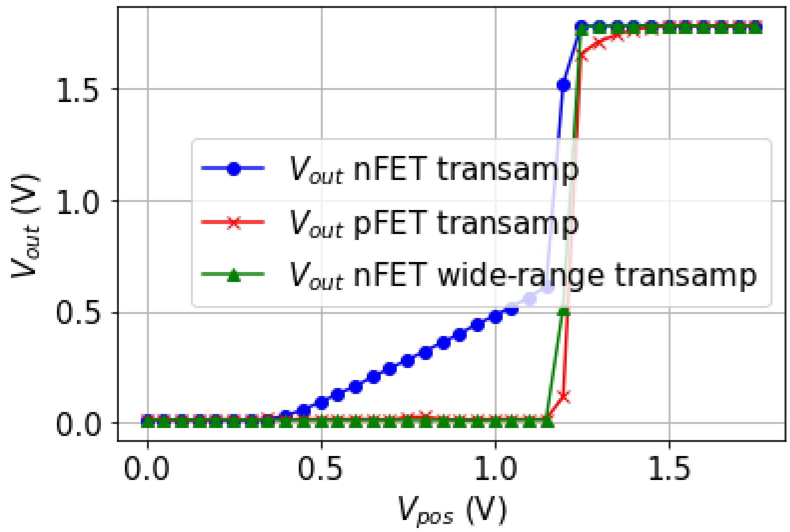


In [76]:

```
# fix Vn > 0.9V, Compare Vout vs Vpos

[V1_sweep_ex5_getset_save,Vout_V1_sweep_ex5_save_1] = np.loadtxt('./data/Vout_sweep_ex5_4.
[V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1] = np.loadtxt('./data/Vout_sweep_ex3_5
[V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1] = np.loadtxt('./data/Vout_sweep_ex4_4
plt.plot(V1_sweep_ex3_getset_save, Vout_V1_sweep_ex3_save_1, 'bo-', label="$V_{out}$ nFET
plt.plot(V1_sweep_ex4_getset_save, Vout_V1_sweep_ex4_save_1, 'rx-', label="$V_{out}$ pFET
plt.plot(V1_sweep_ex5_getset_save, Vout_V1_sweep_ex5_save_1, 'g^-', label="$V_{out}$ nFET
plt.title("Fig. 44: $V_{out}$ vs. $V_{pos}$ with $V_{neg} = 1.2V$ and $I_b = 10 nA$")
```

```
plt.xlabel("$V_{pos}$ (V)")
plt.ylabel("$V_{out}$ (V)")
plt.legend()
plt.grid()
plt.show()
```

Fig. 44: V_{out} vs. V_{pos} with $V_{neg} = 1.2V$ and $I_b = 10nA$



To conclude your observations:

> - nFET: V_{out} steadily rises with a slope of κ until $V_{pos} > V_{neg}$, when it peakes to almost V_{dd}
> .
> - pFET: V_{out} is at nearly GND when $V_{pos} < V_{neg}$, rises nearly instantly when $V_{pos} = V_{neg}$, but then rises with a slope of κ linearly up to V_{dd}.
> - wide range nFET: In contrast to the regular transamp, the wide range transamp behaves like a regular amplifier: its $V_{out} isnearlyatGNDwhen$V_1 > V2 $andriseslearlyinstantlytonearly$V{dd}$when$V_1 < V_2 $. Thismeansthatthegraphisnotdependentonκ.

6 Postlab

1. When we set the output voltage of the transconductance amplifier to a certain value between gnd and Vdd and measured its output current, we found that at some nonzero input voltage (the offset voltage) the output current was zero. Will we get a different input offset voltage if we change the output voltage? Explain why.

> No, I_{out} is always zero at $V_1 = V_2$. Per the following equations, changing V_{out} does not change the point at which I_{out} is zero:
>
> $$I_{out} = I_b \tanh\left(\frac{\kappa}{2U_T}(V_1 - V_2)\right) \tag{1}$$
>
> $$V_{out} = A(V_1 - V_2) \tag{2}$$

1. What are the conditions for keeping M_b in saturation for the P-type transamp? Do they differ from the N-type transamp?

> The requirement is $\max(V_1, V_2) < \frac{V_{dd} - 4U_T}{\kappa} + V_b$.

> This is the exact opposite of the situation for nFET, where the requirement is
> $\max(V_1, V_2) > \frac{V_{dd}-4U_T}{\kappa} + V_b$

1. What are the advantages and disadvantages of the wide-output-range transconductance amplifier vs. a standard transconductance amplifier? Consider layout area, output voltage swing, offset voltage, current asymmetries, and the gain A. Why is the wide-output-range transamp better suited for construction of a high-gain single-stage amplifier? *Hint: think about the necessary symmetries between pairs of transistors.*

> Advantages:
>
> - I_{out} is symmetrical when sweeping V_1 or V_2
> - Way higher gain (even $A > 1000$)
> - Way bigger output voltage swing (nearly GND to nearly V_{dd})
>
>   Disadvantages:
>
> - Bigger layout area
> - More necessary transistors (5 vs 9, nearly twice as many)
> - More current mirrors means more opportunity for mismatches, moving the offset voltage away from $0V$.
>
>   The wide-output-range transamp is better suited for construction of a high-gain single-stage amplifier because it has a much higher gain and actual symmetric output voltage curve. The regular transamp has a linear component on $V_1 < V_2$ for nFET and $V_1 > V_2$ for pFET.

7 What we expect after lab 4 and lab5

Can you sketch a transamp, a wide range transamp, a current correlator, and a bump circuit in both n- and p-type varieties?

How does a differential pair work? How does the common-node voltage change with the input voltages? How can you compute the differential tail currents from the subthreshold equations, and how do you obtain the result in terms of the differential input voltage? How does a current-correlator work? How does a bump circuit work?

The I-V characteristics of a transconductance amplifier below threshold. What's the functional difference between simple and wide-output-range transamp? The subthreshold transconductance gm. The relation between gain A, transistor drain conductances gd, and transconductances gm.

Can you reason through all the node voltages in these circuits? I.e., if we draw the circuit and provide specific power supply and input voltages, can you reason to estimate all the other node voltages, at least to first order approximations, assuming κ = 1?

8 Congratulations

Wish you joy when you look back on your works, beautiful plots and all your efforts!