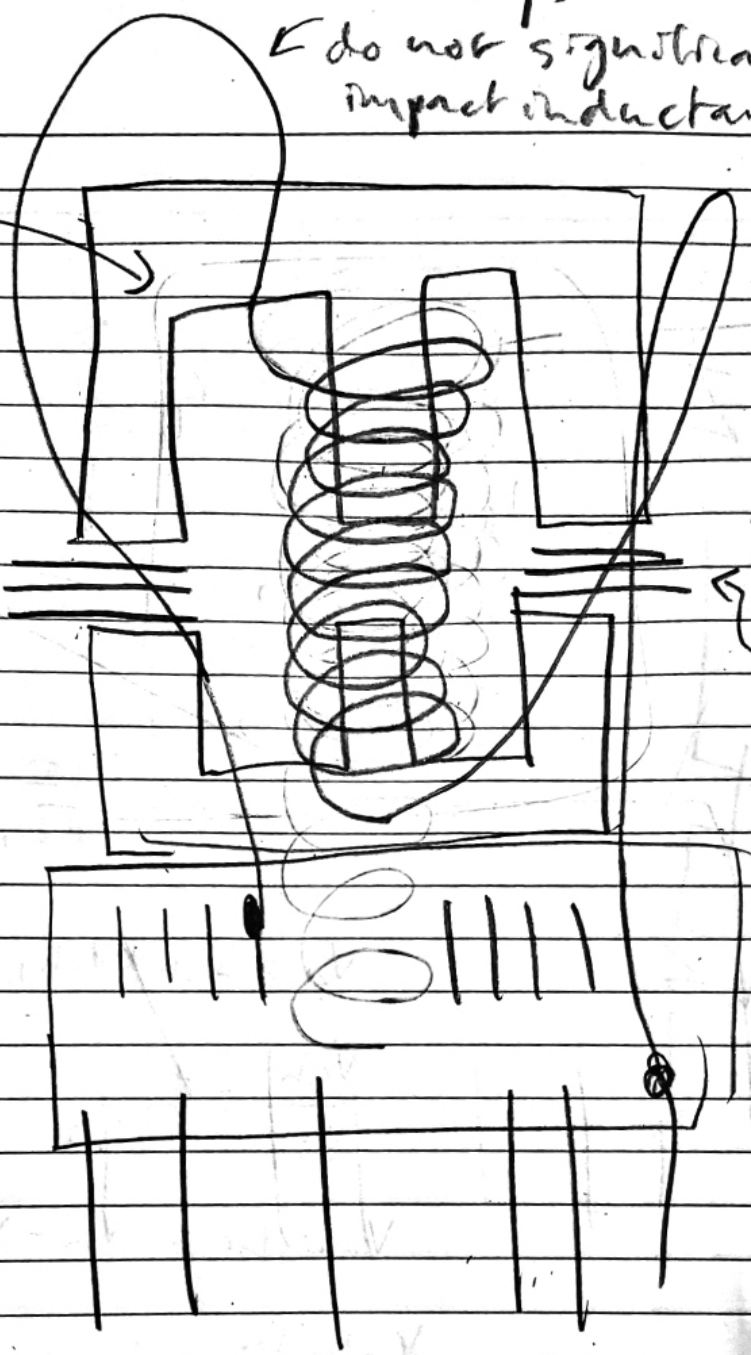
extra loops of wire
← do not significantly impact inductance

PC95 ferrite material

9 complete turns

20AWG magnet wire

3x 0.003" PTFE sheets = 0.009" gap



Actual size

Ferrite core specs

$A_e = 62.6\ mm^2$

$A_{mm} = 59.1\ mm^2$

$L_e = 45.7mm$

Lab 4 Prep: Design and Manufacture Inductor

Lauren Xiong, Ian Eykamp, Melissa Kazazic

In [ ]:
```
# %%capture
# Imports and setup
from pint import UnitRegistry
import math
import numpy

# use pint
units = UnitRegistry()
units.default_format = "~P"
units.load_definitions('units_definition.txt')
```

Out[ ]:
```
{None: ParsedSource(parsed_source=PintRootBlock(opening=BOF(start_line=0, start_co
l=0, end_line=0, end_col=0, raw=None, content_hash=Hash(algorithm_name='blake2b',
hexdigest='fc8618d3ea1c0bbca70f75d4bd6141ec24df680bd120dd216c2b2e09280a971990a1fb7
1c275ca4cdca0f1a5c639dfc3886d3ee06244dc0a296617a0ff969879'), path=WindowsPath('c:/
dev/git/power-electronics/Lab 4/units_definition.txt'), mtime=1678082649.396907),
body=(UnitDefinition(name='tesla', defined_symbol='T', aliases=('teslas',), conver
ter=ScaleConverter(scale=1.0), reference=<UnitsContainer({'amp': -1, 'kilogram':
1, 'second': -2})>, start_line=1, start_col=0, end_line=1, end_col=60, raw='tesla
= kilogram * second ** (-2) * amp ** (-1) = T = teslas'), UnitDefinition(name='web
er', defined_symbol='Wb', aliases=('webers',), converter=ScaleConverter(scale=1),
reference=<UnitsContainer({'meter': 2, 'tesla': 1})>, start_line=2, start_col=0, e
nd_line=2, end_col=40, raw='weber = tesla * meter ** 2 = Wb = webers')), closing=E
OS(start_line=3, start_col=0, end_line=3, end_col=0, raw=None)), config=ParserConf
ig(non_int_type=<class 'float'>))}
```

Pt 1. Determining Turns and Gap Distance

Inductance and Permeance

In [ ]:
```
# Calculating for Inductance and Permeance

target_inductance = 20e-6 * units.henry
print(target_inductance)

perm_air = 4e-7*math.pi*(units.meter*units.kilogram*units.second**(-2)*units.amp**(
eq_area_core = 62.6 * units.millimeter**2
eq_length_core = 45.7 * units.millimeter
rel_perm_core = 3300

# changing variables
eq_length_gap = 0.009 * units.inch
n_turns = 7
```

```
# gap area
lg = eq_length_gap
g = 9 * units.millimeter

eq_area_wing_gap = (2 * ((7 * units.millimeter) + lg) * ((10.65 * units.millimeter)
    (((7 * units.millimeter) + lg) * (numpy.sqrt((g - lg)** 2 - (7 * units.millimet
    ((g - lg) ** 2) * (math.asin((7 * units.millimeter + lg) / (g - lg)))


area_circle = ((7.9 / 2) * units.millimeter + lg)**2 * math.pi
eq_area_gap = (area_circle + 2 * eq_area_wing_gap)/2

print("Circle area:", area_circle)
print("Wing gap area:", eq_area_wing_gap)
print("Total equivalent area of gap:", eq_area_gap)

P_gap = (perm_air*eq_area_gap)/eq_length_gap
P_core = (perm_air*rel_perm_core*eq_area_core)/eq_length_core
P = ((P_core)**(-1)+(P_gap)**(-1))**(-1)
L = P*(n_turns**2)
print("Calculated Permeance:", units.Quantity(P, units.microhenry))
print("Calculated Inductance:",units.Quantity(L, units.microhenry))
```

```
2×10⁻⁵ H
Circle area: 54.85440043768629 mm²
Wing gap area: 46.835775356850235 mm²
Total equivalent area of gap: 74.26297557569337 mm²
Calculated Permeance: 0.38086007700599384 µH
Calculated Inductance: 18.6621437732937 µH
```

Peak Current Density

In [ ]:
```
# Peak current density

awg_val = 20 # AWG
print(awg_val, "AWG")

awg_d = 0.005*92**((36-awg_val)/39)*units.inch
awg_area = math.pi*((awg_d/2)**2)

D = 0.5 # duty cycle
ipeak = 10*units.amp
irms = ipeak * (numpy.sqrt(D/3)) # current root mean square

# current density
current_density = irms/awg_area
print("Current density:", units.Quantity(current_density,units.A/units.millimeter**
```

```
20 AWG
Current density: 7.887038529387198 A/mm²
```

Peak Flux Density

In [ ]:
```
# Peak flux density
```

```
# givens
min_effective_area = 59.1 * units.millimeter**2
Bmax = 200*units.mT

# magnetic potential
mmf = n_turns*irms

# max flux density
flux_max = (mmf*P)/min_effective_area

print("Bmax:",Bmax)
print("Maximum Flux Density:",units.Quantity(flux_max,units.mT))
```

```
Bmax: 200 mT
Maximum Flux Density: 184.16215353729703 mT
```

Pt. 2: Real values

In [ ]:
```
# Recorded values
n_actual = 9.3 #turns
L_actual = 0.0189 * units.millihenry

# Permeance
P_actual = L_actual/(n_actual**2)

print("Measured Turns:", n_actual)
print("Measured Inductance:", units.Quantity(L_actual, units.uhenry))
print("Measured Permeance:", units.Quantity(P_actual, units.uhenry))
```

```
Measured Turns: 9.3
Measured Inductance: 18.9 µH
Measured Permeance: 0.218522372528616 µH
```

In [ ]:
```
# Peak flux density

flux_actual = n_actual*irms*P_actual

flux_density_actual = flux_actual/min_effective_area

print("Actual Maximum Flux Density:",units.Quantity(flux_density_actual,units.mT))
```

```
Actual Maximum Flux Density: 140.38339773605898 mT
```

So, despite needing more turns to meet the required inductance, this also meant that the permeance shrunk, and so the maximum flux density is also around the value that we wanted.

Peak Energy Stored

Using Calculated Values

In [ ]:
```
# Total delta Work
dW_total = (1/2)*(mmf**2)*P
```

```
# Work of Core
dW_core = dW_total * (P/P_core)

# Work of Gap
dW_gap = dW_total * (P/P_gap)

# Print values
print("Estimated total delta Work", units.Quantity(dW_total,units.ujoule))
print("Core delta Work", units.Quantity(dW_core,units.ujoule))
print("Gap delta Work", units.Quantity(dW_gap,units.ujoule))
```

```
Estimated total delta Work 155.5178647774475 µJ
Core delta Work 10.427106244646026 µJ
Gap delta Work 145.09075853280146 µJ
```

Using Measured Value

In [ ]:

```
# Total Work

dW_actual = (1/2)*(mmf**2)*P_actual
print("Actual delta Work", units.Quantity(dW_actual,units.ujoule))
```

```
Actual delta Work 89.22996878251821 µJ
```

To find the inductance of our core, we used the equation $\mathcal{P} = \frac{\mu * A}{l}$ applied to both the ferrite core and the air gap. We had

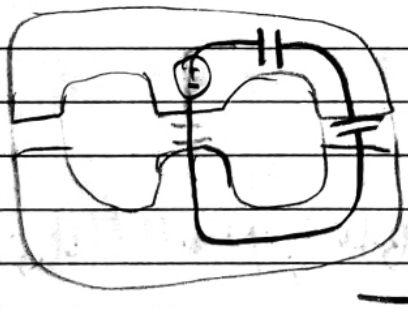
$$\mathcal{P}_{core} = \frac{\mu_{core} * A_{e,core}}{l_{e,core}} \approx \frac{3300 * \mu_0 * 62.6\,mm^2}{45.7\,mm}$$

(μ_{core} ← material data sheet; $A_{e,core}$ ← core geometry data sheet; $l_{e,core}$ ← core geometry data sheet)

all from datasheet

$$\mathcal{P}_{air} = \frac{\mu_0 * A_{e,gap}}{l_{e,gap}} \approx \frac{\mu_0 * 74.26\,mm^2}{l_{gap}}$$

Since the permeances act like capacitors in series, we combine the permeances in parallel.



$$\mathcal{P}_{total} = \mathcal{P}_{core} \parallel \mathcal{P}_{air}$$

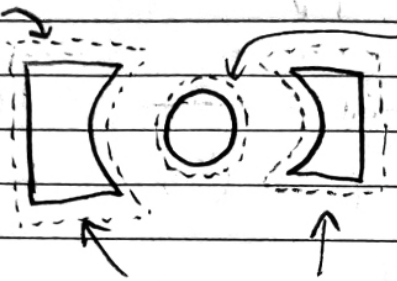
$$\mathcal{P}_{total} = \frac{1}{\left(\frac{1}{\mathcal{P}_{core}} + \frac{1}{\mathcal{P}_{air}}\right)}$$

Because $\mathcal{P}_{core}$ is so much larger than $\mathcal{P}_{air}$ due to the $\mu_r = 3300 * \mu_0$ term, the total permeance becomes almost equal to the smaller permeance, $\mathcal{P}_{air}$, just as the equivalent value of two resistors in parallel is dominated by the smaller one.

Next, we had to find the equivalent cross section of the gap.

It turns out that the entire cross section is important.

There are three pieces:



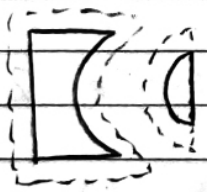
This one is given by πr^2.

Remember to incorporate the expansion of the equivalent cross section by an outside equal to the length of the gap.

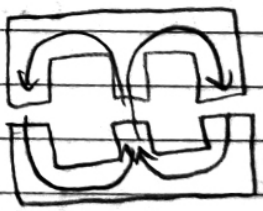
These two are given by a geometric equation

This yielded an area that was more than twice the equivalent area given on the datasheet. We realized that since the flux must flow in a loop through the core, only one-half of the total cross section may be used by flux traveling in one direction in the loop.

Thus, the equivalent cross-sectional area for the gap is equivalent to



even if this is not the actual path the flux takes (in reality it is probably this. ↓


).

Hence, we have $A_{e,gap} \approx 74.26 mm^2$,

which is slightly larger than the core's typical cross-section of 62.6 mm^2, due to the boughing out within the gap.

For the peak current density, we calculated the RMS current in discontinuous conduction mode given a controlled peak current to be

$I_{RMS} = \sqrt{\left(\frac{D}{3}\right)} * I_{peak}$, from the equation sheet in class.

duty cycle

The current density $J = \frac{I_{RMS}}{A_{wire}} = \frac{I_{RMS}}{\pi\left(\frac{D_{wire}}{2}\right)^2} < 10\ A/mm^2$ must not exceed $10\ A/mm^2$.

We chose 20 AWG wire, which yielded $7.8\ A/mm^2$.

We ignored the various skin effects, which would increase the current density by hopefully only a few percent, due to the relatively low switching frequency (50kHz).

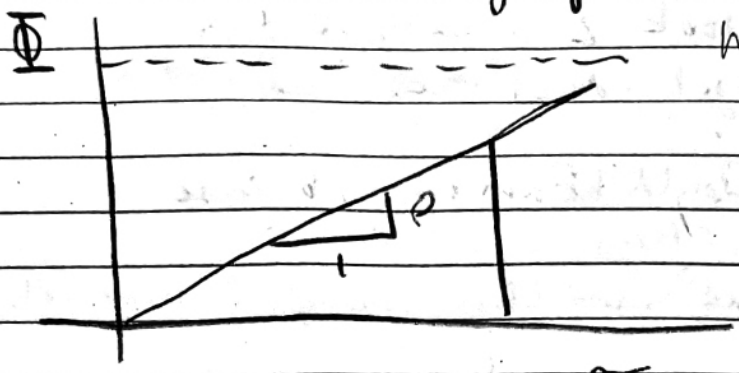
For safety, we could have used one wider winding or multiple smaller windings in parallel.

Our Flux density $\vec{B}$ was not to exceed 200mT.

Our minimum effective area of the core was $59.1\ mm^2$ from the datasheet.

By the equation $\Phi_{max} = \|\vec{B}_{max}\| * A_{min}$, we obtained that our flux Φ should never exceed this threshold (in the worst case scenario where the greatest flux Φ_{max} flows through the smallest cross-sectional area A_{min}).

We have this graph



Φ flux through magnetic cross section can never exceed this threshold.

$$\mathcal{F} = N * I$$

We can calculate our system's MMF as $\mathcal{F} = N * I$ from the inductor windings.

We also know the slope of the curve to be a straight line with slope $\mathcal{P}$. Hence, for a given MMF, we can calculate exactly what the flux is, as $\Phi = \mathcal{F} * \mathcal{P}$.

Thus, the flux $\Phi = \mathcal{F} * \mathcal{P} = N * I * \mathcal{P}$ can never exceed $\Phi_{max} / A_{min} = \vec{B}_{max}$.

Then, $\frac{\Phi_{max}}{A_{min}} = \vec{B}_{max}$, $\frac{\Phi}{A_{min}} = \frac{N * I * \mathcal{P}}{A_{min}} < 200\,mT$.

Since $\mathcal{P} = \frac{L}{N^2}$ and L_{target} is fixed,

we had $\frac{N * I_{max} * \frac{L}{N^2}}{A_{min}} = \frac{L * I_{max}}{N * A_{min}} < 200\,mT$ and $\mathcal{P}_{air} = \frac{\mu_0 * 74.26\,mm^2}{l_{gap}}$

So, N had to be large enough to satisfy $\frac{L * I_{max}}{N * A_{min}} < 200\,mT$, which meant the permeance had to be smaller due to $\mathcal{P} = \frac{L_{fixed}}{N^2}$, so we needed a larger l_{gap}.

In reality, it took $2+\frac{1}{2}$ extra turns to achieve the target inductance.

We figured this would be an issue, because a larger number of turns in isolation increases the maximum flux through the material.

However, the need for additional turns implies that the permeance was actually much smaller to begin with, and the lower permeance actually contributed to a lower magnetic flux density by $\vec{B} = \frac{NIP}{A_{min}} = \frac{I\frac{L}{N}}{A_{min}} = \frac{IL}{N A_{min}}$

I.e. because more turns were required (allowed) to achieve the same inductance, the overall $\vec{B}$ field was actually weaker than predicted, even after factoring in the extra N turns.

→ And also that the relevant gap can be made very small, so as to be able to make the permeance very large over a short distance.

Permeance is nothing more than the reciprocal of reluctance, which is analogous to resistance for MMF, or conductance to the $\vec{B}$ field. The ferrite core conducts the $\vec{B}$ field very well, so the reluctance is due almost entirely to the air gap.

~~If the air gap is made too small (permeance too big), then the reluctance of the core (permeance of the core) will start to dominate.~~

Up to that point, however, we can maintain very fine control over the reluctance (and thus the permeance) by varying the length of air gap over which the $\vec{B}$ field must travel.

There is a constant $\vec{E}$ field inside a resistor such that $\Delta V = -\int \vec{E}\, ds$

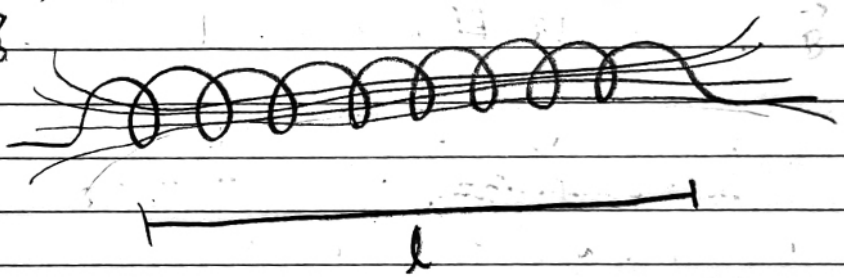
$\frac{kg}{s^2}$ $\frac{kg}{J}$

$= \Delta V = IR$

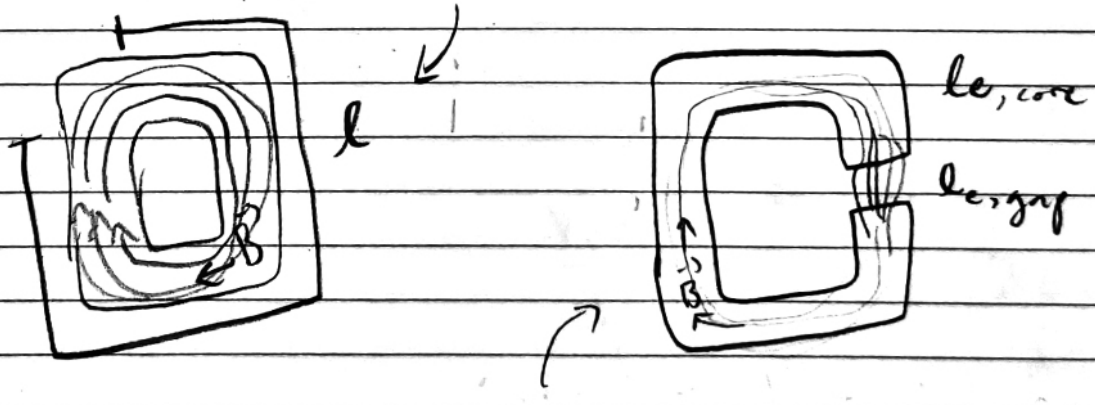
The permeance of a length of material is given by $\frac{N * A_e}{l_e}$.

Thus, it is driven largely by the equivalent length of the inductive area.

For an air core inductor, the length is just the length of the solenoid, since that is where all of the $\vec{B}$ field is concentrated.



For a ferrite core inductor, the equivalent length is calculated around the loop, since that is where the flux is carried.



For a gapped ferrite core inductor, the permeances are added in parallel, and since the permeance is much greater for the ferrite core, its effect on the total permeance is negligible. Thus, the equivalent length is calculated over just the gap with permeability similar to that of air, μ_0.

It is part of the design of gapped core inductors that they give you very fine control over the length of the dominant gap.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
from UliEngineering.EngineerIO import format_value
from si_prefix import si_format
import plecs_helper as helper
%matplotlib
%matplotlib inline
```

```
Using matplotlib backend: TkAgg
```

Lab 4 Report

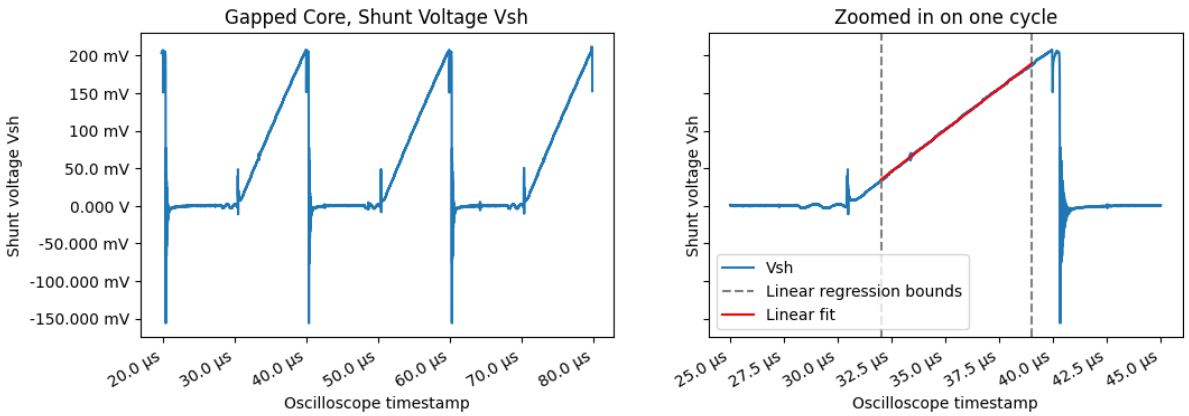
Ian Eykamp

Lab group: Ian Eykamp, Lauren Xiong, Melissa Kazazic

For the gapped core inductor

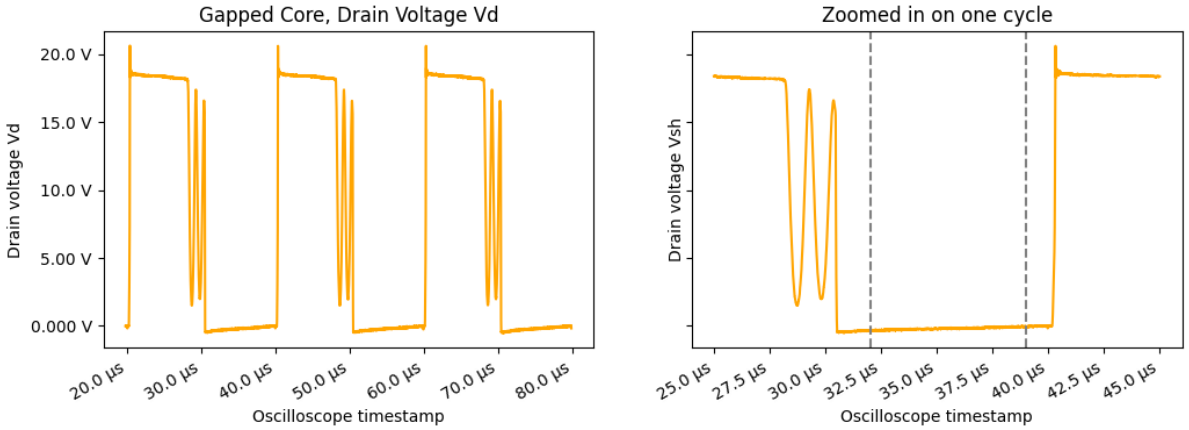
In [ ]:
```
(df, tspan, tstep) = helper.read_rigol_csv("IronCore_Vsh_Vd.csv", ch1 = "Vsh", ch2
df_envelope = df
df_zoom = df[(df["t"] > 25e-6) & (df["t"] < 45e-6)]
linear_ts = (32e-6, 39e-6)
df_linreg = df[(df["t"] > linear_ts[0]) & (df["t"] < linear_ts[1])]

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = False, sharey = True,
fig.autofmt_xdate()
helper.axes_labels("Oscilloscope timestamp", "s", "Shunt voltage Vsh", "V", title =
ax1.plot(df_envelope["t"], df_envelope["Vsh"], label = "Vsh")

helper.axes_labels("Oscilloscope timestamp", "s", "Shunt voltage Vsh", "V", title =
ax2.plot(df_zoom["t"], df_zoom["Vsh"], label = "Vsh")
ax2.axvline(x = linear_ts[0], linestyle = "dashed", color = "grey", label = "Linear
ax2.axvline(x = linear_ts[1], linestyle = "dashed", color = "grey")

x = df_linreg["t"]
y = df_linreg["Vsh"]
A = np.vstack([x, np.ones(len(x))]).T
a, b = np.linalg.lstsq(A, y, rcond=None)[0]
ax2.plot(df_linreg["t"], df_linreg["t"] * a + b, linestyle = "solid", color = "red"
ax2.legend(loc = "lower left")

Vin = 17.8 # V
Vout = 9.08 # V
dIdt = a / 0.05 # ohms
Vinductor = Vin - Vout
L = Vinductor / dIdt
print(f"L: {si_format(L, precision = 2)}H")
```

```
L: 19.75 µH
```



In [ ]:
```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = False, sharey = True,
fig.autofmt_xdate()
helper.axes_labels("Oscilloscope timestamp", "s", "Drain voltage Vd", "V", title =
ax1.plot(df_envelope["t"], df_envelope["Vd"], color = "orange", label = "Vd")

helper.axes_labels("Oscilloscope timestamp", "s", "Drain voltage Vsh", "V", title =
ax2.plot(df_zoom["t"], df_zoom["Vd"], color = "orange", label = "Vd")
ax2.axvline(x = linear_ts[0], linestyle = "dashed", color = "grey", label = "Linear
ax2.axvline(x = linear_ts[1], linestyle = "dashed", color = "grey")
```

Out[ ]:
```
<matplotlib.lines.Line2D at 0x1f7885a2d60>
```

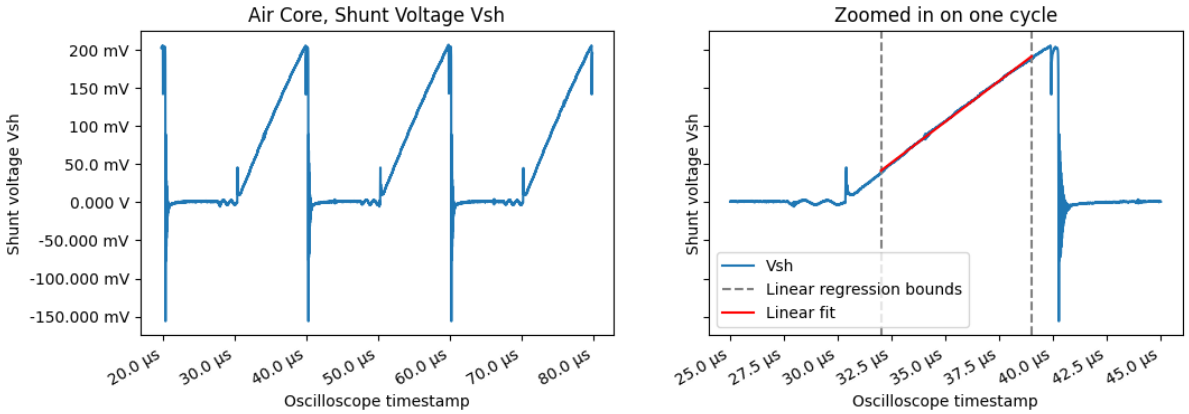


For the triangular air core inductor

In [ ]:
```
(df, tspan, tstep) = helper.read_rigol_csv("triangular_airCore.csv", ch1 = "Vsh", c
df_envelope = df
df_zoom = df[(df["t"] > 25e-6) & (df["t"] < 45e-6)]
linear_ts = (32e-6, 39e-6)
df_linreg = df[(df["t"] > linear_ts[0]) & (df["t"] < linear_ts[1])]

fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = False, sharey = True,
fig.autofmt_xdate()
helper.axes_labels("Oscilloscope timestamp", "s", "Shunt voltage Vsh", "V", title =
ax1.plot(df_envelope["t"], df_envelope["Vsh"], label = "Vsh")

helper.axes_labels("Oscilloscope timestamp", "s", "Shunt voltage Vsh", "V", title =
ax2.plot(df_zoom["t"], df_zoom["Vsh"], label = "Vsh")
ax2.axvline(x = linear_ts[0], linestyle = "dashed", color = "grey", label = "Linear
```

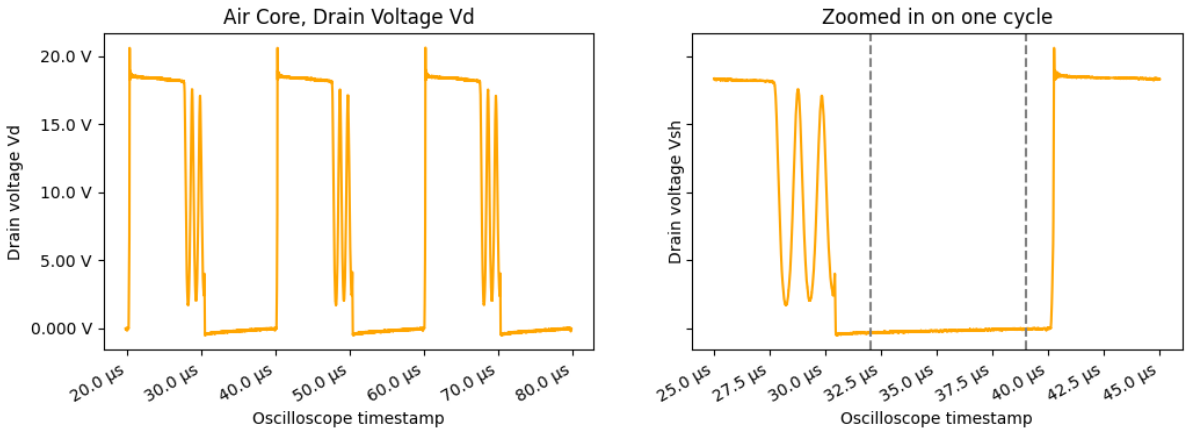
```
ax2.axvline(x = linear_ts[1], linestyle = "dashed", color = "grey")

x = df_linreg["t"]
y = df_linreg["Vsh"]
A = np.vstack([x, np.ones(len(x))]).T
a, b = np.linalg.lstsq(A, y, rcond=None)[0]
ax2.plot(df_linreg["t"], df_linreg["t"] * a + b, linestyle = "solid", color = "red"
ax2.legend(loc = "lower left")

Vin = 17.8 # V
Vout = 9.08 # V
dIdt = a / 0.05 # ohms
Vinductor = Vin - Vout
L = Vinductor / dIdt
print(f"L: {si_format(L, precision = 2)}H")
```

```
L: 20.32 µH
```



In [ ]:
```
fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, sharex = False, sharey = True,
fig.autofmt_xdate()
helper.axes_labels("Oscilloscope timestamp", "s", "Drain voltage Vd", "V", title =
ax1.plot(df_envelope["t"], df_envelope["Vd"], color = "orange", label = "Vd")

helper.axes_labels("Oscilloscope timestamp", "s", "Drain voltage Vsh", "V", title =
ax2.plot(df_zoom["t"], df_zoom["Vd"], color = "orange", label = "Vd")
ax2.axvline(x = linear_ts[0], linestyle = "dashed", color = "grey", label = "Linear
ax2.axvline(x = linear_ts[1], linestyle = "dashed", color = "grey")
```

Out[ ]:
```
<matplotlib.lines.Line2D at 0x1f7887821c0>
```



Results:

The gapped core inductor had an experimental inductance value of 19.75uH at 4A of current. My air core inductor from lesson 5 had an inductance of 20.32uH at a similar current. In both cases, this is a larger inductance than was obtained on the LCR meter (18.96uH and 18.69uH for the gapped core and air core, respectively).

This is reminiscent of Lab 3, in which the calculated inductance value also depended on current, albeit in Lab 3, the inductance decreased with higher current.

Estimating efficiency

This is the entirety of my independent steps (I should have probably done more but I didn't have the time). I researched the Steinmetz equation by reading this app note:

https://www.we-online.com/components/media/o109035v410%20AppNotes_ANP029_AccurateInductorLossDete

I found Pcv from the inductor from the PC95 ferrite material datasheet and showed that at 50kHz, the power loss due to magnetic hysteresis is negligible.

In [ ]:
```
# Imports and setup
from pint import UnitRegistry
import math
import numpy

# use pint
units = UnitRegistry()
units.default_format = "~P"
```

In [ ]:
```
def power_efficiency(Vin, Iin, Vout, Rload = 5 * units.ohm):
    Iout = Vout / Rload
    return (Vout * Iout) / (Vin * Iin)

Iin_gap_core = 1.08 * units.amp
Vin_gap_core = 17.8 * units.volt
Vout_gap_core = 9.33 * units.volt
efficiency_gap_core = power_efficiency(Vin_gap_core, Iin_gap_core, Vout_gap_core)
print(f"Gap core inductor efficiency: {round(efficiency_gap_core.to_base_units() *

Iin_air_core = 1.10 * units.amp
Vin_air_core = 17.8 * units.volt
Vout_air_core = 9.08 * units.volt
efficiency_air_core = power_efficiency(Vin_air_core, Iin_air_core, Vout_air_core)
print(f"Air core inductor efficiency: {round(efficiency_air_core.to_base_units() *
```

```
Gap core inductor efficiency: 90.56%
Air core inductor efficiency: 84.21%
```

In [ ]:

```
core_volume = 2850 * units.millimeter ** 3
Pcv = 30 * units.kilowatt / units.meter ** 3
power_loss = Pcv * core_volume
print(f"Power loss due to magnetic hysteresis: {round(power_loss.to('watt'), 2)}")
print(f"Total power through inductor: {round((Iin_gap_core * Vin_gap_core).to('watt
```

```
Power loss due to magnetic hysteresis: 0.09 W
Total power through inductor: 19.22 W
```