LNG_deal_economics

February 4, 2026

1 LNG Deal Economics

1.1 A. LNG Deal Economics Calculation

1.1.1 1. Imports + Dataclasses

[1]:
```
from dataclasses import dataclass, asdict
from typing import Literal, Dict, Any
```

[2]:
```
@dataclass
class CargoParams:
    deal_type: Literal["FOB", "DES"]
    cargo_mmbtu: float
    sales_price_des: float
    purchase_price_fob: float
    boiloff_rate_voyage: float
    fuel_use_fraction_of_cargo: float
    freight_deduct_usd_per_mmbtu: float
    regas_fee_usd_per_mmbtu: float
    pipeline_tariff_usd_per_mmbtu: float
    hedge_price_usd_per_mmbtu: float
    hedge_volume_mmbtu: float


@dataclass
class ShippingParams:
    distance_nm: float
    speed_knots: float
    daily_charter_rate_usd: float
    boiloff_rate_sea_daily: float
```

1.1.2 2. Helper Functions

[3]:
```
def calc_shipping_days(distance_nm: float, speed_knots: float) -> float:
    return distance_nm / (speed_knots * 24.0)
```

```
def calc_voyage_boiloff(cargo_mmbtu: float, shipping_days: float,␣
↪daily_boiloff_rate: float) -> float:
    return cargo_mmbtu * daily_boiloff_rate * shipping_days


def calc_fuel_use(cargo_mmbtu: float, fuel_use_fraction: float) -> float:
    return cargo_mmbtu * fuel_use_fraction


def calc_freight_cost(daily_charter_rate_usd: float, shipping_days: float) ->␣
↪float:
    return daily_charter_rate_usd * shipping_days


def calc_hedge_pnl(hedge_price: float, physical_price: float, hedge_volume:␣
↪float) -> float:
    return (physical_price - hedge_price) * hedge_volume
```

1.1.3 3. Core LNG P&L Function

[4]:
```
def lng_cargo_economics(cargo: CargoParams, shipping: ShippingParams) ->␣
↪Dict[str, Any]:

    shipping_days = calc_shipping_days(shipping.distance_nm, shipping.
↪speed_knots)
    freight_cost_total = calc_freight_cost(shipping.daily_charter_rate_usd,␣
↪shipping_days)

    voyage_boiloff_mmbtu = calc_voyage_boiloff(
        cargo.cargo_mmbtu, shipping_days, shipping.boiloff_rate_sea_daily
    )
    fuel_use_mmbtu = calc_fuel_use(cargo.cargo_mmbtu, cargo.
↪fuel_use_fraction_of_cargo)

    total_losses_mmbtu = voyage_boiloff_mmbtu + fuel_use_mmbtu
    net_delivered_mmbtu = max(cargo.cargo_mmbtu - total_losses_mmbtu, 0)

    regas_cost_total = net_delivered_mmbtu * cargo.regas_fee_usd_per_mmbtu
    pipeline_cost_total = net_delivered_mmbtu * cargo.
↪pipeline_tariff_usd_per_mmbtu

    if cargo.deal_type == "DES":
        fob_cost_total = cargo.cargo_mmbtu * cargo.purchase_price_fob
        des_revenue_total = net_delivered_mmbtu * cargo.sales_price_des

        gross_margin = des_revenue_total - fob_cost_total - freight_cost_total
        downstream_costs = regas_cost_total + pipeline_cost_total
```

```
        net_margin = gross_margin - downstream_costs

        physical_price_for_hedge = cargo.sales_price_des

    else:  # FOB
        fob_revenue_total = cargo.cargo_mmbtu * cargo.purchase_price_fob
        freight_deduct_total = cargo.cargo_mmbtu * cargo.
↪freight_deduct_usd_per_mmbtu
        netback_total = fob_revenue_total - freight_deduct_total

        gross_margin = netback_total - freight_cost_total
        downstream_costs = regas_cost_total + pipeline_cost_total
        net_margin = gross_margin - downstream_costs

        physical_price_for_hedge = cargo.purchase_price_fob - cargo.
↪freight_deduct_usd_per_mmbtu

    hedge_pnl = calc_hedge_pnl(
        hedge_price=cargo.hedge_price_usd_per_mmbtu,
        physical_price=physical_price_for_hedge,
        hedge_volume=cargo.hedge_volume_mmbtu
    )

    total_pnl = net_margin + hedge_pnl

    return {
        "shipping_days": round(shipping_days, 3),
        "voyage_boiloff_mmbtu": round(voyage_boiloff_mmbtu, 2),
        "fuel_use_mmbtu": round(fuel_use_mmbtu, 2),
        "total_losses_mmbtu": round(total_losses_mmbtu, 2),
        "net_delivered_mmbtu": round(net_delivered_mmbtu, 2),
        "freight_cost_total_usd": round(freight_cost_total, 2),
        "regas_cost_total_usd": round(regas_cost_total, 2),
        "pipeline_cost_total_usd": round(pipeline_cost_total, 2),
        "gross_margin_usd": round(gross_margin, 2),
        "downstream_costs_usd": round(downstream_costs, 2),
        "net_margin_usd": round(net_margin, 2),
        "hedge_pnl_usd": round(hedge_pnl, 2),
        "total_pnl_usd": round(total_pnl, 2),
    }
```

1.1.4 4. Example Scenario

[5]:
```
cargo = CargoParams(
    deal_type="DES",
    cargo_mmbtu=3_000_000,
    sales_price_des=12,
```

```
    purchase_price_fob=9.5,
    boiloff_rate_voyage=0.001,
    fuel_use_fraction_of_cargo=0.02,
    freight_deduct_usd_per_mmbtu=0.20,
    regas_fee_usd_per_mmbtu=0.30,
    pipeline_tariff_usd_per_mmbtu=0.20,
    hedge_price_usd_per_mmbtu=11,
    hedge_volume_mmbtu=2_000_000
)

shipping = ShippingParams(
    distance_nm=9000,
    speed_knots=15,
    daily_charter_rate_usd=80000,
    boiloff_rate_sea_daily=0.001
)

result = lng_cargo_economics(cargo, shipping)
result
```

[5]:
```
{'shipping_days': 25.0,
 'voyage_boiloff_mmbtu': 75000.0,
 'fuel_use_mmbtu': 60000.0,
 'total_losses_mmbtu': 135000.0,
 'net_delivered_mmbtu': 2865000.0,
 'freight_cost_total_usd': 2000000.0,
 'regas_cost_total_usd': 859500.0,
 'pipeline_cost_total_usd': 573000.0,
 'gross_margin_usd': 3880000.0,
 'downstream_costs_usd': 1432500.0,
 'net_margin_usd': 2447500.0,
 'hedge_pnl_usd': 2000000,
 'total_pnl_usd': 4447500.0}
```

[6]:
```
result
```

[6]:
```
{'shipping_days': 25.0,
 'voyage_boiloff_mmbtu': 75000.0,
 'fuel_use_mmbtu': 60000.0,
 'total_losses_mmbtu': 135000.0,
 'net_delivered_mmbtu': 2865000.0,
 'freight_cost_total_usd': 2000000.0,
 'regas_cost_total_usd': 859500.0,
 'pipeline_cost_total_usd': 573000.0,
 'gross_margin_usd': 3880000.0,
 'downstream_costs_usd': 1432500.0,
 'net_margin_usd': 2447500.0,
```

```
 'hedge_pnl_usd': 2000000,
 'total_pnl_usd': 4447500.0}
```

2 B. Sensitivity Tables

2.1 1. DES Price vs Total P&L

[7]:
```
import numpy as np
import pandas as pd

des_prices = np.linspace(8, 20, 25)  # range of DES prices
pnl_list = []

for p in des_prices:
    cargo.sales_price_des = p
    result = lng_cargo_economics(cargo, shipping)
    pnl_list.append(result["total_pnl_usd"])

df_des_pnl = pd.DataFrame({
    "DES Price (USD/MMBtu)": des_prices,
    "Total P&L (USD)": pnl_list
})

df_des_pnl
```

[7]:

	DES Price (USD/MMBtu)	Total P&L (USD)
0	8.0	-15012500.0
1	8.5	-12580000.0
2	9.0	-10147500.0
3	9.5	-7715000.0
4	10.0	-5282500.0
5	10.5	-2850000.0
6	11.0	-417500.0
7	11.5	2015000.0
8	12.0	4447500.0
9	12.5	6880000.0
10	13.0	9312500.0
11	13.5	11745000.0
12	14.0	14177500.0
13	14.5	16610000.0
14	15.0	19042500.0
15	15.5	21475000.0
16	16.0	23907500.0
17	16.5	26340000.0
18	17.0	28772500.0
19	17.5	31205000.0
20	18.0	33637500.0

21	18.5	36070000.0
22	19.0	38502500.0
23	19.5	40935000.0
24	20.0	43367500.0

2.2 2. Freight vs Netback

[8]:
```
freight_values = np.linspace(0.1, 2.0, 25)
netback_list = []

for f in freight_values:
    cargo.freight_deduct_usd_per_mmbtu = f
    result = lng_cargo_economics(cargo, shipping)
    netback_list.append(result["net_margin_usd"])

df_freight_netback = pd.DataFrame({
    "Freight Deduct (USD/MMBtu)": freight_values,
    "Netback (USD)": netback_list
})

df_freight_netback
```

[8]:

	Freight Deduct (USD/MMBtu)	Netback (USD)
0	0.100000	25367500.0
1	0.179167	25367500.0
2	0.258333	25367500.0
3	0.337500	25367500.0
4	0.416667	25367500.0
5	0.495833	25367500.0
6	0.575000	25367500.0
7	0.654167	25367500.0
8	0.733333	25367500.0
9	0.812500	25367500.0
10	0.891667	25367500.0
11	0.970833	25367500.0
12	1.050000	25367500.0
13	1.129167	25367500.0
14	1.208333	25367500.0
15	1.287500	25367500.0
16	1.366667	25367500.0
17	1.445833	25367500.0
18	1.525000	25367500.0
19	1.604167	25367500.0
20	1.683333	25367500.0
21	1.762500	25367500.0
22	1.841667	25367500.0
23	1.920833	25367500.0

```
24                    2.000000     25367500.0
```

2.3 3. Boil-off vs Delivered Volume

[9]:
```
boiloff_values = np.linspace(0.0005, 0.005, 25)
delivered_list = []

for b in boiloff_values:
    shipping.boiloff_rate_sea_daily = b
    result = lng_cargo_economics(cargo, shipping)
    delivered_list.append(result["net_delivered_mmbtu"])

df_boiloff_delivery = pd.DataFrame({
    "Daily Boiloff Rate": boiloff_values,
    "Delivered Volume (MMBtu)": delivered_list
})

df_boiloff_delivery
```

[9]:

	Daily Boiloff Rate	Delivered Volume (MMBtu)
0	0.000500	2902500.0
1	0.000688	2888437.5
2	0.000875	2874375.0
3	0.001063	2860312.5
4	0.001250	2846250.0
5	0.001438	2832187.5
6	0.001625	2818125.0
7	0.001813	2804062.5
8	0.002000	2790000.0
9	0.002188	2775937.5
10	0.002375	2761875.0
11	0.002563	2747812.5
12	0.002750	2733750.0
13	0.002938	2719687.5
14	0.003125	2705625.0
15	0.003313	2691562.5
16	0.003500	2677500.0
17	0.003688	2663437.5
18	0.003875	2649375.0
19	0.004063	2635312.5
20	0.004250	2621250.0
21	0.004438	2607187.5
22	0.004625	2593125.0
23	0.004813	2579062.5
24	0.005000	2565000.0

3 C. Plotly Interactive Charts

3.1 1. DES Price vs Total P&L Curve

[10]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in
/home/engpookw/miniconda3/envs/commercialanalyst_env/lib/python3.10/site-
packages (6.5.2)
Requirement already satisfied: narwhals>=1.15.1 in
/home/engpookw/miniconda3/envs/commercialanalyst_env/lib/python3.10/site-
packages (from plotly) (2.16.0)
Requirement already satisfied: packaging in
/home/engpookw/miniconda3/envs/commercialanalyst_env/lib/python3.10/site-
packages (from plotly) (24.2)
```

[11]:
```
import plotly.express as px

fig = px.line(
    df_des_pnl,
    x="DES Price (USD/MMBtu)",
    y="Total P&L (USD)",
    title="DES Price vs Total P&L"
)
fig.show()
```

3.2 2. Freight vs Netback Curve

[12]:
```
fig = px.line(
    df_freight_netback,
    x="Freight Deduct (USD/MMBtu)",
    y="Netback (USD)",
    title="Freight Deduct vs Netback"
)
fig.show()
```

3.3 3. Boil-off vs Delivered Volume Curve

[13]:
```
fig = px.line(
    df_boiloff_delivery,
    x="Daily Boiloff Rate",
    y="Delivered Volume (MMBtu)",
    title="Boiloff Rate vs Delivered Volume"
)
fig.show()
```

3.4 D. Interactive Plotly Sliders (ipywidgets)

3.5 1. Interactive DES Price Slider

[14]:
```
import ipywidgets as widgets
from ipywidgets import interact

@interact(des_price=widgets.FloatSlider(min=8, max=20, step=0.1, value=12))
def interactive_des(des_price):
    cargo.sales_price_des = des_price
    result = lng_cargo_economics(cargo, shipping)
    print(f"DES Price: {des_price}")
    print(f"Total P&L: {result['total_pnl_usd']:,}")
```

```
interactive(children=(FloatSlider(value=12.0, description='des_price', max=20.0,␣
↪min=8.0), Output()), _dom_cla…
```

3.6 2. Interactive Hedge P&L Visualisation

[15]:
```
@interact(
    hedge_price=widgets.FloatSlider(min=5, max=20, step=0.1, value=11),
    hedge_volume=widgets.FloatSlider(min=0, max=3_000_000, step=50_000,␣
↪value=2_000_000)
)
def hedge_visual(hedge_price, hedge_volume):
    cargo.hedge_price_usd_per_mmbtu = hedge_price
    cargo.hedge_volume_mmbtu = hedge_volume
    result = lng_cargo_economics(cargo, shipping)
    print(f"Hedge Price: {hedge_price}")
    print(f"Hedge Volume: {hedge_volume:,}")
    print(f"Hedge P&L: {result['hedge_pnl_usd']:,}")
    print(f"Total P&L: {result['total_pnl_usd']:,}")
```

```
interactive(children=(FloatSlider(value=11.0, description='hedge_price', max=20.
↪0, min=5.0), FloatSlider(value…
```

4 D. Margin Curve with Interactive DES Price Slider

[16]:
```
@interact(des_price=widgets.FloatSlider(min=8, max=20, step=0.1, value=12))
def margin_curve(des_price):
    cargo.sales_price_des = des_price
    result = lng_cargo_economics(cargo, shipping)

    fig = px.bar(
        x=["Gross Margin", "Net Margin", "Total P&L"],
        y=[result["gross_margin_usd"], result["net_margin_usd"],␣
↪result["total_pnl_usd"]],
        title=f"Margin Breakdown at DES Price = {des_price}"
```

```
    )
    fig.show()
```

```
interactive(children=(FloatSlider(value=12.0, description='des_price', max=20.0,␣
↪min=8.0), Output()), _dom_cla…
```