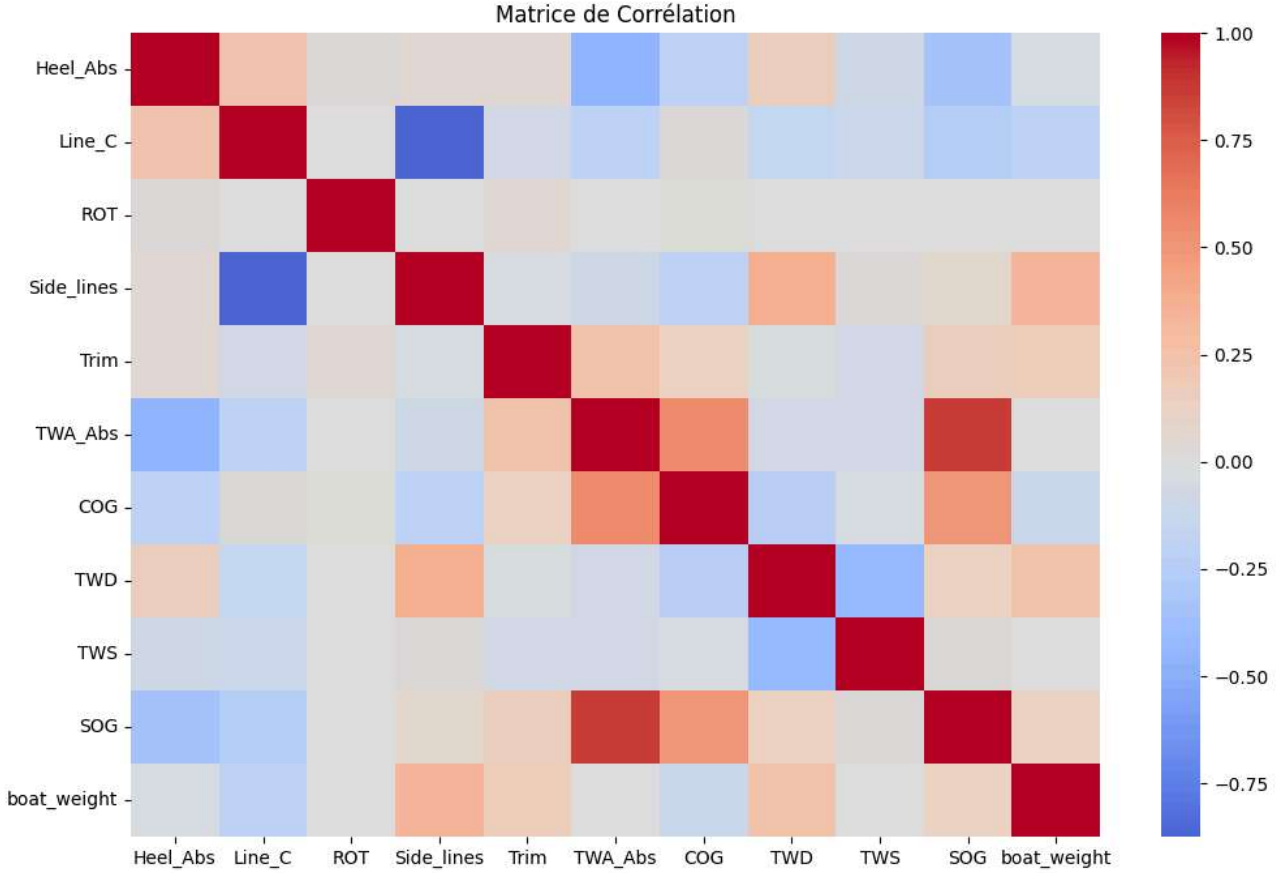
In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from statsmodels.stats.outliers_influence import variance_inflation_factor

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
import statsmodels.formula.api as smf
import statsmodels.api as sm

import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures, StandardScaler, FunctionTransformer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

import warnings
warnings.filterwarnings("ignore")

import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv("all_data.csv")
```

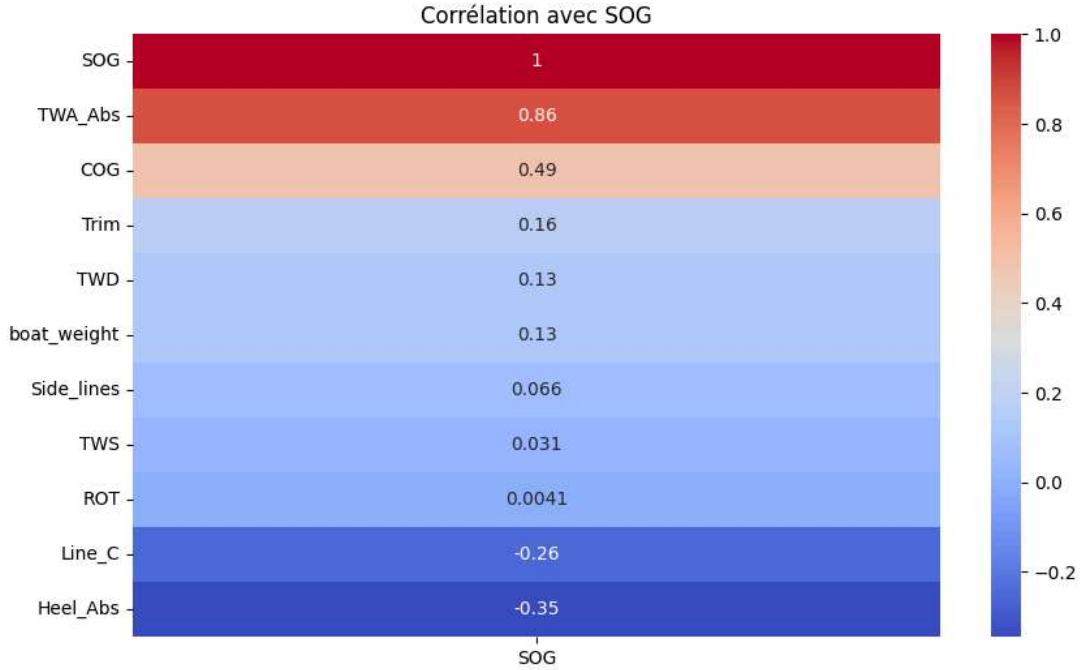
In [2]:
```
def afficher_matrice_correlation(dataframe, taille_figure=(12, 8), cmap="coolwarm"):
    plt.figure(figsize=taille_figure)
    sns.heatmap(dataframe.corr(), cmap=cmap, center=0)
    plt.title("Matrice de Corrélation")
    plt.show()

df_numeric = df.select_dtypes(include=["float64", "int64"]).copy()
drop_cols = [
    "TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE","Total_lines","VMG","gain_forward","gain_lateral", "gain_vmg"
]
df_numeric.drop(columns=[c for c in drop_cols if c in df_numeric.columns], inplace=True)
df_numeric.dropna(subset=["SOG"], inplace=True)
print(f"Variables utilisées:", df_numeric.columns.tolist())
afficher_matrice_correlation(df_numeric)
```

```
Variables utilisées: ['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'COG', 'TWD', 'TWS', 'SOG', 'boat_weight']
```



In [3]:
```python
def afficher_correlation_vmg(df, variable="SOG", taille_figure=(10, 6)):
    corr_with_vmg = df.corr()[variable].sort_values(ascending=False)
    print(f"Corrélation avec {variable} :")
    print(corr_with_vmg)

    plt.figure(figsize=taille_figure)
    sns.heatmap(corr_with_vmg.to_frame(), annot=True, cmap="coolwarm")
    plt.title(f"Corrélation avec {variable}")
    plt.show()

afficher_correlation_vmg(df_numeric, variable="SOG")
```

```
Corrélation avec SOG :
SOG            1.000000
TWA_Abs        0.862650
COG            0.489489
Trim           0.160931
TWD            0.129801
boat_weight    0.128466
Side_lines     0.065668
TWS            0.031173
ROT            0.004052
Line_C        -0.255391
Heel_Abs      -0.345028
Name: SOG, dtype: float64
```



In [4]:
```python
def calculer_anova_quadratique(df, target="SOG"):
    # Make a copy of the DataFrame to avoid modifying the original
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)

    # Print missing value count before imputation
    print("Missing values before imputation:\n", df_copy[cols].isna().sum())

    # --- 1. ANOVA avec imputation ---
    df_impute = df_copy.copy()
    df_impute[cols] = df_impute[cols].apply(lambda x: x.fillna(x.mean()))
    formula_impute = f"{target} ~ " + " + ".join(
        list(cols) +
        [f"I({col}**2)" for col in cols] +
        [f"I({col}**(1/2))" for col in cols]
    )
    model_impute = smf.ols(formula=formula_impute, data=df_impute).fit()
```

```
    anova_impute = sm.stats.anova_lm(model_impute, typ=2)

    # --- 2. ANOVA sans valeurs manquantes ---
    df_no_missing = df_copy.dropna(subset=cols)
    formula_no_missing = f"{target} ~ " + " + ".join(
        list(df_no_missing.columns.drop(target)) +
        [f"I({col}**2)" for col in df_no_missing.columns.drop(target)] +
        [f"I({col}**(1/2))" for col in df_no_missing.columns.drop(target)]
    )
    model_no_missing = smf.ols(formula=formula_no_missing, data=df_no_missing).fit()
    anova_no_missing = sm.stats.anova_lm(model_no_missing, typ=2)

    # --- 3. ANOVA sans les colonnes LineC et SideLines ---
    df_no_linec_sidelines = df_copy.drop(columns=["Line_C", "Side_lines"])
    formula_no_linec_sidelines = f"{target} ~ " + " + ".join(
        list(df_no_linec_sidelines.columns.drop(target)) +
        [f"I({col}**2)" for col in df_no_linec_sidelines.columns.drop(target)] +
        [f"I({col}**(1/2))" for col in df_no_linec_sidelines.columns.drop(target)]
    )
    model_no_linec_sidelines = smf.ols(formula=formula_no_linec_sidelines, data=df_no_linec_sidelines).fit()
    anova_no_linec_sidelines = sm.stats.anova_lm(model_no_linec_sidelines, typ=2)

    return {
        "anova_impute": anova_impute.sort_values("F", ascending=False),
        "anova_no_missing": anova_no_missing.sort_values("F", ascending=False),
        "anova_no_linec_sidelines": anova_no_linec_sidelines.sort_values("F", ascending=False)
    }

resultats_anova = calculer_anova_quadratique(df_numeric)
```

```
Missing values before imputation:
 Heel_Abs          0
Line_C          763
ROT               0
Side_lines     9445
Trim              0
TWA_Abs           0
COG               0
TWD               0
TWS               0
boat_weight       0
dtype: int64
```

In [5]:
```
anova_impute = resultats_anova["anova_impute"]
anova_no_missing = resultats_anova["anova_no_missing"]
anova_no_linec_sidelines = resultats_anova["anova_no_linec_sidelines"]

print("ANOVA avec Imputation:")
display(anova_impute)

print("\nANOVA sans Valeurs Manquantes:")
display(anova_no_missing)

print("\nANOVA sans Line_C et Side_lines:")
display(anova_no_linec_sidelines)
```

```
ANOVA avec Imputation:
```

	sum_sq	df	F	PR(>F)
I(TWA_Abs ** (1 / 2))	1088.761102	1.0	1382.226999	1.686478e-297
TWA_Abs	1081.785978	1.0	1373.371792	1.223524e-295
I(Side_lines ** (1 / 2))	923.814589	1.0	1172.820617	2.166319e-253
Side_lines	795.733582	1.0	1010.216510	5.555243e-219
I(boat_weight ** 2)	680.762482	1.0	864.255971	5.649139e-188
boat_weight	669.549502	1.0	850.020631	6.057039e-185
I(boat_weight ** (1 / 2))	663.787861	1.0	842.705991	2.186551e-183
I(TWA_Abs ** 2)	636.698332	1.0	808.314719	4.640805e-176
I(Side_lines ** 2)	550.984068	1.0	699.496936	7.810476e-153
I(COG ** 2)	148.121549	1.0	188.046399	1.058772e-42
I(Line_C ** 2)	61.553448	1.0	78.144633	9.955580e-19
TWS	47.677951	1.0	60.529120	7.421912e-15
I(TWS ** (1 / 2))	47.299985	1.0	60.049277	9.467372e-15
I(TWD ** 2)	36.699165	1.0	46.591099	8.869273e-12
I(TWS ** 2)	35.989243	1.0	45.689824	1.404248e-11
I(Heel_Abs ** 2)	29.943714	1.0	38.014776	7.087612e-10
TWD	26.036774	1.0	33.054755	9.024555e-09
I(TWD ** (1 / 2))	20.051148	1.0	25.455757	4.546097e-07
Line_C	18.524706	1.0	23.517876	1.242196e-06
Heel_Abs	14.759847	1.0	18.738233	1.503071e-05
COG	9.473944	1.0	12.027561	5.247433e-04
I(COG ** (1 / 2))	9.321881	1.0	11.834511	5.820185e-04
I(Heel_Abs ** (1 / 2))	7.427948	1.0	9.430086	2.135965e-03
I(Trim ** 2)	3.182033	1.0	4.039721	4.444782e-02
I(ROT ** 2)	1.407513	1.0	1.786896	1.813123e-01
I(ROT ** (1 / 2))	1.351290	1.0	1.715518	1.902784e-01
ROT	1.159531	1.0	1.472072	2.250262e-01
I(Line_C ** (1 / 2))	0.406640	1.0	0.516246	4.724515e-01
Trim	0.384540	1.0	0.488190	4.847400e-01
I(Trim ** (1 / 2))	0.185713	1.0	0.235771	6.272801e-01
Residual	31727.211565	40279.0	NaN	NaN

```
ANOVA sans Valeurs Manquantes:
```

	sum_sq	df	F	PR(>F)
I(TWA_Abs ** (1 / 2))	967.813715	1.0	1224.207534	1.115842e-263
TWA_Abs	961.911051	1.0	1216.741132	4.129429e-262
I(boat_weight ** 2)	752.094680	1.0	951.340076	3.785083e-206
boat_weight	741.707856	1.0	938.201568	2.288057e-203
I(boat_weight ** (1 / 2))	736.386933	1.0	931.471022	6.091373e-202
I(TWA_Abs ** 2)	548.938788	1.0	694.363998	1.488105e-151
I(Side_lines ** (1 / 2))	167.620254	1.0	212.026317	6.841096e-48
I(COG ** 2)	141.124874	1.0	178.511764	1.286389e-40
Side_lines	86.192996	1.0	109.027299	1.743964e-25
I(Side_lines ** 2)	80.636260	1.0	101.998469	5.990110e-24
I(TWD ** 2)	33.104264	1.0	41.874266	9.861137e-11
I(TWS ** (1 / 2))	30.542988	1.0	38.634456	5.168067e-10
TWS	30.250910	1.0	38.265000	6.243881e-10
I(Heel_Abs ** 2)	26.519510	1.0	33.545075	7.022253e-09
TWD	24.503356	1.0	30.994800	2.606030e-08
I(TWS ** 2)	20.695451	1.0	26.178103	3.129648e-07
I(TWD ** (1 / 2))	19.461351	1.0	24.617064	7.025525e-07
COG	18.001930	1.0	22.771013	1.832286e-06
Heel_Abs	14.095394	1.0	17.829555	2.422069e-05
I(Heel_Abs ** (1 / 2))	7.873447	1.0	9.959285	1.601749e-03
I(Line_C ** 2)	4.786223	1.0	6.054192	1.387809e-02
I(COG ** (1 / 2))	2.839280	1.0	3.591464	5.808525e-02
I(ROT ** 2)	2.317089	1.0	2.930934	8.690600e-02
I(Trim ** 2)	2.249081	1.0	2.844909	9.167267e-02
Line_C	0.671686	1.0	0.849629	3.566635e-01
I(Trim ** (1 / 2))	0.510684	1.0	0.645974	4.215611e-01
I(Line_C ** (1 / 2))	0.445116	1.0	0.563036	4.530446e-01
I(ROT ** (1 / 2))	0.444994	1.0	0.562882	4.531065e-01
ROT	0.152693	1.0	0.193145	6.603145e-01
Trim	0.108263	1.0	0.136944	7.113408e-01
Residual	27842.853678	35219.0	NaN	NaN

ANOVA sans Line_C et Side_lines:

	sum_sq	df	F	PR(>F)
I(TWA_Abs ** (1 / 2))	1360.880622	1.0	1633.160576	0.000000e+00
TWA_Abs	1272.621467	1.0	1527.242857	0.000000e+00
I(TWA_Abs ** 2)	687.304609	1.0	824.817969	1.411261e-179
I(COG ** 2)	150.594718	1.0	180.725152	4.128824e-41
I(boat_weight ** 2)	146.480537	1.0	175.787821	4.888255e-40
boat_weight	140.997918	1.0	169.208259	1.318107e-38
I(boat_weight ** (1 / 2))	138.188048	1.0	165.836201	7.135984e-38
TWS	72.353226	1.0	86.829392	1.240892e-20
I(TWS ** (1 / 2))	68.890342	1.0	82.673667	1.010726e-19
I(TWS ** 2)	61.661478	1.0	73.998478	8.094993e-18
I(TWD ** 2)	39.215405	1.0	47.061478	6.978628e-12
TWD	26.920686	1.0	32.306877	1.325541e-08
I(Heel_Abs ** 2)	23.615328	1.0	28.340195	1.023046e-07
I(TWD ** (1 / 2))	20.110955	1.0	24.134680	9.017966e-07
I(COG ** (1 / 2))	12.415180	1.0	14.899163	1.135901e-04
Heel_Abs	11.025779	1.0	13.231776	2.755862e-04
COG	7.831294	1.0	9.398150	2.173482e-03
I(Heel_Abs ** (1 / 2))	4.797168	1.0	5.756968	1.642790e-02
I(Trim ** 2)	4.316575	1.0	5.180219	2.285067e-02
ROT	2.751252	1.0	3.301713	6.921511e-02
I(ROT ** (1 / 2))	2.506403	1.0	3.007875	8.286850e-02
I(ROT ** 2)	1.353719	1.0	1.624567	2.024636e-01
Trim	0.593713	1.0	0.712500	3.986200e-01
I(Trim ** (1 / 2))	0.004963	1.0	0.005956	9.384872e-01
Residual	33568.699020	40285.0	NaN	NaN

In [6]:
```python
import numpy as np
import pandas as pd
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.impute import SimpleImputer

def analyser_modele_polynomial_final(df, target="SOG", degree=2, top_coefs=30):
    """
    Version finale qui gère:
    - Les NaN via imputation
    - Les racines carrées sans redondance
    - Les noms de variables clairs
    """
```

```
    cols = df.columns.drop(target)

    X = df[cols].copy()
    X_sqrt = X.apply(lambda x: np.sqrt(np.abs(x))).add_prefix('sqrt_')
    X_combined = pd.concat([X, X_sqrt], axis=1)
    y = df[target].copy()

    imputer = SimpleImputer(strategy='mean')
    X_imputed = pd.DataFrame(imputer.fit_transform(X_combined),
                             columns=X_combined.columns)

    if y.isna().any():
        y = y.fillna(y.mean())

    model = make_pipeline(
        StandardScaler(),
        LinearRegression()
    )

    poly = PolynomialFeatures(degree=degree, include_bias=False)
    X_poly = poly.fit_transform(X_imputed)
    feature_names = poly.get_feature_names_out(X_combined.columns)

    model.fit(X_poly, y)
    y_pred = model.predict(X_poly)
    print(f"R²: {r2_score(y, y_pred):.3f}")
    print(f"Échantillons utilisés: {len(X_imputed)}")

    coef_df = pd.DataFrame({
        'feature': feature_names,
        'coefficient': model.named_steps['linearregression'].coef_
    }).sort_values('coefficient', key=abs, ascending=False)

    def is_valid_feature(name):
        if 'sqrt_' in name:
            base_var = name.split('sqrt_')[-1].split(' ')[0].replace('^2','')
            if f'sqrt_{base_var}^2' in name:
                return False  # Élimine sqrt_x^2
        return True

    valid_coefs = coef_df[coef_df['feature'].apply(is_valid_feature)].head(top_coefs)

    return valid_coefs

resultats = analyser_modele_polynomial_final(df_numeric)
display(resultats)
```

```
R²: 0.896
Échantillons utilisés: 80131
```

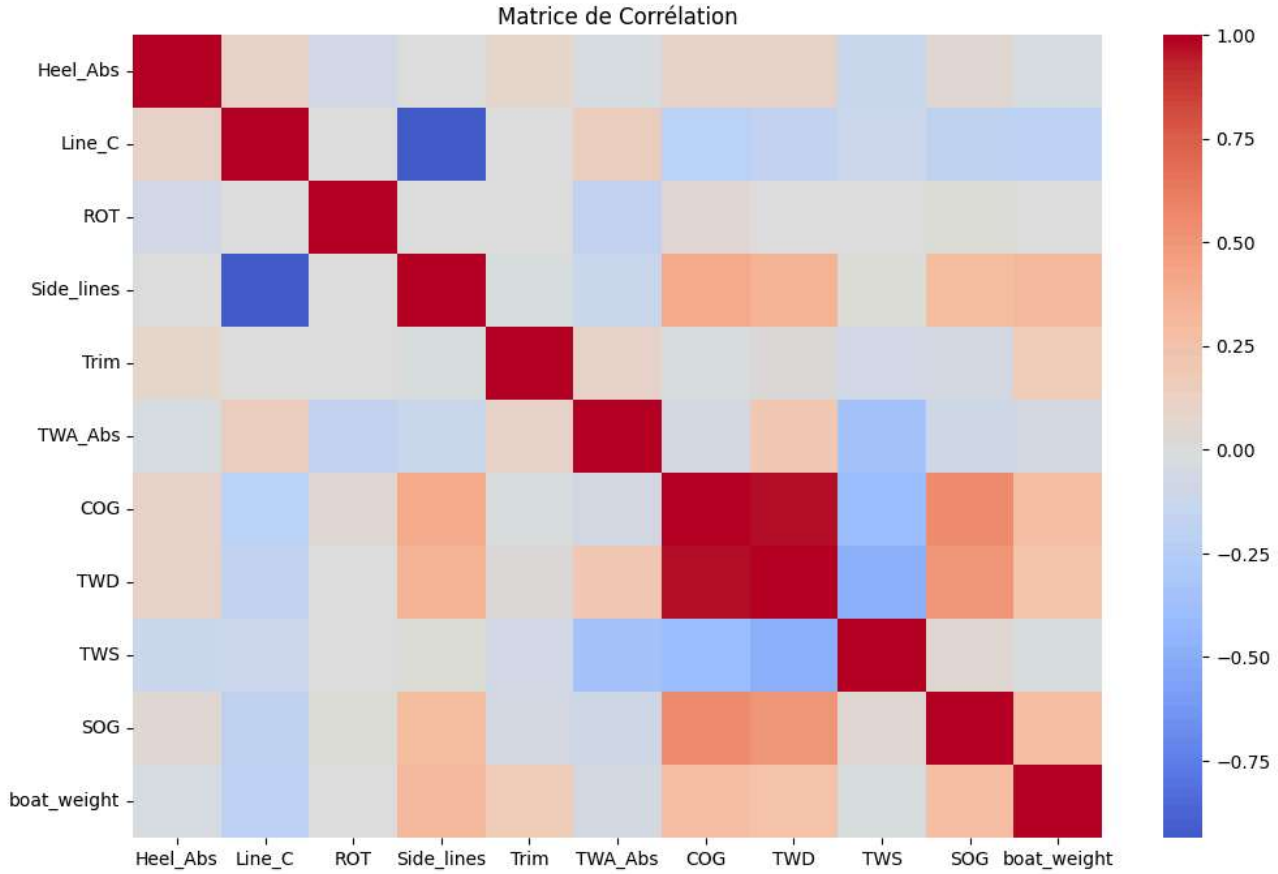
	feature	coefficient
7	TWD	-4.685243e+10
0	Heel_Abs	-5.969228e+09
8	TWS	2.595316e+09
6	COG	1.846250e+09
5	TWA_Abs	1.175460e+09
9	boat_weight	-4.862469e+08
174	boat_weight sqrt_boat_weight	2.365587e+05
164	boat_weight^2	-8.072078e+04
226	sqrt_TWD sqrt_boat_weight	7.540666e+04
151	TWD sqrt_boat_weight	-7.203744e+04
19	sqrt_boat_weight	6.429868e+04
172	boat_weight sqrt_TWD	-4.460727e+04
141	TWD boat_weight	3.893620e+04
17	sqrt_TWD	-3.662378e+04
149	TWD sqrt_TWD	-1.085203e+04
124	TWA_Abs sqrt_boat_weight	-8.670318e+03
223	sqrt_COG sqrt_boat_weight	-8.411285e+03
138	COG sqrt_boat_weight	8.347097e+03
228	sqrt_TWS sqrt_boat_weight	7.805033e+03
93	Side_lines sqrt_boat_weight	7.679783e+03
163	TWS sqrt_boat_weight	-7.667451e+03
219	sqrt_TWA_Abs sqrt_boat_weight	7.053154e+03
208	sqrt_Side_lines sqrt_boat_weight	-6.396072e+03
16	sqrt_COG	6.136401e+03
114	TWA_Abs boat_weight	4.323421e+03
171	boat_weight sqrt_COG	4.188412e+03
173	boat_weight sqrt_TWS	-4.175905e+03
221	sqrt_COG sqrt_TWD	-4.172653e+03
128	COG boat_weight	-4.149939e+03
3	Side_lines	-4.090423e+03

Upwind:

In [7]:
```
# Same analysis, but for downwind and upwind separately
upwind_data = df[df['TWA'] >= 0]
```

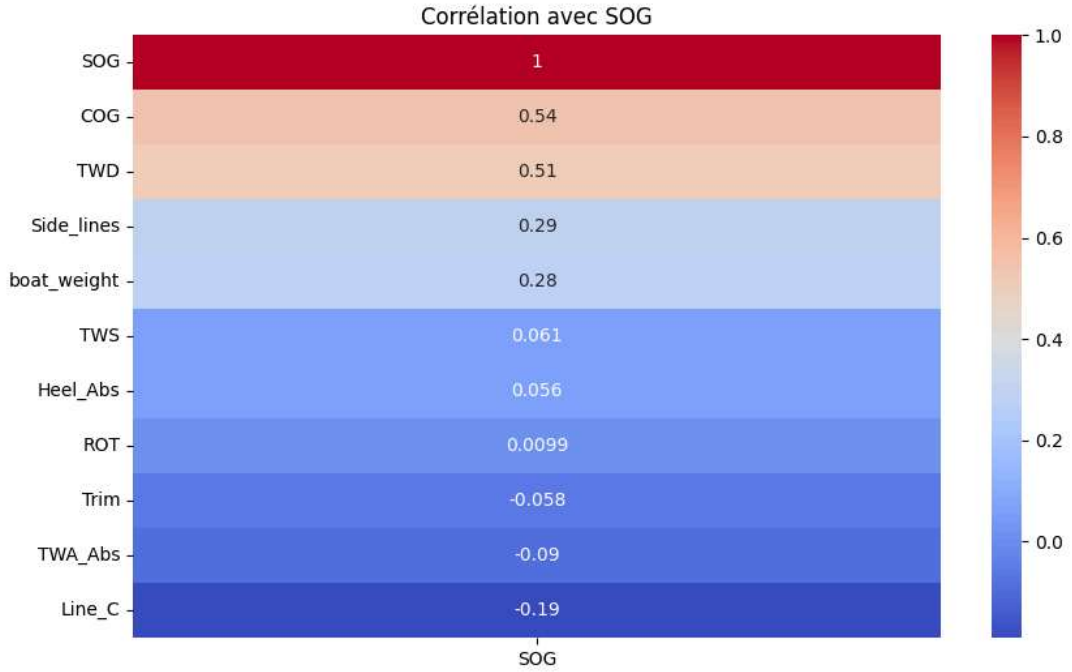
```
# Create a numeric-only version of upwind_data (not df)
df_numeric_upwind = upwind_data.select_dtypes(include=["float64", "int64"]).copy()

"""
drop_cols = [
    "TWA_Abs", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel_Abs", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE"
]
"""
drop_cols = [
    "TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE","Total_lines","VMG","gain_forward","gain_lateral", "gain_vmg"
]
df_numeric_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_upwind.columns], inplace=True)
df_numeric_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Variables utilisées:", df_numeric_upwind.columns.tolist())

afficher_matrice_correlation(df_numeric_upwind)
```

```
Variables utilisées: ['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'COG', 'TWD', 'TWS', 'SOG', 'boat_weight']
```



In [8]:
```
afficher_correlation_vmg(df_numeric_upwind)
```

```
Corrélation avec SOG :
SOG            1.000000
COG            0.542739
TWD            0.510248
Side_lines     0.289328
boat_weight    0.280603
TWS            0.061336
Heel_Abs       0.056305
ROT            0.009906
Trim          -0.058428
TWA_Abs       -0.089706
Line_C        -0.190149
Name: SOG, dtype: float64
```



In [9]:
```
resultats_anova_upwind = calculer_anova_quadratique(df_numeric_upwind)

anova_impute_upwind = resultats_anova_upwind["anova_impute"]
anova_no_missing_upwind = resultats_anova_upwind["anova_no_missing"]
anova_no_linec_sidelines_upwind = resultats_anova_upwind["anova_no_linec_sidelines"]

print("ANOVA avec Imputation:")
display(anova_impute_upwind)

print("\nANOVA sans Valeurs Manquantes:")
display(anova_no_missing_upwind)

print("\nANOVA sans Line_C et Side_lines:")
display(anova_no_linec_sidelines_upwind)
```

```
Missing values before imputation:
 Heel_Abs          0
Line_C           105
ROT                0
Side_lines      5601
Trim               0
TWA_Abs            0
COG                0
TWD                0
TWS                0
boat_weight        0
dtype: int64
ANOVA avec Imputation:
```

	sum_sq	df	F	PR(>F)
I(COG ** (1 / 2))	335.325845	1.0	572.381846	4.617708e-125
I(COG ** 2)	223.252799	1.0	381.079631	3.159185e-84
Side_lines	205.410671	1.0	350.624149	1.079989e-77
I(TWA_Abs ** 2)	190.413997	1.0	325.025693	3.409321e-72
I(Side_lines ** (1 / 2))	186.463817	1.0	318.282963	9.601607e-71
I(Side_lines ** 2)	165.162123	1.0	281.922204	6.432607e-63
I(TWA_Abs ** (1 / 2))	160.143064	1.0	273.354958	4.513307e-61
I(boat_weight ** 2)	101.881105	1.0	173.905161	1.416067e-39
boat_weight	99.774203	1.0	170.308801	8.531709e-39
I(boat_weight ** (1 / 2))	98.690340	1.0	168.458709	2.149568e-38
I(TWD ** (1 / 2))	90.153996	1.0	153.887664	3.126334e-35
I(TWD ** 2)	81.102638	1.0	138.437518	7.122792e-32
I(Line_C ** 2)	37.546230	1.0	64.089246	1.241450e-15
I(TWS ** 2)	24.359035	1.0	41.579466	1.152823e-10
TWS	17.242337	1.0	29.431674	5.847109e-08
I(TWS ** (1 / 2))	16.353819	1.0	27.915023	1.278436e-07
Line_C	9.469168	1.0	16.163322	5.828211e-05
I(ROT ** 2)	3.276896	1.0	5.593473	1.803526e-02
Trim	3.111803	1.0	5.311668	2.119137e-02
TWD	1.810939	1.0	3.091168	7.873074e-02
I(Trim ** (1 / 2))	1.806518	1.0	3.083623	7.909668e-02
COG	1.803677	1.0	3.078772	7.933288e-02
TWA_Abs	1.663262	1.0	2.839092	9.200893e-02
I(Heel_Abs ** 2)	1.438177	1.0	2.454885	1.171727e-01
I(Trim ** 2)	1.135768	1.0	1.938690	1.638235e-01
Heel_Abs	0.969689	1.0	1.655202	1.982645e-01
I(Heel_Abs ** (1 / 2))	0.766685	1.0	1.308688	2.526436e-01
I(ROT ** (1 / 2))	0.325259	1.0	0.555198	4.562087e-01
I(Line_C ** (1 / 2))	0.002679	1.0	0.004573	9.460839e-01
ROT	0.000770	1.0	0.001314	9.710823e-01
Residual	14367.797306	24525.0	NaN	NaN

```
ANOVA sans Valeurs Manquantes:
```

	sum_sq	df	F	PR(>F)
I(COG ** (1 / 2))	217.814307	1.0	371.748282	3.795121e-82
I(boat_weight ** 2)	188.059418	1.0	320.964984	2.914503e-71
boat_weight	185.417566	1.0	316.456078	2.707014e-70
I(boat_weight ** (1 / 2))	184.060435	1.0	314.139835	8.507692e-70
I(TWA_Abs ** 2)	167.732921	1.0	286.273322	8.281322e-64
I(TWA_Abs ** (1 / 2))	141.235115	1.0	241.048956	4.540605e-54
I(COG ** 2)	137.160806	1.0	234.095247	1.435381e-52
I(Side_lines ** 2)	107.257289	1.0	183.058282	1.535806e-41
Side_lines	90.991306	1.0	155.296786	1.596016e-35
I(Side_lines ** (1 / 2))	87.369513	1.0	149.115395	3.504300e-34
I(TWD ** (1 / 2))	71.634982	1.0	122.260938	2.409637e-28
I(TWD ** 2)	64.238313	1.0	109.636887	1.354566e-25
I(Line_C ** 2)	31.131296	1.0	53.132440	3.224872e-13
I(TWS ** 2)	19.245354	1.0	32.846452	1.010552e-08
TWS	12.172294	1.0	20.774711	5.194256e-06
I(TWS ** (1 / 2))	10.834145	1.0	18.490865	1.714661e-05
Line_C	10.831295	1.0	18.486000	1.719039e-05
I(ROT ** 2)	3.559310	1.0	6.074749	1.372047e-02
Trim	1.559044	1.0	2.660854	1.028601e-01
TWD	1.365161	1.0	2.329950	1.269205e-01
COG	1.321170	1.0	2.254868	1.332094e-01
TWA_Abs	1.247514	1.0	2.129159	1.445345e-01
I(Trim ** 2)	0.627895	1.0	1.071642	3.005864e-01
I(Trim ** (1 / 2))	0.614508	1.0	1.048794	3.057965e-01
I(ROT ** (1 / 2))	0.199050	1.0	0.339723	5.599951e-01
I(Heel_Abs ** 2)	0.110409	1.0	0.188438	6.642240e-01
ROT	0.019686	1.0	0.033599	8.545647e-01
I(Line_C ** (1 / 2))	0.008213	1.0	0.014017	9.057587e-01
I(Heel_Abs ** (1 / 2))	0.005020	1.0	0.008568	9.262511e-01
Heel_Abs	0.004481	1.0	0.007648	9.303103e-01
Residual	12707.992858	21689.0	NaN	NaN

```
ANOVA sans Line_C et Side_lines:
```

	sum_sq	df	F	PR(>F)
I(COG ** (1 / 2))	523.872757	1.0	843.531381	2.259958e-182
I(COG ** 2)	368.594589	1.0	593.505005	1.499875e-129
I(TWA_Abs ** 2)	177.015528	1.0	285.027520	1.378348e-63
I(TWD ** (1 / 2))	171.091455	1.0	275.488675	1.565136e-61
I(TWD ** 2)	152.532144	1.0	245.604774	4.365607e-55
I(TWA_Abs ** (1 / 2))	146.627100	1.0	236.096569	4.932547e-53
I(TWS ** 2)	21.010456	1.0	33.830694	6.086892e-09
I(TWS ** (1 / 2))	17.582091	1.0	28.310397	1.042456e-07
TWS	16.881193	1.0	27.181822	1.866923e-07
I(boat_weight ** 2)	6.274014	1.0	10.102316	1.482669e-03
boat_weight	5.851563	1.0	9.422092	2.146209e-03
I(boat_weight ** (1 / 2))	5.632449	1.0	9.069278	2.602043e-03
Trim	3.473781	1.0	5.593426	1.803574e-02
I(Trim ** (1 / 2))	2.682575	1.0	4.319439	3.768972e-02
I(ROT ** 2)	2.665614	1.0	4.292128	3.829964e-02
I(Heel_Abs ** 2)	2.014493	1.0	3.243704	7.171064e-02
Heel_Abs	1.566381	1.0	2.522162	1.122695e-01
TWD	1.562320	1.0	2.515622	1.127360e-01
COG	1.520791	1.0	2.448754	1.176311e-01
I(Heel_Abs ** (1 / 2))	1.317329	1.0	2.121142	1.452910e-01
TWA_Abs	1.308538	1.0	2.106986	1.466407e-01
I(ROT ** (1 / 2))	1.143234	1.0	1.840816	1.748678e-01
I(Trim ** 2)	1.018665	1.0	1.640237	2.003051e-01
ROT	0.280278	1.0	0.451299	5.017252e-01
Residual	15234.907536	24531.0	NaN	NaN

In [10]:
```
resultats_coefs_upwind = analyser_modele_polynomial_final(df_numeric_upwind)
display(resultats_coefs_upwind.head(30))
```

```
R²: 0.647
Échantillons utilisés: 48787
```

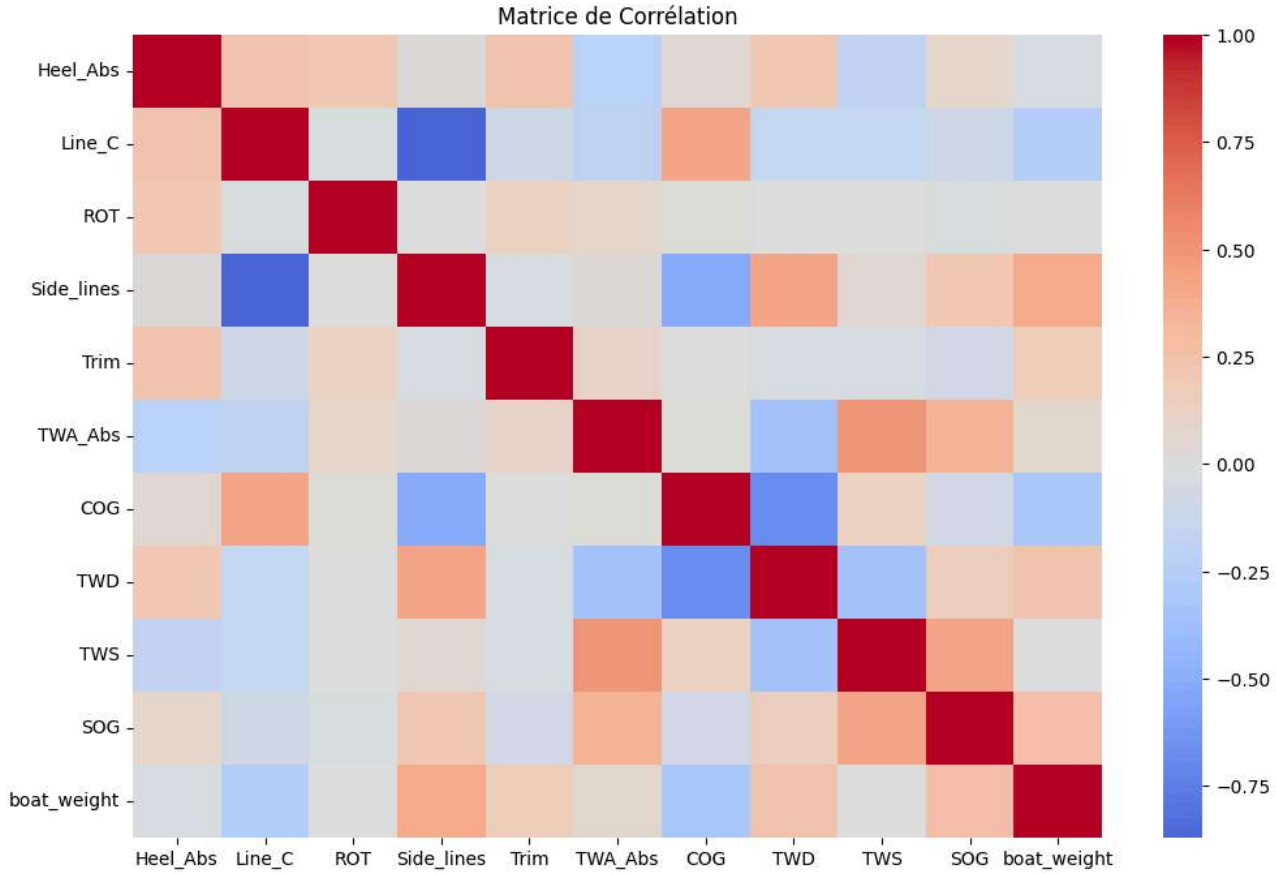
	feature	coefficient
0	Heel_Abs	4.305635e+10
126	COG TWD	-1.835541e+08
139	TWD^2	1.056710e+08
125	COG^2	7.948693e+07
112	TWA_Abs TWD	-3.882476e+07
111	TWA_Abs COG	3.041608e+07
110	TWA_Abs^2	6.311080e+06
151	TWD sqrt_boat_weight	3.559963e+06
138	COG sqrt_boat_weight	-3.514356e+06
141	TWD boat_weight	-1.912187e+06
128	COG boat_weight	1.855349e+06
6	COG	9.033632e+05
7	TWD	-8.917151e+05
148	TWD sqrt_COG	8.891953e+05
124	TWA_Abs sqrt_boat_weight	-8.801822e+05
135	COG sqrt_COG	-7.790435e+05
149	TWD sqrt_TWD	-5.879274e+05
174	boat_weight sqrt_boat_weight	5.784441e+05
136	COG sqrt_TWD	5.093671e+05
114	TWA_Abs boat_weight	4.513818e+05
9	boat_weight	-2.804128e+05
5	TWA_Abs	2.338718e+05
164	boat_weight^2	-1.951727e+05
19	sqrt_boat_weight	1.781001e+05
121	TWA_Abs sqrt_COG	-1.572463e+05
122	TWA_Abs sqrt_TWD	1.170039e+05
150	TWD sqrt_TWS	9.401537e+04
137	COG sqrt_TWS	-9.324541e+04
147	TWD sqrt_TWA_Abs	-7.552248e+04
226	sqrt_TWD sqrt_boat_weight	6.780827e+04

Downwind

In [11]:
```
# Same analysis, but for downwind and upwind separately
downwind_data = df[df['TWA'] <= 0]
```

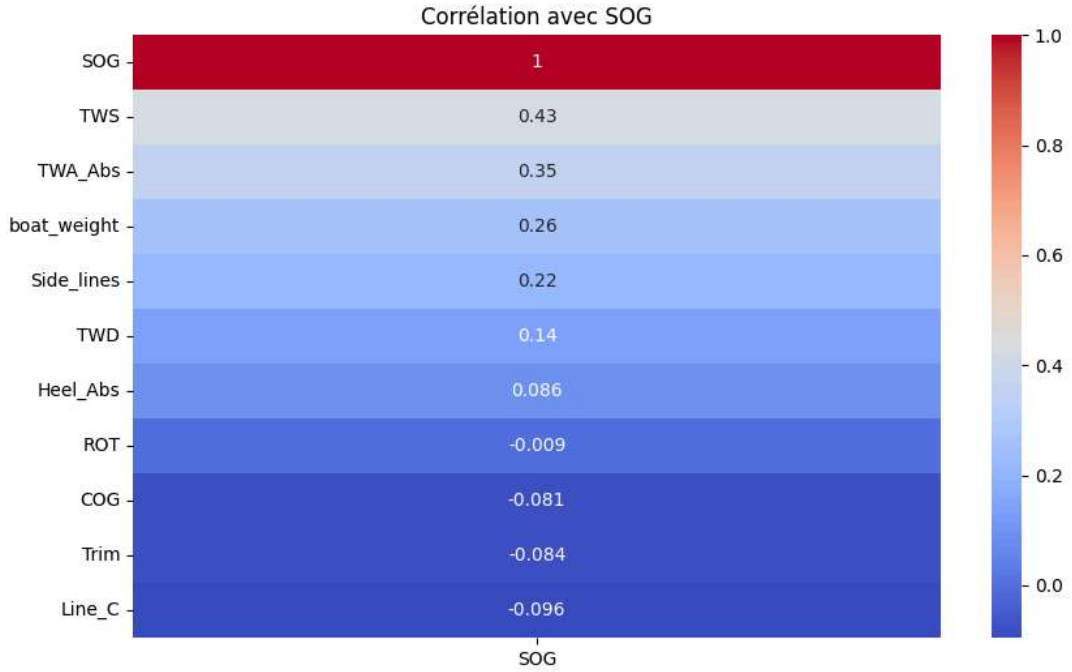
```
# Create a numeric-only version of upwind_data (not df)
df_numeric_downwind = downwind_data.select_dtypes(include=["float64", "int64"]).copy()

"""
drop_cols = [
    "TWA_Abs", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel_Abs", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE"
]
"""
drop_cols = [
    "TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE","Total_lines","VMG","gain_forward","gain_lateral", "gain_vmg"
]
df_numeric_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_downwind.columns], inplace=True)
df_numeric_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Variables utilisées:", df_numeric_downwind.columns.tolist())

afficher_matrice_correlation(df_numeric_downwind)
```

```
Variables utilisées: ['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'COG', 'TWD', 'TWS', 'SOG', 'boat_weight']
```



In [12]:
```
afficher_correlation_vmg(df_numeric_downwind)
```

```
Corrélation avec SOG :
SOG            1.000000
TWS            0.433601
TWA_Abs        0.354902
boat_weight    0.255130
Side_lines     0.215051
TWD            0.140766
Heel_Abs       0.085725
ROT           -0.009001
COG           -0.081012
Trim          -0.083675
Line_C        -0.095684
Name: SOG, dtype: float64
```



In [13]:
```
resultats_anova_downwind = calculer_anova_quadratique(df_numeric_downwind)

anova_impute_downwind = resultats_anova_downwind["anova_impute"]
anova_no_missing_downwind = resultats_anova_downwind["anova_no_missing"]
anova_no_linec_sidelines_downwind = resultats_anova_downwind["anova_no_linec_sidelines"]

print("ANOVA avec Imputation:")
display(anova_impute_downwind)

print("\nANOVA sans Valeurs Manquantes:")
display(anova_no_missing_downwind)

print("\nANOVA sans Line_C et Side_lines:")
display(anova_no_linec_sidelines_downwind)
```

```
Missing values before imputation:
 Heel_Abs          0
Line_C          658
ROT               0
Side_lines     3844
Trim              0
TWA_Abs           0
COG               0
TWD               0
TWS               0
boat_weight       0
dtype: int64
ANOVA avec Imputation:
```

	sum_sq	df	F	PR(>F)
I(boat_weight ** 2)	280.108727	1.0	320.043435	7.139373e-71
boat_weight	276.748166	1.0	316.203764	4.714674e-70
I(boat_weight ** (1 / 2))	274.964706	1.0	314.166039	1.284132e-69
Side_lines	159.502570	1.0	182.242628	2.666022e-41
I(Side_lines ** (1 / 2))	158.028118	1.0	180.557965	6.158616e-41
I(Side_lines ** 2)	148.215567	1.0	169.346452	1.624999e-38
I(Heel_Abs ** 2)	20.962308	1.0	23.950875	9.980473e-07
I(TWS ** 2)	15.316157	1.0	17.499760	2.889031e-05
Heel_Abs	15.208003	1.0	17.376187	3.082854e-05
I(TWD ** 2)	15.130449	1.0	17.287576	3.229838e-05
TWS	13.916168	1.0	15.900177	6.707443e-05
I(Line_C ** 2)	13.010492	1.0	14.865379	1.159225e-04
TWD	12.133361	1.0	13.863197	1.972975e-04
I(TWD ** (1 / 2))	11.342149	1.0	12.959183	3.193352e-04
I(TWS ** (1 / 2))	10.902753	1.0	12.457143	4.175859e-04
I(Heel_Abs ** (1 / 2))	10.418334	1.0	11.903661	5.617075e-04
I(COG ** (1 / 2))	5.625262	1.0	6.427248	1.124786e-02
I(Trim ** (1 / 2))	5.612105	1.0	6.412215	1.134344e-02
COG	3.176865	1.0	3.629786	5.677235e-02
Line_C	3.116111	1.0	3.560370	5.919295e-02
I(TWA_Abs ** 2)	2.942127	1.0	3.361582	6.675283e-02
ROT	1.639721	1.0	1.873493	1.710951e-01
I(Trim ** 2)	1.323313	1.0	1.511975	2.188563e-01
TWA_Abs	1.180188	1.0	1.348446	2.455676e-01
I(COG ** 2)	1.090013	1.0	1.245415	2.644471e-01
I(ROT ** (1 / 2))	0.658359	1.0	0.752221	3.857873e-01
Trim	0.557697	1.0	0.637207	4.247359e-01
I(TWA_Abs ** (1 / 2))	0.487276	1.0	0.556746	4.555854e-01
I(Line_C ** (1 / 2))	0.006269	1.0	0.007163	9.325524e-01
I(ROT ** 2)	0.001765	1.0	0.002016	9.641851e-01
Residual	13761.099348	15723.0	NaN	NaN

```
ANOVA sans Valeurs Manquantes:
```

	sum_sq	df	F	PR(>F)
I(boat_weight ** 2)	100.640081	1.0	112.294438	3.901259e-26
boat_weight	98.650478	1.0	110.074434	1.184568e-25
I(boat_weight ** (1 / 2))	97.618690	1.0	108.923162	2.107506e-25
I(Side_lines ** (1 / 2))	39.114157	1.0	43.643668	4.087165e-11
I(Heel_Abs ** 2)	30.847306	1.0	34.419497	4.546605e-09
Heel_Abs	23.660017	1.0	26.399902	2.814264e-07
I(Side_lines ** 2)	22.300448	1.0	24.882892	6.167828e-07
Side_lines	21.766549	1.0	24.287166	8.397493e-07
I(Heel_Abs ** (1 / 2))	17.797137	1.0	19.858087	8.408060e-06
I(TWD ** 2)	16.649969	1.0	18.578074	1.642412e-05
I(TWS ** 2)	13.920480	1.0	15.532504	8.150749e-05
TWD	12.971683	1.0	14.473835	1.427469e-04
TWS	12.343125	1.0	13.772488	2.071647e-04
I(TWD ** (1 / 2))	11.645643	1.0	12.994236	3.135767e-04
I(TWS ** (1 / 2))	9.449340	1.0	10.543596	1.168759e-03
I(Trim ** (1 / 2))	5.020495	1.0	5.601880	1.795524e-02
I(TWA_Abs ** 2)	3.930117	1.0	4.385234	3.627012e-02
I(COG ** 2)	1.681475	1.0	1.876194	1.707901e-01
TWA_Abs	1.679519	1.0	1.874012	1.710391e-01
I(Trim ** 2)	1.516360	1.0	1.691958	1.933652e-01
I(TWA_Abs ** (1 / 2))	0.816635	1.0	0.911204	3.398123e-01
ROT	0.809945	1.0	0.903738	3.417983e-01
Trim	0.756748	1.0	0.844381	3.581631e-01
I(ROT ** (1 / 2))	0.466956	1.0	0.521031	4.704147e-01
COG	0.240387	1.0	0.268224	6.045338e-01
Line_C	0.145111	1.0	0.161915	6.874057e-01
I(ROT ** 2)	0.106922	1.0	0.119304	7.297957e-01
I(Line_C ** 2)	0.085625	1.0	0.095540	7.572534e-01
I(COG ** (1 / 2))	0.036715	1.0	0.040966	8.396058e-01
I(Line_C ** (1 / 2))	0.031428	1.0	0.035067	8.514574e-01
Residual	12098.020925	13499.0	NaN	NaN

```
ANOVA sans Line_C et Side_lines:
```

	sum_sq	df	F	PR(>F)
I(boat_weight ** 2)	245.543640	1.0	273.834764	5.427237e-61
boat_weight	242.423395	1.0	270.355009	3.020360e-60
I(boat_weight ** (1 / 2))	240.710749	1.0	268.445036	7.750373e-60
I(TWD ** 2)	62.742853	1.0	69.971979	6.514948e-17
TWD	54.815449	1.0	61.131193	5.676352e-15
I(TWD ** (1 / 2))	52.816784	1.0	58.902244	1.753984e-14
I(Heel_Abs ** 2)	19.076756	1.0	21.274748	4.010761e-06
I(TWS ** 2)	16.147274	1.0	18.007736	2.212692e-05
TWS	14.312658	1.0	15.961738	6.493006e-05
Heel_Abs	13.741993	1.0	15.325322	9.087684e-05
I(TWS ** (1 / 2))	11.022658	1.0	12.292670	4.560076e-04
I(COG ** (1 / 2))	9.662462	1.0	10.775755	1.030625e-03
I(Heel_Abs ** (1 / 2))	8.850082	1.0	9.869773	1.683278e-03
COG	5.612489	1.0	6.259151	1.236541e-02
I(Trim ** (1 / 2))	5.485594	1.0	6.117635	1.339446e-02
ROT	2.367400	1.0	2.640168	1.042126e-01
I(COG ** 2)	2.227254	1.0	2.483874	1.150387e-01
I(TWA_Abs ** 2)	1.773284	1.0	1.977599	1.596634e-01
I(ROT ** (1 / 2))	0.880122	1.0	0.981528	3.218372e-01
I(Trim ** 2)	0.868935	1.0	0.969052	3.249318e-01
TWA_Abs	0.343490	1.0	0.383066	5.359756e-01
Trim	0.180329	1.0	0.201106	6.538359e-01
I(TWA_Abs ** (1 / 2))	0.017307	1.0	0.019301	8.895089e-01
I(ROT ** 2)	0.000959	1.0	0.001070	9.739078e-01
Residual	14103.964955	15729.0	NaN	NaN

In [14]:
```
resultats_coefs_downwind = analyser_modele_polynomial_final(df_numeric_downwind)
display(resultats_coefs_downwind.head(30))
```

```
R²: 0.630
Échantillons utilisés: 31344
```

	feature	coefficient
5	TWA_Abs	-1.061884e+12
8	TWS	4.414715e+10
0	Heel_Abs	2.172152e+10
7	TWD	-2.104989e+10
9	boat_weight	-1.888932e+09
6	COG	6.275738e+08
221	sqrt_COG sqrt_TWD	-1.577837e+05
136	COG sqrt_TWD	1.169513e+05
16	sqrt_COG	7.914764e+04
148	TWD sqrt_COG	7.626885e+04
126	COG TWD	-6.467599e+04
174	boat_weight sqrt_boat_weight	-6.386537e+04
149	TWD sqrt_TWD	-3.660666e+04
19	sqrt_boat_weight	-3.190030e+04
216	sqrt_TWA_Abs sqrt_COG	3.010459e+04
138	COG sqrt_boat_weight	2.685859e+04
223	sqrt_COG sqrt_boat_weight	-2.116876e+04
164	boat_weight^2	1.568003e+04
121	TWA_Abs sqrt_COG	-1.528699e+04
217	sqrt_TWA_Abs sqrt_TWD	-1.519586e+04
147	TWD sqrt_TWA_Abs	1.510331e+04
134	COG sqrt_TWA_Abs	-1.505921e+04
139	TWD^2	1.499524e+04
163	TWS sqrt_boat_weight	-1.468097e+04
228	sqrt_TWS sqrt_boat_weight	1.455394e+04
128	COG boat_weight	-1.325189e+04
226	sqrt_TWD sqrt_boat_weight	1.168091e+04
219	sqrt_TWA_Abs sqrt_boat_weight	-1.100284e+04
171	boat_weight sqrt_COG	1.035884e+04
151	TWD sqrt_boat_weight	-9.658316e+03

In [15]:
```
import scipy.stats as stats

# Perform a t-test between upwind and downwind data for the "SOG" variable
def t_test_between_upwind_downwind(df_upwind, df_downwind, target="SOG"):
    # Perform t-test for the target variable "SOG" between the two datasets
    t_stat, p_value = stats.ttest_ind(df_upwind[target].dropna(), df_downwind[target].dropna())
```

```
    print(f"T-statistic: {t_stat:.3f}, p-value: {p_value:.3f}")

    # If p-value is less than 0.05, the difference is statistically significant
    if p_value < 0.05:
        print("The difference is statistically significant, keeping data split.")
        return True  # Keep data split
    else:
        print("The difference is not statistically significant, keeping data combined.")
        return False  # Keep data combined

# Call the t-test function
split_data = t_test_between_upwind_downwind(df_numeric_upwind, df_numeric_downwind)

# Based on the result of the t-test, split or combine data
if split_data:
    # If the t-test shows a significant difference, keep data split
    # The rest of your code for analysis on `df_numeric_upwind` and `df_numeric_downwind` continues
    print("Proceeding with separate analyses for upwind and downwind data.")
else:
    # If the t-test does not show a significant difference, combine the data
    df_numeric_combined = pd.concat([df_numeric_upwind, df_numeric_downwind], axis=0)
    print("Proceeding with combined analysis.")
    # You can call your existing functions on the combined data
    resultats_anova_combined = calculer_anova_quadratique(df_numeric_combined)
    resultats_coefs_combined = analyser_modele_polynomial_final(df_numeric_combined)
    display(resultats_anova_combined)
    display(resultats_coefs_combined)
```

```
T-statistic: -482.303, p-value: 0.000
The difference is statistically significant, keeping data split.
Proceeding with separate analyses for upwind and downwind data.
```

In [ ]: