In [1]:
```
%cd ../../
```

```
/home/jan/FMF/masters
```

In [2]:
```
saved = "ml_hep_sim/analysis/results/spur/"
```

Get pipeline

In [3]:
```
import numpy as np
from ml_hep_sim.analysis.spur_pipeline import get_spur_pipeline
```

In [4]:
```
bonly = False
mc_test = False
use_class = True

nu_bs = np.linspace(10 ** 3, 10 ** 5, 40)
alphas = np.linspace(0.01, 0.1, 10)

pipe = get_spur_pipeline(
    nu_bs,
    alphas,
    bin_range=(0.5, 1.1) if use_class else (0.01, 3.0),
    use_classifier=use_class,
    bonly=bonly,
    mc_test=mc_test,
    scale_by_alpha=True,
)

pipe.fit()

res = pipe.pipes[-1]
```

```
100%|| 10/10 [00:05<00:00,  1.93it/s]
100%|| 10/10 [00:05<00:00,  1.93it/s]
100%|| 10/10 [00:00<00:00, 5997.00it/s]
100%|| 10/10 [00:00<00:00, 5930.02it/s]
100%|| 10/10 [00:00<00:00, 369.83it/s]
100%|| 10/10 [00:00<00:00, 381.35it/s]
```

In [5]:
```
if use_class and not mc_test:
    saved += "class_"
elif use_class and mc_test:
    saved += "class_mc_"
elif mc_test:
    saved += "mbb_mc_"
else:
    saved += "mbb_"

if bonly:
    saved += "bonly_"
```

In [6]:
```
saved
```

Out[6]:
```
'ml_hep_sim/analysis/results/spur/class_'
```

Spurious signal

In [7]:
```
import matplotlib.pyplot as plt
from ml_hep_sim.plotting.style import style_setup, set_size
```

```
set_size()
style_setup(seaborn_pallete=True)
```

In [8]:
```
df = res.parsed_results
```

In [9]:
```
df.head()
```

Out[9]:

	alpha	mu	gamma	mu_err	gamma_err	twice_nll
0	0.01	3.479943e-10	[0.9891947434531224, 0.9900694997953484, 0.989...	0.027828	[0.07246696574720801, 0.0697531480807852, 0.07...	0.989195
1	0.02	2.211284e-10	[0.9914166886884178, 0.9922189682683515, 0.992...	0.073547	[0.07246248979599967, 0.06974168129903813, 0.0...	0.991417
2	0.03	4.527631e-03	[0.9925757702536692, 0.9933350436449526, 0.993...	0.057811	[0.07289850602600895, 0.07014624178921336, 0.0...	0.992576
3	0.04	1.608824e-02	[0.9920709680043004, 0.9928799141293395, 0.992...	0.032697	[0.0730032845301038, 0.0702435452851356, 0.072...	0.992071
4	0.05	2.760562e-02	[0.9915925642128317, 0.992438050547947, 0.9922...	0.033039	[0.07310688463503262, 0.07033922521625902, 0.0...	0.991593

In [10]:
```
df["total_B"] = np.repeat(nu_bs, len(alphas))
df["alpha"] = np.tile(alphas, len(nu_bs))
```

In [11]:
```
df.head()
```

Out[11]:

	alpha	mu	gamma	mu_err	gamma_err	twice_nll	total_B
0	0.01	3.479943e-10	[0.9891947434531224, 0.9900694997953484, 0.989...	0.027828	[0.07246696574720801, 0.0697531480807852, 0.07...	0.989195	1000.0
1	0.02	2.211284e-10	[0.9914166886884178, 0.9922189682683515, 0.992...	0.073547	[0.07246248979599967, 0.06974168129903813, 0.0...	0.991417	1000.0
2	0.03	4.527631e-03	[0.9925757702536692, 0.9933350436449526, 0.993...	0.057811	[0.07289850602600895, 0.07014624178921336, 0.0...	0.992576	1000.0
3	0.04	1.608824e-02	[0.9920709680043004, 0.9928799141293395, 0.992...	0.032697	[0.0730032845301038, 0.0702435452851356, 0.072...	0.992071	1000.0
4	0.05	2.760562e-02	[0.9915925642128317, 0.992438050547947, 0.9922...	0.033039	[0.07310688463503262, 0.07033922521625902, 0.0...	0.991593	1000.0

In [12]:
```
sig_fracs = alphas
lumis = nu_bs
```

In [13]:
```
df["spur"] = df["mu"] * df["total_B"] - df["alpha"] * df["total_B"]
df["spur_ratio"] = np.abs(df["spur"] / df["total_B"])
```

In [14]:
```
idx = -1

df[df["alpha"] == sig_fracs[idx]]
```

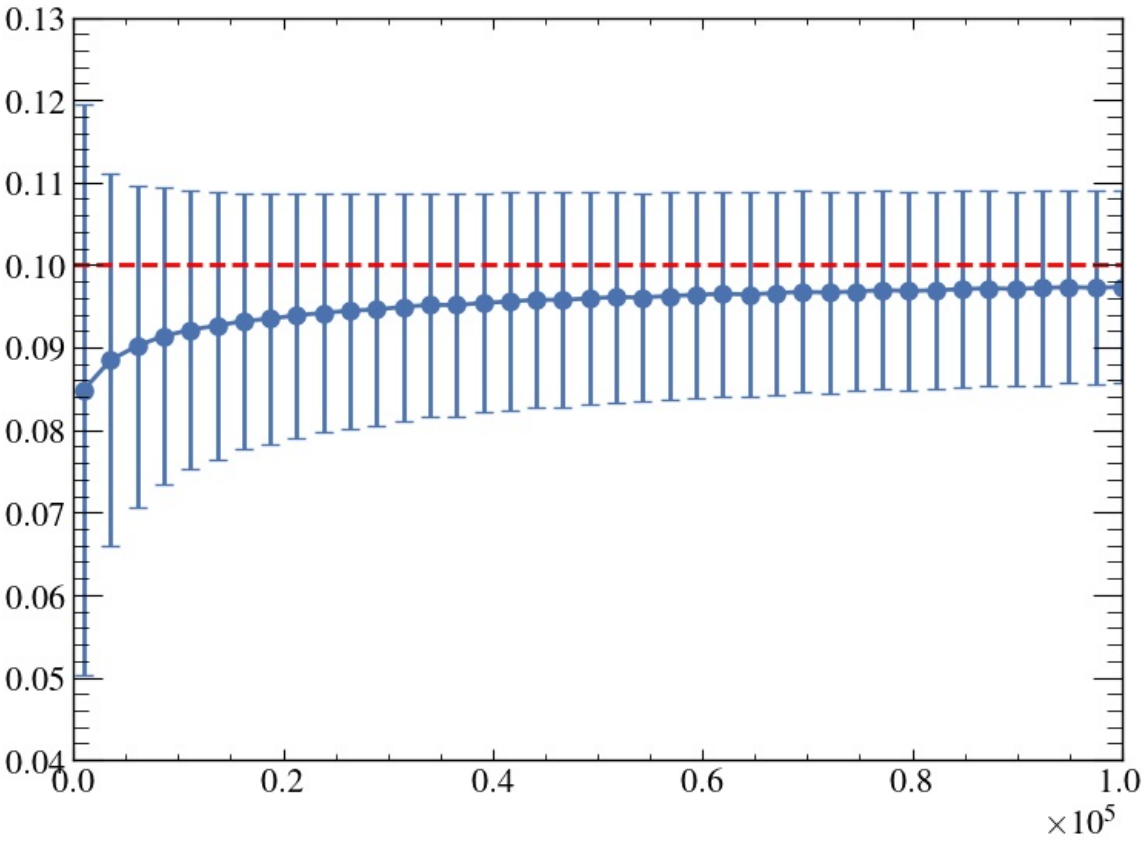
Out[14]:

	alpha	mu	gamma	mu_err	gamma_err	twice_nll	total_B	spur	spur_ratio
9	0.1	0.084877	[0.9892733314358435, 0.9903416132919716, 0.990...	0.034564	[0.07361020524181477, 0.07080299656493089, 0.0...	0.989273	1000.000000	-15.123277	0.015123
19	0.1	0.088462	[0.981487049657194, 0.9813891118236603, 0.9812...	0.022567	[0.04979060754237352, 0.05054835973718125, 0.0...	0.981487	3538.461538	-40.825131	0.011538
29	0.1	0.090186	[0.9784648735048859, 0.9782812311179746, 0.977...	0.019481	[0.040471856999650946, 0.04114162086737483, 0....	0.978465	6076.923077	-59.639438	0.009814
39	0.1	0.091385	[0.9768300147256326, 0.9764966496063827, 0.976...	0.017992	[0.035016303599015475, 0.03569737086262914, 0....	0.976830	8615.384615	-74.222975	0.008615
49	0.1	0.092132	[0.9758295645097146, 0.9754753115812428, 0.975...	0.016902	[0.031329551565266345, 0.0319465702897298, 0.0...	0.975830	11153.846154	-87.757421	0.007868
59	0.1	0.092651	[0.9751602082607805, 0.9747824050930116, 0.974...	0.016187	[0.028642449635762013, 0.02921810392602614, 0....	0.975160	13692.307692	-100.624016	0.007349
69	0.1	0.093193	[0.9746009299906906, 0.9741993395032653, 0.973...	0.015534	[0.026557091561186352, 0.027101052234055067, 0...	0.974601	16230.769231	-110.480388	0.006807
79	0.1	0.093483	[0.9742317640319408, 0.9738150441273611,	0.015243	[0.024916260549012803, 0.025438309407277426,	0.974232	18769.230769	-122.322876	0.006517

			0.973...		0...				
89	0.1	0.093893	[0.973864342766783, 0.9734250288414651, 0.9729...	0.014815	[0.023539635981912832, 0.024041327002521207, 0...	0.973864	21307.692308	-130.133588	0.006107
99	0.1	0.094190	[0.9735923187846299, 0.9731263841567818, 0.972...	0.014509	[0.022382787987029973, 0.02287321733301001, 0....	0.973592	23846.153846	-138.554622	0.005810
109	0.1	0.094448	[0.973350786984674, 0.9728734913068632, 0.9723...	0.014252	[0.021397662368049297, 0.02187322604510883, 0....	0.973351	26384.615385	-146.485857	0.005552
119	0.1	0.094602	[0.9731745086059119, 0.9726811187876436, 0.972...	0.014052	[0.020545168463810826, 0.02101191275165515, 0....	0.973175	28923.076923	-156.135081	0.005398
129	0.1	0.094920	[0.9729383979091906, 0.9724330981321421, 0.971...	0.013776	[0.019779612814325886, 0.020233352469091748, 0...	0.972938	31461.538462	-159.839586	0.005080
139	0.1	0.095154	[0.9727541257592449, 0.972239293183039, 0.9717...	0.013586	[0.019108440977203323, 0.019551592727872102, 0...	0.972754	34000.000000	-164.772077	0.004846
149	0.1	0.095137	[0.9727014776062314, 0.9721838993848494, 0.971...	0.013505	[0.018522177106079007, 0.01895774008113288, 0....	0.972701	36538.461538	-177.681770	0.004863
159	0.1	0.095399	[0.9725195836003727, 0.9719946013544389, 0.971...	0.013299	[0.017971009148711414, 0.018397454962395532, 0...	0.972520	39076.923077	-179.806052	0.004601
169	0.1	0.095560	[0.9723949690001968, 0.9718626463092632, 0.971...	0.013198	[0.017485209454621398, 0.017904938089009448, 0...	0.972395	41615.384615	-184.782705	0.004440
179	0.1	0.095781	[0.9722436928603018, 0.9717038090031127, 0.971...	0.013022	[0.017026877462078316, 0.017439040109925197, 0...	0.972244	44153.846154	-186.305089	0.004219
189	0.1	0.095761	[0.9722182328904863, 0.9716742030361736, 0.971...	0.013014	[0.016633041914382307, 0.01704217508198902, 0....	0.972218	46692.307692	-197.933893	0.004239
199	0.1	0.095951	[0.9720916726353688, 0.9715375639465644, 0.970...	0.012822	[0.01623703640171481, 0.016640444601397908, 0....	0.972092	49230.769231	-199.338416	0.004049
209	0.1	0.096097	[0.9719892512806404, 0.9714297854663078, 0.970...	0.012715	[0.015883740476955843, 0.016281192369222597, 0...	0.971989	51769.230769	-202.070337	0.003903
219	0.1	0.096064	[0.9719765424523157, 0.9714173103140065, 0.970...	0.012659	[0.015563598938822865, 0.015956882753832058, 0...	0.971977	54307.692308	-213.749656	0.003936
229	0.1	0.096241	[0.9718650131577972, 0.9712989610631855, 0.970...	0.012584	[0.01526109881950194, 0.01565037657469881, 0.0...	0.971865	56846.153846	-213.696235	0.003759
239	0.1	0.096381	[0.9717733336623892, 0.9712023944198541, 0.970...	0.012494	[0.014974524883108742, 0.015359178239021054, 0...	0.971773	59384.615385	-214.941857	0.003619
249	0.1	0.096502	[0.971693240641236, 0.9711179690247418, 0.9705...	0.012391	[0.014701975805022527, 0.01508225606022745, 0....	0.971693	61923.076923	-216.625345	0.003498
259	0.1	0.096413	[0.9717150246591534, 0.9711406504112801, 0.970...	0.012386	[0.014464493065167894, 0.014842509830206074, 0...	0.971715	64461.538462	-231.229803	0.003587
269	0.1	0.096600	[0.9716046870094722, 0.9710235612976689, 0.970...	0.012289	[0.014221961894331836, 0.014595999827984374, 0...	0.971605	67000.000000	-227.777541	0.003400
279	0.1	0.096744	[0.9715194168982284, 0.9709317888905377, 0.970...	0.012209	[0.013994926226013005, 0.014365888181096464, 0...	0.971519	69538.461538	-226.443805	0.003256
289	0.1	0.096660	[0.9715447000441898, 0.9709576541380387, 0.970...	0.012158	[0.01378558003561492, 0.014154156992659983, 0....	0.971545	72076.923077	-240.762977	0.003340
299	0.1	0.096780	[0.9714689814673761, 0.9708779332549557, 0.970...	0.012088	[0.013582674065847011, 0.013948145684839486, 0...	0.971469	74615.384615	-240.262536	0.003220
309	0.1	0.096950	[0.9713714311819946, 0.9707745386599653, 0.970...	0.012006	[0.013386785389746358, 0.013748956402444001, 0...	0.971371	77153.846154	-235.301103	0.003050
319	0.1	0.096843	[0.9714106936982658, 0.9708164996899813, 0.970...	0.012018	[0.013219880215665336, 0.013580946861125431, 0...	0.971411	79692.307692	-251.557614	0.003157

329	0.1	0.096936	[0.9713522836859022, 0.970755068931079, 0.9701...	0.011950	[0.013043648825775689, 0.013401936251164037, 0...	0.971352	82230.769231	-251.963851	0.003064
339	0.1	0.097068	[0.9712759111473707, 0.9706740793643296, 0.970...	0.011901	[0.0128791306401434, 0.013235106242633632, 0.0...	0.971276	84769.230769	-248.511585	0.002932
349	0.1	0.097176	[0.9712117139535813, 0.9706061312142195, 0.970...	0.011843	[0.012719470444825132, 0.013073058991439956, 0...	0.971212	87307.692308	-246.553442	0.002824
359	0.1	0.097090	[0.9712437324621592, 0.9706404850038844, 0.970...	0.011834	[0.012577132398236768, 0.01292946910066034, 0....	0.971244	89846.153846	-261.491225	0.002910
369	0.1	0.097194	[0.9711828589078391, 0.9705741375464879, 0.969...	0.011780	[0.012430405958267221, 0.012780949559415777, 0...	0.971183	92384.615385	-259.217688	0.002806
379	0.1	0.097329	[0.9711077405839378, 0.9704943414035999, 0.969...	0.011690	[0.012281385721804539, 0.01262884422182714, 0....	0.971108	94923.076923	-253.517637	0.002671
389	0.1	0.097225	[0.9711492491363148, 0.9705388529937014, 0.969...	0.011717	[0.012164620864706766, 0.012511707790255955, 0...	0.971149	97461.538462	-270.499254	0.002775
399	0.1	0.097341	[0.9710825409003666, 0.970468317029247, 0.9698...	0.011634	[0.012026414579701217, 0.012370957844023478, 0...	0.971083	100000.000000	-265.900048	0.002659

In [15]:
```
if not bonly:
    plt.scatter(lumis, df[df["alpha"] == sig_fracs[idx]]["mu"].to_numpy())
    plt.errorbar(lumis, df[df["alpha"] == sig_fracs[idx]]["mu"].to_numpy(), df[df["alpha"] == sig_fracs[-1]]["m
else:
    plt.scatter(lumis, df[df["alpha"] == sig_fracs[idx]]["mu"].to_numpy())

plt.axhline(sig_fracs[idx], c='r', ls='--')
#plt.xlim(-2000, 1.1e5)

plt.tight_layout()
```



In [16]:
```
use_std = True

r = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].mean(numeric_only=True).to_numpy()

if use_std:
    r_std = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].std(numeric_only=True).to_numpy()
else:
    r_std = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].apply(lambda x: np.sqrt(np.sum(x**2) /
```

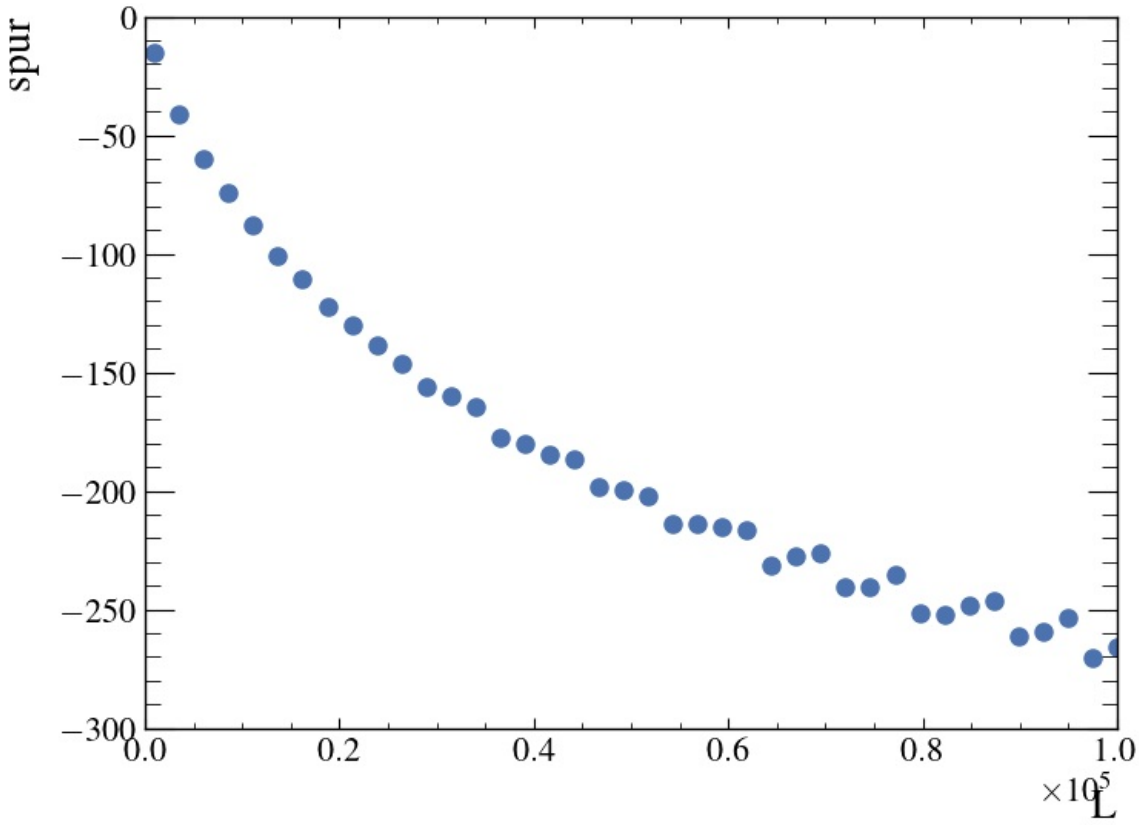
In [17]:
```
r
```

In [17]:
```
r
```

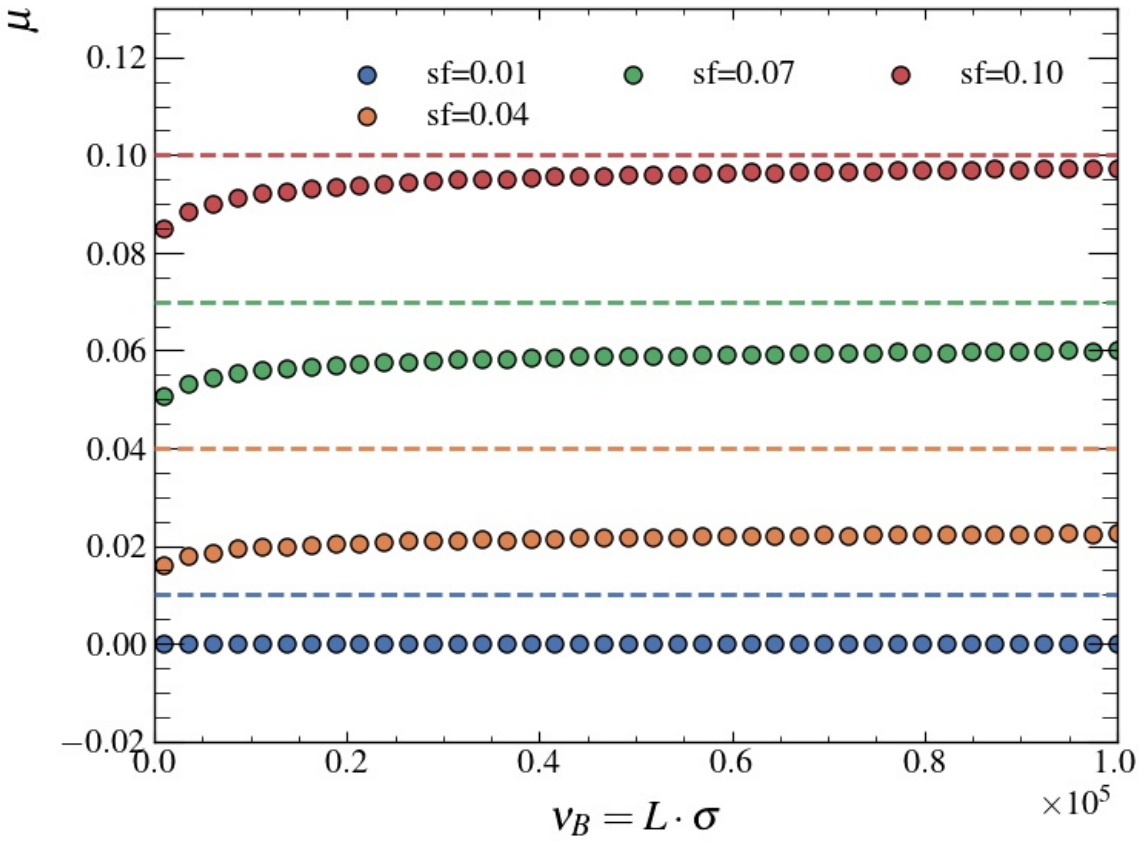
Out[17]:
```
array([ -15.12327693,  -40.82513124,  -59.6394378 ,  -74.22297468,
        -87.75742149, -100.62401565, -110.48038755, -122.32287638,
       -130.13358845, -138.55462192, -146.48585736, -156.13508125,
       -159.83958584, -164.77207699, -177.68176954, -179.80605246,
       -184.78270477, -186.30508904, -197.93389314, -199.33841621,
       -202.07033674, -213.7496555 , -213.69623505, -214.94185657,
       -216.62534533, -231.22980281, -227.77754075, -226.44380479,
       -240.76297721, -240.26253585, -235.30110295, -251.55761376,
       -251.96385106, -248.51158522, -246.55344191, -261.49122521,
       -259.21768754, -253.51763712, -270.49925358, -265.90004756])
```

In [18]:
```
plt.scatter(lumis, r)
# plt.errorbar(lumis, r, r_std, ls="none", capsize=4)

# plt.yscale("log")

plt.xlabel("L")
plt.ylabel("spur")
```

Out[18]:
```
Text(0, 1, 'spur')
```

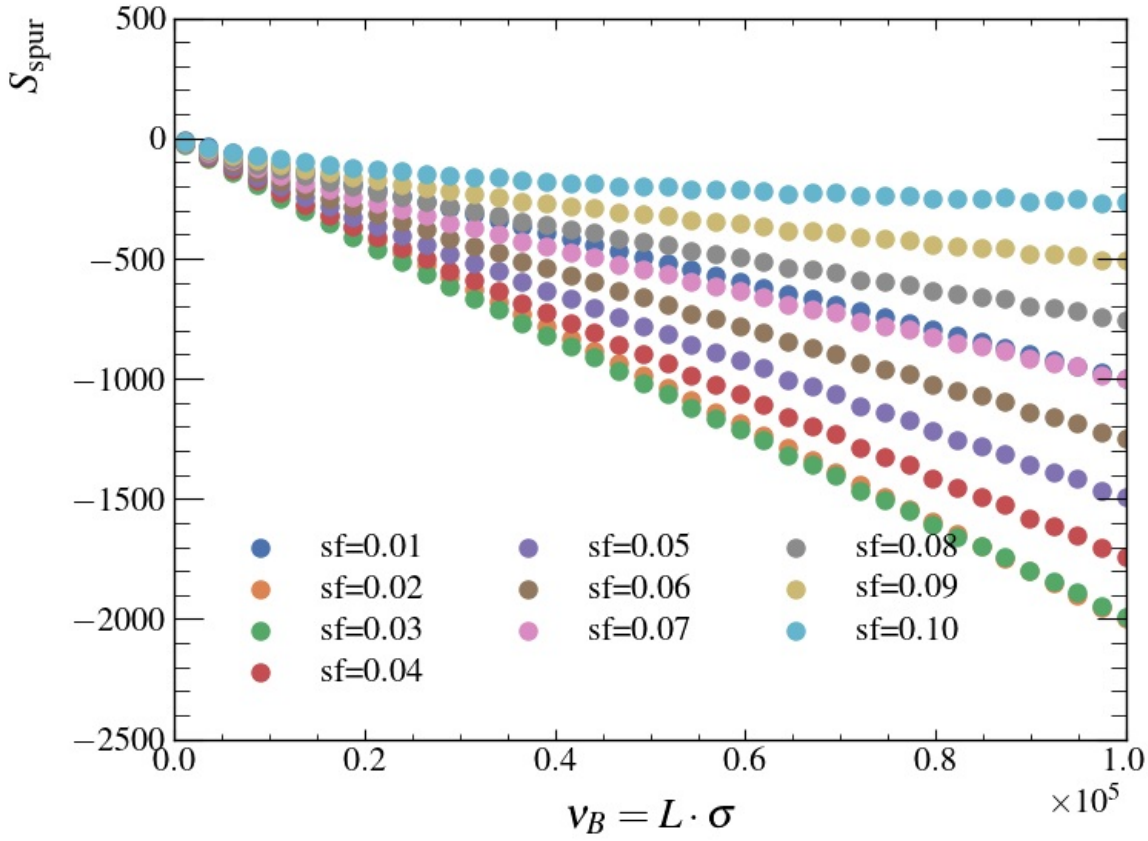


In [19]:
```
for idx, sf in enumerate(sig_fracs[::3]):
    r = df[df["alpha"] == sf].groupby("total_B")["mu"].mean(numeric_only=True).to_numpy()

    if use_std:
        r_std = df[df["alpha"] == sf].groupby("total_B")["mu"].std(numeric_only=True).to_numpy()
    else:
        r_std = df[df["alpha"] == sf].groupby("total_B")["mu"].apply(lambda x: np.sqrt(np.sum(x**2) / len(x))).

    plt.scatter(lumis, r, label="sf={:.2f}".format(sf), edgecolor='k')
    # plt.errorbar(lumis, r, r_std, ls="none", capsize=4)
    plt.axhline(sf, c=f"C{idx}", ls='--', zorder=10)

if bonly:
    plt.yscale("log")

plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")
plt.ylabel("$\mu$")
plt.legend(ncol=3)
plt.ylim(-0.02, 0.13)

plt.tight_layout()
plt.savefig(saved + "mu_vs_L.pdf")
plt.show()
```

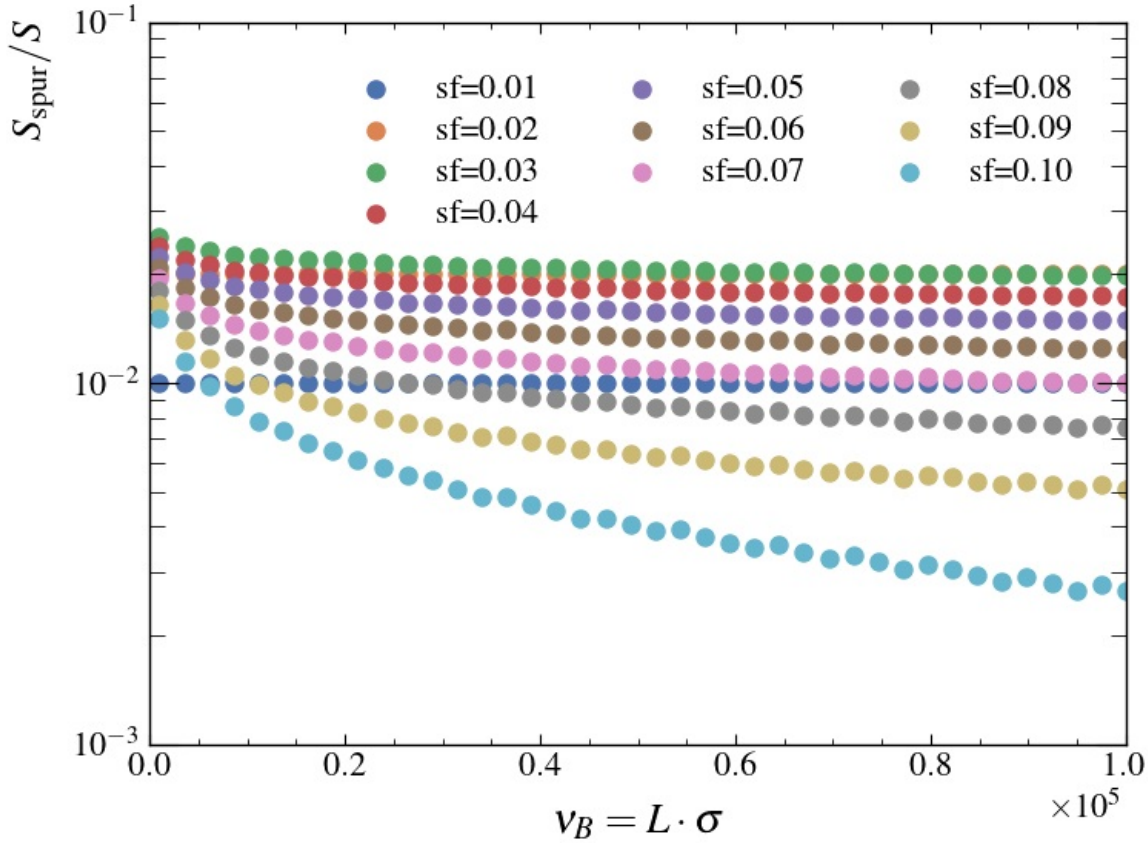


In [20]:

```
for idx, sf in enumerate(sig_fracs):
    r = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].mean(numeric_only=True).to_numpy()

    if use_std:
        r_std = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].std(numeric_only=True).to_numpy()
    else:
        r_std = df[df["alpha"] == sig_fracs[idx]].groupby("total_B")["spur"].apply(lambda x: np.sqrt(np.sum(x**

    plt.scatter(lumis, r, label="sf={:.2f}".format(sf))
    # plt.plot(lumis, r, label="sf={:.2f}".format(sf))

    # plt.errorbar(lumis, r, r_std, ls="none", capsize=4)

#plt.yscale("symlog")
plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")
plt.ylabel(r"$S_{\text{spur}}$")
plt.legend(ncol=3)

plt.tight_layout()
plt.savefig(saved + "spur_vs_L_scenario.pdf")
```

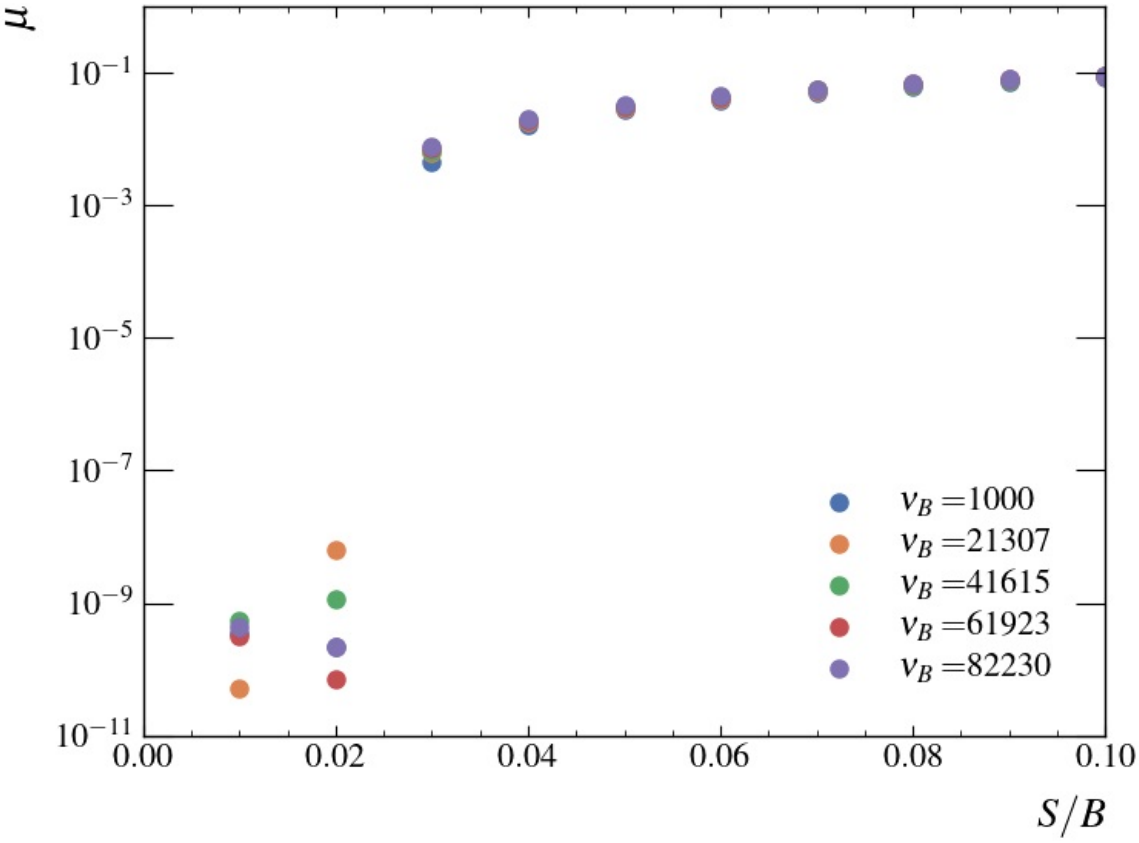


In [21]:
```
for idx, sf in enumerate(sig_fracs):
    r = df[df["alpha"] == sig_fracs[idx]].groupby("total_B").mean(numeric_only=True)["spur_ratio"].to_numpy()
    plt.scatter(lumis, r, label="sf={:.2f}".format(sf))

plt.yscale("log")
plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")
plt.ylabel(r"$S_{\text{spur}} / S$")
plt.legend(ncol=3)

plt.tight_layout()
plt.savefig(saved + "ratio_vs_L.pdf")
```



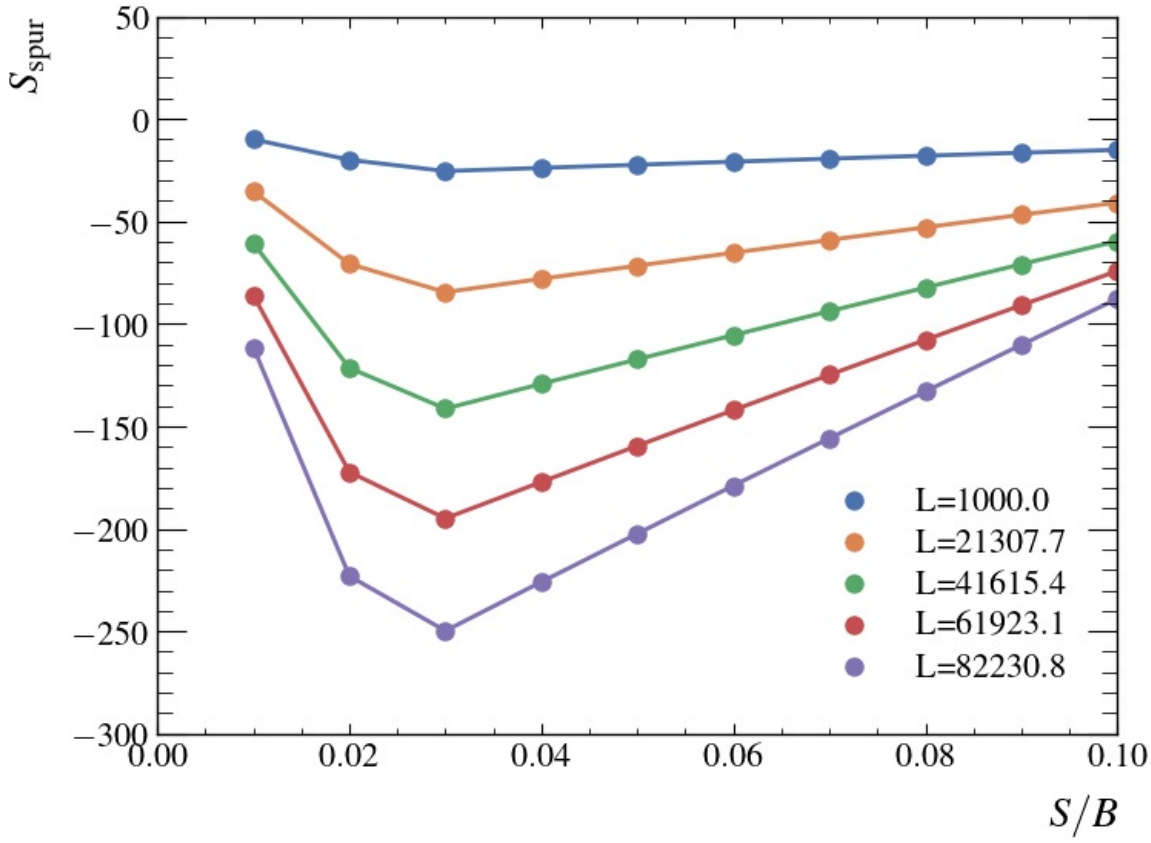
In [22]:
```
for idx, l in enumerate(lumis[::8]):
    r = df[df["total_B"] == lumis[idx]].groupby("alpha").mean(numeric_only=True)["mu"].to_numpy()

    if use_std:
```

```
        r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["mu"].std(numeric_only=True).to_numpy()
    else:
        r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["mu"].apply(lambda x: np.sqrt(np.sum(x**2) / l

    plt.scatter(sig_fracs, r, label=r"$\nu_B=$" + "{}".format(int(l)))
    # plt.errorbar(sig_fracs, r, r_std, ls="none", capsize=4)

    # plt.title(r"L$=${}".format(lumis[idx]))

plt.xlabel(r"$S/B$")
plt.ylabel("$\mu$")

plt.yscale("log")
plt.legend()

plt.tight_layout()
plt.savefig(saved + "mu_vs_sig_frac.pdf")
```



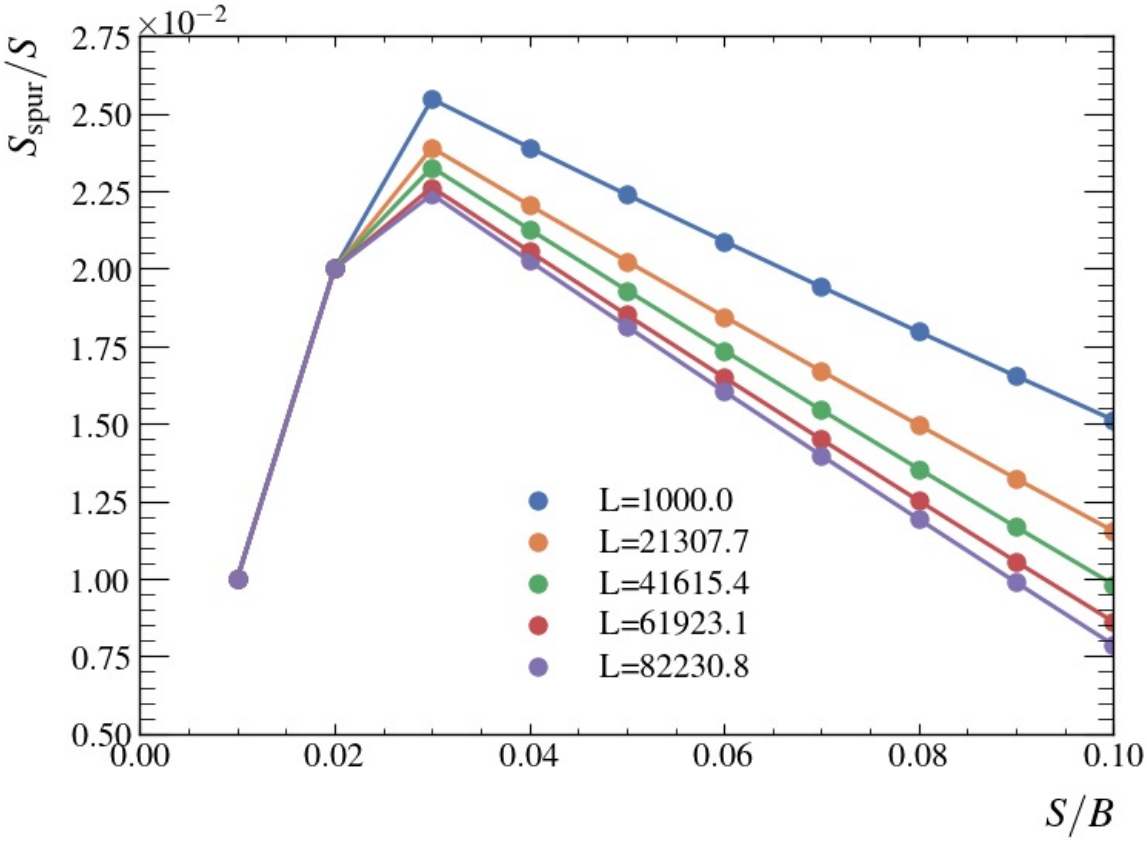
In [23]:

```
for idx, l in enumerate(lumis[::8]):
    r = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur"].mean(numeric_only=True).to_numpy()

    if use_std:
        r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur"].std(numeric_only=True).to_numpy()
    else:
        r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur"].apply(lambda x: np.sqrt(np.sum(x**2) /

    plt.scatter(sig_fracs, r, label="L={:.1f}".format(l))
    plt.plot(sig_fracs, r)
    # plt.errorbar(sig_fracs, r, r_std, ls="none", capsize=4)

    # plt.title(r"L$=${}".format(lumis[idx]))

plt.xlabel(r"$S/B$")
plt.ylabel(r"$S_{\text{spur}}$")

# plt.yscale("log")
plt.legend()

plt.tight_layout()
plt.savefig(saved + "spur_vs_sig_frac.pdf")
```

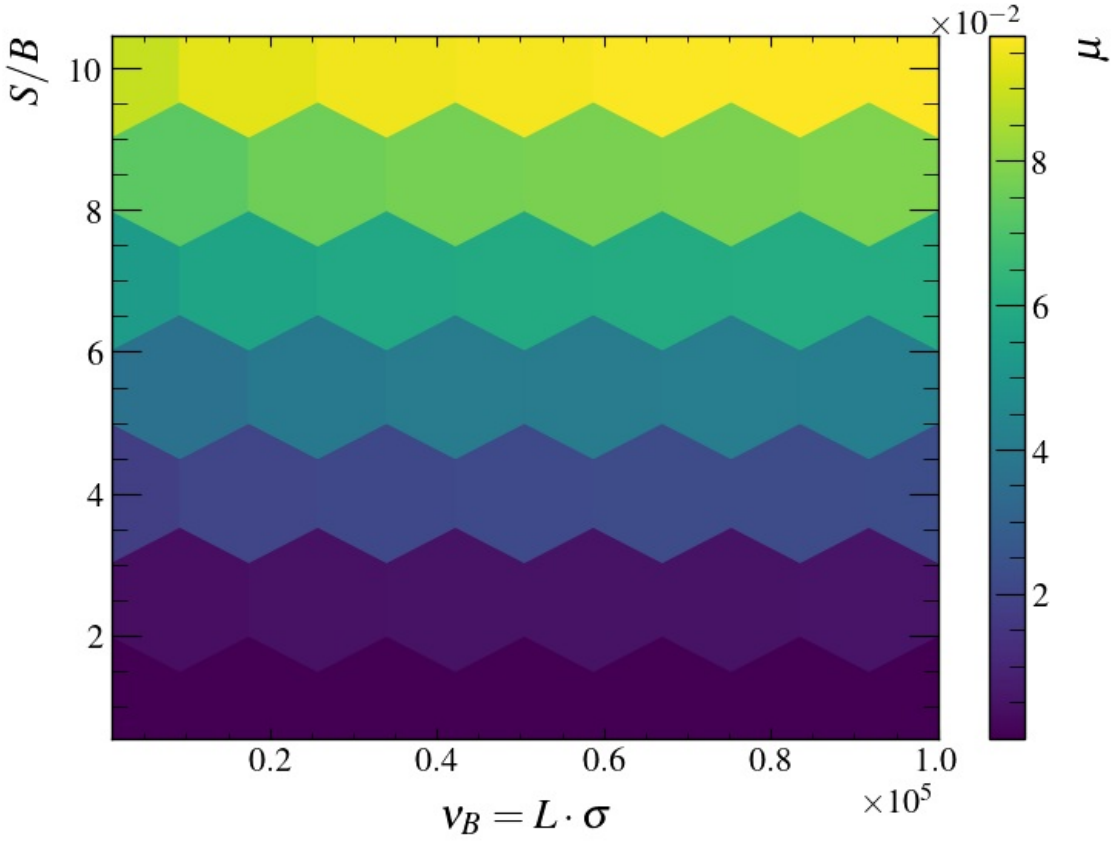


In [24]:

```
if not bonly:
    for idx, l in enumerate(lumis[::8]):
        r = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur_ratio"].mean(numeric_only=True).to_numpy()

        if use_std:
            r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur_ratio"].std(numeric_only=True).to_nu
        else:
            r_std = df[df["total_B"] == lumis[idx]].groupby("alpha")["spur_ratio"].apply(lambda x: np.sqrt(np.s

        plt.scatter(sig_fracs, r, label="L={:.1f}".format(l))
        plt.plot(sig_fracs, r)
        # plt.errorbar(sig_fracs, r, r_std, ls="none", capsize=4)

        # plt.title(r"L$=${}".format(lumis[idx]))

    plt.xlabel(r"$S/B$")
    plt.ylabel(r"$S_{\text{spur}}/S$")

    # plt.yscale("log")
    plt.legend()

    plt.tight_layout()
    plt.savefig(saved + "spur_vs_sig_frac_ratio.pdf")
```

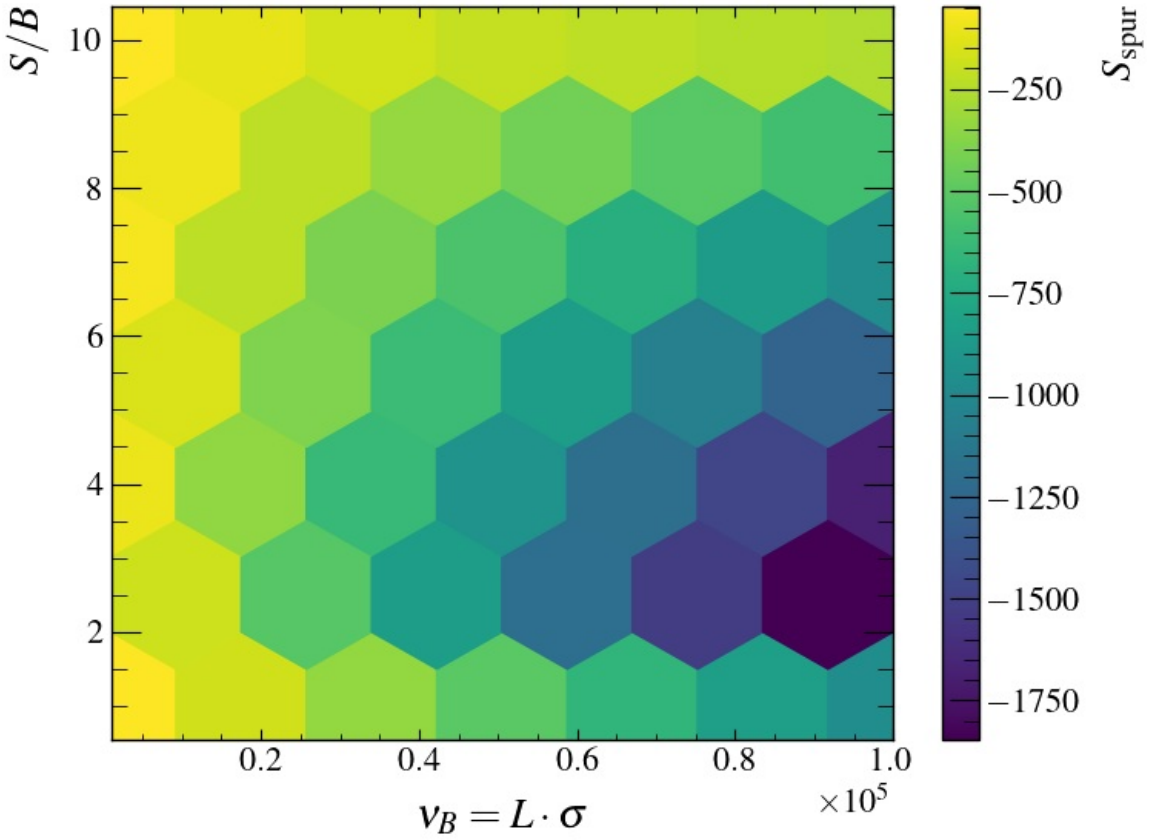


In [25]:
```
y = np.array([float(i) * 100 for i in df["alpha"].values])
x = np.array([float(i) for i in df["total_B"].values])
z = np.array([float(i) for i in df["mu"].values])

plt.ylabel("$S/B$")
plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")

plt.hexbin(x, y, z, gridsize=6)
plt.colorbar(label="$\mu$")
plt.tight_layout()

plt.savefig(saved + "hexbin_sig_frac_L_mu.pdf")
```

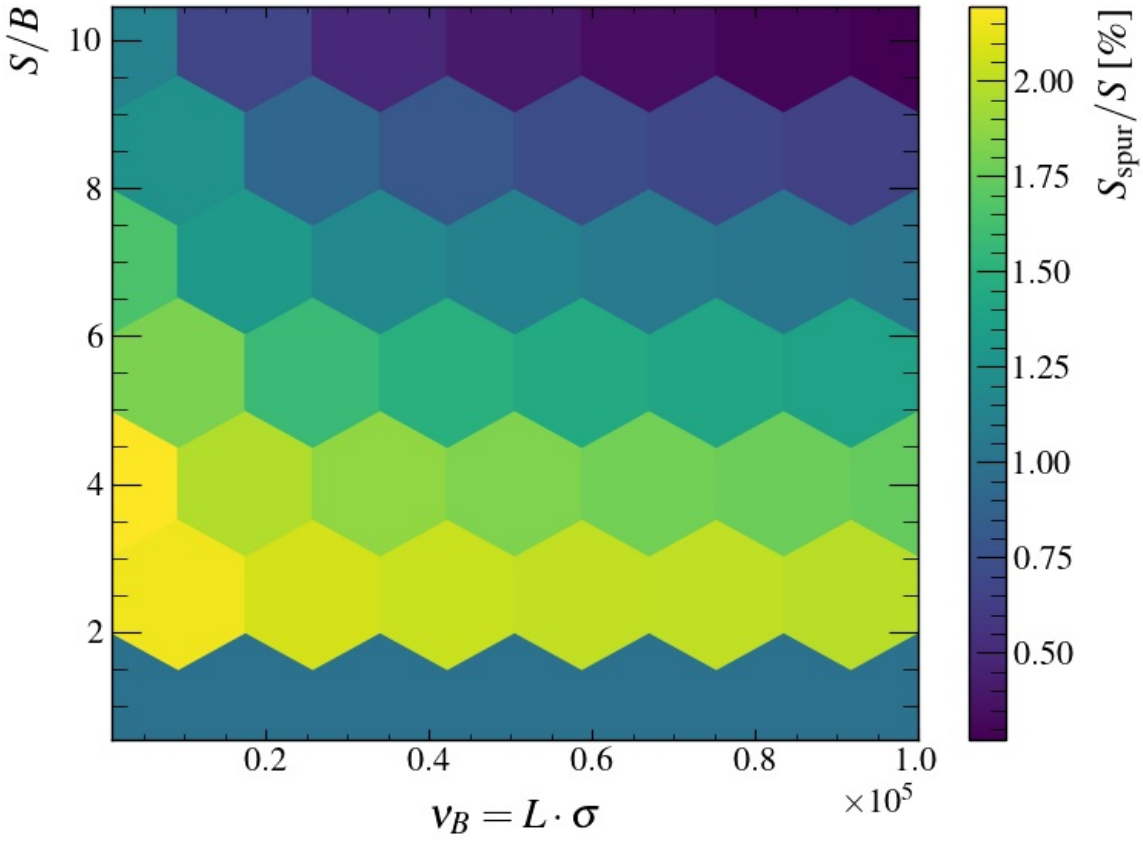


In [26]:
```
y = np.array([float(i) * 100 for i in df["alpha"].values])
x = np.array([float(i) for i in df["total_B"].values])
z = np.array([float(i) for i in df["spur"].values])
```

```
plt.ylabel("$S/B$")
plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")

plt.hexbin(x, y, z, gridsize=6)
plt.colorbar(label=r"$S_{\text{spur}}$")
plt.tight_layout()

plt.savefig(saved + "hexbin_sig_frac_L_spur.pdf")
```



In [27]:
```
if not bonly:
    y = np.array([float(i) * 100 for i in df["alpha"].values])
    x = np.array([float(i) for i in df["total_B"].values])
    z = np.array([float(i) * 100 for i in df["spur_ratio"].values])

    plt.ylabel("$S/B$")
    plt.xlabel(r"$\nu_B=L\cdot\sigma$", loc="center")

    plt.hexbin(x, y, z, gridsize=6)
    plt.colorbar(label=r"$S_{\text{spur}}/S$ [\%]")
    plt.tight_layout()

    plt.savefig(saved + "hexbin_sig_frac_L_spur_ratio.pdf")
```



In [ ]:

In [ ]:

In [ ]: