Chapter 9: continuum limit and phase transitions

In this chapter we consider the behavior of lattice QCD calculations close to the continuum limit $a \to 0$. We focus on pure SU(3) gauge theory with $\beta = 6/g^2$.

Scaling and beta function

Let $O(g,a)$ be a measurement on an ensemble with coupling g and lattice spacing a. If the measurement has a continuum limit, we have

$$\lim_{a\to 0} O(g(a),a) = O_0\,. \tag{1}$$

Therefore for sufficiently small a we must have

$$0 = a\frac{d}{da}O(g(a),a) = \frac{d}{d\log a}O(g(a),a) \tag{2}$$

$$= \left(\frac{\partial g(a)}{\partial \log a}\frac{\partial}{\partial g} + \frac{\partial}{\partial \log a}\right) O(g,a) \tag{3}$$

This expression separates the explicit lattice scale dependence of the measurement from the one introduced through the indirect dependence of the coupling constant g on the lattice spacing. In a more general theory, such as QCD with dynamical quarks, we will have more couplings to tune which all also contribute to this equation similarly to g. For now, we continue to focus on the pure QCD case.

As an example, consider measuring the mass of a particle in lattice units m^{lat}. Then the corresponding mass in physical units, say GeV, is given by

$$O_m(g,a) = a^{-1}m^{\mathrm{lat}}(g)\,. \tag{4}$$

For this measurement we therefore have

$$\frac{\partial}{\partial \log a}O_m(g,a) = -O_m(g,a)\,. \tag{5}$$

In general, for an operator O_λ with mass dimension $[O_\lambda] = \lambda$, we obtain

$$\frac{\partial}{\partial \log a}O_\lambda(g,a) = -\lambda O_\lambda(g,a)\,. \tag{6}$$

This term therefore accounts for the naive scaling dimension λ.

It is now useful to define the **beta function**

$$\beta(g(a)) = \frac{\partial g(a)}{\partial \log a^{-1}} = -\frac{\partial g(a)}{\partial \log a} \tag{7}$$

which encapsulates the scale dependence of the coupling constant g. One can show in perturbation theory that for gauge group SU(N) with n_f massless fermions one has

$$\beta(g(a)) = -g^3\beta_0(N,n_f) - g^5\beta_1(N,n_f) + O(g^7) \tag{8}$$

with

$$\beta_0(N,n_f) = \frac{1}{(4\pi)^2}\left(\frac{11}{3}N - \frac{2}{3}n_f\right)\,, \tag{9}$$

$$\beta_1(N,n_f) = \frac{1}{(4\pi)^4}\left(\frac{34}{3}N^2 - \frac{10}{3}Nn_f - \frac{N^2-1}{N}n_f\right)\,. \tag{10}$$

For $N=3$ and $n_f = 0$, we therefore have

$$\beta_0(3,0) = \frac{11}{(4\pi)^2} \approx 0.0696583\,, \tag{11}$$

$$\beta_1(3,0) = \frac{51}{128\pi^4} \approx 0.00409035\,. \tag{12}$$

Dimensional transmutation

We can solve the differential equation for g, Eq. 7,

$$\frac{dg}{d\log a} = g^3\beta_0 + g^5\beta_1 + O(g^7) \tag{13}$$

by separation of variables

$$d\log a = \frac{dg}{g^3\beta_0 + g^5\beta_1} + O(g) \tag{14}$$

$$= dg\left(\frac{1}{g^3\beta_0} - \frac{\beta_1}{g\beta_0^2}\right) + O(g) \tag{15}$$

integrating with integration constant c_0 yields

$$\log a = c_0 - \frac{1}{2}\frac{1}{g^2\beta_0} - \frac{\beta_1}{\beta_0^2}\log g + O(g^2) \tag{16}$$

We can re-write this expression as

$$a = \exp(c_0)\exp\left(-\frac{1}{2}\frac{1}{g^2\beta_0}\right)\left(g^2\right)^{-\frac{\beta_1}{2\beta_0^2}}\left(1+O(g^2)\right) \tag{17}$$

$$= \exp(c_0)(\beta_0)^{\frac{\beta_1}{2\beta_0^2}}\exp\left(-\frac{1}{2}\frac{1}{g^2\beta_0}\right)\left(\beta_0 g^2\right)^{-\frac{\beta_1}{2\beta_0^2}}\left(1+O(g^2)\right) \tag{18}$$

$$= \frac{1}{\Lambda_L}\exp\left(-\frac{1}{2}\frac{1}{g^2\beta_0}\right)\left(\beta_0 g^2\right)^{-\frac{\beta_1}{2\beta_0^2}}\left(1+O(g^2)\right) \tag{19}$$

with

$$\frac{1}{\Lambda_L} = \exp(c_0)(\beta_0)^{\frac{\beta_1}{2\beta_0^2}}, \tag{20}$$

where $[\Lambda_L] = 1$. The integration constant therefore has introduced a new dimensional variable to the theory that had no explicit dimensional parameters in the action before! The effect of relating the dimensionless parameter g and the dimensionful parameter a is called **dimensional transmutation**.

Asymptotic freedom

We can also re-write the above equation as

$$g(a)^{-2} = \beta_0 \log(a^{-2}\Lambda_L^{-2}) + \frac{\beta_1}{\beta_0}\log(\log(a^{-2}\Lambda_L^{-2})) + O(1/\log(a^2\Lambda_L^2)). \tag{21}$$

This scale dependence of the coupling constant or **running coupling** is a feature of the quantization of the field theory. We have already observed the scale dependence of the coupling in chapter 7 and noticed that the coupling gets weaker for shorter distances. The fact that the coupling vanishes in the limit $a \to 0$, which is equivalent to considering a free theory, is referred to as **asymptotic freedom**.

For sufficiently small coupling g it suffices to look at the leading-order perturbative result such that as long as $\beta_0 > 0$, we are driven to a free theory. For a general gauge group N and number of massless fermions n_f, this holds as long as

$$\frac{11}{3}N > \frac{2}{3}n_f\,. \tag{22}$$

For QCD (N=3) we therefore have asymptotic freedom as long as

$$n_f < \frac{33}{2} \tag{23}$$

or as long as we have at maximum 16 massless quarks. We have so far found 6 quark flavors in nature.

Another aspect worth highlighting is that due to asymptotic freedom, the continuum limit $a \to 0$ of QCD corresponds to $g \to 0$. Note that in a general quantum field theory the continuum limit may be at non-trivial couplings as we will discuss again below.

Universality of two-loop perturbation theory

One can define different coupling constants g' in addition to the bare lattice coupling g. In continuum perturbation theory, e.g., one typically uses the $\overline{\text{MS}}$ scheme and $g^{\overline{\text{MS}}}(\mu)$ at renormalization energy scale μ. There are also regulator independent momentum-subtraction schemes (RI-MOM), the Wilson-flow coupling, or a coupling defined in short-distance perturbation theory of the static quark potential.

They all have in common that one can relate two different coupling constant definitions g_1 and g_2 in perturbation theory using

$$g_2 = g_1 + c g_1^3 + O(g_1^5) \tag{24}$$

where c is computable in perturbation theory.

Homework: Using this equation, show that β_0 and β_1 are universal, i.e., they are the same for the β function of g_1 and g_2. Start by writing

$$\frac{\partial g_i(a)}{\partial \log a} = g_i^3\beta_0^{(i)}(N,n_f) + g_i^5\beta_1^{(i)}(N,n_f) + O(g_i^7) \tag{25}$$

and relate $\beta_{0,1}^{(1)}$ to $\beta_{0,1}^{(2)}$.

Higher orders in the beta expansion will no longer be universal. In addition, it should be noted that Λ_L depends on the definition of the coupling g and that the ratio of Λ_L for different couplings can be calculation in perturbation theory.

Comparing the perturbative results to lattice data

Next, let us compare the perturbative predictions to actual lattice data. We now understand that for sufficiently small lattice spacing a, perturbation theory must become a good approximation, however, we do not yet know how fast this limit is approached.

One way to make progress is to measure a for a large number of β values in a lattice calculation, extending our exercise of chapter 7. This was done, e.g., in this paper, where (2.6) of said paper finds

$$a = r_0 \exp(-1.6804 - 1.7331(\beta-6) + 0.7849(\beta-6)^2 - 0.4428(\beta-6)^3) \tag{26}$$

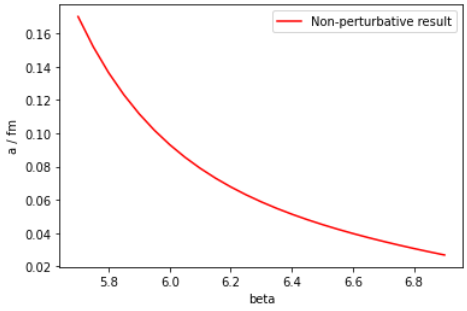
to approximate the lattice results with less than one per-cent error for $\beta \in [5.7, 6.92]$.

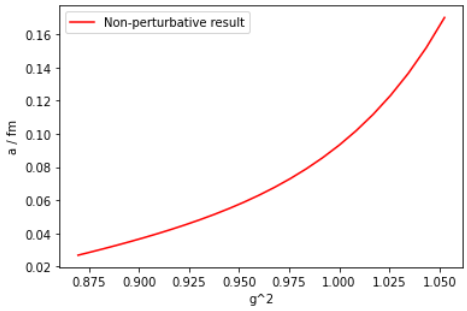
Let us now compare this to the perturbative results.

In [275…

```
import numpy as np
```

```
from matplotlib import pyplot as plt

def lattice_spacing_in_fm_wilson_action(beta):
    assert beta >= 5.7 and beta <= 6.92
    return 0.5*np.exp(-1.6804
                      -1.7331*(beta - 6)
                      +0.7849*(beta - 6)**2
                      -0.4428*(beta-6)**3)

betas = np.arange(5.7,6.92,0.05)
fig, ax = plt.subplots()
plt.xlabel("beta")
plt.ylabel("a / fm")
ax.plot(betas, [lattice_spacing_in_fm_wilson_action(beta) for beta in betas],
        ls='-', c='red', label="Non-perturbative result")
plt.legend()
plt.show()

fig, ax = plt.subplots()
plt.xlabel("g^2")
plt.ylabel("a / fm")
ax.plot([6/b for b in betas], [lattice_spacing_in_fm_wilson_action(beta) for beta in betas],
        ls='-', c='red', label="Non-perturbative result")
plt.legend()
plt.show()
```

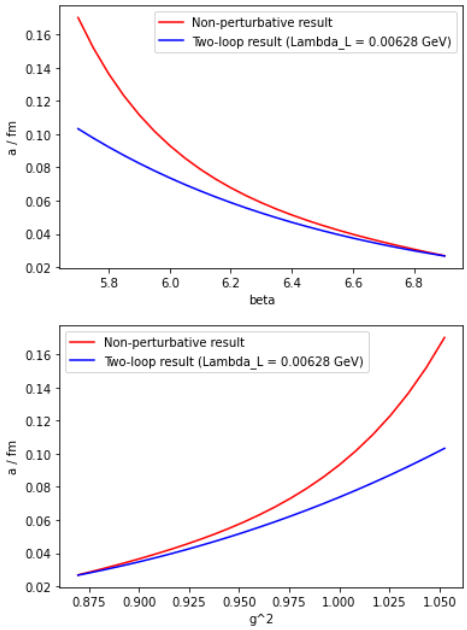




In [282…

```
from scipy.optimize import minimize

def lattice_spacing_in_fm_two_loop(beta, lambda_l_in_GeV):
    Nc = 3
    beta0 = 1./(4.*np.pi)**2.*(11./3.*Nc)
    beta1 = 1./(4.*np.pi)**4.*(34./3.*Nc**2.)
    inv_lambda_l_in_fm = 0.1973/lambda_l_in_GeV
    g2 = 2*Nc / beta
    return inv_lambda_l_in_fm*(beta0*g2)**(-beta1/2./beta0**2.)*np.exp(-1./2./beta0/g2)

# determine the Lambda parameter by minimizing the difference
beta_match = 6.92
lambda_l_in_GeV=minimize(lambda l_GeV: abs(lattice_spacing_in_fm_two_loop(beta_match, l_GeV)
                                - lattice_spacing_in_fm_wilson_action(beta_match)), 0.05,
                         method = 'Nelder-Mead', tol=1e-15).x[0]

print("Wilson Gauge Lambda_L / GeV",lambda_l_in_GeV)


betas = np.arange(5.7,6.92,0.05)
fig, ax = plt.subplots()
plt.xlabel("beta")
plt.ylabel("a / fm")
ax.plot(betas, [lattice_spacing_in_fm_wilson_action(beta) for beta in betas],
        ls='-', c='red', label="Non-perturbative result")
ax.plot(betas, [lattice_spacing_in_fm_two_loop(beta, lambda_l_in_GeV) for beta in betas],
        ls='-', c='blue', label=f"Two-loop result (Lambda_L = {lambda_l_in_GeV:.3g} GeV)")
plt.legend()
plt.show()

fig, ax = plt.subplots()
plt.xlabel("g^2")
plt.ylabel("a / fm")
ax.plot([6/b for b in betas], [lattice_spacing_in_fm_wilson_action(beta) for beta in betas],
        ls='-', c='red', label="Non-perturbative result")
ax.plot([6/b for b in betas], [lattice_spacing_in_fm_two_loop(beta, lambda_l_in_GeV) for beta in betas],
        ls='-', c='blue', label=f"Two-loop result (Lambda_L = {lambda_l_in_GeV:.3g} GeV)")
plt.legend()
plt.show()
```

```
Wilson Gauge Lambda_L / GeV 0.0062821932183028065
```



We first note that even for $a = 0.03$ fm or $a^{-1} \approx 6.6$ GeV, this expansion converges rather poorly. We should, however, also note that the expansion in terms of one coupling constant g_1 can converge differently from the expansion in a different coupling constant g_2. So it may be worth exploring expanding, e.g., in the $\overline{\text{MS}}$ coupling instead!

By matching perturbation theory with lattice at $\beta = 6.92$ we find $\Lambda_L \approx 0.00628$ GeV. This is vastly different to what one finds in the $\overline{\text{MS}}$ scheme!

One can show that the poor convergence of lattice perturbation theory, i.e., the expansion in the bare lattice coupling g and the large hierarchy of Λ parameters are related, see this paper, where we also find

$$\Lambda_{\overline{\text{MS}}} \approx 29 \Lambda_L \tag{27}$$

and

$$g^2 = g^2_{\overline{\text{MS}}}(\mu)(1 - g_{\overline{\text{MS}}}(\mu)(\beta_0 \log(\pi/a/\mu)^2 + 0.3088(1)) + O(g^4_{\overline{\text{MS}}}) \tag{28}$$

(Note the additional 4π factor absorbed in β_0 in this paper compared to here.)

For concrete numbers and $\mu = \pi/a$ we find

$$g^2 = g^2_{\overline{\text{MS}}}(\pi/a)(1 - 0.3088(1) g^2_{\overline{\text{MS}}}(\pi/a)) + O(g^4_{\overline{\text{MS}}}) \tag{29}$$

such that the $\overline{\text{MS}}$ coupling is actually larger compared to the bare coupling g. In general, however, the coefficients in terms of this larger coupling constant have a better convergence property.

Phase transitions

We have already established that correlation functions of operators on the lattice that are displaced by n sites, after subtraction of possibly non-vanishing vacuum expectation values, decay exponentially as $e^{-n/\xi}$ for a theory for which the first non-vacuum eigenstate has a finite energy gap to the vacuum. Such a gap is typically related to the mass of a particle and we therefore refer to it as a **mass gap**.

Note that ξ is given in lattice units, i.e., it gives the typicall falloff distance in terms of lattice spacings. The continuum limit as we have discussed it so far, therefore corresponds to a diverging correlation length ξ.

Such diverging correlation lengths are typical for a system undergoing a second order phase transition. Interestingly, at a second order phase transition, one also observes **universality**, i.e., the details of the interactions at the lattice scale become irrelevant and the system is dominated by its symmetry structure. Different lattice regulators therefore fall into different **universality classes** depending on which symmetry they have or restore in the $a \to 0$ limit.

In our subsequent discussion of the renormalization group, we will see how such universality can arise.

Exercise: continuum limit of improved gauge theory

As a warmup exercise for the remaining topics in this chapter, let us consider modifying the Wilson action to

$$S = \beta \sum_x \sum_{\mu<\nu} \left(c_0 (1 - P_{\mu\nu}(x)) + c_1 (2 - R_{\mu\nu}(x) - R_{\nu\mu}(x)) \right) \tag{30}$$

with scalar plaquette and 2x1 rectangle terms

$$P_{\mu\nu}(x) = \frac{1}{N_c} \text{Re Tr } U^{(1x1)}_{\mu\nu}(x) \,, \tag{31}$$

$$R_{\mu\nu}(x) = \frac{1}{N_c} \text{Re Tr } U^{(2x1)}_{\mu\nu}(x) \,, \tag{32}$$

$$U^{(1x1)}_{\mu\nu}(x) = U_\mu(x) U_\nu(x + a\hat{\mu}) U^\dagger_\mu(x + a\hat{\nu}) U^\dagger_\nu(x) \,, \tag{33}$$

$$U^{(2x1)}_{\mu\nu}(x) = U_\mu(x) U_\mu(x + a\hat{\mu}) U_\nu(x + 2a\hat{\mu}) U^\dagger_\mu(x + a\hat{\mu} + a\hat{\nu}) U^\dagger_\mu(x + a\hat{\nu}) U^\dagger_\nu(x) \,, \tag{34}$$

and

$$\beta = \frac{2N_c}{g^2}\,. \tag{35}$$

We remember from chapter 6 that

$$U_{\mu\nu}^{(1x1)}(x) = \exp\left(ia^2 g F_{\mu\nu}(x) + O(a^3)\right) \tag{36}$$

and find in a straightforward computation that

$$U_{\mu\nu}^{(2x1)}(x) = \exp\left(i2a^2 g F_{\mu\nu}(x) + O(a^3)\right)\,. \tag{37}$$

Therefore in S the coefficient of the $F_{\mu\nu}^2$ operator has a prefactor

$$c_0 + 8c_1\,. \tag{38}$$

Since one of the three parameters β, c_0, and c_1 is redundant, we often fix

$$c_0 + 8c_1 = 1 \tag{39}$$

and consider the theory with parameters (β, c_1).

We now generate ensembles with different values for β and c_1 and study the resulting lattice spacing and string tensions.

In [12]:
```
import gpt as g;
import numpy as np;
from matplotlib import pyplot as plt;

def generate_ensemble(Ls, Lt, beta, c1, Nthermalize, Nmax, Nskip):
    L = [Ls, Ls, Ls, Lt]

    grid = g.grid(L, g.double)

    w = g.qcd.gauge.action.improved_with_rectangle(beta, c1)

    rng = g.random("test", "vectorized_ranlux24_24_64")
    U = g.qcd.gauge.unit(grid)
    Nd = len(U)

    # improved actions with rectangle term have a larger
    # footprint, so need masks that accommodate this
    mask = [g.complex(grid) for i in range(4)]
    assert all([l % len(mask) == 0 for l in L])
    pos = g.coordinates(grid)
    for i, m in enumerate(mask):
        m[:] = 0
        m[pos[np.sum(pos, axis=1) % len(mask) == i]] = 1

    g.default.set_verbose("su2_heat_bath", False)
    markov = g.algorithms.markov.su2_heat_bath(rng)
    plaquette_heatbath = []
    for it in range(Nmax):
        plaq = g.qcd.gauge.plaquette(U)
        plaquette_heatbath.append(plaq)
        if it % (Nmax//20) == 0:
            g.message(f"SU(2)-subgroup heatbath {it} has P = {plaq}")
        for m in mask:
            for mu in range(Nd):
                markov(U[mu], w.staple(U, mu), m)
        if it > Nthermalize and it % Nskip == 0:
            g.default.push_verbose("io", False)
            g.save(f"{Ls}c{Lt}_PR_{beta}_{c1}/su3.{it}",U)
            g.default.pop_verbose()


    fig, ax = plt.subplots()
    ax.plot(range(len(plaquette_heatbath)), plaquette_heatbath, marker='+', ls='', c='red', label="Heatbath")
    plt.legend()
    plt.show()

#for beta in [5.5, 5.6, 5.7, 5.8, 5.9]:
#    for c1 in [-0.01,-0.005,0.0,0.001]:
#        if not os.path.exists(f"8c8_PR_{beta}_{c1}/su3.105"):
#            print(beta, c1)
#            generate_ensemble(8, 8, beta, c1, 50, 300, 5)
```

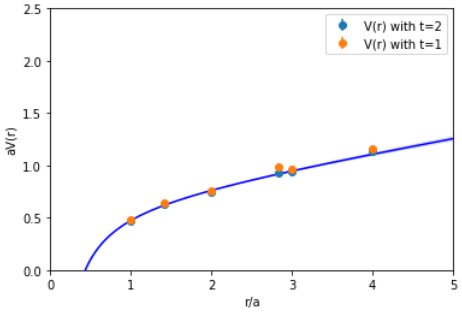
In [4]:
```
from scipy.optimize import curve_fit

g.default.set_verbose("io", False)

def jackknife_covariance_estimator(f, x):
    N = len(x)
    X = sum(x) / N
    mean = f(X)
    def outer_sqr(a):
        return np.outer(a,a)
    return sum([outer_sqr(f((N*X - x[j])/(N-1)) - mean) for j in range(N)])*(N-1)/N

def V(t0, configurations):
    U = g.load(configurations[0])

    Wt = [{},{}]

    for t in [t0,t0+1]:
```

```
        for r in range(1,5):
            Wt[t-t0][r] = g.qcd.gauge.transport(U,[g.qcd.gauge.path().f(3,t).f(nu,r).b(3,t).b(nu,r)
                                    for nu in range(3)])
        for r in [1,2]:
            Wt[t-t0][np.sqrt(2.)*r] = g.qcd.gauge.transport(U,
                                        [g.qcd.gauge.path().f(3,t).f(nu,r).f(rho,r).b(3,t).b(rho,r).b(nu,r)
                                         for nu in range(3) for rho in range(3) if nu != rho])

    Wlc = []
    for conf in configurations:
        U = g.load(conf)

        for i in range(4):
            U = g.qcd.gauge.smear.stout(U, rho=1.0/12.0,orthogonal_dimension=3)

        print(conf, end="\r")
        Wlc.append([[sum([g.sum(g.trace(l)).real for l in wti[r](U)]) for wti in Wt] for r in Wt[0]])

    Wlc = np.array(Wlc)
    Wl_mean = np.mean(Wlc,axis=0)
    V_mean = [np.log(Wl_mean[i,0]/Wl_mean[i,1]) for i in range(len(Wl_mean))]
    rs = [r for r in Wt[0]]
    V_cov = jackknife_covariance_estimator(lambda x: np.array([np.log(x[i,0]/x[i,1]) for i in range(len(Wl_mean))]),
                                           Wlc)
    V_err = [V_cov[i,i]**0.5 for i in range(len(V_mean))]
    print()
    return rs, V_mean, V_err, V_cov

def lattice_spacing(configurations):
    def model(r,*p):
        return p[0] + p[1]/r + p[2]*r

    def model_gradient(r,*p):
        return np.array([r**0,r**-1,r]).T

    rs1, Vm1, Ve1, Vc1 = V(1, configurations)
    rs2, Vm2, Ve2, Vc2 = V(2, configurations)

    par_mean, par_cov = curve_fit(model, rs2, Vm2, p0=[0.5,0.5,0.5], sigma=Vc2, absolute_sigma=True, jac=model_gradient)
    a_in_fm = ((par_mean[1] + 1.65)/par_mean[2])**-0.5 * 0.5

    a_in_fm_dpar1 = -0.5 * ((par_mean[1] + 1.65)/par_mean[2])**-1.5 * 0.5 * 1.0 / par_mean[2]
    a_in_fm_dpar2 = 0.5 * ((par_mean[1] + 1.65)/par_mean[2])**-1.5 * 0.5 * 1.0 * (par_mean[1] + 1.65) / par_mean[2]**2
    a_in_fm_dpar = np.array([0,a_in_fm_dpar1,a_in_fm_dpar2])

    a_in_fm_err = np.dot(a_in_fm_dpar,np.dot(par_cov,a_in_fm_dpar))**0.5
    print(a_in_fm, a_in_fm_err)

    def err_f(r,p,cov_p):
        return np.dot(model_gradient(r,*p), np.dot(cov_p, model_gradient(r,*p)))**0.5

    rrange = np.arange(0.1, 5.0, 0.01)
    fy = np.array([model(r,*par_mean) for r in rrange])
    fyerr = np.array([err_f(r,par_mean,par_cov) for r in rrange])

    fig, ax = plt.subplots()

    plt.ylim(0,2.5)
    plt.xlim(0,5)

    plt.xlabel("r/a")
    plt.ylabel("aV(r)")
    ax.fill_between(rrange,fy-fyerr,fy+fyerr,alpha=0.1,color="blue")
    ax.plot(rrange,fy,c="blue")

    ax.errorbar(rs2, Vm2, Ve2, fmt="o", label="V(r) with t=2")
    ax.errorbar(rs1, Vm1, Ve1, fmt="o", label="V(r) with t=1")

    plt.legend()
    plt.show()

    return (a_in_fm, a_in_fm_err, rs2, Vm2, Ve2, par_mean, par_cov)
```

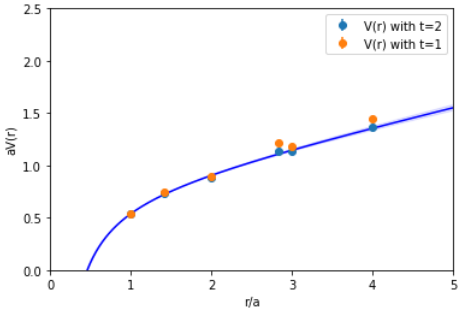
In [15]:

```
betas = [5.5, 5.6, 5.7, 5.8, 5.9]
c1s = [-0.01,-0.005,0.0,0.001]
lattice_spacings = {}
for beta in betas:
    for c1 in c1s:
        lattice_spacings[(beta,c1)] = lattice_spacing([f"8c8_PR_{beta}_{c1}/su3.{it}" for it in range(105,300,5)])
```

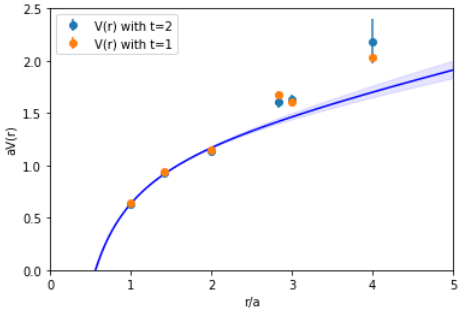
```
8c8_PR_5.5_-0.01/su3.295
8c8_PR_5.5_-0.01/su3.295
0.15810282203051035 0.003498859247574835
```



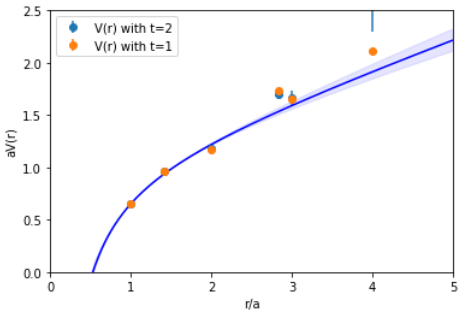
8c8_PR_5.5_-0.005/su3.295
8c8_PR_5.5_-0.005/su3.295
0.1868994068833518 0.004208694244922055



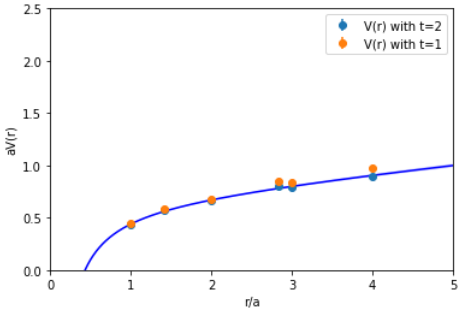
8c8_PR_5.5_0.0/su3.295
8c8_PR_5.5_0.0/su3.295
0.21718319816941065 0.013083434554633596



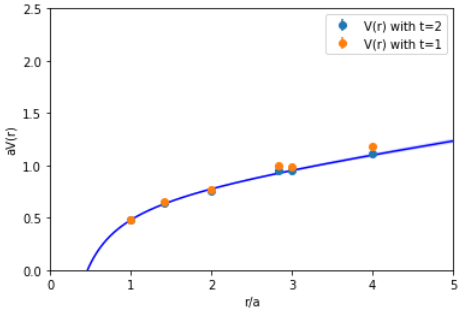
8c8_PR_5.5_0.001/su3.295
8c8_PR_5.5_0.001/su3.295
0.2539060665234701 0.010557821408040645



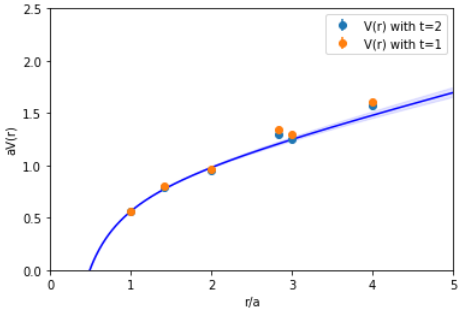
8c8_PR_5.6_-0.01/su3.295
8c8_PR_5.6_-0.01/su3.295
0.12164264923548306 0.00292567534000254



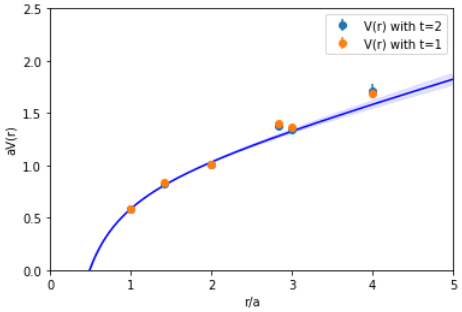
8c8_PR_5.6_-0.005/su3.295
8c8_PR_5.6_-0.005/su3.295
0.15047127976527147 0.0034932025893514157



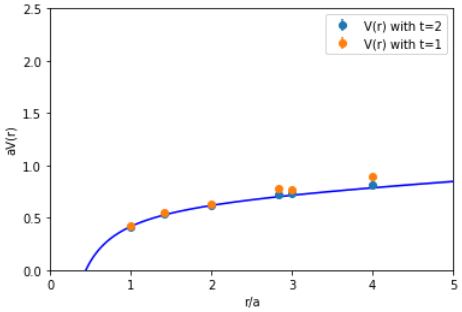
8c8_PR_5.6_0.0/su3.295
8c8_PR_5.6_0.0/su3.295
0.20086459794990957 0.00634641716209961



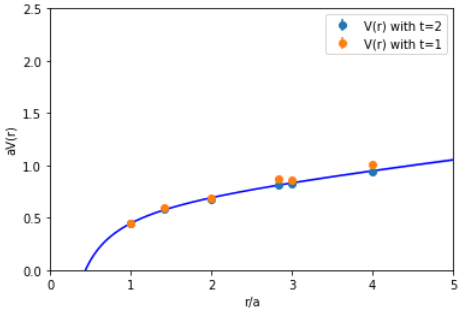
8c8_PR_5.6_0.001/su3.295
8c8_PR_5.6_0.001/su3.295
0.2138736834711988 0.006997106174368788



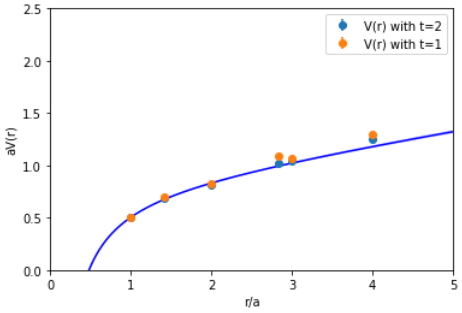
8c8_PR_5.7_-0.01/su3.295
8c8_PR_5.7_-0.01/su3.295
0.09192037765225937 0.0026468316372939005



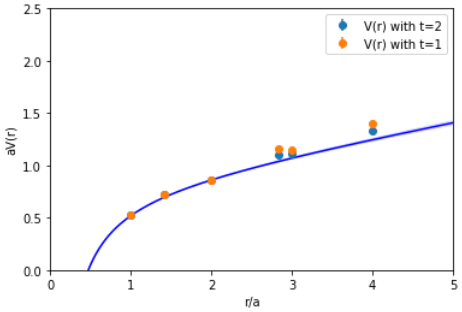
8c8_PR_5.7_-0.005/su3.295
8c8_PR_5.7_-0.005/su3.295
0.1293335414227953 0.0024809239670201167



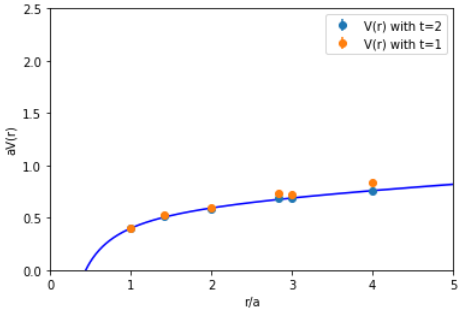
8c8_PR_5.7_0.0/su3.295
8c8_PR_5.7_0.0/su3.295
0.1573308742079209 0.0025845724865758432



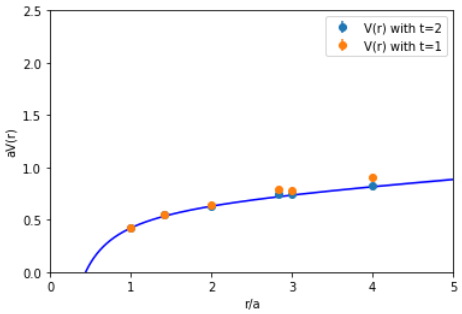
8c8_PR_5.7_0.001/su3.295
8c8_PR_5.7_0.001/su3.295
0.1687132194522321 0.003445718886308644



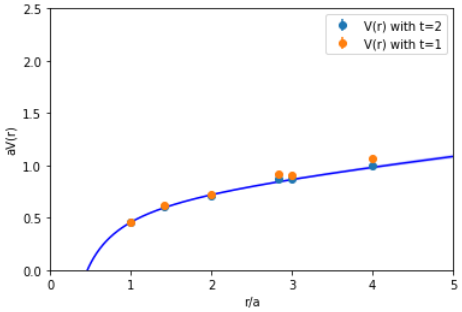
8c8_PR_5.8_-0.01/su3.295
8c8_PR_5.8_-0.01/su3.295
0.09203400893716623 0.0029059888297853723



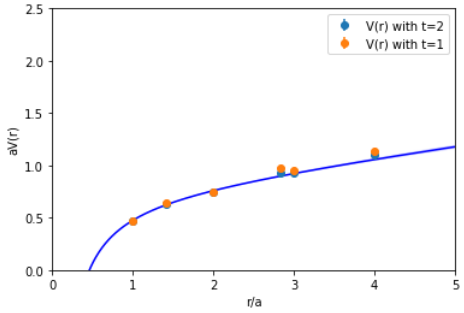
8c8_PR_5.8_-0.005/su3.295
8c8_PR_5.8_-0.005/su3.295
0.1006433813360252 0.0025320263861270975



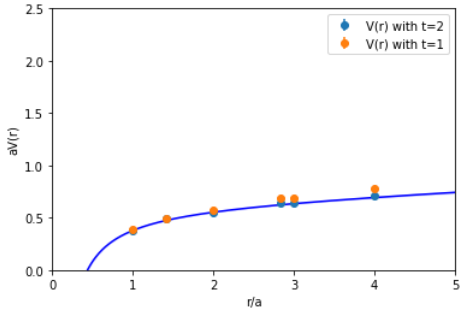
8c8_PR_5.8_0.0/su3.295
8c8_PR_5.8_0.0/su3.295
0.1294152500355272 0.0033037499656259793



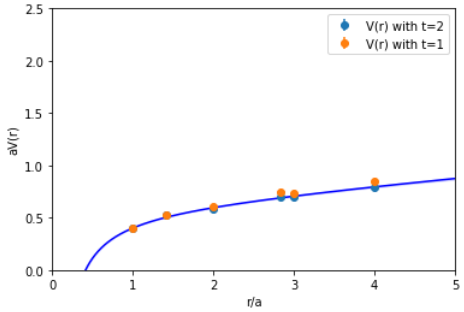
8c8_PR_5.8_0.001/su3.295
8c8_PR_5.8_0.001/su3.295
0.14188732621769773 0.003144769408094298



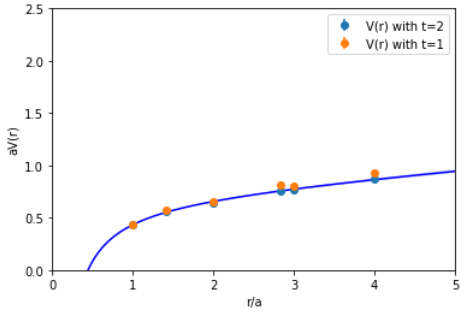

```
8c8_PR_5.9_-0.01/su3.295
8c8_PR_5.9_-0.01/su3.295
0.08010883389432717 0.0029945879102213032
```



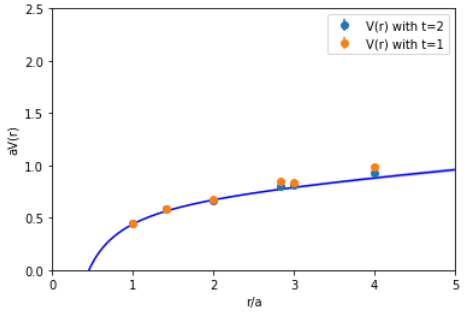

```
8c8_PR_5.9_-0.005/su3.295
8c8_PR_5.9_-0.005/su3.295
0.10991342324028512 0.0024540176462637463
```




```
8c8_PR_5.9_0.0/su3.295
8c8_PR_5.9_0.0/su3.295
0.10996826346055882 0.003386320680349076
```




```
8c8_PR_5.9_0.001/su3.295
8c8_PR_5.9_0.001/su3.295
0.10904119309010185 0.003992690555767243
```

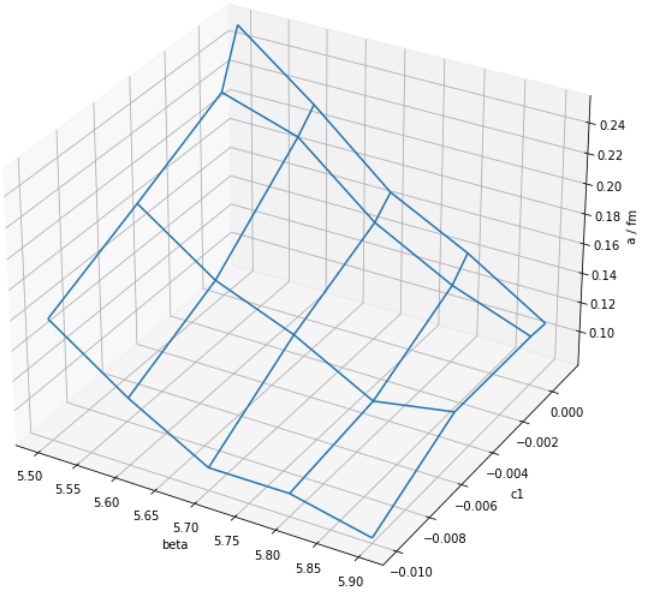


In [52]:

```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')

ax.set_xlabel('beta')
```

```
ax.set_ylabel('c1')
ax.set_zlabel('a / fm')
X = np.array([[beta for beta in betas] for c1 in c1s])
Y = np.array([[c1 for beta in betas] for c1 in c1s])
Z = np.array([[lattice_spacings[(beta, c1)][0] for beta in betas] for c1 in c1s])
ax.plot_wireframe(X, Y, Z)

print("No error bars here as they are difficult to see and errors are sufficiently small")
plt.show()
```

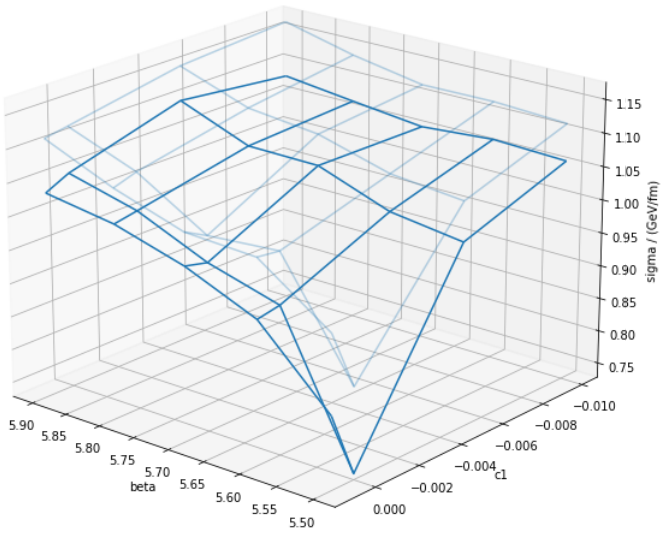
```
No error bars here as they are difficult to see and errors are sufficiently small
```



In [65]:

```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')

ax.set_xlabel('beta')
ax.set_ylabel('c1')
ax.set_zlabel('sigma / (GeV/fm)')
X = np.array([[beta for beta in betas] for c1 in c1s])
Y = np.array([[c1 for beta in betas] for c1 in c1s])
Z = np.array([[
    lattice_spacings[(beta, c1)][5][2]*0.1973/lattice_spacings[(beta, c1)][0]**2.
    for beta in betas]
    for c1 in c1s
])
Z_err = np.array([[
    lattice_spacings[(beta, c1)][6][2,2]**0.5*0.1973/lattice_spacings[(beta, c1)][0]**2.
    for beta in betas]
    for c1 in c1s
])

print("Note the rotated angle.  Continuum limit now in the back.  1sigma shifted as transparent.")
ax.view_init(elev=20., azim=130)
ax.plot_wireframe(X, Y, Z)
ax.plot_wireframe(X, Y, Z + Z_err, alpha=0.3)
plt.show()
```

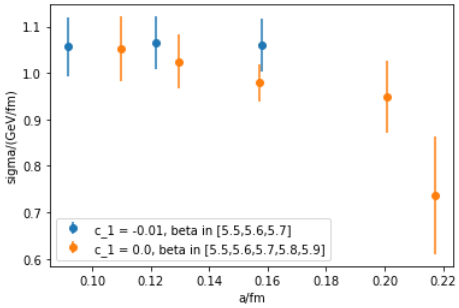
```
Note the rotated angle.  Continuum limit now in the back.  1sigma shifted as transparent.
```



The additional parameter c_1 leads to the same theory in the continuum limit but may reduce/enhance discretization errors. The additional operator corresponding to c_1 is therefore irrelevant to the physics of the continuum limit.

We also show the string tension σ as a function of a for the case $c_1 = -0.01$, which has a flatter continuum extrapolation compared to the $c_1 = 0$ case of chapter 7:

In [85]:
```
fig, ax = plt.subplots()
plt.xlabel("a/fm")
plt.ylabel("sigma/(GeV/fm)")
#plt.ylim(0.69,1.2)

# merge all
a_in_fm = []
sigma_in_GeV = []
sigma_in_GeV_err = []
for b in [5.5, 5.6, 5.7]:
    ame = lattice_spacings[(b, -0.01)]
    a_in_fm.append(ame[0])
    sigma_in_GeV.append(ame[5][2]/ame[0]**2.*0.1973)
    sigma_in_GeV_err.append(ame[6][2,2]**0.5/ame[0]**2.*0.1973)

ax.errorbar(a_in_fm, sigma_in_GeV, sigma_in_GeV_err, fmt="o", label=f"c_1 = {-0.01}, beta in [5.5,5.6,5.7]")

a_in_fm = []
sigma_in_GeV = []
sigma_in_GeV_err = []
for b in [5.5, 5.6, 5.7, 5.8, 5.9]:
    ame = lattice_spacings[(b, 0.0)]
    a_in_fm.append(ame[0])
    sigma_in_GeV.append(ame[5][2]/ame[0]**2.*0.1973)
    sigma_in_GeV_err.append(ame[6][2,2]**0.5/ame[0]**2.*0.1973)

ax.errorbar(a_in_fm, sigma_in_GeV, sigma_in_GeV_err, fmt="o", label=f"c_1 = {0.0}, beta in [5.5,5.6,5.7,5.8,5.9]")

plt.legend()
plt.show()
```



Homework: Repeat the discussion of the continuum limit of the Coulomb coefficient in the static quark potential with $c_1 = -0.01$ and $\beta \in [5.2, 5.3, 5.4, 5.5, 5.6]$. You may use the above code to generate the needed ensembles.

Continuum effective theory and Symanzik improvement

We constructed our lattice gauge theory by considering the similarities with a continuum theory with Lagrange density

$$\mathcal{L} = \frac{1}{2} \mathrm{Tr}\, F_{\mu\nu} F^{\mu\nu} \,. \tag{40}$$

In chapter 5, we considered the construction of this theory based on the symmetries of the continuum limit of QCD. Extending this line of reasoning, we can also write down a **continuum effective theory** that has the continuum symmetries for $a = 0$ but the reduced lattice symmetries for $a \neq 0$. We can do this by adding higher-dimensional operators to the action with explicit lattice spacing factors a such that the total mass dimension of the Lagrange density remains unchanged.

If we do this for concreteness in the Euclidean space version, we find to order a^2

$$\begin{aligned}\mathcal{L} = & \frac{1}{2}\sum_{\mu,\nu} \mathrm{Tr}\, F_{\mu\nu}^2 \\ & + r_1 a^2 \sum_{\mu,\nu} \mathrm{Tr}\, F_{\mu\nu} D_\mu^2 F_{\mu\nu} \\ & + r_2 a^2 \sum_{\mu,\nu} \mathrm{Tr}\, D_\mu F_{\nu\sigma} D_\mu F_{\nu\sigma} \\ & + r_3 a^2 \sum_{\mu,\nu} \mathrm{Tr}\, D_\mu F_{\mu\sigma} D_\nu F_{\nu\sigma} \\ & + \dots .\end{aligned} \tag{41}$$

What about terms linear in a? Such terms have to violate parity symmetry and are therefore absent. In general, we can only write down terms that are consistent with the symmetries of the lattice action.

We can now compute a sufficient number of observables both in lattice gauge theory and using such a continuum effective theory to determine the corresponding coefficients r_i explicitly. Since both theories need to yield the same results for all observables, we first see that pure gauge theory on the lattice does not have linear a errors und that by modifying the lattice gauge action, we may be able to set $r_i = 0$ in the corresponding continuum effective action. Adding additional operators to the lattice action to achieve this is also referred to as **Symanzik improvement**. If one were to perform a tree-level perturbation theory calculation, one finds that it would be sufficient to set $c_1 = -1/12$ in our above example to remove all order a^2 errors. Of course, loop effects violate this statement such that there are $\alpha_s a^2$ errors left with $\alpha_s = g^2/(4\pi)$.

If we consider matrix elements of operators different than the Hamiltonian of the system, we also need to consider a continuum effective version of this operator with higher-dimensional terms added to account for the effects of higher powers in a. Therefore it is not sufficient to improve the action, one also has to improve the operators for which one intends to measure matrix elements.

Finally, when we perform the matching between the continuum effective theory and a lattice theory, we find that the coefficients r_i can depend on $\alpha_s(a)$ themselves. Since the coupling constant itself is scale dependent, a general functional ansatz for a continuum extrapolation in pure gauge theory is

$$f(a) = f_0 + a^2 f_1 + a^2 \alpha_s(a) f_2 + O(a^4, a^2\alpha_s^2)\,. \tag{42}$$

Renormalization group

We have already seen above that adding a rectangle term to the Wilson gauge action was an **irrelevant** to the continuum limit of the theory but modified discretization errors and the mapping of β to the lattice spacing a. One could, however, also imagine theories for which additional parameters have a **relevant** role to play in the continuum limit of the theory. An example would be that after adding quarks with mass m to the theory, we must be able to find a family of well-defined continuum limits corresponding to the different physical quark masses. We will now bring more order to this discussion.

Consider integrating out short distances (**blocking**) or high momenta following a given scheme in our theory without violating its symmetries. The long-distance physics will not be modified by such changes on very short distances (see the universality discussion above) such that

$$a \to a' = \lambda a\,, \tag{43}$$
$$m_{\mathrm{lat}} \to m'_{\mathrm{lat}} = \lambda m_{\mathrm{lat}} \tag{44}$$

for all measured masses/correlation lengths m_{lat}. In addition discretization errors may also change such that more generally we can consider the effect of such a blocking operation to define a flow in the space of actions

$$S \to S' \tag{45}$$

where we could consider matching both actions S and S' to a continuum effective action

$$S = \int dx c_i O_i(x)\,, \tag{46}$$
$$S' = \int dx c'_i O_i(x) \tag{47}$$

and to consider the effect on the coefficients c_i of the corresponding operators O_i in these actions. Such a blocking operation therefore defines a flow of coefficients

$$c_i \to c'_i\,. \tag{48}$$

This operation has the structure of a semigroup and is referred to as the **Wilson renormalization group**. We can consider writing a corresponding flow with lattice spacing for each coefficient

$$c_i(a)\,. \tag{49}$$

If c_i keeps decreasing as $a \to 0$, we call O_i **irrelevant**, if c_i keeps increasing as $a \to 0$, we call O_i **relevant**. In other cases, we call O_i **marginal**.

The marginal/relevant parameters in the $a \to 0$ limit define the **critical surface** of the theory due to the second-order phase transition at $a \to 0$ described above. The fact that the critical surface is typically low-dimensional is they key to understanding **universality** of the quantum field theory continuum limit. All details of the lattice regulators must correspond to irrelevant operators and we can study different theories on the critical surface. (Again for pure QCD the critical surface is a single point, QCD with n_f quark flavors has a n_f dimensional critical surface corresponding to the physical quark masses.)

Finally, we note that the blocking operation can also have a **fixed point** with $\lambda = 1$. Such a fixed point may be found at $a = 0$ (**ultraviolet fixed point**) or $a \to \infty$ (**infrared fixed point**) or also in between. Note that the neighborhood of a fixedpoint can have attractive and repulsive sub-regions as $a \to 0$.

Continuum trajectory

Once we understand how many relevant/marginal operators we have in a theory, we can define a **continuum trajectory** as a scheme that fixes all coefficients of relevant/marginal operators by defining a condition for the lattice spacing a and a sufficient number of physical measurements to tune the remaining parameters. In pure QCD we only have one relevant operator and only need one condition for the lattice spacing. In QCD with one quark flavor m, we have two relevant/marginal operators and we need both a condition for the lattice spacing and an additional condition to tune the quark mass such as the ratio of a corresponding meson mass to the string tension.

We then can consider the subset of coefficients that lie on this continuum trajectory, which will define a one-dimensional flow in the lattice spacing a.