[x] Read through Luscher's note
[x] What do optimal coefficients look like?
[x] Where in the code are optimal coefficients computed?
[x] compare notation to paper https://arxiv.org/pdf/1806.04350.pdf
[x] work out the residues explicitly
[x] show explicitly why we're solving $(Q_{\mathrm{w}}^2 + \mu_{2k}^2)\,\psi_k = \Lambda_m^2 \phi$,
[x] Change notation of C^2 to some other thing, maybe Λ^2? So as not to confuse with the normalization constant for the Dirac operator
[x] Read through `fermion.qlua` & understand how it corresponds
[] Read through `D_deriv_D` , `D_deriv_D_prop`, and `DxD` in `inner_act` & understand how things are implemented
[] Read through `rational.qlua` to see how the coefficients are scaled; check my work.
[] Do I need to write my own multi-shift solver, or is there something readily available with Python that I can use? Look @ Tej's code in `solver.qlua`.

Effective fermion action with pseudofermions:

To simulate the action with pseudofermions, that is, to write

$$\det \mathcal{M}[U] = \int \mathcal{D}\,[\phi]\mathcal{D}\left[\phi^\dagger\right] \exp\left(-\phi^\dagger\,\mathcal{M}^{-1}[U]\,\phi\right), \tag{1}$$

we need to find the inverse and the determinant of the fermion matrix, $M = \mathcal{M}[U]$, for each configuration U. To guarantee that the matrix has a real spectrum, we generally write it as

$$K = M^\dagger M, \tag{2}$$

which is positive semi-definite by construction. Given a Wilson Dirac operator, for single flavor of fermions the corresponding effective fermion action reads

$$S_{\mathrm{F}}^{\mathrm{eff}}[U] = \phi^\dagger\,K^{-1/2}\,\phi = \phi^\dagger\,(a_{\mathrm{s}}a_{\mathrm{t}})^{-1}(D_{\mathrm{w}}^\dagger[U]\,D_{\mathrm{w}}[U])^{-1/2}\,\phi, \tag{3}$$

Where sum over lattice sites is implicit. Fields ϕ, $\phi^\dagger$ are dimensionless and have the same structure as their fermion counterparts: in the case of staggered $U(1)$ fermions, that's just spacetime indices.

Using γ_5-hermiticity of the Wilson Dirac operator, let us also define a Hermitian operator

$$Q_{\mathrm{w}} \equiv \gamma_5 D_{\mathrm{w}}, \tag{4}$$

so that $Q_{\mathrm{w}}^\dagger = Q_{\mathrm{w}}$. This definition will be necessary for the pseudofermion initialization step, and we'll use both D_{w} and Q_{w} throughout. In terms of Q_{w}, we have

$$K = (a_{\mathrm{s}}a_{\mathrm{t}})^2\,Q_{\mathrm{w}}^\dagger[U]\,Q_{\mathrm{w}}[U]. \tag{5}$$

Rational functions to approximate the square root:

In practice, to approximate the square root of an operator, we will use rational functions, writing $K^{-1/2} \approx R(K/\Lambda_m^2)\,\Lambda_m^2$ for some positive constant Λ_m^2, where Λ_m is on the order of the largest eigenvalue of K on a given configuration. It will be convenient to use slightly different approximations at different points in the (rational) HMC algorithm: pseudofermion sampling, MD evolution, and MCMC accept-reject step can employ slightly different approximations.

For domains over real numbers, the accuracy of a rational approximation of $y^{-\alpha}$ by a rational function $y(x)$ on some interval $\epsilon \le x \le 1$ may be expressed as

$$\delta = \max_{\epsilon \le y \le 1} |1 - y^\alpha R(y)|, \tag{6}$$

and present the same kinds of errors as the ones introduced by floating-point arithmetic. The maximum error Δ falls of exponentially with the degree d of the approximation used. Typically, a given rational function will be characterized by an effective range, outside of which the accuracy of approximation may decrease dramatically.

For rational approximations of Hermitian positive-definite matrices, the effective range of a rational approximation should encompass the eigenvalue range $[\lambda_{\min}, \lambda_{\max}]$. As the matrix K is a function of gauge field configurations, it may therefore be necessary during the simulation to dynamically scale rational approximations $R(K)$ to geometrically center the effective range of R on the eigenvalue range of K: to evaluate $f^{\alpha}\, R(f\, K)$ instead --- hence the factor Λ_m.

A rational function $R(y)$ of degree $[n,\, n]$ can be expressed as a product or in terms of partial fractions

$$R(y) = a_0 \prod_{k=1}^{n} \frac{y + a_{2k-1}}{y + a_{2k}} = a_0 \left(1 + \sum_{k=1}^{n} \frac{r_{2k}}{y + a_{2k}} \right) \tag{7}$$

To derive the factors r_{2k}, let $a_{2k-1} \to a_k$, $a_{2k} \to b_k$, and define

$$\begin{aligned} P_0(y) &= (y + a_1)(y + a_2) \ldots (y + a_n) \\ Q(y) &= (y + b_1)(y + b_2) \ldots (y + b_n). \end{aligned} \tag{8}$$

Here, $\deg(P_0(y)) = \deg(Q(y))$, and to apply a partial fractions decomposition directly, we need $\deg(P_0(y))y < \deg(Q(y))$. Note that since $\deg(P(y) - Q(y)) = n - 1$, we may write

$$P_0 = Q + ((-P_0 + y^n) + (Q - y^n)) = Q + (P_0 - Q), \tag{9}$$

so that

$$R(y) = a_0 \frac{P_0(y)}{Q(y)} + a_0 \frac{Q(y) + (P_0(y) - Q(y))}{Q(y)} = a_0 \left(1 + \frac{P_0(y) - Q(y)}{Q(y)} \right) \equiv a_0 \left(1 + \frac{P(y)}{B(y)} \right) \tag{10}$$

Since $\deg P(y) = n - 1 < \deg Q(y)$, now we may apply a partial fractions decomposition to $\frac{P(y)}{Q(y)}$, using the residues of this rational function:

$$\frac{P(y)}{Q(y)} = \sum_{k=1}^{n} \frac{P(-b_k)}{Q'(-b_k)} \frac{1}{y - b_k} \tag{11}$$

where

$$\begin{aligned} \frac{P(-b_k)}{Q'(-b_k)} &= \frac{P^0(-b_k) - Q(-b_k)}{Q'(-b_k)} \\ &= \frac{\prod_{l=1}^{n}(-b_k + a_l) - \prod_{l=1}^{n}(-b_k + b_l)}{\prod_{l \neq k}^{n}(-b_k + b_l)} \\ &= \frac{\prod_{l=1}^{n}(-b_k + a_l)}{\prod_{l \neq k}^{n}(-b_k + b_l)} \\ &\to \frac{\prod_{l=1}^{n}(-a_{2k} + a_{2l-1})}{\prod_{l \neq k}^{n}(-a_{2k} + a_{2l})} \\ &\equiv r_{2k}. \end{aligned} \tag{12}$$

Finally we get the expression of the form

$$R(y) = a_0 \prod_{k=1}^{n} \frac{y + a_{2k-1}}{y + a_{2k}} = a_0 \left(1 + \sum_{k=1}^{n} \frac{r_{2k}}{y + a_{2k}} \right), \tag{13}$$

where

$$r_{2k} = \frac{\prod_{l=1}^{n}(-a_{2k} - a_{2l-1})}{\prod_{l\neq k}^{n}(-a_{2k} - a_{2l})} \tag{14}$$

Since $a_i \in \mathbb{R}$, we have $r_i \in \mathbb{R}$. Similar decompositions can be introduced over complex numbers and extended to the case of matrix. This formL is convenient especially when computing the pseudofermion contribution to the driving force in the MD evolution, or when sampling pseudofermions.

There exist analytic solutions for the optimal form of the coefficients a_i for $\alpha = 1/2$ --- that is, for $r(y)$ approximating $\frac{1}{\sqrt{y}}$, --- of degrees $[n,\, n]$ and $[n-1,\, n]$. They are referred to as Zolotarev rational approximations. The minimizing coefficients are all positive. In particular, for the approximation of degree $[n, n]$, they satisify

$$a_1 > a_2 > \ldots > a_{2n} > 0. \tag{15}$$

Explicitly, optimal coefficients in the Zolotarev rational approximation involve the Jacobi elliptic functions $\mathrm{sn}(u,\, k)$, $\mathrm{cn}(u,\, k)$, and the complete elliptic integral $K(k)$:

$$\begin{aligned} a_r &= \frac{1-c_r}{c_r}, \quad r = 1,\, 2,\, \ldots,\, 2n, \\ a_0 &= \frac{2\sqrt{\delta}}{d}\,\frac{c_1 c_3 \ldots c_{2n-1}}{c_2 c_4 \ldots c_{2n}}, \end{aligned} \tag{16}$$

with the error δ given by

$$\delta = \frac{d^2}{(1+\sqrt{1-d^2})^2}, \tag{17}$$

where

$$\begin{aligned} d &= k^{2n+1}(c_1 c_3 \ldots c_{2n-1})^2 \\ c_r &= \mathrm{sn}^2(rv,\, k), \quad r = 1,\, 2,\, \ldots,\, 2n, \\ v &= \frac{K(k)}{2n+1}, \\ k &= \sqrt{1-\epsilon}, \end{aligned} \tag{18}$$

and ϵ is the smallest eigenvalue of $(a_\mathrm{s}\, a_\mathrm{t})\, Q_\mathrm{w}^2[U]/\Lambda_m^2$, while Λ_m^2 is the largest eigenvalue of $(a_\mathrm{s}\, a_\mathrm{t})\, Q_\mathrm{w}^2[U]$.

This is very nice: say the spectrum of Q_w^2 is fully contained in $[\epsilon\Lambda_m^2, \Lambda_m^2]$ for some positive constant Λ_m^2. Then we want to obtain a rational approximation of $Q_\mathrm{w}^2/\Lambda_m^2$, which will have poles at $-a_{2k}$, so that Q_w^2 has poles at $-\mu_{2k}^2 \equiv -\Lambda_m^2 a_{2k}$. For the Zolotarev rational approximation, we now that these poles are positive, decreasing with the increasing order of the approximation. Since in the pseudofermion action, we are looking to evaluate

$$S_\mathrm{F}^\mathrm{eff}[U] = \phi^\dagger\, K^{-1/2}\, \phi = \phi^\dagger\, (a_\mathrm{s} a_\mathrm{t})^{-1} (Q_\mathrm{w}^\dagger[U]\, Q_\mathrm{w}[U])^{-1/2}\, \phi \approx \phi^\dagger\, R(K/\Lambda_m^2)\, \Lambda_m^{-1} \phi, \tag{19}$$

which requires finding $R(K/\Lambda_m^2)\phi$.

$R(K/\Lambda_m^2)$ approximates $(a_\mathrm{t} a_\mathrm{s})^{-1}((Q_\mathrm{w}^\dagger[U]/\Lambda_m)\,(Q_\mathrm{w}[U]/\Lambda_m))^{-1/2} = \Lambda_m\, |a_\mathrm{t} a_\mathrm{s} Q_\mathrm{w}[U]|^{-1}$; the poles of the rational approximation are given by $-\mu_{2k}^2/\Lambda_m^2$; we may think of it as

$$R(K/\Lambda_m^2) = a_0 \left(1 + \sum_{k=1}^{n} \Lambda_m^2 r_{2k}\, (K + \mu_{2k}^2)^{-1}\right), \tag{20}$$

with

$$r_{2k} = \frac{\prod_{l=1}^{n}(-a_{2k} - a_{2l-1})}{\prod_{l\neq k}^{n}(-a_{2k} - a_{2l})} \tag{21}$$

so that finding $R(K/\Lambda_m^2)\phi$ essentially amounts to solving the equations

$$\left((a_\mathrm{s}a_\mathrm{t})^2 Q_\mathrm{w}^2[U] + \mu_{2k}^2\right)\psi = \phi, \tag{22}$$

where $\mu_r = \Lambda_m\sqrt{a_r} > 0$. The higher the order of the approximation, the larger the number of equations that need to be solved simultaneously. Because the right-hand side of all equations is the same, numerically this may be done with a multi-mass solver. Since masses get small at higher order of Zolotarev approximation, with growing k the equations get harder to solve.

With solutions to these equations given by $\psi_{2k} \equiv \psi(U,\, \mu_{2k}^2)$, the effective fermion action assumes the form

$$\begin{aligned} S_\mathrm{F}^\mathrm{eff}[U] &\approx \phi^\dagger\, R(K/\Lambda_m^2)\, \Lambda_m^{-1}\phi \\ &= \phi^\dagger \frac{a_0}{\Lambda_m}\left(\phi + \sum_{k=1}^{n} \Lambda_m^2 r_{2k}\, \psi(U,\, \mu_{2k},\, \Lambda_m)\right) \\ &= \phi^\dagger \frac{a_0}{\Lambda_m}\left(\phi + \sum_{k=1}^{n} \Lambda_m^2 r_{2k}\, \psi_{2k}\right). \end{aligned} \tag{23}$$

In `lattlib`, rescaled constant $\frac{a_0}{\Lambda_m}$, rescaled residues $\Lambda_m^2 r_{2k}$, and rescaled poles/masses μ_{2k}^2 and the error δ are returned as `a0, r, musq, delta` from a function call `negative_sqrt_coeffs(eps * Lambdasq_m, Lambdasq_m, n)`.

Fermions $\psi_{2k} \equiv \psi(U,\, \mu_{2k})$ can be obtained a call to `stupid_multishift_cg(K, musq, phi)` , where `K` is the K matrix provided as a `bsr` (block-sparse-row) matrix, `musq` are the μ_{2k}^2, and `phi` is the pseudofermion field ϕ.

To compute $S_\mathrm{F}^\mathrm{eff}[U]$, this inversion is done inside the call to `pf_action_r(M, Mdag, a0, r, musq, phi)`. Here $K = M^\dagger M$.

In `rhmc-qlua`, contribution at each site is computed via the function `actionDensity` which takes U as an argument `U`. The whole action, computed via the function `action` (with the same signature), is computed at `actionDensity(U):sum()`.

The contribution at each site n in `actionDensity` is computed as

$$\begin{aligned} (S_\mathrm{F}^\mathrm{eff}[U])_n &= \phi_n^\dagger \frac{a_0}{\Lambda_m}\left(\phi_n + \sum_{k=1}^{n} r_{2k}\, (\psi_k)_n\right) \\ &= \sum_{\alpha,\, a} [\phi_n^*]_a^\alpha \underbrace{\left[\frac{a_0}{\Lambda_m}\left(1 + \sum_{k=1}^{n} \Lambda_m^2 r_{2k}\, (K + \mu_{2k}^2)^{-1}\right)\phi_n\right]_a^\alpha}_{\text{irsq\_phi(U)(phi)}}, \end{aligned} \tag{24}$$

awhich is an element-wise multiplication of two vectors which can be computed with `qcd.dot` of ϕ stored as `phi` and the second vector, called `irsq_phi`, computed as `irsq(U)(phi)`.

The function call `irsq(U)` returns a function f. The returned function f is produced by calling `rDxD(U, coeffs_irsq, eps)` , where `eps` is a variable for inverter's precision. In turn, this function, if the code is run on CPUs, calls `rDxD_CPU(U, coeffs_irsq, eps)`. Finally, that call returns the desired inverter function $f = f(\phi)$, taking as argument an inversion source `phi`. Given `phi`, it sums over k, adding up the contributions as

$$f(\phi) = \underbrace{\frac{a_0}{\Lambda_m}}_{\text{coeffs\_irsq.a[0]}} \times \phi + \sum_{k=1}^{n} \underbrace{\frac{a_0}{\Lambda_m} \times \Lambda_m^2 r_{2k}}_{\text{coeffs\_irsq.a[k]}} \times \underbrace{(K + \mu_{2k}^2)^{-1}}_{\text{xs[k]}} \times \phi, \tag{25}$$

Where `xs` stores the output of a call to

```
xs = solver.multshift_CG(
        inner_act.DxD(U),
        coeffs_irsq.b,
        eps,
        max_iter,
        should_print
    )
```

Coefficients `coeffs_irsq.b` most likely store $\mu_{2k} = \Lambda_m^2\, a_{2k}$?

Generate pseudofermion fields:

Since the pseudofermions are distributed according to

$$P[\phi] \propto \exp\left(-\phi^\dagger (A\phi)\right), \tag{26}$$

where

$$A = K^{-1/2}(U) = (a_\mathrm{s} a_\mathrm{t})^{-1} \left(D_\mathrm{w}^\dagger[U]\, D_\mathrm{w}[U]\right)^{-1/2} = (a_\mathrm{s} a_\mathrm{t})^{-1} \left(Q_\mathrm{w}^\dagger[U]\, Q_\mathrm{w}[U]\right)^{-1/2}, \tag{27}$$

Use

$$\begin{gathered} P[\phi] = P[\chi] \det\left[\frac{\mathrm{d}\phi(\chi)}{\mathrm{d}\chi}\right] \propto \exp\left(-\chi^\dagger \chi\right) \det\left[\frac{\mathrm{d}\phi(\chi)}{\mathrm{d}\chi}\right], \\ \chi(\phi) = K(U)^{-1/4}\,\phi \;\Rightarrow\; \phi(\chi) = K(U)^{1/4}\,\chi. \end{gathered} \tag{28}$$

In this step, gauge fields U are held fixed, and so do not change with χ: $\det\left[\frac{\mathrm{d}\phi(\chi)}{\mathrm{d}\chi}\right] = \det\left[K(U)^{1/2}\right]$. The Jacobian does not depend on χ, and so the transformed probability measure does not depend on χ, and can be safely ignored (as a counter-example think of generating numbers uniformly distributed on the unit sphere from Gaussian-distributed Cartesian components of the vector; there, the Jacobian depends on where you are on the sphere).

Because of this, we can draw χ from $P[\chi]$, and then find ϕ as $\phi = K(U)^{1/4}\chi$.

For a general case, denote this as

$$\phi_i = B\chi_i, \tag{29}$$

where a representative ensemble of pseudofermion fields ϕ_i, with $i = 1, \ldots, N$, is drawn. The operator B has to satisfy

$$A = (B^\dagger B)^{-1} \tag{30}$$

In practice, we will use a rational-function approximation $\Lambda_m^{1/2} S[K/\Lambda_m^2] \approx B$, such that $\Lambda_m\, S^\dagger[K/\Lambda_m]\, S[K/\Lambda_m^2] \approx \Lambda_m (K/\Lambda_m^2)^{+1/2} = \Lambda_m\, R^{-1}(K/\Lambda_m^2)$. Thus, we want

$$S^\dagger(K/\Lambda_m^2)\, S(K/\Lambda_m^2) = R^{-1}(K/\Lambda_m^2). \tag{31}$$

For that one may take

$$
\begin{aligned}
S(K/\Lambda_m^2) &= a_0^{-1/2}\frac{(a_s a_t)(Q_w[U]+i\mu_2)\dots((a_s a_t)\,Q_w[U]+i\mu_{2n})}{(a_s a_t)(Q_w+i\mu_1)\dots((a_s a_t)\,Q_w[U]+i\mu_{2n-1})} \\
&= a_0^{-1/2}(1+\sum_{k=1}^{n}\Lambda_m\,(i\,r'_{2k-1})((a_s a_t)\,Q_w[U]+i\mu_{2k-1})^{-1}) \\
&= a_0^{-1/2}\left(1+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k-1})((a_s a_t)\,Q_w[U]-i\mu_{2k-1})(K+\mu_{2k-1}^2)^{-1}\right),
\end{aligned}
\tag{32}
$$

where

$$
r'_{2k-1} = \frac{\prod_{l=1}^{n}(-\sqrt{a_{2k-1}}-\sqrt{a_{2l}})}{\prod_{l\neq k}^{n}(-\sqrt{a_{2k-1}}-\sqrt{a_{2l-1}})}.
\tag{33}
$$

Thus, initializing pseudofermions also amounts to the application of a multi-mass solver, to find

$$
((a_s a_t)^2\,Q_w^2[U]+\mu_{2k-1}^2)\,\zeta = \chi.
\tag{34}
$$

Call solutions to these equations $\zeta_{2k-1}=\zeta(U,\mu_{2k-1})$. Then, with χ fields initialized, we can obtain the pseudofermion fields ϕ as

$$
\begin{aligned}
\phi &= B\chi = \Lambda_m^{1/2}S[K/\Lambda_m^2]\,\chi \\
&= \sqrt{\frac{\Lambda_m}{a_0}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k-1})((a_s a_t)\,Q_w[U]-i\mu_{2k-1})\left[(K+\mu_{2k-1}^2)^{-1}\chi\right]\right) \\
&= \sqrt{\frac{\Lambda_m}{a_0}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k-1})((a_s a_t)\,Q_w[U]-i\mu_{2k-1})\,\zeta(U,\mu_{2k-1})\right) \\
&= \sqrt{\frac{\Lambda_m}{a_0}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k-1})((a_s a_t)\,Q_w[U]-i\mu_{2k-1})\,\zeta_{2k-1}\right) \\
&= \sqrt{\frac{\Lambda_m}{a_0}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k-1})(-i\,\mu_{2k-1})\,\zeta_{2k-1}+\sum_{k=1}^{n}[(a_s a_t)\,Q_w[U]]\left[(i\,\Lambda_m\,r'_{2k-1})\,\zeta_{2k-1}\right]\right).
\end{aligned}
\tag{35}
$$

In `lattlib`, rescaled constant $\sqrt{\frac{\Lambda_m}{a_0}}$, rescaled residues $i\Lambda_m r'_{2k}$, and rescaled poles/masses $i\mu_{2k-1}$ and the error δ' are returned as `a0, r, musq, delta` from a function call `negative_sqrt_coeffs(eps * Lambdasq_m, Lambdasq_m, n)`.

Fermions $\zeta_{2k-1}\equiv\zeta(U,\,\mu_{2k-1}^2)$ can be obtained a call to `stupid_multishift_cg(K, musq, phi)` , where `K` is the K matrix provided as a `bsr` (block-sparse-row) matrix, `musq` are the μ_{2k-1}^2, and `phi` is the lattice complex field χ.

Pseudofermons ϕ are with a function call to `sample_pf_r(M, Mdag, a0, r, musq, ia0, ir, imu)`, which does the inversions.

Another way to obtain ϕ is to express it as

$$
\begin{aligned}
\phi &= K^{+1/4}\chi = K^{1/2}(K^{-1/4}\chi) \\
&= [(a_s a_t)|Q_w[U]|]\,\Lambda_m^{-1/2}\,S^{-1}[K/\Lambda_m^2]\,\chi \\
&= [(a_s a_t)|Q_w[U]|]\,\sqrt{\frac{a_0}{\Lambda_m}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k})\,((a_s a_t)\,Q_w[U]-i\mu_{2k})\left[(K+\mu_{2k}^2)^{-1}\chi\right]\right) \\
&= [(a_s a_t)|Q_w[U]|]\,\sqrt{\frac{a_0}{\Lambda_m}}\left(\chi+\sum_{k=1}^{n}(i\,\Lambda_m\,r'_{2k})\,((a_s a_t)\,Q_w[U]-i\mu_{2k})\,\zeta_{2k}\right).
\end{aligned}
\tag{36}
$$

In `rhmc-qlua`, ϕ is obtained via the function `initTraj` which takes U and a random Gaussian field generator as arguments `U` and `rndGen`. The generator is used to produce χ, stored in variable `chi`, and ϕ is obtained as `phi` as

$$\phi = [(a_{\rm s}a_{\rm t})\,|Q_{\rm w}[U]|] \left[\underbrace{\sqrt{\frac{a_0}{\Lambda_m}}}_{\text{coeffs\_r.a[0]}}\chi + \underbrace{\sum_{k=1}^{n} \underbrace{\left(\Lambda_m\sqrt{\frac{a_0}{\Lambda_m}}\,r'_{2k}\right)}_{\text{coeffs\_r.a[k]}} \times \underbrace{(\mu_{2k})}_{\text{coeffs\_r.b[k]}} \times \underbrace{(K+\mu_{2k}^2)^{-1}\chi}_{\text{xs[k]}}}_{\chi_1} \right. \\ \left. + \underbrace{\sum_{k=1}^{n} i \times [(a_{\rm s}a_{\rm t})\,Q_{\rm w}] \left[\underbrace{\left(\Lambda_m\sqrt{\frac{a_0}{\Lambda_m}}\,r'_{2k}\right)}_{\text{coeffs\_r.a[k]}} \times \underbrace{(K+\mu_{2k}^2)^{-1}\chi}_{\text{xs[k]}} \right]}_{\chi_2} \right]. \tag{37}$$

The function call `r(U)` returns a function *g*. The returned *g* takes a pseudofermion field χ as an argument `chi`, and returns the whole expression on the right-hand side above.

(*Filtering*, incomplete section...)

Fluctuations in ϕ can be large if the condition number of $K(U)$ is large, which would happen for small bare fermion masses. To avoid this, one possible solution is to factorize the determinant

$$|\det Q_{\rm w}|^2 = \det(Q_{\rm w}^\dagger Q_{\rm w} + \mu^2) \times \det\left(\frac{Q_{\rm w}^\dagger Q_{\rm w}}{Q_{\rm w}^\dagger Q_{\rm w} + \mu^2}\right), \tag{38}$$

to separate the contribution of the eigenvalues larger than μ^2 from those smaller than μ^2.

Compute pseudofermion contribution to the driving force:

If approximating the effective fermion acting with pseudofermions, we may express the driving force as

$$\begin{aligned} (F[U,\phi])^\mu &= \partial^\mu (S_{\rm G}[U] + \phi^\dagger\, K^{-1/2}[U]\,\phi) \\ &= \frac{\partial}{\partial\,\omega_\mu}\Big(S_{\rm G}[e^{i\omega}U] + \phi^\dagger\, K^{-1/2}[e^{i\omega}U]\,\phi\Big)\Big|_{\omega=0} \\ &= \frac{\partial}{\partial\,\omega_\mu}\Big(S_{\rm G}[e^{i\omega}U] + \phi^\dagger\,\Lambda_m^{-1}\,K^{-1/2}[e^{i\omega}U/\Lambda_m^2]\,\phi\Big)\Big|_{\omega=0} \end{aligned} \tag{39}$$

The fermion contribution may be evaluated with a partial fractions expansion as

$$\begin{aligned} (F_{\rm F}[U,\phi])^\mu &= \frac{a_0}{\Lambda_m}\,\phi^\dagger \sum_k \frac{\partial}{\partial\omega_\mu}\left(1 + \frac{\Lambda_m^2 r_{2k}}{K+\mu_{2k}^2}\right)\phi \\ &= -\frac{a_0}{\Lambda_m}\,\phi^\dagger \sum_k (\Lambda_m^2\,r_{2k})\,(K+\mu_{2k}^2)^{-1}\,\left[(a_{\rm s}a_{\rm t})^2\frac{\partial\,Q_{\rm w}^2}{\partial\omega_\mu}[U]\right](K+\mu_{2k}^2)^{-1}\phi \end{aligned} \tag{40}$$

where

$$\begin{aligned}\frac{\partial Q_{\mathrm{w}}^2}{\partial \omega_\mu} &= \frac{\partial D_{\mathrm{w}}^\dagger}{\partial \omega_\mu}\gamma_5\gamma_5 D_{\mathrm{w}} + D_{\mathrm{w}}^\dagger \gamma_5\gamma_5 \frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu} \\ &= \frac{\partial Q_{\mathrm{w}}^\dagger}{\partial \omega_\mu}\gamma_5 D_{\mathrm{w}} + Q_{\mathrm{w}}^\dagger \gamma_5 \frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu} \\ &= \frac{\partial Q_{\mathrm{w}}}{\partial \omega_\mu}\gamma_5 D_{\mathrm{w}} + Q_{\mathrm{w}} \gamma_5 \frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}\end{aligned} \tag{41}$$

and, since $\frac{\partial D_{\mathrm{w}}^\dagger}{\partial \omega_\mu} = \frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}$, we have

$$\begin{aligned}(F_{\mathrm{F}}[U,\phi])^\mu &= -2 \times \frac{a_0}{\Lambda_m}\sum_k (\Lambda_m^2 r_{2k})\,(a_{\mathrm{s}}a_{\mathrm{t}})^2\,\mathrm{Re}\left[\psi_{2k}^\dagger\, Q_{\mathrm{w}}\gamma_5\,\frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}\psi_{2k}\right] \\ &= -2 \times \frac{a_0}{\Lambda_m}\sum_k (\Lambda_m^2 r_{2k})\,(a_{\mathrm{s}}a_{\mathrm{t}})^2\,\mathrm{Re}\left[\psi_{2k}^\dagger\, D_{\mathrm{w}}^\dagger[U]\,\left(\frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}[U]\,\psi_{2k}\right)\right] \\ &= -2 \times \frac{a_0}{\Lambda_m}\sum_k (\Lambda_m^2 r_{2k})\,(a_{\mathrm{s}}a_{\mathrm{t}})^2\,\mathrm{Re}\left[(D_{\mathrm{w}}[U]\,\psi_{2k})^\dagger\,\left(\frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}[U]\,\psi_{2k}\right)\right]\end{aligned} \tag{42}$$

In `lattlib`, rescaled constant $\frac{a_0}{\Lambda_m}$, rescaled residues $\Lambda_m^2 r_{2k}$, and rescaled poles/masses μ_{2k}^2 and the error δ are returned as `a0, r, musq, delta` from a function call `negative_sqrt_coeffs(eps * iLambdasq_m, Lambdasq_m, n)`.

Fermions $\psi_{2k} \equiv \psi(U,\,\mu_{2k})$ can be obtained a call to `stupid_multishift_cg(K, musq, phi)`, where `K` is the K matrix provided as a `bsr` (block-sparse-row) matrix, `musq` are the μ_{2k}^2, and `phi` is the pseudofermion field ϕ.

To compute $(F_{\mathrm{F}}[U,\phi])^\mu$, this inversion is done inside the call to `pf_force_r(M, Mdag, cfg, phi, kappa, a0, r, fermion_bc)`. Here $K = M^\dagger M$. The argument `cfg` is a gauge configuration field U.

In `rhmc-qlua`, the pseudofermion contribution to the driving force is computed via function `force`, taking in the gauge field U as an argument `U`. Contributions are aggregated for each k, and for each k and μ, represented as `k` and `mu`, the contribution is given by

$$(F_{\mathrm{F}}[U,\phi])_k^\mu = \underbrace{\left(\frac{a_0}{\Lambda_m}\Lambda_m^2\, r_{2k}\right)}_{\text{coeffs\_irsq.a[k]}} \times \left(-2 \times \underbrace{\mathrm{Re}\left[\psi_k^\dagger\, D_{\mathrm{w}}^\dagger\,\left(\frac{\partial D_{\mathrm{w}}}{\partial \omega_\mu}\psi_k\right)\right]}_{\text{Fi[mu]=inner\_act.D\_deriv\_D(U, psi)}}\right) \tag{43}$$

Accept-Reject Step

If using pseudofermions, accept if a random number $x \in [0,1)$ is smaller than

$$\exp\left(\frac{1}{2}P^2 - \frac{1}{2}P'^2 + S_{\mathrm{G}}(U) - S_{\mathrm{G}}(U') + \phi^\dagger\,(K^{-1/2}[U]\,\phi - K^{-1/2}[U']\,\phi).\right) \tag{44}$$

In the last step, $K^{-1/2}[U_n]\,\phi$ may be reused from the force computation for the final step of MD evolution.

In practice for large lattices, we will use a rational-function approximation of $K^{-1/2}$. It may differ from the one we use in the MD evolution step (in principle, we would want to be more precise to account for errors accumulated over MD evolution step at the accept-reject stage).

Finally, we could further weight the accept probability to account for the inaccuracies from using a determinant of a rational approximation: