Stout smearing for openQCD

Jonas Rylund Glesaaen March 2018

1 Parameters for stout smearing

The stout smearing module is activated by adding the following section to your program configuration files for the `qcd1, ym1, ms1, ms2` & `ms4` executables.

```
[Smearing parameters]
n_smear       2
rho_s         0.14
rho_t         0.00
gauge         0
fermion       1
```

The block has been made optional to be compatible with openQCD 1.6 configuration files. If no such section is specified, smearing will not be used. The parameters have the following interpretations:

`n_smear (integer)`
The number of smearing steps to apply to the gauge field. The same as n in eq. (6)

`rho_s (double)`
Parameter $\rho_{\mu\nu}$ as defined in eq. (1) for both μ and ν spatial directions ($\in \{1, 2, 3\}$)

`rho_t (double)`
Parameter $\rho_{\mu\nu}$ where either μ or ν is a temporal direction ($= 0$)

`gauge (boolean)`
Turns on or off smearing for the gauge action

`fermion (boolean)`
Turns on or off smearing for all the fermionic actions

One should note that one cannot set `rho_s == 0.0 && rho_t != 0.0` as this would never be what one wants to do.

2 Smearing of the gauge field

We will closely follow the equations and setup described in the paper by Morningstar and Peardon [2], with slight modifications to align with the design choices behind openQCD.

We start by defining the sum of staples

$$C_\mu = \sum_{\mu\neq\nu} \rho_{\mu\nu} \Big(U_\nu(x) U_\mu(x+\hat{\nu}) U_\nu^\dagger(x+\hat{\mu}) + U_\nu^\dagger(x-\hat{\nu}) U_\mu(x-\hat{\nu}) U_\nu(x-\hat{\nu}+\hat{\mu}) \Big). \tag{1}$$

We want to use these staples to smear the gauge field; however due to the fact that adding SU(3) matrices does not result in a new SU(3) matrix we need some way of creating one from C_μ,

$$\Omega_\mu(x) = C_\mu(x) U_\mu^\dagger(x) \tag{2}$$

$$X_\mu(x) = \mathcal{P}_{\mathfrak{su}(3)}\{\Omega_\mu(x)\} \tag{3}$$

Where the projection operator $\mathcal{P}$ takes an arbitrary complex 3×3 matrix and returns a traceless and anti-hermitian matrix

$$\mathcal{P}_{\mathfrak{su}(3)}\{M\} = \frac{1}{2}\Big(M - M^\dagger\Big) - \frac{1}{6}\operatorname{tr}\Big(M - M^\dagger\Big). \tag{4}$$

A single stout smearing step is then defined by

$$U_\mu^{(k+1)}(x) = \exp\left(X_\mu^{(k)}(x)\right) U_\mu^{(k)}(x), \tag{5}$$

where obviously $\exp(X) \in \mathrm{SU}(3)$, and thus so is every step $U^{(k)}$. The full *stout* link is then defined by applying this step n times

$$\tilde{U}_\mu(x) \equiv U_\mu^{(n)}(x), \tag{6}$$

where n is a parameter in the simulation setup.

2.1 *The matrix exponential*

To compute the matrix exponential we leverage the results from the Cayley-Hamilton (*CH*) theorem. For a traceless anti-hermitian matrix X, its characteristic polynomial is

$$X^3 + tX + id = 0, \tag{7}$$

where t and d are two real parameters

$$t = -\tfrac{1}{2}\,\mathrm{tr}\left(X^2\right), \qquad d = i\det X. \tag{8}$$

Recursively using the characteristic polynomial of a matrix to rewrite powers of it in terms of lower order powers, any analytic function can be rewritten in terms of three scalar coefficients

$$f(X) = f_0 I + f_1 X + f_2 X^2. \tag{9}$$

The coefficients are basis independent and only depend on the two parameters, t, d. Specifically for the exponential we introduce

$$\exp(X) \approx p_0 I + p_1 X + p_2 X^2, \tag{10}$$

where the p_i coefficients are defined in [1].

2.2 *Programming implementation*

We will go though some details of the programming implementation.

Cayley-Hamilton coefficients

The openQCD software already has an implementation of the matrix exponential utilising the Cayley-Hamilton theorem. The function in question is:

```
void expXsu3(double eps, su3_alg_dble *X, su3_dble *u);
```

which computes

$$u' = \exp(\epsilon X) u. \tag{11}$$

In the computation the program first normalises the matrix X by defining $Y = (0.5)^n X$ with n chosen so that $||Y|| < 1$. It then computes

$$\exp(Y) = p_{0,0} I + p_{1,0} Y + p_{2,0} Y^2 \equiv E, \tag{12}$$

and finally

$$\exp(X) = \begin{cases} (EE)^n & \text{if } n > 0, \\ E & \text{if } n = 0. \end{cases} \tag{13}$$

Unfortunately, using this method we lose the CH coefficients p_i, which are needed later in the computation of the Molecular Dynamics (*MD*) forces. This can be rectified by using the characteristic polynomial to carry out the matrix squaring rather than a normal matrix product. The equation of interest reads

$$E_2 \equiv (EE)^2 = (p_{0,0} I + p_{1,0} Y + p_{2,0} Y^2)^2 = p_{0,1} I + p_{1,1} Y + p_{2,1} Y^2 \tag{14}$$

which we can solve

$$
\begin{aligned}
p_{0,1} &= p_{0,0}^2 - 2id_Y p_{1,0} p_{2,0}, \\
p_{1,1} &= 2p_{0,0}p_{1,0} - id_Y p_{2,0}^2 - 2t_Y p_{1,0} p_{2,0}, \\
p_{2,1} &= 2p_{0,0}p_{2,0} + p_{1,0}^2 - t_Y p_{2,0}^2,
\end{aligned}
$$

where t_Y, d_Y are the coefficients of the CH polynomial for the rescaled matrix Y. The coefficients $p_{i,1}$ can then in turn be used to compute $p_{i,2}$, and so on until we have $p_{i,n}$ which give us

$$
\exp(X) = p_{0,n} I + p_{1,n} Y + p_{2,n} Y^2. \tag{15}
$$

The final step is to rescale the parameters, which will give us p_i

$$
d_X = 2^{3n} d_Y,\ t_X = 2^{2n} t_Y,\ p_0 = p_{0,n},\ p_1 = 0.5^n p_{1,n},\ p_2 = 0.5^{2n} p_{2,n}. \tag{16}
$$

In the program we use this recursive algorithm to define a variant of `expXsu3` which also returns the CH coefficients

```
void expXsu3_w_factors(double eps, su3_alg_dble *X, su3_dble *u,
                       ch_drv0_t *s_in)
```

Although the CH matrix exponential is already implemented in openQCD, and we use this method for exponentiating matrices, we will follow the derivation of [2] in the implementation of the MD forces, and we will therefore need to define a couple more quantities which will be reused later

$$
d_{\max} = 2\left(\frac{t}{3}\right)^{3/2}, \tag{17}
$$

$$
\theta = \arccos\left(d/d_{\max}\right), \tag{18}
$$

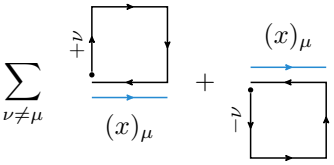
$$
u = \sqrt{\tfrac{1}{3}t}\cos\left(\tfrac{1}{3}\theta\right), \tag{19}
$$

$$
w = \sqrt{t}\sin\left(\tfrac{1}{3}\theta\right). \tag{20}
$$

Computation of the staples and multi-process communication

We will first go through how we compute the Ω matrix defined in eq. (2) before looking at how this is communicated to neighbouring processes. The matrix Ω is a sum of all closed staples for each link on the lattice:

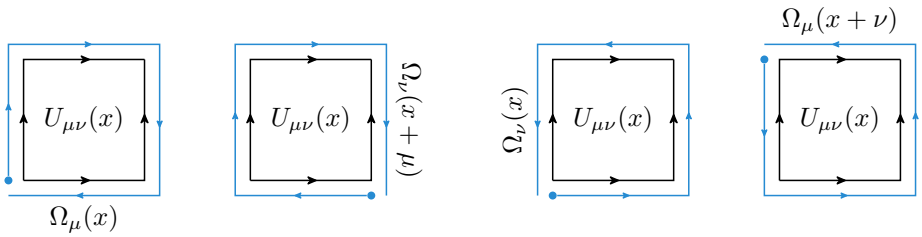
$$
\Omega_\mu(x) = \sum_{\nu \neq \mu}
$$



(21)

In the graphical notation the dot • indicates the starting point of the matrix multiplication and the blue line tells us for which link the contribution counts towards.

Due to the way the halo is stored it is more advantageous to loop over all plaquettes than to loop over all links; this will have the added benefit of reducing the number of computations as every plaquette contributes to the four Ω matrices on its boundary.



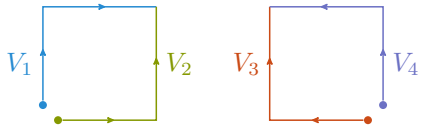
Thus for every plaquette we have the following additive contribution to the four neighbouring Ω matrices

$$\Omega_\mu(x) \mathrel{+}= U_\nu(x)U_\mu(x+\nu)U_\nu^\dagger(x+\mu)U_\mu^\dagger(x), \tag{22a}$$
$$\Omega_\nu(x+\mu) \mathrel{+}= U_\mu^\dagger(x)U_\nu(x)U_\mu(x+\nu)U_\nu^\dagger(x+\mu), \tag{22b}$$
$$\Omega_\nu(x) \mathrel{+}= U_\mu(x)U_\nu(x+\mu)U_\mu^\dagger(x+\nu)U_\nu^\dagger(x), \tag{22c}$$
$$\Omega_\mu(x+\nu) \mathrel{+}= U_\nu^\dagger(x)U_\mu(x)U_\nu(x+\mu)U_\mu^\dagger(x+\nu). \tag{22d}$$

We also see that both $\Omega_\mu(x)$ and $\Omega_\nu(x)$, and $\Omega_\nu(x+\mu)$ and $\Omega_\mu(x+\nu)$, share common factors, and therefore define

$$V_1 = U_\nu(x)U_\mu(x+\nu), \tag{23a}$$
$$V_2 = U_\mu(x)U_\nu(x+\mu), \tag{23b}$$
$$V_3 = U_\mu^\dagger(x)U_\nu(x), \tag{23c}$$
$$V_4 = U_\nu(x+\mu)U_\mu^\dagger(x+\nu), \tag{23d}$$

or graphically



which simplifies the eqs. (22)

$$\Omega_\mu(x) \mathrel{+}= V_1V_2^\dagger, \tag{24a}$$
$$\Omega_\nu(x+\mu) \mathrel{+}= V_3V_4^\dagger, \tag{24b}$$
$$\Omega_\nu(x) \mathrel{+}= V_2V_1^\dagger, \tag{24c}$$
$$\Omega_\mu(x+\nu) \mathrel{+}= V_3^\dagger V_4, \tag{24d}$$

and also minimises matrix multiplication.

After the program has looped over all plaquettes $(x)_{\mu\nu}$ of the form where x is in the bulk of the current sublattice and μ and ν are both positive directions, the only incomplete Ω matrices will be those close to the negative boundaries whose negative directional plaquettes are on the negative facing halo. However, because we include a positive facing halo for the Ω field the missing contribution will already have been computed on the positive halo of the neighbouring lattice in the negative direction. Thus by taking the Ω matrix stored on the positive facing halo and adding them to the corresponding links on the neighbouring lattice we finalise the computation. This is illustrated in figure 1.

3 Integration with Molecular Dynamics

To compute the HMC trajectory we need the MD force in order to update the gauge momenta. The gauge force is defined as:

$$\Sigma_\mu(x) = \left(\frac{\partial S[U]}{\partial U_\mu(x)}\right)^T. \tag{25}$$

However, since we smeared the gauge configuration in the previous section, in practice we only have access to the gauge force with respect to the stout links

$$\tilde{\Sigma}_\mu(x) = \left(\frac{\partial S_{\text{st}}[\tilde{U}]}{\partial \tilde{U}_\mu(x)}\right)^T \equiv \Sigma_\mu^{(n)}(x), \tag{26}$$

while what we need is

$$\Sigma_\mu(x) = \left(\frac{\partial S_{\text{st}}[\tilde{U}]}{\partial U_\mu(x)}\right)^T. \tag{27}$$

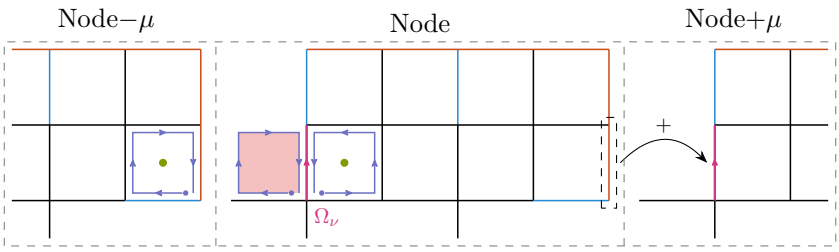


Figure 1: Figure showing a 4×2 two dimensional lattice and its neighbours. The links in black indicate links in the bulk, blue links are type 1 boundary links and orange links are type 2 boundary links. The figure illustrates the communication issue when computing Ω (here Ω_μ) on a link for which one or more contribution comes from links not available on the current sublattice (contribution filled with red). As one can see from the figure the contribution exists on a neighbouring sublattice and is stored in a type 2 boundary link for this particular example.

In this we will follow the steps of [2] to introduce a recursive mapping

$$\left\{\Sigma^{(k)}, U^{(k-1)}\right\} \longrightarrow \Sigma^{(k-1)}, \tag{28}$$

where

$$\Sigma^{(k)} = \left(\frac{\partial S_{\text{st}}[\tilde{U}]}{\partial U_\mu^{(k)}(x)}\right)^T. \tag{29}$$

The first step in this process is to compute the derivative of the action with respect to the fictitious simulation time of the HMC algorithm

$$\frac{\mathrm{d}}{\mathrm{d}\tau} S_{\text{st}}[\tilde{U}] = 2 \sum_{x,\mu} \operatorname{re}\operatorname{tr}\left\{\Sigma_\mu^{(k)} \frac{\mathrm{d}}{\mathrm{d}\tau} U_\mu^{(k)}(x)\right\}, \tag{30}$$

where $k \in [0, n]$. We continue by using the chain rule on smeared links with $k > 0$

$$\frac{\mathrm{d}}{\mathrm{d}\tau} U_\mu^{(k)}(x) = e^{X^{(k-1)}} \left(\frac{\mathrm{d}}{\mathrm{d}\tau} U_\mu^{(k-1)}(x)\right) + \left(\frac{\mathrm{d}}{\mathrm{d}\tau} e^{X^{(k-1)}}\right) U_\mu^{(k-1)}(x). \tag{31}$$

From now on we will simplify the notation by saying that (k) quantities are primed, and $(k-1)$ quantities are unprimed. We can compute the derivative of the matrix exponential by using the CH formula (10),

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}\tau} e^X &= \frac{\mathrm{d}}{\mathrm{d}\tau}\left(p_0 I + p_1 X + p_2 X^2\right), \\ &= \dot{p_0} I + \dot{p_1} X + \dot{p_2} X^2 + p_1 \dot{X} + p_2 \dot{X} X + p_2 X \dot{X}, \end{aligned} \tag{32}$$

where $\dot{x} \equiv \frac{\mathrm{d}x}{\mathrm{d}\tau}$. The derivatives of the CH parameters can be written in terms of derivatives of the two characteristic parameters t and d which gives

$$\dot{p_j} = \beta_{1,j} \operatorname{tr}\left(X\dot{X}\right) + \beta_{2,j} \operatorname{tr}\left(X^2 \dot{X}\right). \tag{33}$$

The coefficients $\beta_{i,j}$ are

$$\beta_{1,j} = -\frac{2u\rho_j^{(1)} + (3u^2 - w^2)\rho_j^{(2)} - 2(15u^2 + w^2)p_j}{2(9u^2 - w^2)^2}, \tag{34}$$

$$\beta_{2,j} = i\frac{\rho_j^{(1)} - 3u\rho_j^{(2)} - 24up_j}{2(9u^2 - w^2)^2}, \tag{35}$$

and the ρ's

$$\rho_0^{(1)} = 2\Big(u + i(u^2 - w^2)\Big)e^{2iu} + 2e^{-iu}\Big\{4u(2 - iu)\cos w + i\big(9u^2 + w^2 - iu(3u^2 + w^2)\big)\xi_0(w)\Big\}, \tag{36}$$

$$\rho_1^{(1)} = -2i(1 + 2iu)e^{2iu} + ie^{-iu}\Big\{2(1 - iu)\cos w - i\big(6u + i(w^2 - 3u^2)\big)\xi_0(w)\Big\}, \tag{37}$$

$$\rho_2^{(1)} = -2e^{2iu} + ie^{-iu}\big\{3(1 - iu)\xi_0(w) - \cos w\big\}, \tag{38}$$

$$\rho_0^{(2)} = -2e^{2iu} + 2iue^{-iu}\Big\{\cos w + (1 + 4iu)\xi_0(w) + 3u^2\xi_1(w)\Big\}, \tag{39}$$

$$\rho_1^{(2)} = -e^{-iu}\big\{\cos w + (1 + 2iu)\xi_0(w) - 3u^2\xi_1(w)\big\}, \tag{40}$$

$$\rho_2^{(2)} = -e^{-iu}\big\{\xi_0(w) - 3iu\xi_1(w)\big\}. \tag{41}$$

Where the two auxiliary functions are

$$\xi_0(w) = \operatorname{sinc}(w) \equiv \frac{\sin(w)}{w}, \tag{42}$$

$$\xi_1(w) = \frac{\cos(w)}{w^2} - \frac{\sin(w)}{w^3}. \tag{43}$$

For reference we have defined the coefficients only slightly differently to [2]. One can convert from one set of coefficients to the other using the following formula

$$\beta_{1,j}(\rho_j, p_j) = -b_{1,j}(r_j, f_j), \quad \beta_{2,j}(\rho_j, p_j) = ib_{1,j}(r_j, f_j), \tag{44}$$

$$\rho_0^{(i)} = r_0^{(i)}, \quad \rho_1^{(i)} = -ir_1^{(i)}, \quad \rho_2^{(i)} = -r_2^{(i)}. \tag{45}$$

We can now combine equations (32) and (33) to rewrite the derivative of the exponential as

$$\frac{\mathrm{d}}{\mathrm{d}\tau}e^X = \operatorname{tr}\big(X\dot{X}\big)B_1 + \operatorname{tr}\big(X^2\dot{X}\big)B_2 + p_1\dot{X} + p_2\dot{X}X + p_2X\dot{X}, \tag{46}$$

where we have defined

$$B_i = \beta_{i,0}I + \beta_{i,1}X + \beta_{i,2}X^2. \tag{47}$$

Inserting this result into (30) we get

$$\sum_{x,\mu}\operatorname{re}\operatorname{tr}\big(\Sigma'\dot{U}'\big) = \sum_{x,\mu}\operatorname{re}\operatorname{tr}\big(\Sigma' e^X\dot{U}\big) + \sum_{x,\mu}\operatorname{re}\operatorname{tr}\Big\{\underbrace{\Big(\operatorname{tr}\big(\Sigma' B_1 U\big)X + \operatorname{tr}\big(\Sigma' B_2 U\big)X^2 + p_1 U\Sigma' + p_2 U\Sigma' X + p_2 XU\Sigma'\Big)}_{\Gamma}\dot{X}\Big\}. \tag{48}$$

Next we want to rewrite $\dot{X}$ in terms of $\dot{U}$; the first step in this process is getting it as a function of $\dot{\Omega}$. Using eqs. (3) and (4) we see that

$$\operatorname{re}\operatorname{tr}\big(\Gamma\dot{X}\big) = \operatorname{re}\operatorname{tr}\Big\{\Gamma\frac{\mathrm{d}}{\mathrm{d}\tau}\Big(\tfrac{1}{2}\big(\Omega - \Omega^\dagger\big) - \tfrac{1}{6}\operatorname{tr}\big(\Omega - \Omega^\dagger\big)\Big)\Big\} = \operatorname{re}\operatorname{tr}\Big\{\underbrace{\Big(\tfrac{1}{2}\big(\Gamma - \Gamma^\dagger\big) - \tfrac{1}{6}\operatorname{tr}\big(\Gamma - \Gamma^\dagger\big)\Big)}_{\mathcal{P}_{\mathfrak{su}(3)}\{\Gamma\}\,\equiv\,\Lambda}\dot{\Omega}\Big\}. \tag{49}$$

When computing $\frac{\mathrm{d}}{\mathrm{d}\tau}S$ we can choose our d.o.f. as we please and therefore

$$\frac{\mathrm{d}}{\mathrm{d}\tau}S_{\mathrm{st}} = 2\sum_{x,\mu}\operatorname{re}\operatorname{tr}\big(\Sigma'\dot{U}'\big) = 2\sum_{x,\mu}\operatorname{re}\operatorname{tr}\big(\Sigma\dot{U}\big), \tag{50}$$

which means that (48) gives

$$\sum_{x,\mu} \mathrm{re\,tr}\left(\Sigma \dot{U}\right) = \sum_{x,\mu} \mathrm{re\,tr}\left(\Sigma' e^{X} \dot{U}\right) + \sum_{x,\mu} \mathrm{re\,tr}\left(\Lambda \dot{\Omega}\right). \tag{51}$$

Finally we need to compute $\dot{\Omega}$ in terms of $\dot{U}$. Since there is a sum over x and μ in the derivative of the action, we can use translational and rotational invariance to collect all terms that are of the form $\frac{\mathrm{d}}{\mathrm{d}\tau} U_\mu^{(k)}(x)$

$$\begin{aligned}
\sum_{x,\mu} \mathrm{re\,tr}\left(\Lambda_\mu(x) \frac{\mathrm{d}}{\mathrm{d}\tau} \Omega_\mu(x)\right) &= \sum_{x,\mu} \mathrm{re\,tr}\left(\Lambda_\mu(x) \dot{C}_\mu(x) U_\mu^\dagger(x) - C_\mu^\dagger(x) \Lambda_\mu(x) \dot{U}_\mu(x)\right), \\
&= \sum_{x,\mu} \mathrm{re\,tr}\Bigg(\Bigg[\sum_{\nu\neq\mu} \Big\{ \rho_{\nu\mu} U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) \Lambda_\nu(x) \\
&\qquad + \rho_{\mu\nu} U_\nu^\dagger(x+\hat{\mu}-\hat{\nu}) U_\mu^\dagger(x-\hat{\nu}) \Lambda_\mu(x-\hat{\nu}) U_\nu(x-\hat{\nu}) \\
&\qquad - \rho_{\nu\mu} U_\nu^\dagger(x+\hat{\mu}-\hat{\nu}) U_\mu^\dagger(x-\hat{\nu}) \Lambda_\nu(x-\hat{\nu}) U_\nu(x-\hat{\nu}) \\
&\qquad - \rho_{\nu\mu} \Lambda_\nu(x+\hat{\mu}) U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) U_\nu^\dagger(x) \\
&\qquad + \rho_{\mu\nu} U_\nu(x+\hat{\mu}) U_\mu^\dagger(x+\hat{\nu}) \Lambda_\mu(x+\hat{\nu}) U_\nu^\dagger(x) \\
&\qquad + \rho_{\nu\mu} U_\nu^\dagger(x+\hat{\mu}-\hat{\nu}) \Lambda_\nu(x+\hat{\mu}-\hat{\nu}) U_\mu^\dagger(x-\hat{\nu}) U_\nu(x-\hat{\nu}) \Big\} \\
&\qquad - C_\mu^\dagger(x) \Lambda_\mu(x)\Bigg] \dot{U}_\mu(x)\Bigg), \\
&\equiv \sum_{x,\mu} \mathrm{re\,tr}\left(\Xi_\mu(x) \dot{U}_\mu(x)\right),
\end{aligned} \tag{52}$$

where we have used $\Lambda^\dagger = -\Lambda$. With this we finally have an expression w.r.t. $\dot{U}$ only

$$\sum_{x,\mu} \mathrm{re\,tr}\left(\Sigma_\mu(x) \dot{U}_\mu(x)\right) = \sum_{x,\mu} \mathrm{re\,tr}\left(\Sigma'_\mu(x) e^{X_\mu(x)} \dot{U}_\mu(x) + \Xi_\mu(x) \dot{U}_\mu(x)\right), \tag{53}$$

which we can manipulate to

$$\sum_{x,\mu} \mathrm{re\,tr}\left(\left[\Sigma_\mu(x) - \Sigma'_\mu(x) e^{X_\mu(x)} - \Xi_\mu(x)\right] \dot{U}(x)\right) = 0. \tag{54}$$

By setting the expression in [] to zero we get our final equation

$$\Sigma_\mu^{(k-1)}(x) = \Sigma_\mu^{(k)}(x) e^{X_\mu^{(k-1)}(x)} + \Xi_\mu^{(k)}(x), \tag{55}$$

and as $\Xi^{(k)}$ is a function of $\left[\Sigma^{(k)}, U^{(k-1)}\right]$ only, we see that we have fulfilled the requirements of eq. (28).

3.1 *Relationship with the openQCD force*

So far we have followed the derivation of the MD force update as calculated by Morningstar & Peardon in [2], however, the final implementation detail is to relate their definition of the gauge link force (25), to the one used in openQCD. In openQCD all forces are members of the algebra of SU(3), and are represented by 8 real numbers. They are formally defined as

$$F_{\mu;a}(x) = \lim_{h\to 0} \frac{1}{h}\Big(S\big[e^{hT_a} U_\mu(x)\big] - S\big[U_\mu(x)\big]\Big), \tag{56}$$

where *only* the link $U_\mu(x)$ is modified while the others remain constant. Using this definition we can write down $\frac{\mathrm{d}}{\mathrm{d}\tau}S$

$$\frac{\mathrm{d}}{\mathrm{d}\tau}S = \sum_{x,\mu,a} \mathrm{re}\Big(F_{\mu;a}(x)\dot{\omega}_{\mu;a}(x,\tau)\Big), \tag{57}$$

where we have represented the time-dependence of $U_\mu(x)$ through

$$U_\mu(x;\tau) = e^{\omega_{\mu;a}(x,\tau)T_a}U_\mu(x). \tag{58}$$

By comparing with e.q. (30) we see that we want to relate

$$\sum_a F_{\mu;a}(x)\dot{\omega}_{\mu;a}(x,\tau) \quad \longleftrightarrow \quad 2\,\mathrm{tr}\Big(\Sigma_\mu(x)\dot{U}_\mu(x)\Big). \tag{59}$$

The left hand side of the relation is equal to

$$F_{\mu;a}(x) = 2\,\mathrm{tr}\big(F_\mu(x)T_a\big), \qquad \text{where} \qquad F_\mu(x) = \sum_a F_{\mu;a}(x)T_a, \tag{60}$$

while the right hand side can be rewritten using our definition of $U(\tau)$ in e.q. (58)

$$\mathrm{tr}\Big(\Sigma_\mu(x)\dot{U}_\mu(x)\Big) = \sum_a \mathrm{tr}\Big(\Sigma_\mu(x)T_aU_\mu(x)\dot{\omega}_{\mu;a}(x)\Big). \tag{61}$$

By inserting these two equations into e.q. (59) we get

$$2\sum_a \mathrm{tr}\Big(F_\mu(x)T_a\Big)\dot{\omega}_{\mu;a}(x,\tau) \quad \longleftrightarrow \quad 2\sum_a \mathrm{tr}\Big(\Sigma_\mu(x)T_aU_\mu(x)\Big)\dot{\omega}_{\mu;a}(x,\tau), \tag{62}$$

from which we see that

$$F_\mu(x) = U_\mu(x)\Sigma_\mu(x). \tag{63}$$

This means that if we want unsmearing to be in terms of F instead of Σ the expression for Γ changes to

$$\Gamma = \mathrm{tr}\big(UU'^\dagger F'B_1\big)X + \mathrm{tr}\big(UU'^\dagger F'B_2\big)X^2 + p_1UU'^\dagger F' + p_2UU'^\dagger F'X + p_2XUU'^\dagger F', \tag{64}$$

and the main unsmearing relationship becomes

$$F_\mu^{(k-1)}(x) = \mathcal{P}_{\mathfrak{su}(3)}\Big\{U_\mu^{(k-1)}(x)U_\mu^{(k)\dagger}(x)F_\mu^{(k)}(x)e^{X_\mu^{(k-1)}(x)} + U_\mu^{(k-1)}(x)\Xi_\mu^{(k)}(x)\Big\}. \tag{65}$$

Unfortunately the induction relationship in e.q. (28) has been transformed to requiring the links at both level (k) and $(k-1)$

$$\Big\{F^{(k)}, U^{(k)}, U^{(k-1)}\Big\} \longrightarrow F^{(k-1)}. \tag{66}$$

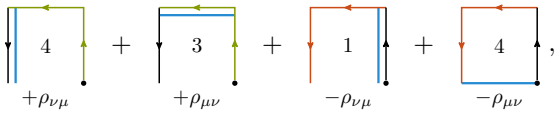
This is however only a setback in terms of elegance as the data is already there and readily available.

3.2 *Programming implementation*

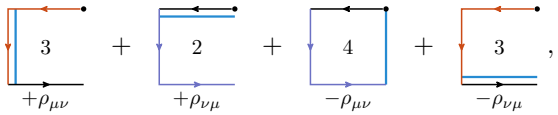
Computing Ξ

The main difficulty with respect to carrying out the force unsmearing computation lies in computing Ξ, eq. (52). Similar to how we carried out the computation of the Ω matrix at the smearing step, we will compute

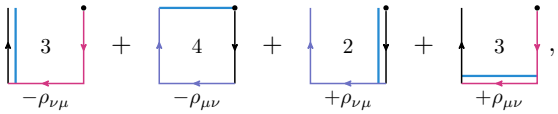
Ξ through additive contributions from plaquettes. If we consider a single plaquette we can graphically write down all contributions to Ξ from eq. (52)

$\Xi_\mu(x)$ += 

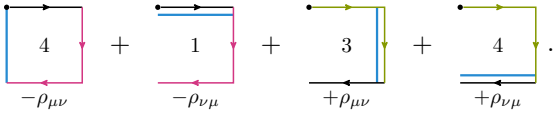
, (67a)

$\Xi_\nu(x+\mu)$ += 

, (67b)

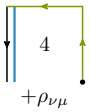
$\Xi_\nu(x)$ += 

, (67c)

$\Xi_\mu(x+\nu)$ += 

. (67d)

Here the dot indicates the starting point of the multiplication, the blue line (with no arrow) indicates multiplication by a Λ matrix, and the number in the staple indicates which position in the product of the four matrices the Λ matrix sits. E.g. a 3 implies we have the product $UU\Lambda U$. The coloured wedges indicate recurring expressions that we compute once and reuse in the other contributions. Below every staple is also noted the overall prefactor in terms of $\rho_{\mu\nu}$. Lastly an example to illustrate,



$= \rho_{\nu\mu} U_\nu(x+\mu) U_\mu^\dagger(x+\nu) U_\nu^\dagger(x) \Lambda_\nu(x)$. (68)

Modifications to the integrator

We have also had to change the order of the molecular dynamics integrator steps in order to not have to unnecessarily smear or unsmear the fields. An example output of the `print_mdint()` function after this change is

```
TS: smear fields,              eps = 0.00e+00
TP: force  1,                  eps = 6.25e-03
TP: force  2,                  eps = 2.50e-02
TS: unsmear fields and forces, eps = 0.00e+00
TP: force  0,                  eps = 6.25e-03
TP: force  3,                  eps = 2.50e-02
TU:                            eps = 6.25e-03
```

In this example the actions 1 and 2 are smeared while actions 0 and 3 are not. In general the smeared forces will be computed first, the unsmearing algorithm is then applied to the accumulated smeared force, finally the forces of unsmeared actions are added to this.

4 Memory footprint and communication overhead

In this ultimate section we will summarise the memory footprint of the smearing module as well as the MPI communication overhead.

4.1 *Memory footprint*

The application allocates additional memory for the following fields when stout smearing is used

Field	type	typesize	elements
$U^{(k)}$	su3_dble	144B	$n(4V+B)$
$X^{(k)}$	su3_alg_dble	64B	$4nV$
coeffs$^{(k)}$	ch_drv0_t	64B	$4nV$
Ω	su3_dble	144B	$4V+B$
Λ	su3_alg_dble	64B	$4V+B$
Ξ	su3_dble	144B	$4V+B$

Here V is the number of lattice points on the global lattice and B is the total gauge link halo for the parallelisation scheme of choice. These two numbers can be computed with the following formulae based on the input given to the build system. First the volume is given by

$$\texttt{V = (L0 * NPROC0) * (L1 * NPROC1) * (L2 * NPROC2) * (L3 * NPROC3).} \tag{69}$$

We will also reiterate how to compute the gauge halo B. This is also available in the `README.global` file in the `main/` directory, but we will include it here for completeness. To compute B we first need the number of points on the μ'th face of the local lattice, this is given by

$$\texttt{FACE}\mu = \begin{cases} \prod_{\nu\neq\mu} \texttt{L}\nu & \text{if } \texttt{NPROC}\mu \neq 1, \\ 0 & \text{else.} \end{cases} \tag{70}$$

The full boundary (positive and negative) of a local lattice is then

$$\texttt{BNDRY = 2 * (FACE0 + FACE1 + FACE2 + FACE3).} \tag{71}$$

To get the gauge link halo we have to count the type 1 and type 2 links. The type 1 links are links are the ones perpendicular to the faces, there are $\frac{1}{4}$`BNDRY` of these (one factor of $\frac{1}{2}$ because we only look at the positive boundary, another $\frac{1}{2}$ because these are only connected to the odd boundary points). The type 2 links are the ones parallel with the face and point in their respective positive direction; there are `3 * FACE`μ of these per face and this $\frac{3}{2}$ `* BNDRY` of these in total. The total number of gauge links in the halo of a local lattice is thus $\frac{7}{4}$ `* BNDRY`, and thus B is

$$\texttt{B = 7 * (NPROC0 * NPROC1 * NPROC2 * NPROC3) * BNDRY / 4.} \tag{72}$$

4.2 *Communication*

The stout smearing step of the algorithm is performed as described by the following pseudocode

Algorithm 1 Smearing routine

for $i \leftarrow 0, n-1$ **do**
  $smeared\_field[i] \leftarrow udfld()$
  $\Omega \leftarrow 0$
  for all $pl \in plaquettes$ **do**
    $\Omega(pl) \leftarrow \Omega(pl) + \Omega\_contrib\big(udfld()(pl)\big)$
  end for
  $add\_boundaries(\Omega)$ ▷ communicate su3_dble field
  for all $l \in links$ **do**
    $X[i][l] \leftarrow \mathcal{P}_{\mathfrak{su}3}\big\{\Omega[l]\big\}$
    $udfld()[l]$, $coeffs[i][l] \leftarrow$ `expXsu3_w_factors`$\big(udfld()[l], X[i][l]\big)$
  end for
  $copy\_boundaries(udfld())$ ▷ communicate su3_dble field
end for

The unsmearing step used to compute unsmeared molecular dynamics forces is as follows

Algorithm 2 Molecular dynamics force unsmearing routine

$smeared\_field[n] \leftarrow udfld()$ ▷ no actual assignment happening
for $i \leftarrow n-1, 0$ **do**
  for all $l \in links$ **do**
    $\Sigma \leftarrow smeared\_field[i+1][l]^\dagger \times force[l]$
    $\Lambda[l] \leftarrow \mathcal{P}_{\mathfrak{su}3}\{\Lambda\_contrib(\Sigma,\ smeared\_field[i][l],\ coeffs[i][l])\}$
  end for
  copy_boundaries(Λ) ▷ communicate `su3_alg_dble` field
  $\Xi \leftarrow 0$
  for all $pl \in plaquettes$ **do**
    $\Xi(pl) \leftarrow \Xi(pl) + \Xi\_contrib(smeared\_field[i](pl),\ \Lambda(pl))$
  end for
  add_boundaries(Ξ) ▷ communicate `su3_dble` field
  for all $l \in links$ **do**
    $\Sigma \leftarrow smeared\_field[i+1][l]^\dagger \times force[l]$
    $expX \leftarrow$ `ch2mat`$(X[i][l], coeffs[i][l])$
    $force[l] \leftarrow \mathcal{P}_{\mathfrak{su}3}\{smeared\_field[i][l] \times (\Sigma \times expX + \Xi[l])\}$
  end for
end for

We thus see that every smearing step requires the communication between two `su3_dble` fields, while the unsmearing step requires communication of one `su3_dble` field and one `su3_alg_dble`.

4.3 *Example*

Let us consider a 32^4 lattice divided into blocks of size 8^4 and $n = 2$. In this case $V = 1048576 \approx 1.0 \times 10^6$ and $B = 1834008 \approx 1.8 \times 10^6$. The additional memory footprint due to smearing of this run will then be

Field	type	typesize	elements	localsize	globalsize
$U^{(k)}$	su3_dble	144B	$n(4V+B)$	6598K	1656M
$X^{(k)}$	su3_alg_dble	64B	$4nV$	2040K	512M
coeffs$^{(k)}$	ch_drv0_t	64B	$4nV$	2040K	512M
Ω	su3_dble	144B	$4V+B$	3312K	828M
Λ	su3_alg_dble	64B	$4V+B$	1472K	368M
Ξ	su3_dble	144B	$4V+B$	3312K	828M
			Total	18.4M	4.7G

We can also compute the communication required for a pair of calls to the smearing- and unsmearing routines.

Field	type	typesize	local commsize	global commsize
Ω	su3_dble	144B	1008K	252M
$U^{(k)}$	su3_dble	144B	1008K	252M
Λ	su3_alg_dble	64B	448K	112M
Ξ	su3_dble	144B	1008K	252M
		Total (one smear)	3.5M	868M
		Total	6.8M	1.7G

The memory footprint and communication overhead is reduced in the case of `rho_t == 0.0` as we do not need to communicate the temporal gauge links in this case.

References

[1] Martin Lüscher. *SU(3) matrix functions*. **doc/su3_fcts.pdf**.

[2] Colin Morningstar and Mike J. Peardon. "Analytic smearing of SU(3) link variables in lattice QCD". In: *Phys. Rev.* D69 (2004), p. 054501. DOI: 10.1103/PhysRevD.69.054501. arXiv: hep-lat/0311018 [hep-lat].