Fieldshifts

Ahmidas Development Team

October 27, 2008

Starting from the following example lattice

$$\begin{matrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{matrix}$$

where the X direction is taken to be horizontal and Y is taken to be vertical, in memory this is stored consequtively as `123456789`. With the direction X as the direction for 123, the up direction is taken as from 1 to 3, and denoted as +1. That means that 1 is the down neighbour of 2, and `lattice[0] = 1` (with `[]` denoting physical indexing operators, see the section on indexing). A field shift in the X down direction results then in

$$\begin{matrix} 8 & 9 & 7 \\ 5 & 6 & 4 \\ 2 & 3 & 1 \end{matrix}$$

Such that `lattice[0]` is now 2, and `lattice[2]` is now 1. We don't want to copy the field around in memory, so we retain `123456789` consequtively in memory. To be able to identify the physical location, we have offsets to point to the proper location in memory. The new offset in the X direction should point to the second lattice element in memory, so `memory[1]`. The original offset in memory was 0, the new offset in memory should be 1. To get the new offset, we need to add 1 to the old offset. Or more precisely, we need to subtract the direction of our shift (which is `+1`). Doing the same shift again, we get:

$$\begin{matrix} 9 & 8 & 7 \\ 6 & 4 & 5 \\ 3 & 1 & 2 \end{matrix}$$

With the proper offset now being 2, which can again be obtained by subtracting the shift direction. So far so good with the subtraction of the shift

direction. Repeating the shift again, we get our initial lattice back:

$$\begin{matrix} 7 & 8 & 9 \\ 4 & 5 & 6 \\ 1 & 2 & 3 \end{matrix}$$

So now the offset should be again 0. We therefore need a mod with the `localSize` of the dimension in which we shift (3 in this case). This ensures that we know how to handle down shifts. Now shift in the up direction. A field shift in the X up direction results then in

$$\begin{matrix} 9 & 7 & 8 \\ 6 & 4 & 5 \\ 3 & 1 & 2 \end{matrix}$$

Such that `lattice[0]` is now 3, and `lattice[1]` is now 1. Again we need to change the offsets to point to the proper location in memory. The new offset in the X direction should point to the lattice element 3, so `memory[2]`. The original offset was 0, the new offset should be 2. But if we just subtract the direction as before, we end up at -1. Modding with the dimension won't help, so we need to add the `localSize` of the dimension to get a positive number. This does not break the procedure for the shifts in the down direction and all the other shifts in the up directions, since we anyway mod afterwards. One can check that this procedure is identical for shifts in the other dimensions, and that repeated shifting in alternating dimensions also works. The implementation in the code therefore is:

```
d_offsets[idx] += d_weave.localSize(idx) - dir;
d_offsets[idx] %= d_weave.localSize(idx);
```