Please refer to the power-point presentation for context. The following guide describes how to access and fine-tune the modifications but assumes the user knows about the basic motivations for programming a separate a domain decomposition algorithm for the constraints.

TCL-Level

The obstacle domain-decomposition (ODD) is going to be used by default at all times when this flavour of Espresso is compiled. At the TCL level, there are currently only three variables that will affect how / when the ODD is performed. They can be viewed or set using the **`setmd`** command;

Number of cells with `ddObs_nCell`

Syntax

| **`setmd ddObs_nCell`** *nCell_x nCell_y nCell_z*

Description

Allows to set the number of cells for the ODD in $x\ y\ z$. By default this initialises to 10 10 10, the user should choose an adequate division according to their geometry. Changing these values will also set the **`ddObs_reinit`** flag to 1 so that Espresso will re-run the ODD algorithm with the correct divisions.

Syntax

| **`setmd ddObs_nCell`**

Description

Will simply output number of cell divisions for the ODD in $x\ y\ z$

Initialisation state of the domain decomposition with `ddObs_reinit`

Syntax

| **`setmd ddObs_reinit`** *state*

Description

Sets the flag **`ddObs_reinit`** to the boolean value given by *state* (the only available option is 1-- pending initialisation). When it is set, a domain decomposition algorithm about the obstacles is run on the next integration call. The value is always 1 when Espresso is first launched and turns to 0 after the space has been cut out. Since the obstacles usually do not move this rarely needs to be changed (it is however sometimes the case--- perhaps for warm-up reasons). When the obstacles have been modified the space needs to be cut out again before an an integration call is made. By setting the **`ddObs_reinit`** flag to 1, the Espresso will re-init the ODD before performing the integration.

Syntax

```
| setmd ddObs_reinit
```

Description
Will simply output the current state of the ODD; 0=initialised and 1=pending initialisation.

Domain decomposition algorithm with `ddObs_special`

There are some cases where there are quite many spherical constraints and few of the other types. This is mostly because exotic objects are usually constructed out of spheres. When this is the situation, it is sometimes faster to first cycle through all of the spheres and place them in their corresponding spacial cells. This way, when cycling through the cells to include constraints, the vast majority of them (the spheres) can be overlooked. The user can choose this method by changing the flag **`ddObs_special`** to 1. For this trick to be robust, one must make sure that the interaction distance between the spheres is less than one half of the cell size (in all spacial directions).

Syntax

```
| setmd ddObs_special state
```

Description

Sets the flag **`ddObs_special`** to the boolean value given by *state* (the available options are 1- uses the special algorithm and 0- using the standard algorithm). When it is changed, a domain decomposition algorithm about the obstacles is run on the next integration call. The default value always 0 when Espresso is first launched. Changing this value will also set the **`ddObs_reinit`** flag to 1 so that Espresso will re-run the ODD using the different algorithm.

Syntax

```
| setmd ddObs_special
```

Description
Will simply output the current state of the ODD algorithm; 0- walk though the spacial cells and include close obstacles (default) OR 1- cycle though the spheres first before cycling through the spacial cells.

ESPREesSo-Level

These were developped using espresso-2.1.2j (www.espresso.mpg.de)

List of files that has been modified;

`constraint.h`
`global.h`
`global.c`
`verlet.c`
`verlet.h`

List of functions that has been added;

`constraint.h` : `MDINLINE void add_constraints_forces_DDCON(Particle *p1)`

`verlet.h` : `int ddObs_reinit_callback(Tcl_Interp *interp, void *_data);`
`verlet.c` : `int ddObs_reinit_callback(Tcl_Interp *interp, void *_data);`

`verlet.h` : `int ddObs_nCell_callback(Tcl_Interp *interp, void *_data);`
`verlet.c` : `int ddObs_nCell_callback(Tcl_Interp *interp, void *_data);`

`verlet.h` : `int ddObs_special_callback(Tcl_Interp *interp, void *_data);`
`verlet.c` : `int ddObs_special_callback(Tcl_Interp *interp, void *_data);`

List of functions that has been modified;

`verlet.c` : `void calculate_verlet_ia()`
`verlet.c` : `void build_verlet_lists_and_calc_verlet_ia()`