

## Developers:Operations\_by\_blocks

The key to good performance, especially in parallel, is to write code that operates simultaneously over a block of vectors.

Unfortunately Octopus was originally written to work with one vector at a time, but the code has been slowly converted to operation by blocks.

Of course, not all algorithms allow you to do things by blocks. For example, this is the case for the cg eigensolver, where one eigenvector must be computed before calculating the next one. But other operations, such as exponentials or the rmdiiis eigensolver, are ideally suited for operation by blocks.

Octopus has some data structures to make operations by blocks easier. The main object is `batch_t`, a container for a set of two-dimensional vectors. It contains also some extra information, as the dimension of the second index and an integer label of each vector (this is not related to the position in the batch).

It can be initialized in two ways:

- Initialize it from a contiguous array:

```
batch_init(this, dim, ist_start, ist_end, array(:, :, :))
```

- Or initialize it by passing individual vectors:

```
batch_init(this, dim, nst)
```

and then add `nst` vectors with

```
batch_add_state(this, ist, vec(:, :))
```

.

To destroy the object use `batch_end`.

There are some operations that can be performed over a batch:

- `hamiltonian_apply_batch`
- `preconditioner_apply_batch`
- `exponential_apply_batch`
- `nl_operator_operate_batch`
- `mf_dotp_batch`: performs a dot product between all the components of two batches, it is much faster than doing it individually.

