

Comment vectoriser les fonctions de calculs et en y incluant des opérations de type fma

Pour cela je vais vous montrer une instruction de calcul d'une des 6 fonctions vectorisés :

Dans la fonction `hFil_forward(int t, int i, int j)`, nous pouvons voir ce calcul :

```
return HPHY(t - 1, i, j) + alpha * (HFIL(t - 1, i, j) - 2 * HPHY(t - 1, i, j) + HPHY(t, i, j));
```

Un calcul qui représente à la fois des soustractions, additions et multiplication (ce qui nous rappelle le fma)

J'ai donc décidé de couper le calcul de cette manière :

```
return HPHY(t - 1, i, j) + alpha * (HFIL(t - 1, i, j) - 2 * HPHY(t - 1, i, j) + HPHY(t, i, j));
```

The diagram illustrates the vectorization of the calculation using SSE intrinsics. It shows the expression being broken down into three nested operations:

- `_mm256_fmsub_pd`: This intrinsic handles the innermost calculation, $(\text{HFIL}(t-1, i, j) - 2 * \text{HPHY}(t-1, i, j) + \text{HPHY}(t, i, j))$.
- `_mm256_fmadd_pd`: This intrinsic handles the multiplication of the result of the inner calculation by `alpha`.
- `_mm256_add_pd`: This intrinsic handles the final addition of `HPHY(t-1, i, j)` to the result of the previous operation.

Ce qui donne une fois implémenté, cette ligne de code assez complexe :

```
_mm256_add_pd((_mm256_load_pd(&HPHY(t - 1, i, j))), (_mm256_fmadd_pd((_mm256_load_pd(&alp[0])), (_mm256_load_pd(&HFIL(t - 1, i, j))),  
    (_mm256_fmsub_pd(_mm256_load_pd(&two[0]), (_mm256_load_pd(&HPHY(t - 1, i, j))), _mm256_load_pd(&HPHY(t, i, j))))))));
```

Ceci est un exemple parmi plusieurs autres qui sont quelques fois bien plus complexes, mais se sont montrés très efficaces.