1. Optimizaciones.

1.1. Inversin de ciclos.

El mtodo consiste en invertir los ciclos exteriores del mtodo bsico. Al realizar la multiplicacion A x B = C con el mtodo bsico, las matrices A y C se recorren por filas y la B por columnas. Al intercambiar el orden de los bucles, se cambia el sentido de recorrido de cada matriz, es decir, A y C por columnas y B por filas. Esto permite un mejor aprovechamiento de la localidad espacial que provee la dispocicion de los elementos contiguos en la memoria bajo el criterio Column Major Order.

1.2. Multiplicacin por columnas.

Este mtodo realiza la multiplicacin de matrices en forma incremental, efectuando todas las operaciones posibles con los elementos disponibles de una de las matrices y sumando los resultados parciales en la matriz resultado.

Utilizando las propiedades de las matrices se consiguen las siguientes equivalencias que proporcionanan fundamento matemtico al mtodo. Por simplicidad, se utilizan matrices de 2x2.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

El producto del miembro izquierdo se puede escribir como

$$\left(\begin{bmatrix} a_{11} & 0 \\ a_{21} & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{12} \\ 0 & a_{22} \end{bmatrix} \right) \left(\begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \end{bmatrix} + \begin{bmatrix} 0 & b_{12} \\ 0 & b_{22} \end{bmatrix} \right)$$

Y expandiendo los parntesis resulta

$$\begin{bmatrix} a_{11} & 0 \\ a_{21} & 0 \end{bmatrix} \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{12} \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & 0 \\ a_{21} & 0 \end{bmatrix} \begin{bmatrix} 0 & b_{12} \\ 0 & b_{22} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} \\ 0 & a_{22} \end{bmatrix} \begin{bmatrix} 0 & b_{12} \\ 0 & b_{22} \end{bmatrix}$$

As, la matriz producto se puede descomponer en la siguiente suma

$$\begin{bmatrix} a_{11}b_{11} & 0 \\ a_{21}b_{11} & 0 \end{bmatrix} + \begin{bmatrix} a_{12}b_{21} & 0 \\ a_{22}b_{21} & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{11}b_{12} \\ 0 & a_{21}b_{12} \end{bmatrix} + \begin{bmatrix} 0 & a_{12}b_{22} \\ 0 & a_{22}b_{22} \end{bmatrix}$$

Como se puede apreciar, las tres matrices se acceden por columnas. Esto resulta en una mejor utilizacin de la memoria cache.

Para que este mtodo funcione correctamente, se requiere que la matriz donde se almacena el resultado haya sido inicializada en 0. Sin embargo, como esto no es tenido en cuenta en el calculo del tiempo (ya que se realiza para todos los metodos), no presenta una desventaja.

Otro factor importante es el grado de asociatividad de la cache, ya que para que no se produzca trashing tiene que tener como minimo 4 vas para evitar que se reemplacen entre si las columnas de las distintas matrices.

Por ltimo, si el tamano de la matriz a multiplicar las columnas de cada matriz no se pueden mapear completmente en cada lnea de la cach obteniendo menor rendimiento.

1.3. Multiplicacin por bloques

Generalizando el mtodo anterior, es posible dividir la matriz en bloques de tamao arbitrario y, procediendo de la misma forma, se obtiene una suma de matrices que permiten calcular la multiplicacin progresivamente, empleando un conjunto de datos reducido.

1.4. Bsico con trasposicin.

Al igual que en el mtodo anterior, la trasposicin de la primera matriz del producto permite que posiciones contiguas de memoria sean accedidas para las tres matrices, optimizando el uso de memoria cache.

Sin embargo, el desempeo de este mtodo se ve impactado por la penalidad de calcular la matriz traspuesta, que requiere movimientos de grandes cantidades de datos.