Poisson operator in tensor product geometry

The Laplacian and other similar operators represented on a tensor-product grid can be inverted **directly** at a cost $O(N_x^3 + N_y^3)$ for pre-processing, a cost of $O(N_x N_y (N_x + N_y))$ for each solve, and storage of size $O(N_x^2 + N_y^2)$. Let D_{xx} be the $N_x \times N_x$ matrix representing ∂_{xx} and D_{yy} be the $N_y \times N_y$ matrix representing ∂_{yy}. We diagonalize each, at a cost of $N_x^3 + N_y^3$:

$$D_{xx} = E_x \Lambda_x E_x^{-1} \qquad D_{yy} = E_y \Lambda_y E_y^{-1}$$

where E_x and E_y are the $N_x \times N_x$ and $N_y \times N_y$ matrices of eigenvectors, and Λ_x and Λ_y are the diagonal matrices containing the eigenvalues. We write $f(x, y)$ as an $N_x \times N_y$ matrix.

$$\begin{aligned}
\partial_{xx} f(x_i, y_j) &= \sum_k D_{xx}(i,k) f(x_k, y_j) = (D_{xx} f)(x_i, y_j) \\
\partial_{yy} f(x_i, y_j) &= \sum_k D_{yy}(j,k) f(x_i, y_k) = (f D_{yy}^T)(x_i, y_j) \\
\partial_{xx} f &= D_{xx} f = E_x \Lambda_x E_x^{-1} f \\
\partial_{yy} f &= f D_{yy}^T = f (E_y \Lambda_y E_y^{-1})^T = f (E_y^{-1})^T \Lambda_y E_y^T \\
\nabla^2 f &= g \\
E_x \Lambda_x E_x^{-1} f + f (E_y^{-1})^T \Lambda_y E_y^T &= g \\
\Lambda_x \left[E_x^{-1} f (E_y^T)^{-1} \right] + \left[E_x^{-1} f (E_y^{-1})^T \right] \Lambda_y &= \left[E_x^{-1} g (E_y^T)^{-1} \right] \\
\Lambda_x(i) \left[E_x^{-1} f (E_y^T)^{-1} \right](i,j) + \left[E_x^{-1} f (E_y^{-1})^T \right](i,j) \Lambda_y(j) &= \left[E_x^{-1} g (E_y^T)^{-1} \right](i,j) \\
\left[E_x^{-1} f (E_y^T)^{-1} \right](i,j) &= \frac{\left[E_x^{-1} g (E_y^T)^{-1} \right](i,j)}{\Lambda_x(i) + \Lambda_y(j)} \\
f &= E_x \left[E_x^{-1} f (E_y^T)^{-1} \right] E_y^T
\end{aligned}$$

g	$\rightarrow$	$E_x^{-1} g$	$O(N_x^2 N_y)$
$E_x^{-1} g$	$\rightarrow$	$E_x^{-1} g (E_y^T)^{-1}$	$O(N_x N_y^2)$
g	$\rightarrow$	$\left[E_x^{-1} g (E_y^T)^{-1} \right](i,j) / (\Lambda_x(i) + \Lambda_y(j))$	$O(N_x N_y)$
$E_x^{-1} f (E_y^T)^{-1}$	$\rightarrow$	$f (E_y^T)^{-1}$	$O(N_x^2 N_y)$
$f (E_y^T)^{-1}$	$\rightarrow$	f	$O(N_x N_y^2)$
Total cost			$O(N_x N_y (N_x + N_y))$

The notation hides the simplicity: one diagonalizes in the x and y directions, one then divides by the sum of the eigenvalues, then one diagonalizes back. The principle is the same one as in a 2D Fourier transform. Only matrices of size $N_x \times N_x$ and $N_y \times N_y$ are needed and the operations are quadratic only in N_x or N_y and linear in the other direction, never quadratic in $N_x N_y$.

Precisely the same approach can be used in three dimensions. In two dimensions, the operators can be written in compact mathematical notation with the use of transposes, but in three dimensions, we must introduce other notation, such as an operator P_{12} which permutes the first and second arguments, and combined indices.

$$\begin{aligned}
\partial_{xx} f(x_i, y_j, z_k) &= \sum_\ell D_{xx}(i,\ell) f(x_\ell, y_j, z_k) = (D_{xx} f)(x_i, y_j, z_k) \\
\partial_{yy} f(x_i, y_j, z_k) &= \sum_\ell D_{yy}(j,\ell) f(x_i, y_\ell, z_k) = \sum_\ell D_{yy}(j,\ell)(P_{12} f)(y_\ell, x_i, z_k) \\
&= (D_{yy} P_{12} f)(y_j, x_i, z_k) = (P_{12} D_{yy} P_{12} f)(x_i, y_j, z_k) \\
\partial_{zz} f(x_i, y_j, z_k) &= \sum_\ell D_{zz}(k,\ell) f(x_i, y_j, z_\ell) = \sum_\ell f(x_i, y_j, z_\ell) D^T_{zz}(\ell, k) = (f D^T_{zz})(x_i, y_j, z_k)
\end{aligned}$$

g	$\rightarrow$	$g_1(x_i, y_j, z_k) \equiv \sum_\ell E_x^{-1}(i,\ell) g(x_\ell, y_j, z_k)$	$O(N_x^2 N_y N_z)$
	$\rightarrow$	$g_2(x_i, y_j, z_k) \equiv \sum_\ell E_y^{-1}(j,\ell) g_1(x_i, y_\ell, z_k)$	$O(N_x N_y^2 N_z)$
	$\rightarrow$	$g_3(x_i, y_j, z_k) \equiv \sum_\ell E_z^{-1}(k,\ell) g_2(x_i, y_j, z_\ell)$	$O(N_x N_y N_z^2)$
	$\rightarrow$	$f_3(x_i, y_j, z_k) \equiv g_3(x_i, y_j, z_k)/(\Lambda_x(i) + \Lambda_y(j) + \Lambda_z(k))$	$O(N_x N_y N_z)$
	$\rightarrow$	$f_2(x_i, y_j, z_k) \equiv \sum_\ell (E_x)^{-1}(i,\ell) f_3(x_\ell, y_j, z_k)$	$O(N_x^2 N_y N_z)$
	$\rightarrow$	$f_1(x_i, y_j, z_k) \equiv \sum_\ell (E_y)^{-1}(j,\ell) f_2(x_i, y_\ell, z_k)$	$O(N_x N_y^2 N_z)$
	$\rightarrow$	$f(x_i, y_j, z_k) \equiv \sum_\ell (E_z)^{-1}(k,\ell) f_1(x_i, y_j, z_\ell)$	$O(N_x N_y N_z^2)$
Total cost			$O(N_x N_y N_z (N_x + N_y + N_z))$

Additional costs are incurred by transposes, where necessary, before and after carrying out operations in the middle (y) direction. This is the same operation count as for multidimensional Fourier transforms, and for precisely the same reason, i.e. one is carrying out a sequence of one-dimensional operations.