Finite element local convolution integrals

Lucas Myers

August 16, 2023

1 Convolution definition

In these kinds of problems where there's a long-wavelength distortion in an otherwise periodic field, we have the need to take an average over the system period. To do this in a smooth way, one may define the local averaging as:

$$\tilde{X}(\mathbf{r}) = \int_{\Omega} d\mathbf{r}' \, \frac{X(\mathbf{r}')}{(2\pi a_0^2)^{d/2}} \exp\left(-\frac{(\mathbf{r}-\mathbf{r}')^2}{2a_0^2}\right) \tag{1}$$

where Ω is the domain, and a_0 is the lattice spacing. Typically the averaging is denoted by $\langle X \rangle$, but we have use for the angle brackets when taking the inner product of functions. Note that, since the integrand decays exponentially away from $\mathbf{r}$, we may define some cutoff Λ at which distance away from $\mathbf{r}$ we truncate the integral. Then this integral is done over a domain $C(\Lambda, \mathbf{r})$, a circle of radius Λ centered at $\mathbf{r}$.

Approximating the local average in terms of Lagrange elements ϕ yields:

$$\tilde{X} \approx \sum_j \tilde{X}_j \phi_j \tag{2}$$

Taking the inner product with other Lagrange elements ϕ_i yields:

$$\sum_j \tilde{X}_j \langle \phi_i, \phi_j \rangle = \left\langle \phi_i, \tilde{X} \right\rangle \tag{3}$$

where we are now taking this as the definition of $\tilde{X}_j$. We may easily solve this since $M_{ij} = \langle \phi_i, \phi_j \rangle$ is symmetric and positive-definite. However, to implement this numerically we will need to query $\tilde{X}$ at every quadrature point, which corresponds to calculating (1) at every quadrature point. This is actually tractable by introducing a cutoff, but we still have to keep track of $\tilde{X}$ at every quadrature point.

Explicitly the numerical computation of the right-hand side is given by:

$$\begin{aligned}
\left\langle \phi_i, \tilde{X} \right\rangle &= \sum_q \phi_i^{(q)} \tilde{X}^{(q)} \, (J \times W)^{(q)} \\
&= \sum_q \phi_i^{(q)} \sum_{q'} \frac{1}{(2\pi a_0^2)} X^{(q')} \exp\left(-\frac{(\mathbf{r}^{(q)} - \mathbf{r}^{(q')})^2}{2a_0^2}\right) (J \times W)^{(q')} \, (J \times W)^{(q)} \\
&= \frac{1}{(2\pi a_0^2)} \sum_{q'} \sum_q \phi_i^{(q)} X^{(q')} \exp\left(-\frac{(\mathbf{r}^{(q)} - \mathbf{r}^{(q')})^2}{2a_0^2}\right) (J \times W)^{(q')} \, (J \times W)^{(q)}
\end{aligned} \tag{4}$$

where here q and q' are indices of the two sets of quadrature points, a superscript of each of them corresponds to indexing by them, and $(J \times W)$ is the corresponding quadrature weight (scaled by the Jacobian from the unit cell). So then we must do one of two things in the multi-MPI process implementation: either we must send $\mathbf{r}^{(q)}$ to each process, calculate $\sum_q' X^{(q')} \exp\left(-(\mathbf{r}^{(q)} - \mathbf{r}^{(q')})^2/2a_0^2\right) (J \times W)^{(q')}$ on the other processes, then reassociate them with each quadrature point q so that the q summation can be carried out; or we may send over a set of $\phi_i^{(q)}$, $\mathbf{r}^{(q)}$, and $(J \times W)^{(q)}$, evaluate both sums on the external process, and then just reassociate with the local DoF i. I think it is probably easiest to do the former, and also it avoids having to send `n_dofs` $\times$ `n_quad_points` test functions.