A DATA EFFICIENT, CQM-BASED BEM APPROACH FOR ELASTODYNAMICS

Bernhard Kager\(^1\), Martin Schanz\(^2\)

\(^1\) Graz University of Technology, Technikerstraße 4, 8010 Graz, Austria, bernhard.kager@tugraz.at, www.mech.tugraz.at
\(^2\) Graz University of Technology, Technikerstraße 4, 8010 Graz, Austria, m.schanz@tugraz.at, www.mech.tugraz.at

**Key words:** Boundary Element Method, Fast Algorithms, Elastodynamics, Convolution Quadrature, Data Efficiency.

The numerical treatment of wave propagation through linear elastic continua with the Boundary Element Method (BEM) inherently involves the proper approximation of temporal convolution integrals. This talk introduces and discusses a new data efficient formulation that is based on Lubich’s convolution quadrature method [1].

**Convolution Integral and Convolution weights**

Given a time interval \((0, T) \in \mathbb{R}^+\) and its subdivision into \(M\) equidistant timesteps \(\Delta t\), such that \(t_n = n\Delta t, n = 0..M\). Consider a domain \(\Omega \in \mathbb{R}^3\), two points \(\mathbf{x}, \mathbf{y} \in \partial \Omega\) with \(\mathbf{r} = \mathbf{y} - \mathbf{x}\) and a vector-valued field \(\mathbf{\phi}(\mathbf{x}, t)\). With the displacement fundamental solution \(\hat{U}(\mathbf{r}, t)\), the occurring convolution integral is consequently defined and approximated by

\[
\int_0^{t_n} U(\mathbf{r}, t_n - \tau) \mathbf{\phi}(\mathbf{y}, \tau) d\tau \approx \sum_{m=1}^{n} \omega^{n-m}(\mathbf{r}) \mathbf{\phi}(\mathbf{y}), \quad \omega^{n}(\mathbf{r}) = \frac{\partial^n}{n!\partial \xi^n} \left[ \hat{U}(\mathbf{r}, \frac{\gamma(\xi)}{\Delta t}) \right]_{\xi=0}.
\]

A crucial prerequisite for this approach is the existence of the fundamental solution \(\hat{U}(\mathbf{r}, s)\) in Laplace domain and the selection of an appropriate multistep method with characteristic polynomial \(\gamma(\xi)\). Commonly, Cauchy’s Integral Formula is used to compute the \(n\)-th partial derivative with respect to \(\xi\). Contrary to that, as is proposed by [2] for acoustics, we present a direct, recursive evaluation of the weights for the elastodynamic case. \(\hat{U}(\mathbf{r}, s)\) is a linear combination of expressions

\[
P_n^\alpha(r, s) := s^{-n} \exp\left(-\frac{s}{rc_\alpha}\right), \quad n = 0, 1, 2 \text{ and } \alpha = 1, 2
\]

with \(c_\alpha\) being the wave velocities. Based on the BDF-2 scheme, Hackbusch, Kress and Sauter [2] provide exact expressions for the \(n\)-th partial derivative of \(P_n^\alpha(r, \gamma(\xi)/\Delta t)\) involving Hermite polynomials. Monegato [3] extends this to higher BDF schemes and
provides recurrence relations. In this presentation, we derive similar expressions for the computation of the \(n\)-th partial derivatives of the two remaining expressions \(P_1^r(r, \gamma(\xi)/\Delta t)\) and \(P_2^r(r, \gamma(\xi)/\Delta t)\) for BDF-2. This finally results in a direct evaluation of the convolution weights for the single and regularized double layer potential.

**Weight Interpolation Scheme** Since the spatial discretization is inevitably bounded with \(r \leq r_{\text{max}}\), a Hermite interpolation scheme is employed for the weight functions in \([0, r_{\text{max}}]\) to speed up the kernel evaluation.

**Data Efficiency** The weight functions that exhibit local support behaviour in \(r\) act as kernel functions for the spatial integration. To utilize the local support information, a Principal Component Analysis is performed to compute subdomains of \(\Gamma\). Subsequently, utilizing a numerical support detection, zero matrix blocks are determined prior to any matrix evaluation.

**Results** A rod of dimensions \(3\,\text{m} \times 1\,\text{m} \times 1\,\text{m}\) is investigated (Figure 1). The rod is clamped on one side and on the opposing side a heaviside traction load is applied. Parameters are set to \(\beta = \frac{c_1 \Delta t}{r_e} = 1\), \(c_1 = 1\,\text{m/s}\), \(c_2 = \sqrt{0.5}\,\text{m/s}\) and the interval detection accuracy \(\varepsilon_{\text{supp}} = 10^{-6}\). The results are shown in Table 1.

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{lvl} & \# \text{elem.} & n_t & \text{mem}_{\text{to.}} & \text{mem}_{\text{th.}} & \text{ratio} \\
\hline
0 & 112 & 38 & - & - & 0.81 \\
1 & 448 & 56 & 0.18 & 0.23 & 0.78 \\
2 & 1792 & 87 & 4.21 & 5.91 & 0.71 \\
3 & 7168 & 143 & 89.68 & 157.16 & 0.57 \\
\hline
\end{array}
\]

Table 1: Memory Consumption results

![Figure 1: Surface mesh (1792 elements)](image)

**REFERENCES**

