Implicit Symplectic Integrator

For simulating the dynamics there are many possible approaches. From what I've implemented the most successful technique has been an implicit scheme that uses a symplectic integration scheme and preserves the Lie group structure (the rotation matrices) using an RKMK technique. The implicit scheme has the benefit over explicit schemes because it isn't restricted to small time steps by the CFL condition which makes it much easier to achieve real-time simulations; however, implicit schemes can become numerically ill-conditioned at small time steps. The symplectic integrator is for the sake of not introducing any numerical damping and getting overall better qualitative behavior in the simulations. The preservation of the Lie group structure is to make sure that the determinant of the rotation matrices always remains 1 up to numerical precision.

One downside of implicit schemes is they can be computationally expensive to integrate due to having to solve large systems of equations; however, it was noted that under some conditions we can eliminate the majority of the equations and only have to solve one equation at each time step (a 6-component vector equation). To meet these conditions we need an implicit scheme in time and an explicit scheme and space and the solution procedure uses a shooting method to integrate at each time step.

This is not necessarily the best approach, but it is the best I've tried though there are some clear places to improve or try new things. The RKMK technique works for general Lie groups, but considering we are concerned with rotations there are likely more convenient and efficient schemes (RKMK can be hard to generalize to high accuracy), the most likely to be useful is quaternions rather than rotation matrices. I have a way to generalize the implicit time scheme to arbitrary order and know some properties about it and in implementing it it appears to work fine. Generalizing the explicit space scheme is pretty straightforward, but I haven't yet implemented a convenient generalized programming approach for it. There may also be more elegant proofs or alternative ways to do the generalizing, also some proofs could be argued to lack necessary depth (primarily symplecity even though experimentally it seems correct). These approaches all use finite differences, but finite elements, (pseudo)spectral methods, or hybrids like pseudospectral elements may have benefits that weren't noted or skipped due to some intuition that said it would be difficult/inefficient, but not shown.

Since there are many possible considerations, I will be going over the scheme

used and implemented so far which should be the simplest of these class of integrators. It will use the implicit midpoint method in time and the explicit Euler method in space. One interesting thing is that the implicit midpoint and trapezoidal methods appear to be equivalent or nearly equivalent for this approach. The main interesting point would be the implicit midpoint is in general symplectic, while the trapezoidal method is not, but in this approach they both are symplectic. I haven't been able to quite show that they are strictly different or the same, but this seems like it could provide 2 different starting points for generalizing the schemes where I've already generalized midpoint like approaches.

To start going over the tutorial go to: Initial