Representing Rotations as Quaternions

When looking at the computational efficiency of the simulations one thing that can jump out is the use of `expm`. During the computations we use `expm` to step the kinematics, but this can be a relatively expensive thing to compute. We can specialize `expm` to work on $SE(3)$ and $SO(3)$ to improve the efficiency a bit, but it can still be expensive. So, we want to improve this and avoid expensive `expm` calls.

The reason we needed `expm` in the first place is because rotation matrices are not closed under addition. When we step the rotations using a Runge-Kutta integrator we add the current step and a term based on the derivative to get the next step, but doing this causes the determinant of the rotation to drift making it no longer a pure rotation. One way to avoid this was the RKMK integration scheme, but we can change the representation to get more efficient computations. If we represent rotations as quaternions we no longer need the RKMK scheme, and therefore don't use `expm`, because quaternions are closed under addition. One issue is that unit quaternions are rotations, but with addition they won't remain unital. This can be avoided by having normalization be included in the change from quaternion to rotation matrix form. This ought to result in a more computationally efficient implementation of the kinematics.

The main piece the quaternions change is the derivative in space and time, everywhere else in the computations we want to keep the rotation matrix form. To convert a quaternion to a rotation matrix we have:

$$q = q_0 + q_1 i + q_2 j + q_3 k = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{1}$$

$$R = I + \frac{2}{q^T q} \begin{bmatrix} -q_2^2 - q_3^2 & q_1 q_2 - q_3 q_0 & q_1 q_3 + q_2 q_0 \\ q_1 q_2 + q_3 q_0 & -q_1^2 - q_3^2 & q_2 q_3 - q_1 q_0 \\ q_1 q_3 - q_2 q_0 & q_2 q_3 + q_1 q_0 & -q_1^1 - q_2^2 \end{bmatrix} \tag{2}$$

where q is the quaternion using the typical i, j, k basis.

So, now we need to see how the derivative and integration changes. For rotations we originally had:

$$R' = R\hat{\omega}$$

With quaternions we instead have:

$$q' = \frac{1}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} q$$