Numerical Lecture #2: Dimensionless Equations, Leapfrog Timestepping, and Coding Algorithm Choices

I. Dimensionless Equations

A. 1. Numerical codes are best implemented using dimensionless equations normalized to characteristic scales for time, length, velocity, mass, charge, etc.

2. Publications based on numerical results should explicitly state normalizations.
   a) Preferentially, all quantities should be plotted with the normalization explicitly given
   b) This allows people unfamiliar with the code to properly interpret results easily in terms of physical, rather than numerical, quantities.

B. Dimensionless Equations for $\underset{\sim}{E} \times \underset{\sim}{B}$ drift motion

1. There is no single "correct" choice for dimensionless normalization.
   $\Rightarrow$ It often depends on the problem of interest.
2. In some problems, a natural plasma physics scale appears.

Lorentz Force Law:

3. a. $$m \frac{d\underset{\sim}{v}}{dt} = q(\underset{\sim}{E} + \underset{\sim}{v} \times \underset{\sim}{B})$$

b. $$\frac{d\underset{\sim}{v}}{dt} = \underbrace{\frac{qB_0}{m}}_{=\Omega} \left(\frac{\underset{\sim}{E}}{B_0} + \underset{\sim}{v} \times \frac{\underset{\sim}{B}}{B_0} \right)$$

Angular cyclotron frequency naturally arises!

I. B.3. (Continued)

c. $\frac{d\underset{\sim}{v}}{d(\Omega t)} = \frac{\underset{\sim}{E}}{B_0} + \underset{\sim}{v} \times \frac{\underset{\sim}{B}}{B_0}$ ⇒ Units of velocity

d. Recall Larmor radius definition $r_L = \frac{v_\perp}{\Omega}$

e. Choose $v_\perp$ as characteristic velocity:

$$\frac{d\left(\frac{\underset{\sim}{v}}{v_\perp}\right)}{d(\Omega t)} = \frac{\underset{\sim}{E}}{v_\perp B_0} + \frac{\underset{\sim}{v}}{v_\perp} \times \frac{\underset{\sim}{B}}{B_0}$$

Dimensionless Normalization

f. Thus $\boxed{\frac{d\underset{\sim}{v}'}{dt'} = \underset{\sim}{E}' + \underset{\sim}{v}' \times \underset{\sim}{B}'}$ where $\boxed{v' = \frac{\underset{\sim}{v}}{v_\perp} \quad E' = \frac{\underset{\sim}{E}}{v_\perp B_0} \quad t' = \Omega t \quad B' = \frac{\underset{\sim}{B}}{B_0}}$

4. Position:

a. $\frac{d\underset{\sim}{x}}{dt} = \underset{\sim}{v} \Rightarrow \frac{d\left(\frac{\Omega}{v_\perp}\underset{\sim}{x}\right)}{d(\Omega t)} = \frac{\underset{\sim}{v}}{v_\perp}$

b. Recall $r_L = \frac{v_\perp}{\Omega}$, so $\boxed{\frac{d\underset{\sim}{x}'}{dt'} = \underset{\sim}{v}'}$ where $\boxed{x' = \frac{x}{r_L} \quad v' = \frac{\underset{\sim}{v}}{v_\perp} \quad t' = \Omega t}$

5. Dimensions of the problem:

4 independent dimensional quantities

a. Time/Frequency: $\Omega \equiv \frac{qB_0}{m}$

b. Velocity: $v_\perp$

c. Length: $r_L \equiv \frac{v_\perp}{\Omega}$

Since $r_L = \frac{v_\perp}{\Omega}$, only two independent parameters

d. Magnetic Field B_0

e. Electric Field $v_\perp B_0$

II. Leapfrog Timestep

A. Second-Order Accuracy in Time

1. The first-order Euler method requires extremely small timesteps to obtain accurate results over long integration times.
2. A higher-order method is an easy method to achieve much better accuracy, and consequently use fewer timesteps to obtain a fixed accuracy.

3. The leapfrog timestep uses the derivative at a time centered between the previous step and next step.
a. Consider [redacted] $\frac{dx}{dt} = v$

b. Discretizing, we use $\left.\frac{dx}{dt}\right|_{t_j} = \frac{x_{j+1} - x_{j-1}}{t_{j+1} - t_{j-1}} = v_j$

c. Thus, for constant timesteps Δt, we obtain

$$\frac{x_{j+1} - x_{j-1}}{2\Delta t} = v_j \Rightarrow \overset{\text{Leapfrog Timestep}}{\boxed{x_{j+1} = x_{j-1} + 2\Delta t\, v_j}}$$

4. Error Computation
a. Using a Taylor expansion about t_j for x_{j+1} and x_{j-1}, (HW) we can show $e \sim \mathcal{O}(\Delta t^3) \Rightarrow \boxed{\text{2nd-order Accuracy}}$

III. Coding Algorithm Choices

A. Good Programming Practice

1. When adding new capabilities to a working code, it is best, whenever possible, to add these new capabilities as new options in the code.

2. This enables results with the new method to be tested directly against the old method.

3. Do not delete code until you have tested and validated the new code
   ⇒ There is nothing worse than "upgrading" an old code, only to break, and having to spend significant time debugging just to get back to a code that works at all.

4. Using a switch-case structure in Matlab, or a select-case structure in Fortran, is a nice way to include new options in a code.

5. Appropriately modular programming can often yield very concise and efficient code that boasts numerous options.