Mathematics 420 Scientific Computing
Test I Topics (Wed Mar 10 in class)

Be sure to turn in any missing or makeup HWs for half credit at the time of the test. Remember HW counts 40% of the grade. If you made ≤ 50% on any HW, you may turn in a makeup at the time of the test for a maximum of half credit.

• HWs 1–6
• newton.c, BE_TR.c, & van.c (know what every line of code does)
• \( \epsilon_M = 2^{-52} \approx 2 \times 10^{-16} \) (64 bit double precision)
• Finite difference derivatives (forward difference \( du/dt \), central differences \( du/dx \) & \( d^2u/dx^2 \))
• Order of accuracy (consistency) & stability of forward & backward Euler, TR, & RK4 methods
  \[ \text{LTE} = \Delta t \tau \] defined in HW3
  Growth factor \( G \): \( u_{n+1} = Gu_n \)
  For stability \( |G| \leq 1 \) for model problem \( u' = au, a < 0 \)
• Statement of Equivalence Theorem (omit proof): For consistent numerical methods, stability is equivalent to convergence.
  convergence means
  \[ \lim_{n \to \infty} |u(t_n) - u_n| = 0, \quad n\Delta t = t_n \text{ fixed} \]

Memorize:

\[
\begin{align*}
  u(t \pm \Delta t) &= u(t) \pm \Delta t u'(t) + \frac{\Delta t^2}{2!} u''(t) \pm \frac{\Delta t^3}{3!} u'''(t) + \frac{\Delta t^4}{4!} u^{(4)}(t) \pm \cdots \\
  \equiv u \pm \Delta t u' + \frac{\Delta t^2}{2!} u'' \pm \frac{\Delta t^3}{3!} u''' + \frac{\Delta t^4}{4!} u^{(4)} \pm \cdots
\end{align*}
\]
HW5: (1) Explain in a short paragraph what the following code fragment from van.c does & why:

```c
u0 = u[step-1];
v0 = v[step-1];
/* u2, v2 step (2 * dt/2) */
RK(dt/2., u0, v0, &u1, &v1, epsilon);
RK(dt/2., u1, v1, &u2, &v2, epsilon);
/* u1, v1 step (dt) */
RK(dt, u0, v0, &u1, &v1, epsilon);
compute_dt(timestep, u0, v0, u1, v1, u2, v2);
if (!timestep->REDO_STEP) {
    u[step] = u2;
    v[step] = v2;
}
```

Set the initial conditions \((u_0, v_0)\) for the timestep = \((u[step-1], v[step-1])\). Using the RK4 method, compute a provisional new solution \((u_2, v_2)\) with two steps of \(dt/2\) (storing the intermediate solution in \((u_1, v_1)\)). Then again using RK4, compute a provisional new solution \((u_1, v_1)\) with one step of \(dt\). Now call compute_dt to see if the LTE is sufficiently small, setting REDO_STEP in the timestep structure. If REDO_STEP is NO, copy \((u_2, v_2)\) into the new solution \((u[step], v[step])\).