Wednesday 730.20 in 2076/350

- Basic plan for today: finish as much of Session 3/4 as possible. On Monday we'll do Session 5.
  - Leftover bits of 3/4 are covered in Assignment #3.

- New handouts: Assignment #2, "Integrals with Singularities...", "Simple Points", "Using a Plot File with Gnuplot", pages 100.

- Comments on Assignment #2
  - Follow-up to Session 3/4 on your own. If you blaze through the session, just start on the assignment.
  - Quantitatively analyze the errors.
  - (Bonus) is a way to even out the workload for different programming experience. I strongly recommend that you consider doing items is you already have experience.
  - You can always send me codes that you are stuck on or want feedback on.
  - You can always "upgrade" your solution.
  - Deadlines are not firm but don't put off to the last minute.
  - Office hours & help.

- Comments on Assignment #1 (pages 1)
  - In general codes look good — I'll make individual comments.
  - "Type casting"
    - A common problem with summing 1/n is flat.
    - double a = 1/n; => a=0.
    - A little test (from "Practical C++")
      | expression | result | result type |
      |------------|--------|-------------|
      | 19/10      | 1.0    | Integer     |
      | 19.0/10.0  | 1.9    | Floating Point |
      | 19.0/10.0  | 1.9    | Floating Point |
      | 19/10.0    | 1.9    | Floating Point |

It is possible to use coercion, but not advised.

- better to

  1) use decimal points when floats or doubles are involved; 1.0 or 1.0, not 1

  - I define
    
    ```
    const double one = 1.0;
    const double two = 2.0;
    ```

  2) always type cast

    ```
    int integer = (int) floating;
    ```

    - C-like

    ```
    double sum_up = 1.0/(float) i;
    ```

    - prefix notation

    ```
    double sum_down = 1.0/(float) (N-i+1);
    ```

    - functional notation

    ```
    double sum_up = 1.0/ float(i);
    ```

    ```
    sum_down = 2.0/ float(N-i+1);
    ```

    What about:

    ```
    for (double n = 1.; n < N; n += 10.)
    ```

    ```
    sum_up = 1.0/n
    ```

    Works, but tricky part is the n<=N comparison. Round-off error means n=N may or may not be included.

    BE CAREFUL

    - Note, n *= 10 vs. n += 1

    For log plots, multiplicative scale makes most sense.
1/21/04

Graphs:

- adjust plot range so the curve fills up the plot
- unless there are other motivations for the range
- don't draw lines between fluctuation points
- how to check fluctuations:
  - maybe you just got lucky/unlucky?
  - adjust your multiplicative factor and see if very different

- when fitting, be careful about specifying range of fit
- check robustness of your answer

⇒ now quantitative error analysis.

• Comment: gnuplot plot files
  - very convenient for going back and reproducing
    a plot that you want to edit.
  - saves typing and typos
  - encourages greater documentation

• Comment: gnuplot variables
  - plot using 1:2
  - plot using (1): (2)
    plot using (3): ((3)**2)
    plot using (log10($1))): (log10(abs($3-$2))/abs($3))