Computational Physics (6810): Session 5

Dick Furnstahl
Nuclear Theory Group
OSU Physics Department

February 1, 2017

- See “Distribution of Round-Off Errors” handout (cf. PS#1b bonus)
- Once more on round-off error for derivatives $|\epsilon, \epsilon'| \sim \mathcal{O}(\epsilon_m)$

$$\frac{f_c(x+h) - f_c(x)}{h} = \frac{f(x+h)(1+\epsilon) - f(x)(1+\epsilon')}{h} = \frac{f(x+h) - f(x)}{h} + \frac{(\epsilon - \epsilon')f(x)}{h}$$

- `NaN` $\Longrightarrow$ “not a number”; `inf` $\Longrightarrow$ “infinity”
  - Session 5: check the arithmetic of these (quick!)
  - Examples:

- Matrix timing: how should matrix multiplication scale with size N?
  - If we *assume* a dense matrix (compare sparse matrix):

$$\begin{pmatrix} & \\ & \end{pmatrix} \times \begin{pmatrix} & \\ & \end{pmatrix} = \begin{pmatrix} & \\ & \end{pmatrix}$$

- Diagonalization in coordinate representation
  - Solve $l = 0$ Schrödinger equation:

$$-\frac{\hbar^2}{2m}\frac{d^2u^{(n)}(r)}{dr^2} + V(r)u^{(n)}(r) = E_n u^{(n)}(r) \text{ with } u^{(n)}(r=0) = 0$$

  - normalization: $\int_0^\infty |u^{(n)}(r)|^2\, dr = 1$
- Solve as *matrix* problem: $H\Psi = E\Psi$ in discrete r basis
- If we use the approximation:

$$\frac{d^2u}{dr^2} \approx \frac{u(r+h) - 2u(r) + u(r-h)}{h^2} + \mathcal{O}(h^2)$$

  what do the kinetic energy and potential matrices look like?

Pointers once more

- You can stop listening if you are ok with pointers . . .
- In `session05.zip` the file `derivative_test_new.cpp` gives a possible solution to the “Pointer Games” exercise
- Void pointers once more:

```
void * params_ptr; // pointer to something (holds an address)
double x = 5;
double * x_ptr = &x; // define as pointer; set to address of x
params_ptr = &x; // knows address of x but not the type
cout << *x_ptr << endl;  // dereference --> prints 5
```

- recover in subroutine:

```
double x_passed;  // NOT a pointer
\\ read the next statement from right to left:
\\ 1) "cast" as pointer to a double, 2) dereference,
\\ 3) assign to x_passed
x_passed = * (double *) params_ptr;
```

Pointers once more (cont.)

- Now go through an example with structures . . .

```
typedef struct
{
  double a;
  double b;
} new_struct;

struct new_struct my_struct;  // new_struct is the data type;
                              //  my_struct is the "instance"
```

- recover in subroutine:

```
double a_passed;  // NOT a pointer
\\ read the next statement from right to left [note ()'s]:
\\ 1) dereference a pointer to a structure with ->,
\\ 2) "cast" as pointer to a new_struct structure,
\\ 3) assign to a_passed
a_passed = ((new_struct *) params_ptr)->a;
```