Chandler Justice - A02313187

Introduction to OpenMP

Preprocessor Directives, variables, functions

- #pragma omp parallel num_threads(n)
  This initializes a parallel region. This means OpenMP will try to automatically optimize the region for parallel execution. **Note:** The region that needs to be parallelized will need to be surrounded by curly braces.
- #include <omp.h>
  This directive tells the linker to pull in the OMP function signatures. This allows us to use openMP in our applicaiton
- omp_get_thread_num()
  This will return the rank of the thread running the application. This is the id of the rank running a parallel region
- omp_set_num_threads(n)
  Requests n threads at runtime.

fork-join parapllelism
This is a paradigm wherein a main thread spawns a team of threads as needed. this number of spawned threads can either be manually specified, or OpenMP can automatically allocate a number of threads aligned with your system specifications.

The main thread will fork out a job into multiple ranks/threads (they mean the same thing), and each worker thread will compute a smaller distrubition of the work. Once the work is completed, the threads will all "join" their results back to the main thread. This means the main thread will facilitate allocation and collection of work from all ranks. This process is all facilitated automatically by the #prama omp parallel preprocessor directive.

misc. notes and insights

- OpenMP is **non-deterministic**. This means that there is no guarantee in the order which threads will execute code. For example, one execution the first thread (rank 0) could run a line of code first, and then the next execution of the same program the third thread (rank 2) could run the same line of code first.
- Since we are working with multiple threads, the possibility of **race conditions** exist. This means two different threads (ranks) are trying to write to the same memory address. This causes undefined behavior.

Example: Approximating π
We know π can be approximated using:
$\int_0^1 \frac{4.0}{(1+x^2)} dx = \pi$

serial implementaiton example

```
static long num_steps = 10000;
double step;
void main()
{ int i; double x, pi, sum = 0.0;
step = 1.0/(double) num_steps;

for(int i = 0; i ¡ num_steps; i++){ x = (i + 0.5) * step;
sum += 4.0 / (1.0 + x * x);
}
pi = step * sum;
}
```

November 13

Leslie Matrix

$n_{t+1} = Ln_t$

$$\begin{bmatrix} f_0 & f_1 & f_2 & ... & f_{n-1} \\ s_0 & 0 & 0 & ... & 0 \\ 0 & s_1 & 0 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & s_{\omega-2} & 0 \end{bmatrix}$$

properties:

- f_x is the feumdity of the age class
- s_x is the proportion of individiuals who share from $x \rightarrow x+1$
- n_t is the vector containing K_1 of methods in each plan(?)

example: Find the largest eigenvalue & eigenvector
$n_1 = Ln_0 \in \mathbb{R}^\omega$
$\omega x \omega$ matrix
$n_0 = \alpha_1 v_1 + \alpha_2 v_2 + ... +_n v_n$
$n_k = l^k n_0$

We can use the **power method** to find the eigenvalues of the Leslie matrix

$$Ln_0 = \begin{bmatrix} \sum_{j=0}^{w-1} f_j (n_u)_j \\ s_0 (n_0)_0 \\ s_1 (n_0)_1 \\ s_2 (n_0)_2 \\ ... \\ s_{\omega-2} (n_0)_{\omega-2} \end{bmatrix}, \text{ For example } n_0 = \begin{bmatrix} 10 \\ 9 \\ 7 \\ 12 \\ ... \end{bmatrix} \rightarrow (n_0)_0 = 10; (n_0)_1 = 9$$

Companion Matrix

$$\begin{bmatrix} a_{n-1} & a_{n-2} & ... & a_0 \\ 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ ... & ... & 1 & ... \\ 0 & 0 & ... & 0 \end{bmatrix}$$

example: $\frac{d^2y}{dx^2} + 3\frac{dy}{dx} + \sin(x)y = 0$

$\frac{d^2y}{dx^2} = \frac{y^{x+h} - 2y(x) + y(x+h)}{h^2}$

$\frac{dy}{dx} = \frac{y(x+h) - y(x-h)}{2h}$