

# Projectile Motion

Fall Semester

## Topic Outline

- Historical Perspective
  - ABC and ENIAC
  - Ballistics tables
- Projectile Motion
  - Air resistance
  - Euler's method
  - Parameter search
- Serial to Parallel
  - Moore's Law
  - Amdahl's Laws
- Message Passing Interface
  - Manager-Worker
  - Boilerplate MPI
  - Send and receive
  - Avoiding deadlock
  - Communication groups
- Visualization
  - Gnuplot
  - Scripting
- Up Next: Fractal Generation

## Electronic Numerical Integrator And Computer (1)

<http://www.cs.iastate.edu/jva/jva-archive.shtml>

- “The Atanasoff-Berry Computer was the world’s first electronic digital computer. It was built...at Iowa State...during 1937-42.”

<http://www.upenn.edu/almanac/v42/n18/eniac.html>

- “When the Allied forces landed in North Africa in 1943, they found themselves operating ordnance in terrain that was entirely different from anything they had previously encountered.”
- “With pressure to produce firing tables for new artillery continually mounting, the need to find faster ways to perform ballistics computations became increasingly urgent.”

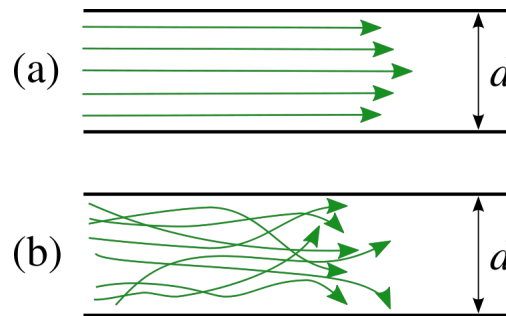
## Electronic Numerical Integrator And Computer (2)

<http://www.upenn.edu/almanac/v42/n18/eniac.html>

- “Calculating a trajectory could take up to 40 hours using a desk-top calculator. ... This military emergency provided the final impetus for large-scale experimentation in the field of electronic digital computers.”
- “In fact, as a digital device the computer would solve differential equations...the first real task assigned to ENIAC...involved millions of discrete calculations associated with top-secret studies of thermonuclear chain reactions—the hydrogen bomb.”
- “This creative tradition of building on the best of the past...”

## Air Resistance for 1-D Motion in the Absence of Wind

- Model with a force that opposes motion by acting in the direction opposite velocity. This means that if we're going up then it's pushing down and if we're going down then it's pushing up.
- Vertical wind? Swirling, or skyscraper with a chimney effect.
- Magnitude of the force is proportional to velocity for laminar flow and proportional to velocity-squared for turbulent flow.



## Euler's Method (1)

- Input. Angle of elevation  $\theta_0$  and initial velocity  $v_0$ .
- Without air resistance, parabola, range is  $v_0^2 \sin(2\theta_0) / g$ .
- With air resistance, run simulation to determine range:
  - Discretization, time-stepping.
  - Acceleration determined by mass, air resistance, gravity ( $a_y$ ).
  - Velocity determined by previous velocity and acceleration.
  - Position determined by previous position and velocity.
- Euler's Method. Simplest possible scheme, assume acceleration and velocity are constant for the entire time-step.

## Euler's Method (2)

- Drag coefficient determined experimentally.
- Assume  $rf \propto \rho C_d A$ , for air density  $\rho$ , drag coefficient  $C_d$ , and cross-sectional area  $A$ . Stop loop when  $y$  is 0. The range is  $x$ .

```
loop:
```

```
    ax = Fax(vw, vx, rf)/m;
```

```
    ay = Fay(g, vy, rf)/m;
```

```
    vx += ax*dt;
```

```
    vy += ay*dt;
```

```
    x += vx*dt;
```

```
    y += vy*dt;
```

## Parameter Search in 2-D with only Two Variables

- Given  $x_0$ ,  $y_0$ ,  $v_0$ ,  $g$ ,  $rf$ ,  $m$ ,  $dt$ , and assuming horizontal wind.
- During battle, measure wind velocity  $v_w$  and target range  $x_t$ .
- Set angle of elevation to corresponding  $\theta_0$  and fire artillery.
- Before the battle, vary  $\theta_0$  and  $v_w$  to determine various  $x_t$ .
- Get  $v_0$  from manufacturer and use  $v_w$  for relative velocity.

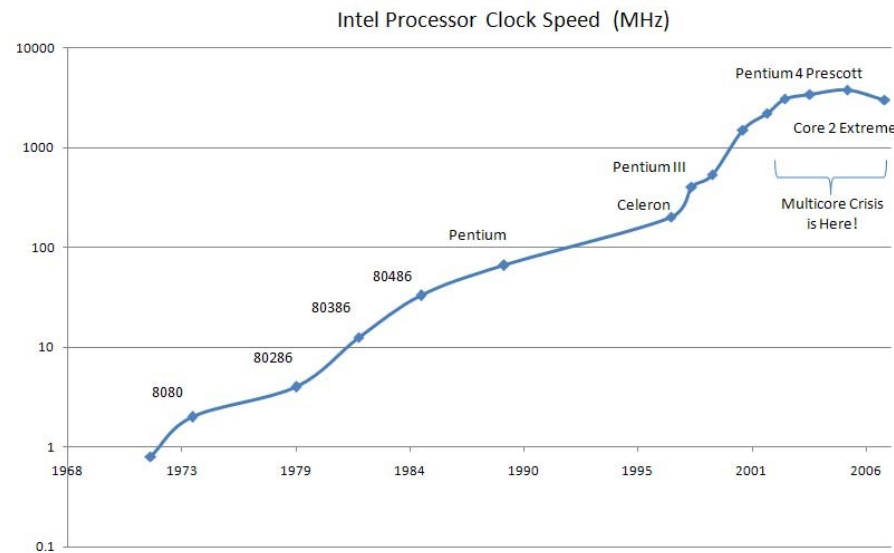
$\theta_0$	$v_w$	$x_t$	$\theta_0$	$v_w$	$x_t$	$\theta_0$	$v_w$	$x_t$	$\theta_0$	$v_w$	$x_t$
15.0	-5.0		30.0	-5.0		45.0	-5.0		60.0	-5.0	
15.0	0.0		30.0	0.0		45.0	0.0		60.0	0.0	
15.0	5.0		30.0	5.0		45.0	5.0		60.0	5.0	
15.0	10.0		30.0	10.0		45.0	10.0		60.0	10.0	

## Moore's Law

<http://www.intel.com/technology/mooreslaw>

- “...transistors on a chip will double about every two years.”

<http://smoothspan.wordpress.com/2007/09/06>



## Amdahl's Laws

- Quote<sup>1</sup>:

- “The speedup  $S = 1 / ((1 - f) + f/k)$ , where  $f$  is the fraction of work performed by the new [parallel] component and  $k$  is the speedup of the component.” This puts a bound on  $S$ .
- If  $k \rightarrow \infty$  then  $S = 1 / (1 - f) \not\rightarrow \infty$  for  $f < 100\%$ , latency.
- If  $f = 0.5$  then the best case is  $S = 2$  and in reality  $S < 2$ .

- Quote<sup>2</sup>:

- “In a balanced...system, the hertz of CPU speed, the bytes of RAM, and the bits per second...bandwidth [!] are all equal.”

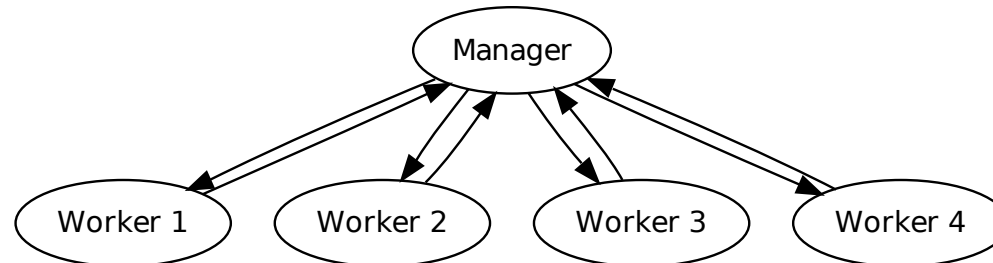
---

<sup>1</sup>Null, Linda, and Julia Lobur. *The Essentials of Computer Organization and Architecture*. Sudbury, MA: Jones and Bartlett Publishers, 2003. 274.

<sup>2</sup>Adams, Joel, and Timothy Brom. “Microwulf: A Beowulf Cluster For Every Desk”. *Proc. of the 33rd SIGCSE Tech. Symposium*. Portland, OR: 2008. 411-415.

## Manager-Worker

- Ideal communication scheme for “embarrassingly parallel” problems in which each task is independent of all other tasks.
- Parameter search, task of calculating  $x_t$  for one set of  $(\theta_0, v_w)$  is independent of calculating  $x_t$  for any other  $(\theta_0, v_w)$ .
- Manager gives different tasks to each worker. Worker performs task and sends back results. Ideal if manager sends two numbers and worker computes for an hour then reports one number.



## Message Passing Interface (1)

- Early 1990s. See <http://www.mpi-forum.org> for documents.
- *De facto* standard for programming on a distributed system.
- Implementations for C, C++, Fortran, Python, Java, etc.
  - MPICH from William Gropp and Argonne National Lab.
  - LAM from Ohio Supercomputer Center and Notre Dame.
- No shared memory so data must be passed from node to node.
- Single program multiple data (SPMD). Use rank and if-else.
- Compare to OpenMP, Co-Array Fortran, XMT-C, CUDA, etc.
- See also, Lin and Snyder page 202.

## Message Passing Interface (2)

```
#include "mpi.h"
```

```
main:
```

```
    int size; // How many nodes are in this run?
```

```
    int rank; // Which zero-indexed node am I?
```

```
    MPI_Init(&argc,&argv); // Call this first!
```

```
    MPI_Comm_size(MPI_COMM_WORLD,&size);
```

```
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
```

```
    ...
```

```
    MPI_Finalize(); // Call this last!
```

## Message Passing Interface (3)

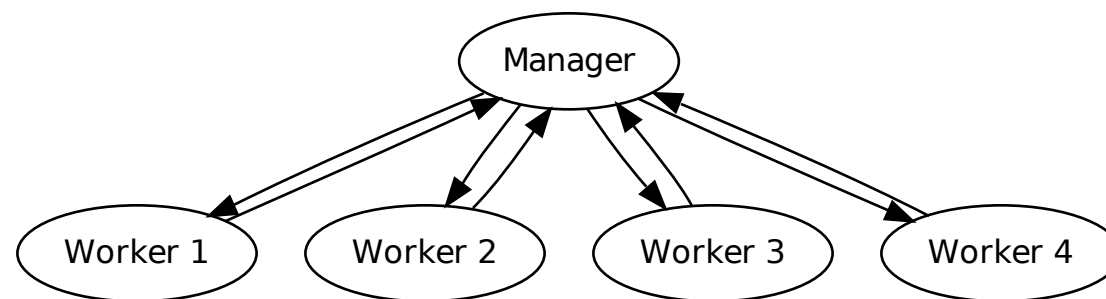
```
int size,rank,j,k,tag=0; // don't care about tag
MPI_Status status; // don't care about status
double pi=3.14159265;
...
j=(rank+1)%size; k=(rank+size-1)%size;
MPI_Send(&pi,1,MPI_DOUBLE,j,tag,MPI_COMM_WORLD);
MPI_Recv(&pi,1,MPI_DOUBLE,k,tag,MPI_COMM_WORLD,
                                               &status);
printf("node=%d,pi=%e\n",rank,pi); // sent to root
```

## Avoiding Deadlock

```
if(rank==0) // manager
    for(j=1;j<size;j++)
    {
        MPI_Send(...,j,tag,MPI_COMM_WORLD);
        MPI_Recv(...,j,tag,MPI_COMM_WORLD,&status);
    }
else // worker
    MPI_Send(...,0,tag,MPI_COMM_WORLD); // lazy
// sequencing of send/recv --> two loops better!
```

## Communication Groups

- Assume one big happy family: `MPI_COMM_WORLD`
- MPI allows us to create more complicated schemes but these are not necessary for simple Manager-Worker communication.
- Similarly, tagged messages are also unnecessary but students have made clever use of this feature in the past.
- See also, Lin and Snyder page 217.

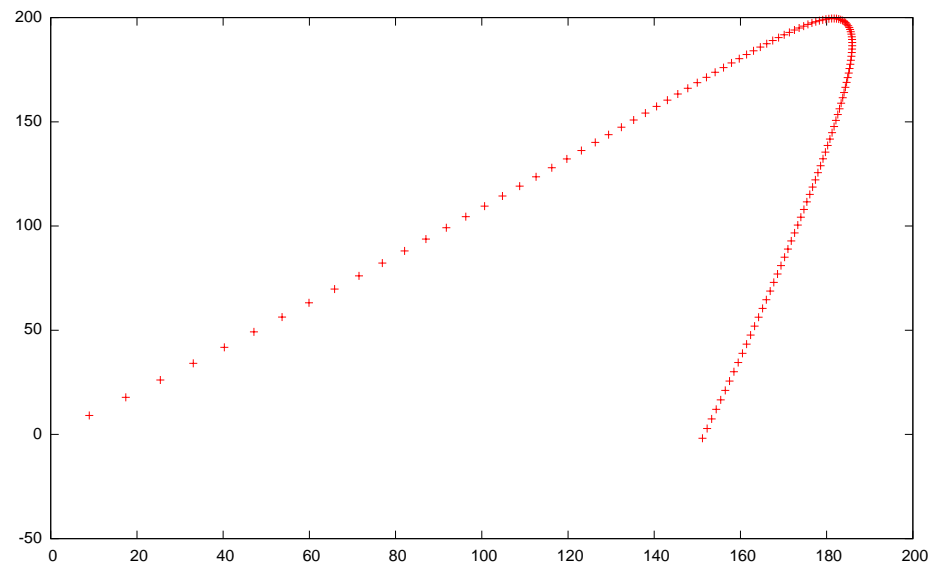


## Batch Jobs versus One-at-a-Time with Kill Signal

- How many times does the Manager talk to each Worker? Just once at the beginning and once at the end? Or over and over and over again as each small task is finished?
- In the first case communication overhead may be lower but you need to partition the parameter space in a way that makes sense and some options are better than others. You also risk getting nothing if a node goes down before reporting back its results.
- In the second case you need some way for the Manager to tell the Worker that there are no more tasks, some kind of kill signal that will have to be checked for as each new task is started.

## Gnuplot

- To run: `gnuplot` or `gnuplot filename.p`
- Easy command: `plot "xy.txt"` or `splot "xyz.txt"`
- Can output as EPS, PNG, PDF, etc.



Example Program: Euler, `projmot_0.c`

- Headwind.
- Free fall.

Example Program: MPI, `mpi_demo_0.c`

- Manager-Worker scheme with simple send and receive.
- Compile: `mpicc -lm mpi_demo_0.c`
- Run: `mpirun -np 5 -machinefile hosts.txt a.out`

Example Script: Gnuplot `plot*.p`

- Simple 2-D plot.
- Simple 2-D plot written to a PDF.

## Lab Assignment: Projectile Motion

- Generate a 3-D surface plot of  $(\theta_0, v_w, x_t)$  where:
  - The  $\theta_0 \in [1.0^\circ, 179.0^\circ]$  and  $\Delta_{\theta_0} = 1.0$  is the step size.
  - The  $v_w \in [-40.0, 40.0]$  and  $\Delta_{v_w} = 1.0$  is the step size.
- Category 1 hurricane, 40 m/s is approximately 90 mph.
- Plot the runtime to generate the  $x_t$  for:
  - Serial code.
  - Parallel code with only one Worker, thus two nodes.
  - Parallel code with  $n = 2, 3, 4, 5, 6, 7, 8, 16, 32$  Workers.
- Gnuplot: `set dgrid3d 179,81` and `splot`