CS 470/570 Homework 4
Due Monday November 13

Homework 4

- Implement a two dimensional n-body simulation program using pthreads
- The program expects one command line argument. The argument is the name of a file that contains the number of bodies, the mass, initial position and initial velocity of each body, total time for the simulation and the length of time between recalculation of the position and velocity of each body.
- The output for the program should be written to standard output. The output includes the final position and velocity for each body.

Homework 4 Input File Format

- The input file will be a text file containing the following information.
  - The number of bodies (an int)
  - The total time in seconds (an int)
  - Delta time in seconds (an int)
  - The mass, initial position (x, y values) and initial velocity (x, y values) of each body. (one line for each body. The values are separated by spaces. The values are floats)

Homework 4

- N-Body Problem
- Sequential Code segments
- Euler's method to approximate a function
- The material in the following slides come from *Parallel Programming* by Peter Pacheco

N-body Problem

- Given a collection of particles with characteristics, such as position, mass or velocity, (some of) which vary over time find the characteristics of the particles after some time T.
- Gravitational n-body problem: given the mass, position, and velocity of a collection of particles (planets, stars, etc.) use Newton's second law of motion and law of universal gravitation to determine the particles positions and velocities after some time T

N-body Problem Formulas

(6.1)

$$\mathbf{f}_{qk}(t) = -\frac{Gm_q m_k}{\left|\mathbf{s}_q(t) - \mathbf{s}_k(t)\right|^3}\left[\mathbf{s}_q(t) - \mathbf{s}_k(t)\right]$$

(6.2)

$$\mathbf{F}_q(t) = \sum_{\substack{k=0 \\ k\neq q}}^{n-1} \mathbf{f}_{qk} = -Gm_q \sum_{\substack{k=0 \\ k\neq q}}^{n-1} \frac{m_k}{\left|\mathbf{s}_q(t) - \mathbf{s}_k(t)\right|^3}\left[\mathbf{s}_q(t) - \mathbf{s}_k(t)\right]$$

N-body Problem Formulas

(6.3)

$$\mathbf{s}_q''(t) = -G \sum_{\substack{j=0 \\ j \neq q}}^{n-1} \frac{m_j}{\left|\mathbf{s}_q(t) - \mathbf{s}_j(t)\right|^3} \left[\mathbf{s}_q(t) - \mathbf{s}_j(t)\right]$$

$$t = 0, \Delta t, 2\Delta t, \ldots, T\Delta t$$

Gravitational Constant

- $G = 6.67408 \times 10^{-11}\ m^3\ kg^{-1}\ s^{-2}$

Sequential Pseudo-Code

```
Get input data;
for each timestep {
   if (timestep output) Print positions and velocities of particles;
   for each particle q
      Compute total force on q;
   for each particle q
      Compute position and velocity of q;
}
Print positions and velocities of particles;
```

Sequential Computation of Forces

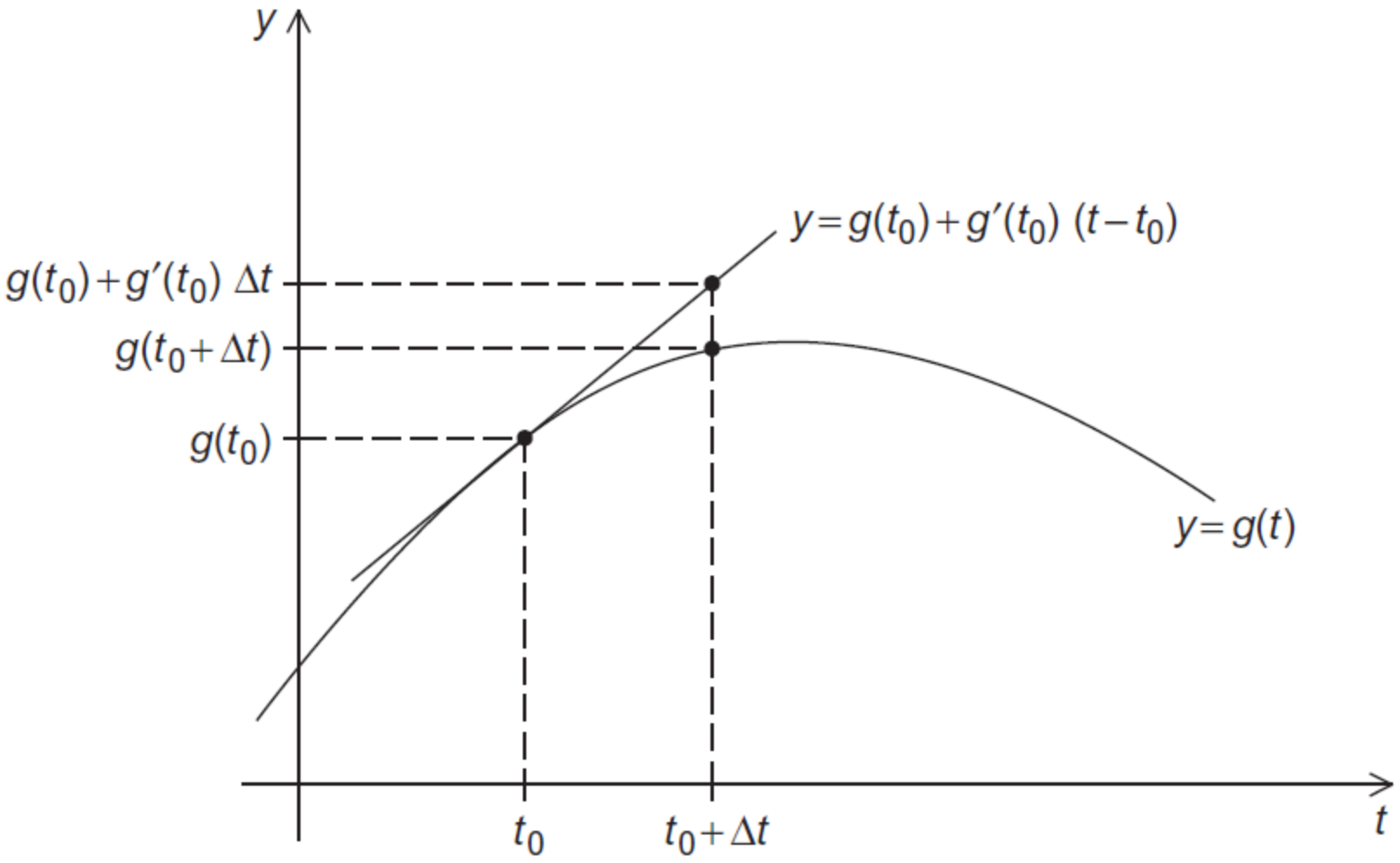
```
for each particle q {
    for each particle k != q {
        x_diff = pos[q][X] - pos[k][X];
        y_diff = pos[q][Y] - pos[k][Y];
        dist = sqrt(x_diff*x_diff + y_diff*y_diff);
        dist_cubed = dist*dist*dist;
        forces[q][X] -= G*masses[q]*masses[k]/dist_cubed * x_diff;
        forces[q][Y] -= G*masses[q]*masses[k]/dist_cubed * y_diff;
    }
}
```

Sequential Computation of Forces Taking Advantage of Symmetry

```
for each particle q
   forces[q] = 0;
for each particle q {
   for each particle k > q {
      x_diff = pos[q][X] - pos[k][X];
      y_diff = pos[q][Y] - pos[k][Y];
      dist = sqrt(x_diff*x_diff + y_diff*y_diff);
      dist_cubed = dist*dist*dist;
      force_qk[X] = G*masses[q]*masses[k]/dist_cubed * x_diff;
      force_qk[Y] = G*masses[q]*masses[k]/dist_cubed * y_diff

      forces[q][X] += force_qk[X];
      forces[q][Y] += force_qk[Y];
      forces[k][X] -= force_qk[X];
      forces[k][Y] -= force_qk[Y];
   }
}
```

Euler Method

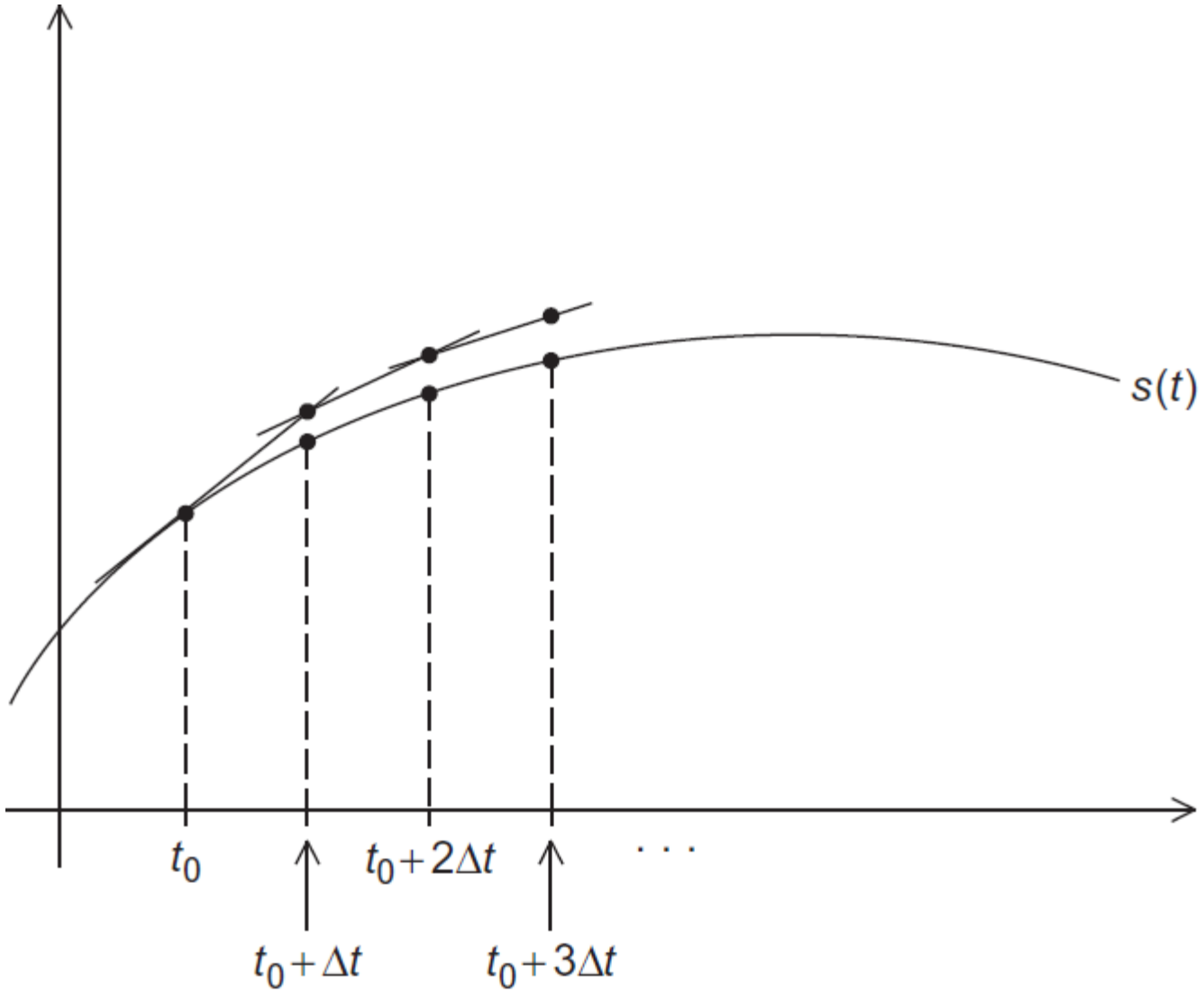


Formula For Approximation

- $g(t+\Delta t) \approx g(t_0) + \Delta t g'(t_0)$
- $s_q(\Delta t) \approx s_q(0) + \Delta t s'_q(0) = s_q(0) + \Delta t v_q(0)$
- $v_q(\Delta t) \approx v_q(0) + \Delta t v'_q(0) = v_q(0) + \Delta t a_q(0)$

$$= v_q(0) + \Delta t(1/m_q)F_q(0)$$

Euler Method



Formulas

- Use approximation to $s_q(\Delta t)$ and $v_q(\Delta t)$ to approximate $s_q(2\Delta t)$ and $v_q(2\Delta t)$ etc.

Program Formulas

- pos[q][X] = pos[q][X] + delta_t*vel[q][X]
- pos[q][Y] = pos[q][Y] + delta_t*vel[q][Y]
- vel[q][X] = vel[q][X] + delta_t/masses[q]*forces[q][X]
- vel[q][Y] = vel[q][Y] + delta_t/masses[q]*forces[q][Y]

Pseudo-code Parallel Version 1

```
   for each timestep {
      if (timestep output) Print positions and velocities of particles;
#     pragma omp parallel for
      for each particle q
         Compute total force on q;
#     pragma omp parallel for
      for each particle q
         Compute position and velocity of q;
   }
```

Parallel Version 1

```
#  pragma omp parallel for
   for each particle q {
      forces[q][X] = forces[q][Y] = 0;
      for each particle k != q {
         x_diff = pos[q][X] - pos[k][X];
         y_diff = pos[q][Y] - pos[k][Y];
         dist = sqrt(x_diff*x_diff + y_diff*y_diff);
         dist_cubed = dist*dist*dist;
         forces[q][X] -= G*masses[q]*masses[k]/dist_cubed * x_diff;
         forces[q][Y] -= G*masses[q]*masses[k]/dist_cubed * y_diff;
      }
   }
```

Parallel Version 1

```
#  pragma omp parallel for
   for each particle q {
      pos[q][X] += delta_t*vel[q][X];
      pos[q][Y] += delta_t*vel[q][Y];
      vel[q][X] += delta_t/masses[q]*forces[q][X];
      vel[q][Y] += delta_t/masses[q]*forces[q][Y];
   }
```

Pseudo-code Parallel Version 2

```
#  pragma omp parallel
   for each timestep {
      if (timestep output) Print positions and velocities of particles;
#     pragma omp for
      for each particle q
         Compute total force on q;
#     pragma omp for
      for each particle q
         Compute position and velocity of q;
   }
```

Pseudo-code Parallel Version 3

```
#   pragma omp parallel
    for each timestep {
        if (timestep output) {
#           pragma omp single
            Print positions and velocities of particles;
        }
#       pragma omp for
        for each particle q
            Compute total force on q;
#       pragma omp for
        for each particle q
            Compute position and velocity of q;
    }
```