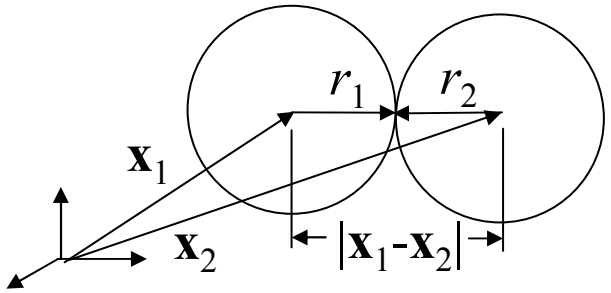
DEM Modeling: Lecture 03

The Hard-Particle Algorithm
Collision Detection

When Two Spheres Collide…

- A collision will occur between two spheres when the magnitude of the position of one sphere relative to the other is equal to the sum of sphere radii.



$$\left|\mathbf{x}_1 - \mathbf{x}_2\right| = (r_1 + r_2)$$

- Two approaches for marching forward in time
  - Time-Step Driven: time proceeds in small increments
  - Event Driven: time proceeds from collision to collision

Time Step Driven Algorithm

- Increment the particle velocities and positions in small time steps (Hopkins and Louge, 1991)

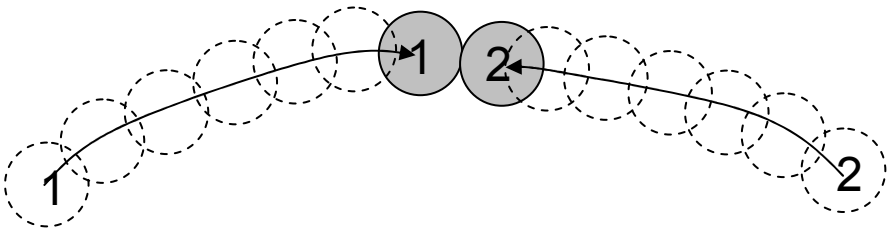
$$\dot{\mathbf{x}}_{n+\frac{1}{2}} = \dot{\mathbf{x}}_{n-\frac{1}{2}} + \ddot{\mathbf{x}}_{n-1}\Delta t$$

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \dot{\mathbf{x}}_{n+\frac{1}{2}}\Delta t$$

(Verlet/leapfrog algorithm – other time integration schemes to be discussed in a different lecture)

- Collisions occur when:

$$\left|\mathbf{x}_1 - \mathbf{x}_2\right| \le (r_1 + r_2)$$



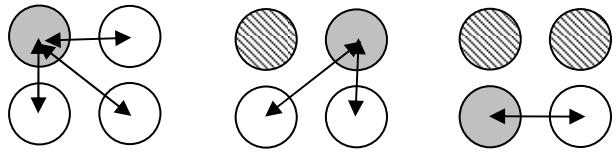
Time Step Driven Algorithm…

- Choose time step so that a particle moves a fraction of its diameter during each time step.
  - a function of particle size and speed, not a function of particle material properties (as is the case with the soft-particle method)
  - larger time steps $\Rightarrow$ faster sims, but larger overlaps and more error (more on this topic later in this lecture)
  - can include non-collision forces (e.g. aerodynamic, electrostatic, gravitational) acting on particles between collisions by determining a particle's acceleration using Newton's Laws
- Still need to check for collisions between particles

Brute Force Coarse Contact Detection

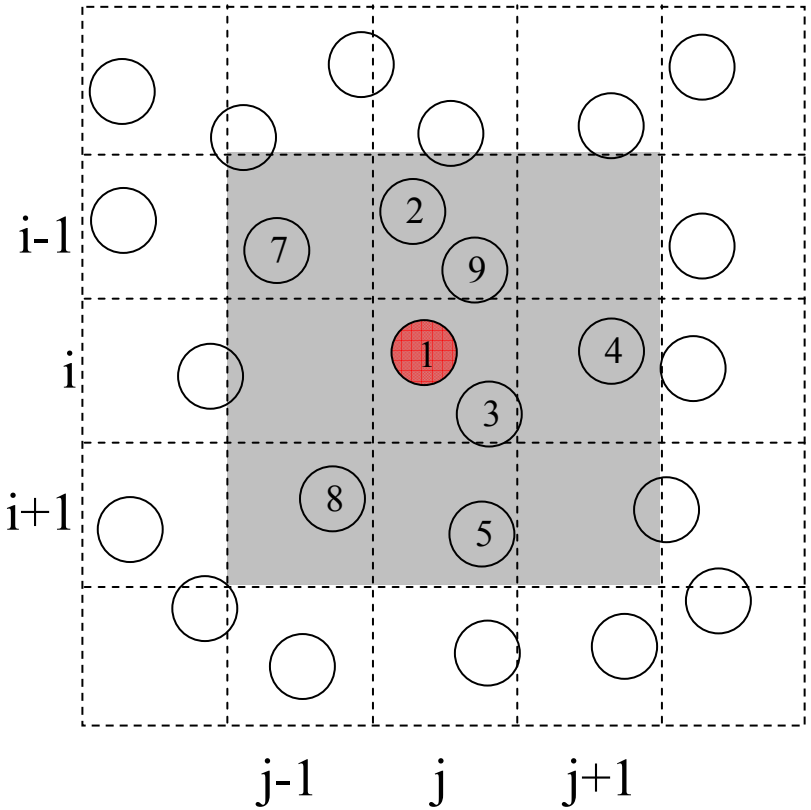
- Assume a system contains *N* particles
- To determine if contact occurs between any two particles
  - could check for contacts between all possible particle pairs:
    - particle 1: *N*-1 contact checks
    - particle 2: *N*-2 contact checks
    - particle *N*-1: 1 contact check
    - particle *N*: 0 contact checks
    - total # of contact checks:

      $(N-1)+(N-2)+\ldots+1 = N(N-1)/2 \sim \mathbf{O(N^2)}$



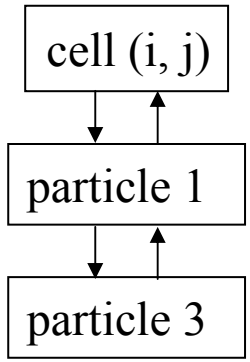
  - aka “naïve” contact detection
- There are more efficient ways of checking for contacts!
  - neighboring-cell contact detection scheme
  - nearest-neighbor contact detection scheme
  - sweep and prune

Neighboring Cell Coarse Contact Detection



- divide the workspace into a grid of cells
- for each cell, maintain a list of the particles contained within that cell
- for a given particle, only check for contact between other particles in its own cell and neighboring cells
- cell size may be smaller than particle size, a single particle may occupy multiple cells

(More on coarse contact detection in a different lecture.)



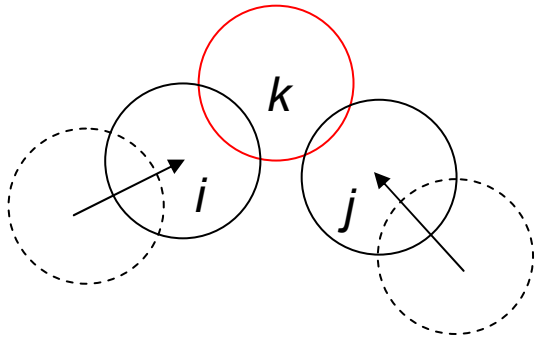
double linked lists are often used to maintain the cell lists

For particle 1, in cell (i, j), check for contact against:

cell (i-1, j-1):	particle 7
cell (i-1, j):	particles 2 and 9
cell (i-1, j+1):	-
cell (i, j-1):	-
cell (i, j):	particle 3
cell (i, j+1):	particle 4
cell (i+1, j-1):	particle 8
cell (i+1, j):	particle 5
cell (i+1, j+1):	-

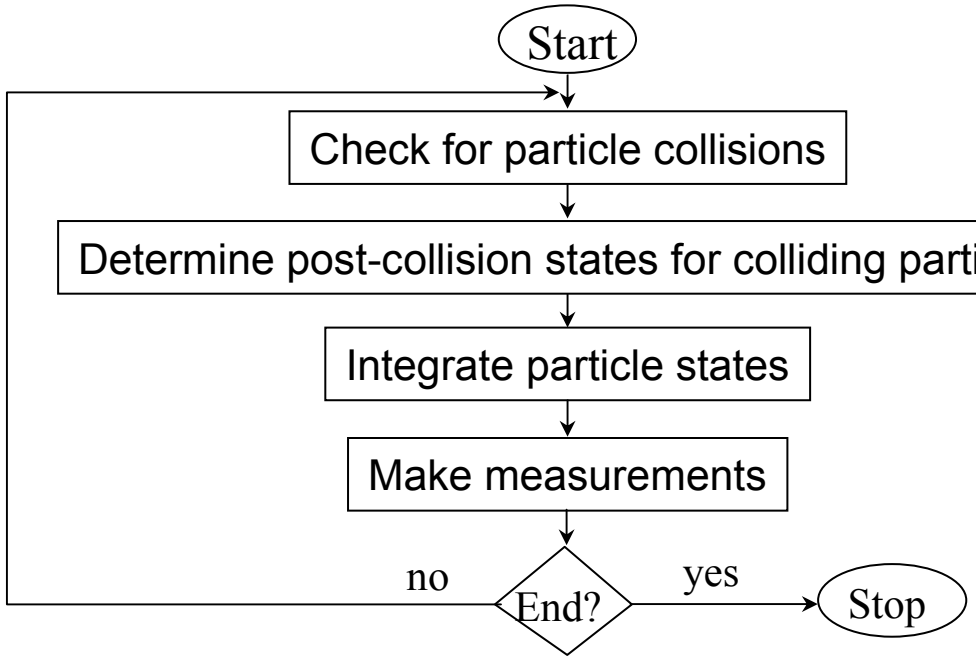
A Time Step Algorithm Issue

- It is possible to have more than two particles colliding simultaneously using a time step algorithm.



  - likelihood decreases as time step decreases
  - could use a multi-time step approach – move backwards in time if multiple overlaps occur and then move forward again with a smaller time step
  - could perform two hard particle collisions, one after the other (collisions in rapid succession), and accept the error associated with the calculation

Time Step Driven Flowchart



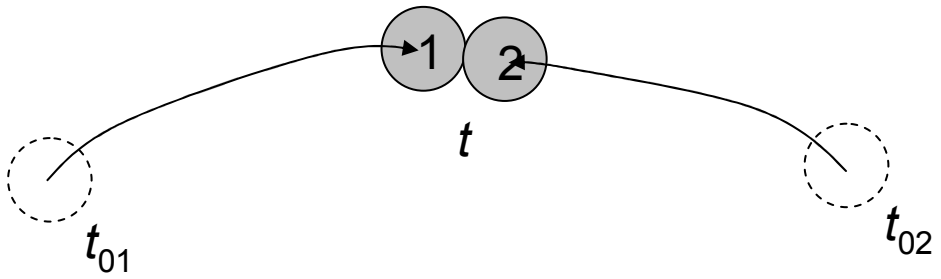
Event Driven Algorithm

- Assume particles move in ballistic trajectories between impacts

$$\mathbf{x}_1(t) = \frac{1}{2}\ddot{\mathbf{x}}_{01}(t-t_{01})^2 + \dot{\mathbf{x}}_{01}(t-t_{01}) + \mathbf{x}_{01}$$

$$\mathbf{x}_2(t) = \frac{1}{2}\ddot{\mathbf{x}}_{02}(t-t_{02})^2 + \dot{\mathbf{x}}_{02}(t-t_{02}) + \mathbf{x}_{02}$$

where the subscript "0" represents the conditions immediately *after* the previous collision



Event Driven Algorithm…

- A collision occurs when

$$\left|\mathbf{x}_2(T)-\mathbf{x}_1(T)\right|=(r_2+r_1)$$

where T is the time when the collision occurs.

- Substituting and simplifying gives:

$$a_4T^4+a_3T^3+a_2T^2+a_1T+a_0=0$$

where

$$a_4=|\mathbf{A}|^2$$
$$a_3=2(\mathbf{A}\cdot\mathbf{B})$$
$$a_2=2(\mathbf{A}\cdot\mathbf{C})+|\mathbf{B}|^2$$
$$a_1=2(\mathbf{B}\cdot\mathbf{C})$$
$$a_0=|\mathbf{C}|^2-(r_1+r_2)^2$$

and

$$\mathbf{A}=\tfrac{1}{2}(\ddot{\mathbf{x}}_{01}-\ddot{\mathbf{x}}_{02})$$
$$\mathbf{B}=-\ddot{\mathbf{x}}_{01}t_{01}+\dot{\mathbf{x}}_{01}+\ddot{\mathbf{x}}_{02}t_{02}-\dot{\mathbf{x}}_{02}$$
$$\mathbf{C}=\tfrac{1}{2}\ddot{\mathbf{x}}_{01}t_{01}^2-\dot{\mathbf{x}}_{01}+\mathbf{x}_{01}-\tfrac{1}{2}\ddot{\mathbf{x}}_{02}t_{02}^2+\dot{\mathbf{x}}_{02}-\mathbf{x}_{02}$$

Event Driven Algorithm…

- Use Bairstow's Method to solve the quartic equation (see, for example, Hoffman, 2001).
  - factors out quadratic equations
  - uses an iterative Newton's approach to determining the coefficients for the quadratic equations

$$a_4T^4 + a_3T^3 + a_2T^2 + a_1T + a_0 = 0 \Rightarrow \left(m_2T^2 + m_1T + m_0\right)\left(\ n_2T^2 + n_1T + n_0\ \right) = 0$$

- Note that when the accelerations are identical, then:

$$\ddot{\mathbf{x}}_{01} = \ddot{\mathbf{x}}_{02} \Rightarrow \begin{aligned} \mathbf{A} &= \mathbf{0} \\ \mathbf{B} &= \dot{\mathbf{x}}_{01} - \dot{\mathbf{x}}_{02} \\ \mathbf{C} &= \tfrac{1}{2}\ddot{\mathbf{x}}_{01}t_{01}^2 - \dot{\mathbf{x}}_{01} + \mathbf{x}_{01} - \tfrac{1}{2}\ddot{\mathbf{x}}_{02}t_{02}^2 + \dot{\mathbf{x}}_{02} - \mathbf{x}_{02} \end{aligned} \qquad \Rightarrow \begin{aligned} a_4 &= 0 \\ a_3 &= 0 \\ a_2 &= \left|\mathbf{B}\right|^2 \\ a_1 &= 2\left(\mathbf{B}\cdot\mathbf{C}\right) \\ a_0 &= \left|\mathbf{C}\right|^2 - \left(r_1 + r_2\right)^2 \end{aligned}$$

$$a_2T^2 + a_1T + a_0 = 0$$

Event Driven Algorithm…

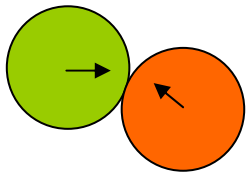
- A collision will occur when:

$$T = \frac{-a_1 - \sqrt{a_1^2 - 4a_2 a_0}}{2a_2}$$

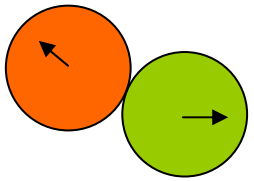
  - complex values for $T \Rightarrow$ particles never collide with each other
  - $T < \max(t_{01}, t_{02}) \Rightarrow$ particles collide in the past
  - The larger positive root to the equation

$$T = \frac{-a_1 + \sqrt{a_1^2 - 4a_2 a_0}}{2a_2}$$

    is not considered since it corresponds to the time when $|\mathbf{x}_2 - \mathbf{x}_1| = (r_2 + r_1)$ when particles are allowed to pass through each other

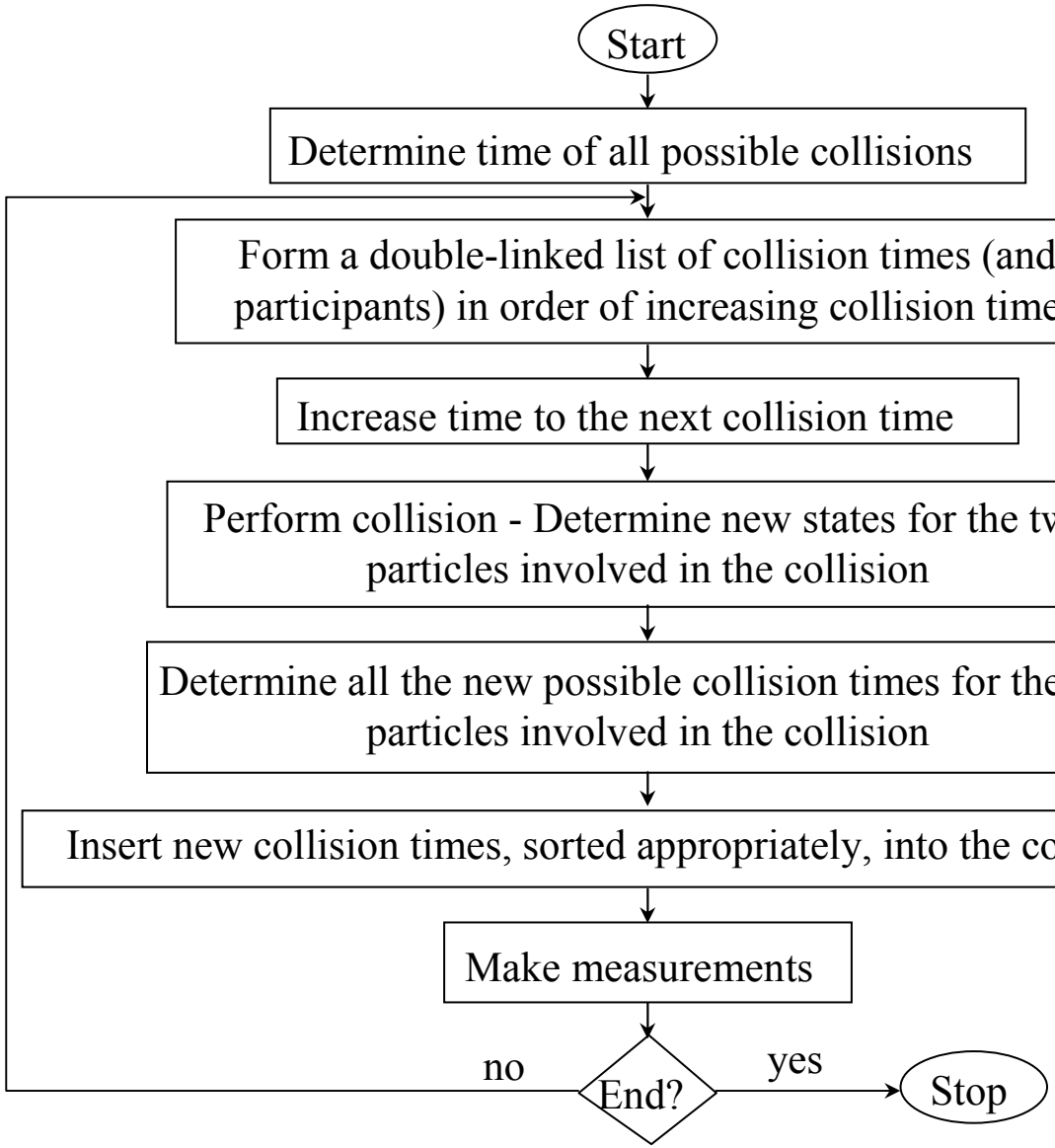


at time T_{small}



at time T_{large}
(particles have passed through each other)

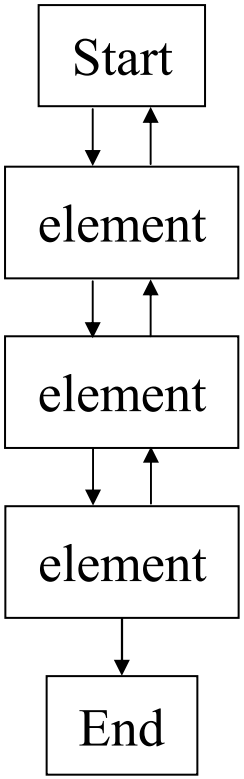
Event Driven Algorithm…



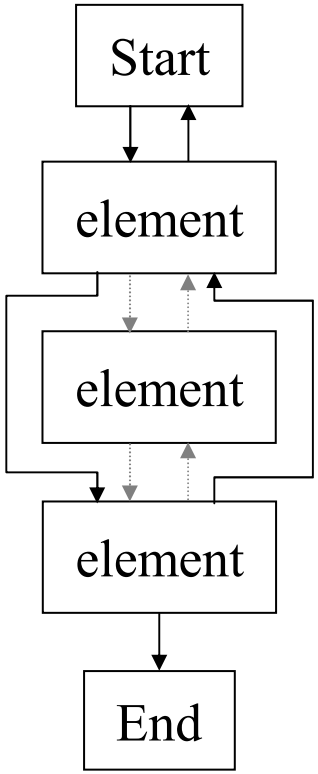
- The list of collision times and participants is referred to as the "collision list."

Double Linked Lists

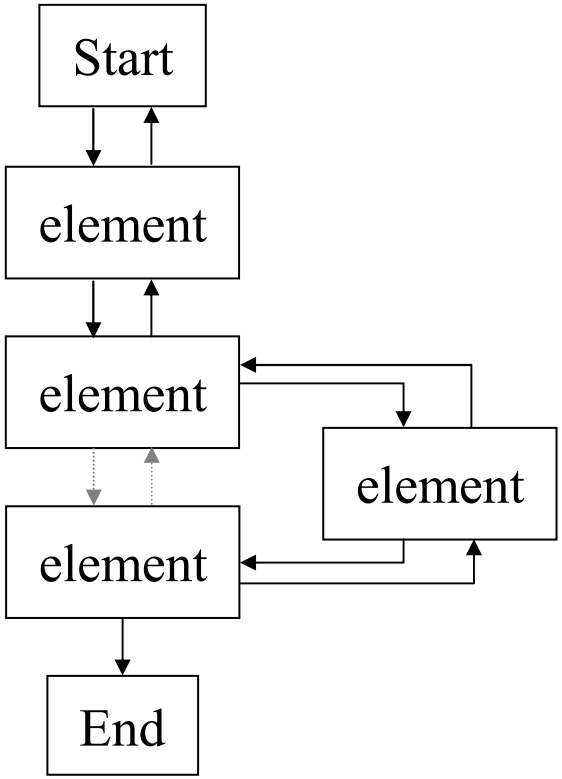
Original List with Links



Removing an Element



Adding an Element



Double Linked Lists…

- C++ Standard Template Library (STL) has a double-linked list class
  - highly recommend for use – minimize chance of memory allocation/leak errors when programming
- Examples of miscellaneous commands:

```
#include <list.h>
list<double> L;
double value1 = 1.0, value2 = 2.0, value 3 = 3.0;
L.clear();
L.push_front(value1);
L.push_front(value3);
L.push_back(value2);
L.sort();
```

- To learn more:
  - Google "C++ STL list"

An Example Collision List Scenario

1. Example collision list prior to collision resolution

(6,8): $T = 3.2$
(2,5): $T = 3.4$
(4,6): $T = 4.0$
(1,9): $T = 4.1$
(8,9): $T = 4.3$
(5,6): $T = 4.4$

2. Perform collision resolution for particles in the first element in list.

$$\left(\mathbf{x}_6, \dot{\mathbf{x}}_6, \mathbf{x}_8, \dot{\mathbf{x}}_8\right)^- \Rightarrow \left(\mathbf{x}_6, \dot{\mathbf{x}}_6, \mathbf{x}_8, \dot{\mathbf{x}}_8\right)^+$$

3. Remove references to particles involved in the collision.

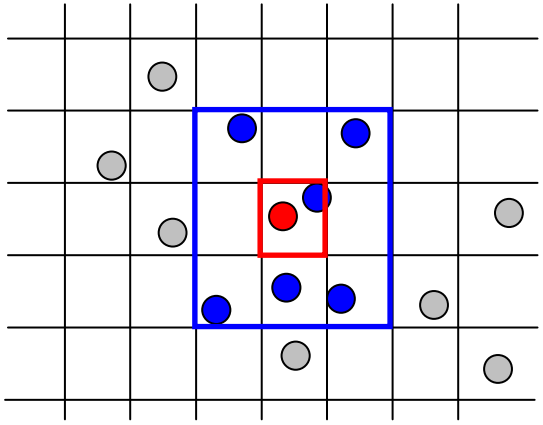
(2,5): $T = 3.4$
(1,9): $T = 4.1$

An Example Collision List Scenario…

4. Calculate new collision times for the particles involved in the collision using the post-collision states. Is potentially a $2N$ calculation, *e.g.* check 6 against 1, 2, 3,… and 8 against 1, 2, 3, …
5. Add new collisions to collision list (only those with collision times greater than the current time) and sort in ascending order based on collision time.

(4,6): $T = 3.3$
(2,5): $T = 3.4$
(8,9): $T = 3.9$
(1,9): $T = 4.1$

Another Collision Detection Approach



- Rather than perform a $2N$ collision detection step after each collision, use a neighboring cell approach.
- The collision list should only include particle collisions with particles in the current cell and neighboring cells (*e.g.* red checked with blue).
- The collision list will include "collisions" between a particle's center and the grid walls (red cell). For non-accelerating particles, this is a simple linear calculation for collision time.
- When a particle/grid wall "collision" occurs, do not change the particle state, but instead move the particle to a new cell and re-calculate collision times with the new walls and particles within the neighboring cells.
- Use a cell size equal to the particle diameter to minimize the number of collision detection calculations.

See, for example, Lasinski *et al*. (2004)

Solving for Quartic Roots

```
// Initial guesses for roots.
 if (t > 0.0) {
   r = 2.0*t;
   s = -t*t;
 } else {
   r = s = 1.0;
 }

 // Solve by applying Bairstow's method.
 count = 0;
 do {
   flag = 0;

   b4 = a4;
   b3 = a3 + r*b4;
   b2 = a2 + r*b3 + s*b4;
   b1 = a1 + r*b2 + s*b3;
   b0 = a0 + r*b1 + s*b2;

   c4 = b4;
   c3 = b3 + r*c4;
   c2 = b2 + r*c3 + s*c4;
   c1 = b1 + r*c2 + s*c3;

   denom = c2*c2 – c1*c3;

   if ((denom != 0.0) && (count < 1000)) {
     delta_r = (-b1*c2 + b0*c3)/denom;
     delta_s = (-b0*c2 + b1*c1)/denom;
     r += delta_r;
     s += delta_s;
     count++;
     if ((fabs(delta_r) > tol) || (fabs(delta_s) > tol)) {
       flag = 1;
     }
   } else {
     // perturb the r and s values and start again
     r = 500.0*(0.5-(double) rand()/(double) RAND_MAX);
     s = 500.0*(0.5-(double) rand()/(double) RAND_MAX);
     count = 0;
     flag = 1;
   }
 } while (flag != 0);

 m2 = 1.0;
 m1 = -r;
 m0 = -s;
 n2 = a4;
 n1 = a3+a4*r;
 n0 = -a0/s;
```

Solving a Quadratic Equation

The normal approach:

$$T = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2a_0}}{2a_2}$$

but if $a_1^2 >> 4a_2a_0$, then

$$T = \frac{-a_1 \pm a_1'}{2a_2} \quad \text{where} \quad a_1' = \sqrt{a_1^2 - 4a_2a_0} \approx a_1$$

and cancellation will occur for the "-" root.

A better approach to avoid cancellation error:

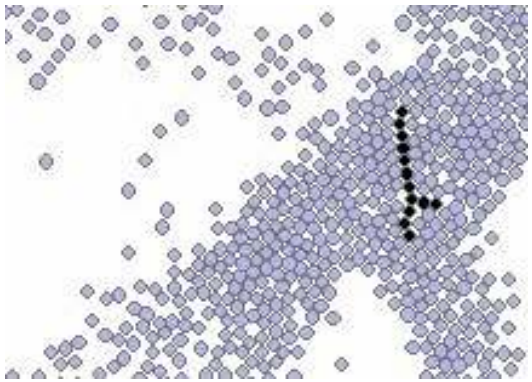
$$q = -\tfrac{1}{2}\left[a_1 + \operatorname{sgn}(a_1)\sqrt{a_1^2 - 4a_2a_0} \right]$$

$$T = \frac{q}{a_2}, \frac{a_0}{q}$$

```
if (a2 != 0.0) {  // check to see if a linear equation
  if ((temp = a1*a1 – 4.0*a2*a0) < 0.0)
    root1 = root2 = NaN;  // imaginary roots
  else {
    if (a1 == 0.0) {
      root1 = sqrt(-a0/a2);
      root2 = -root1;
    } else {
      if (a1 < 0.0)
        q = -0.5*(a1 - sqrt(temp));
      else
        q = -0.5*(a1+sqrt(temp));
      root1 = q/a2;
      root2 = a0/q;
    }
  }
 } else {  // a linear eqn, not a quadratic
  if (a1 != 0.0)
    root1 = root2 = -a0/a1;
  else  // a0 = 0
    root1 = root2 = NaN;  // no roots to solve for
}
```

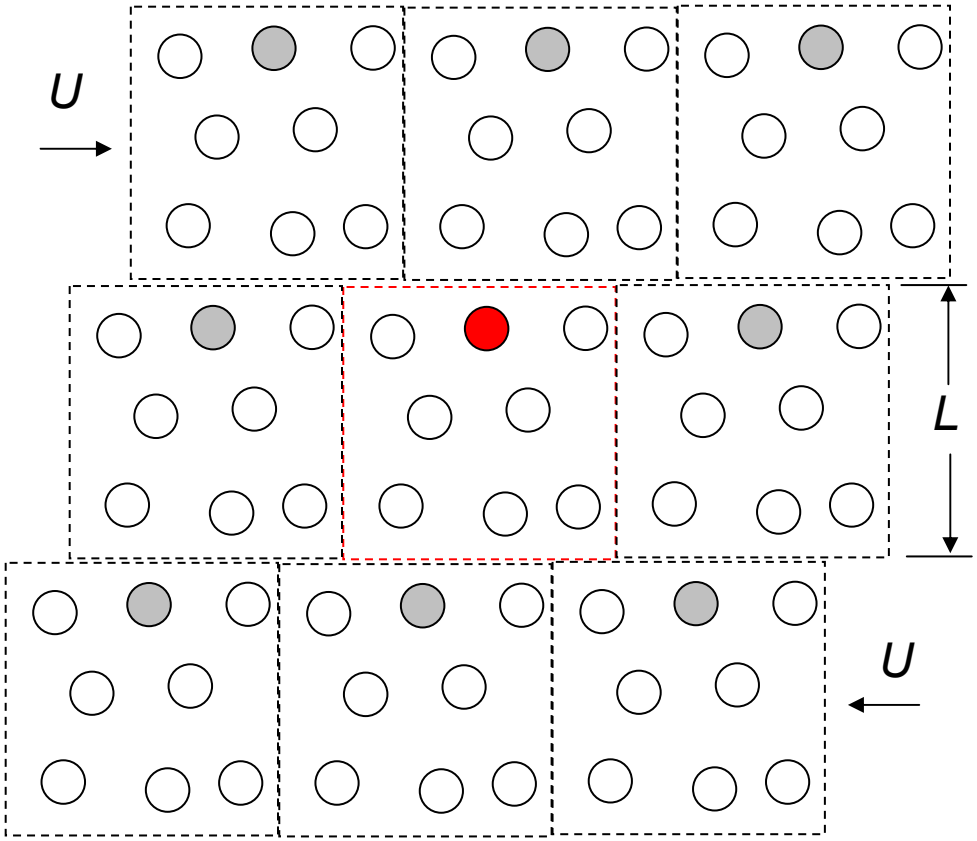
Inelastic Collapse

- *e.g.*, McNamara and Young (1992)
- Occurs when collisions occur in rapid succession
  - e.g., a particle coming to rest on a surface
  - $\Rightarrow$ an infinite number of collisions occurs in finite time
- Problematic for an event driven approach
  - inefficient when many collisions occur in a short amount of time
  - $\Rightarrow$ can't be used to simulate granular materials with long lasting contacts without computational algorithm "fixes" or using a time-step approach
- Inelastic collapse is more likely to occur when
  - normal coeff. of rest. is small (e.g. $\varepsilon_N <\approx 0.6$)
  - the solids fraction is large (less significant than ε_N)



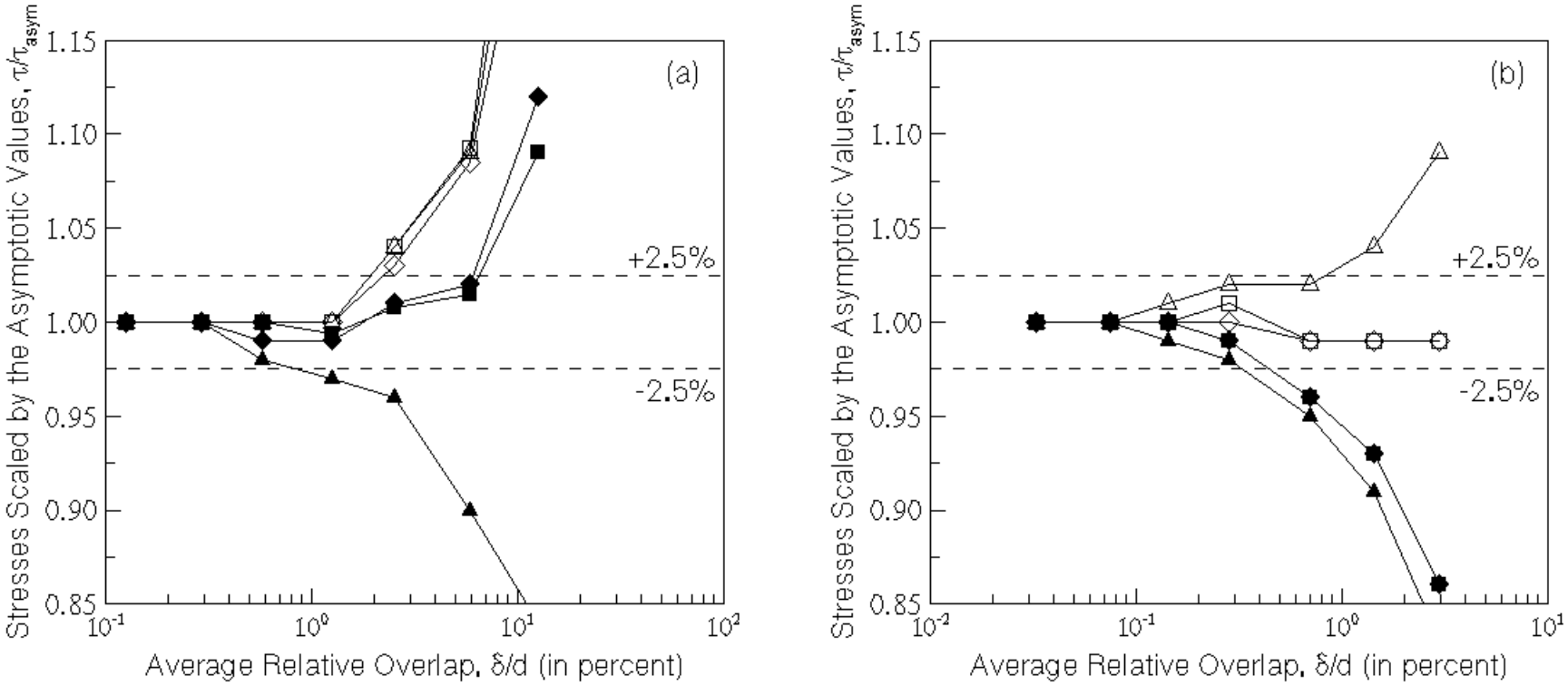
Time Step vs. Event Driven

- Consider the pure shear, 2D simulations by Ketterhagen *et al*. (2005)
  - Lees-Edwards (1972) boundary conditions with shear rate $\gamma = U/L$
  - calculate stresses in the domain
  - use time step and event driven approaches
  - compare results to kinetic theory predictions
  - relative overlap, δ/d, proportional to time step, Δt



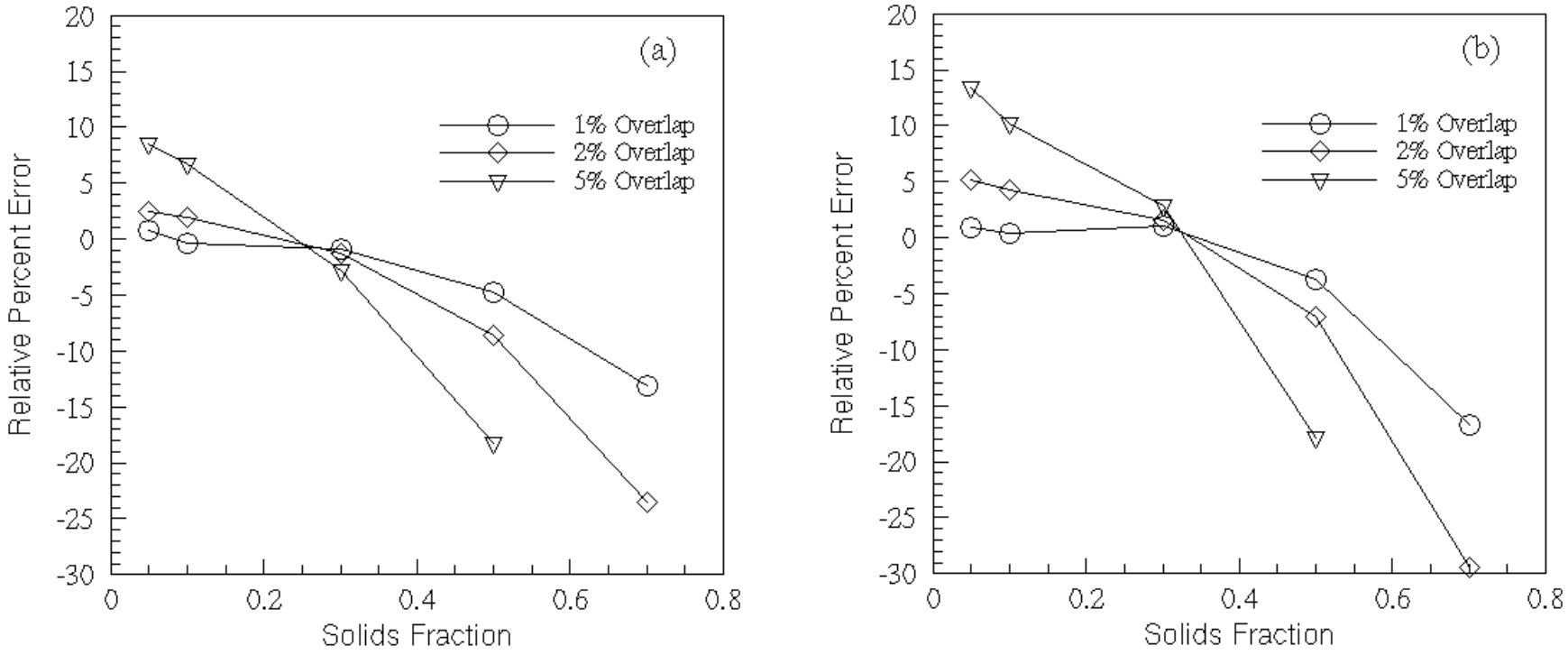
$$\Delta\dot{\mathbf{x}}_C \sim \gamma d \quad \text{and} \quad \delta \sim \Delta\dot{\mathbf{x}}_C \Delta t \quad \Rightarrow \quad \frac{\delta}{d} \sim \gamma \Delta t$$

Time Step vs. Event Driven…



Stress results from the time step driven algorithm for ε = 0.9 (a) ν = 0.1, and (b) ν = 0.5. For sufficiently small time steps (relative overlap is proportional to Δt), the stresses approach an asymptotic value. As the time step increases (*i.e*. overlap increases), the error increases. The horizontal lines show relative error thresholds of ± 2.5%. Open symbols: kinetic contribution, closed symbols: collisional contribution. Squares: xx component, triangles: -xy component, and diamonds: yy component.

Time Step vs. Event Driven…



Percent error in stress results as compared to the asymptotic values (equivalent to the event driven results) for the time step driven model at (a) $\varepsilon = 0.9$ and (b) $\varepsilon = 0.5$ and a range of solid fractions.

Time Step vs. Event Driven…

- The time step driven approach can be more computationally efficient than the event-driven approach at large solid fractions where frequent collisions occur.
  - larger errors, however, as solid fraction, relative impact speed, and time step increase
- As the time step decreases, results from the time step driven algorithm approach those from event driven algorithm
- One cannot easily model the effects of other forces such as electrostatic or aerodynamic forces if an event driven approach is used. These effects can be modeled using a time step driven approach.

Summary

- **Hard-particle simulations are either:**
  - time step driven
    - time proceeds in sufficiently small increments
    - can incorporate forces on particles between collisions
    - is not subject to inelastic collapse
    - slower for dilute systems, faster for dense systems
    - may have multi-particle collisions
    - errors due to overlaps increase as solid fraction, time step, and relative impact speed increase
  - event driven
    - time proceeds from collision to collision
    - computational algorithm utilizes a “collision list”
    - simulation can suffer from inelastic collapse
    - slower for dense systems, faster for dilute systems

References

Hoffman, J.D., 2001, *Numerical Methods for Engineers and Scientists*, Marcel-Dekker, New York.

Hopkins, M.A. and Louge, M.Y., 1991, “Inelastic microstructure in rapid granular flows of smooth disks,” *Physics of Fluids A*, Vol. 3, No. 1, pp. 47 – 57.

Ketterhagen, W.R., Curtis, J.S., and Wassgren, C.R., 2005, “Stress results from two-dimensional granular shear flow simulations using various collision models,” *Physical Review E*, Vol. 71, Article 061307.

Lasinski, M.E., Curtis, J.S., and Pekny, J.F., 2004, “Effect of system size on particle-phase stress and microstructure formation,” *Physics of Fluids*, Vol. 16, No. 2, pp. 265 – 273.

Lees, A.W. and Edwards, S.F., 1972, “Computer study of transport processes under extreme conditions,” *Journal of Physics C*, Vol. 5, No. 15, pp. 1921 – 1929.

McNamara, S. and Young, W.R., 1992, “Inelastic collapse and clumping in a one-dimensional granular medium,” *Physics of Fluids A*, Vol. 4, No. 3, pp. 496 – 504.