CS 294-73 (CCN 27241)
Software Engineering for Scientific Computing

http://www.cs.berkeley.edu/~colella/CS294

Homework 4 Notes

Discretizing ODEs

Fourth-order Runge-Kutta:

$$
\begin{aligned}
k_1 &= F(Q^n, t^n) \\
Q^{n,(1)} = Q^n + \frac{\Delta t}{2} k_1 \ , \quad k_2 &= F(Q^{n,(1)}, t^{n+\frac{1}{2}}) \\
Q^{n,(2)} = Q^n + \frac{\Delta t}{2} k_2 \ , \quad k_3 &= F(Q^{n,(2)}, t^{n+\frac{1}{2}}) \\
Q^{n,(3)} = Q^n + \Delta t k_3 \ , \quad k_4 &= F(Q^{n,(3)}, t^{n+1})
\end{aligned}
$$

$$
Q^{n+1} = Q^n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)
$$

$$
\frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) = \frac{1}{\Delta t} \int_{t^n}^{t^n + \Delta t} f(Q(t), t) dt + O(\Delta t)^4
$$

Generalizes Simpson's rule for integrals.

Rewriting RK4

```
K := 0; delta := 0;

K:= dt*F(Q+K,t); delta+=K; K*= ½;

K:= dt*F(Q+K,t+½*dt); delta+=2*K; K*=½;

K:= dt*F(Q+K,t+½*dt); delta+=2*K;

K:=dt*F(Q+K,t+dt); delta += K;

Q += delta/6;
```

$$k_1 = F(Q^n, t^n)$$

$$Q^{n,(1)} = Q^n + \frac{\Delta t}{2} k_1 \ , \ k_2 = F(Q^{n,(1)}, t^{n+\frac{1}{2}})$$

$$Q^{n,(2)} = Q^n + \frac{\Delta t}{2} k_2 \ , \ k_3 = F(Q^{n,(2)}, t^{n+\frac{1}{2}})$$

$$Q^{n,(3)} = Q^n + \Delta t k_3 \ , \ k_4 = F(Q^{n,(3)}, t^{n+1})$$

$$Q^{n+1} = Q^n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

Programming ODE Methods

Generic programming: only want to do it once.

```
template <class X, class F, class dX>
class RK4
{
public:
  ~RK4();
   void advance(double a_time, double a_dt, X& a_state);
protected:
  dX m_k, m_delta;
  F m_f;
};
```

ParticleVelocities

For this assignment, you will declare this class with the template parameters

```
RK4<ParticleSet,ParticleVelocities,ParticleShift> rk4;
```

`ParticleSet` and `ParticleShift` are defined for you, except for `ParticleSet::rebin`, which you must implement.

`ParticleVelocities` has a single member function you must implement:

```
void ParticleVelocities::operator()
       (ParticleShift& a_kOut,
        const Real& a_time, const Real& a_dt,
        const ParticleSet& a_state
        const ParticleShift& a_kIn)
```

It implements the operation of evaluating `dt*F(Q+K,t)` in our reformulation of `RK4` above.

`a_state <-> Q, a_kIn <-> K , a_dt <-> dt, a_time<-> t, a_kOut <-> dt*F(Q+K,t).`

`ParticleVelocities`

You will implement this class for two functions:

- Rigid-body rotation: $\vec{u}(x, y) = (-y, x)$
- The MLC algorithm described in the lecture.