Features of various types of methods

	Advantages	Disadvantages
1. **explicit**:	just evaluation, simplest, easiest	poor stability, must restrict Δt for stiff
implicit:	must solve equ. at each timestep	good stability, best for stiff, expensive
2. **single-step**:	self-starting	more evaluations for high order
multi-step:	fewer evals for high order, good for adaptive, best for stiff, good for high order, good for predictor-corrector	not self-starting (need a singlestep to start)
3. **Runge-Kutta**:	efficient (for non-stiff), RKF45 for adaptive	difficult to construct very high order

4. **Symplectic**: best for long-time integration of Hamiltonian systems
preserve "volume" in phase-space 

best: **Verlet** (1967) for $ma = F$ written as $\begin{cases} \dot{x} = v \\ \dot{v} = \frac{1}{m} F(t, x) \end{cases}$

velocity Verlet scheme:
explicit, 2^{nd} order,
self-starting

$$x_{n+1} = x_n + \Delta t \cdot v_n + \frac{\Delta t^2}{2} a_n$$

$$v_{n+1/2} = v_n + \frac{\Delta t}{2} a_n$$

$$a_{n+1} = \frac{1}{m} F(t_{n+1}, x_{n+1})$$

$$v_{n+1} = v_{n+1/2} + \frac{\Delta t}{2} a_{n+1} \equiv v_n + \frac{\Delta t}{2}(a_n + a_{n+1})$$

5. ETD: Exponential Time Differencing schemes for $y' = \underset{\text{linear}}{Ly} + \underset{\text{nonlinear}}{N(y)}$
see wikipedia article   good for stiff

6. PEER methods: RK type, up to order 9

7. SSP schemes: Strong Stability Preserving; for conservation laws (PDEs)

ODE systems

of 1^{st} order ODEs arise most often in applications,
and IVPs for higher order ODEs can be re-written as 1^{st} order systems.

All ODE integrators naturally generalize to systems. $\begin{cases} \vec{y}\,' = \vec{F}(t, \vec{y}) \\ \vec{y}(t_0) = \vec{y}^{\,0} \end{cases}$
and software are written for systems.

e.g. Euler scheme for 2×2 system: $\begin{cases} y_1' = f_1(t, y_1, y_2), \ y_1(t_0) = y_1^0 \\ y_2' = f_2(t, y_1, y_2), \ y_2(t_0) = y_2^0 \end{cases}$

Let $Y1n \approx y_1(t_n)$, $Y2n \approx y_2(t_n)$.

Euler scheme: $\Delta t = \dfrac{t_{end} - t_0}{Nsteps}$

function y1p = f1(t, y1, y2)
function y2p = f2(t, y1, y2)

initialize: $Y1n = y_1^0$, $Y2n = y_2^0$, $tn = t_0$

timestepping: for n = 1 : Nsteps

$Y1n = Y1n + \Delta t \cdot f1(t_n, Y1n, Y2n)$
$Y2n = Y2n + \Delta t \cdot f2(tn, Y1n, Y2n)$
$tn = tn + n \cdot \Delta t$
print ----

end

Software for ODE solvers   many exist!

1. Matlab: ode23 , ode113
   ode45 (= RKF45, best adaptive code for non-stiff)
   for stiff: ode15s
   ode23s , ode23t , ode23tb

   EXPINT: exponential integrator, Matlab code, explicit

2. netlib.org: software repository, mostly Fortan, C
   odepack / has several codes of LSODE family:
   lsoda.f ; auto selects stiff/nonstiff solver

   ode/ some of best codes: rksuite, vode, dassl

3. GSL = Gnu Scientific Lib: C codes, some for ODEs

4. Intel ODE solvers, in C

5. Maple: has all main methods; as choices in 'dsolve':
   method = classical rkf45 dverk78 (RK)
   gear, lsode ...

6. Ode by Keith Briggs: RK(8) by Hairer, up to 20 ODEs: $a' = f(x, a)$
   has command-line interface, provides error estimate
   e.g. $a' = 3a^2 - x^3/4$, $a(0) = 9$, up to $t_{end} = .5$, print 5 steps:
   ode "3*a^2 - x^3/4" 9 0 .5 5