

Numerical Optimization Applied to Space-Related Problems

Robert J. Vanderbei

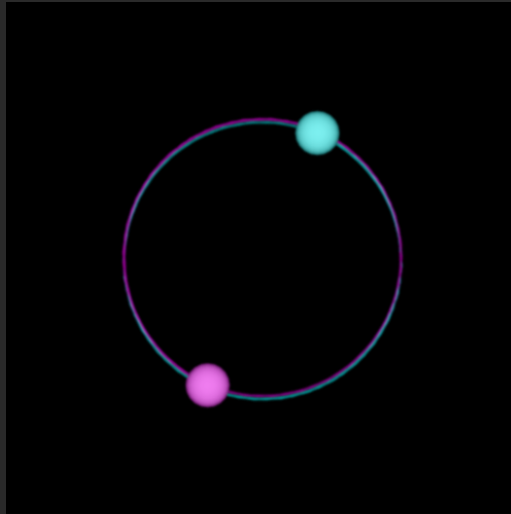
2015 July 16

MOPTA 2015
Lehigh University

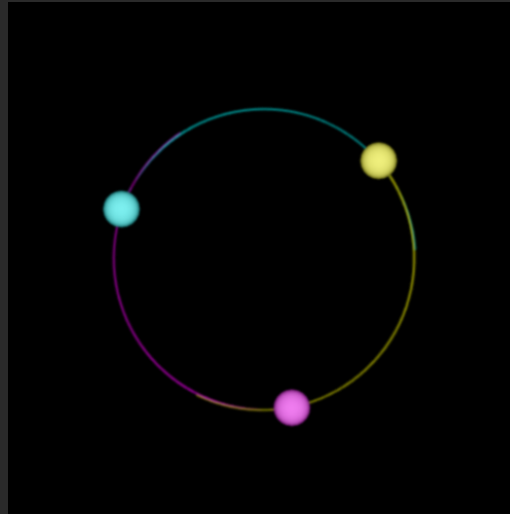
<http://www.princeton.edu/~rvdb>

Lagrange Style Orbits

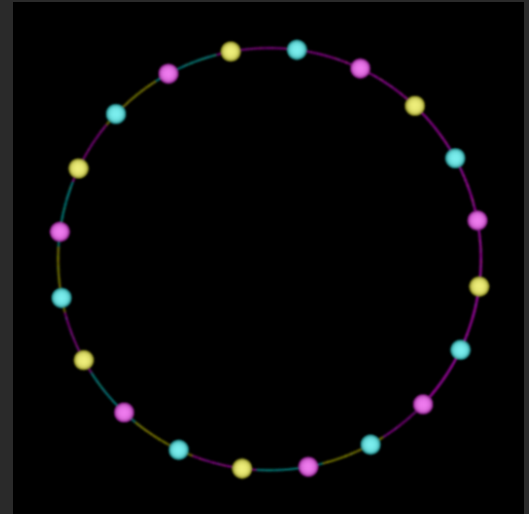
Click anywhere below for WebGL simulation



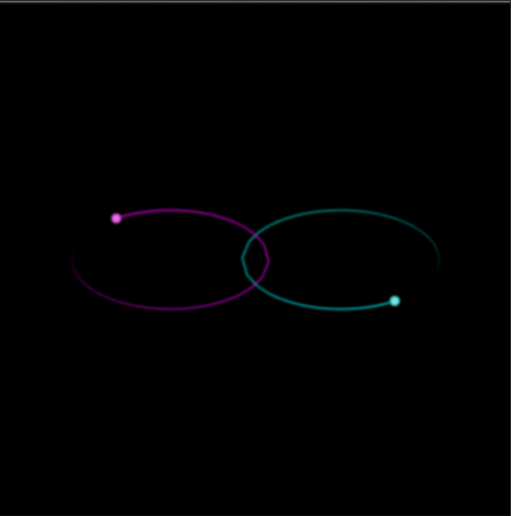
Reset Pause/Run Select an orbit: Lagrange2



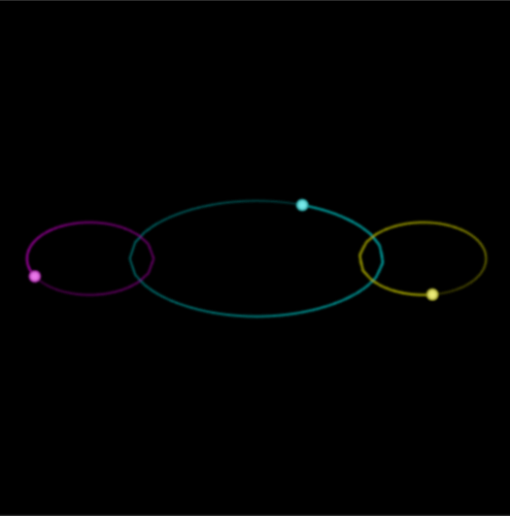
Reset Pause/Run Select an orbit: Lagrange3



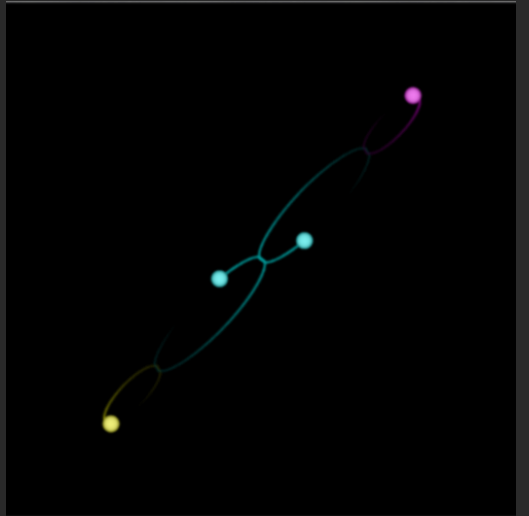
Reset Pause/Run Select an orbit: Lagrange20



Reset Pause/Run Select an orbit: Double Ellipse

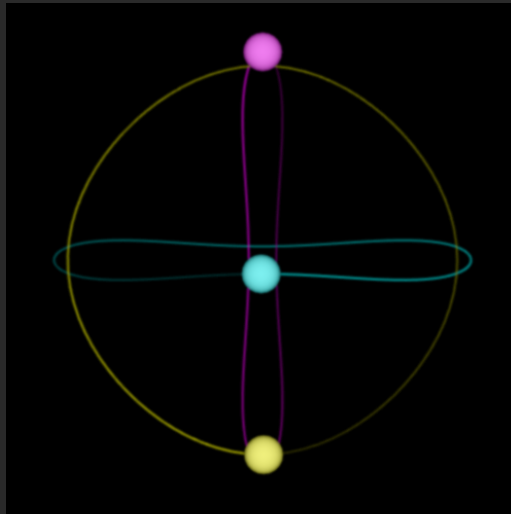


Reset Pause/Run Select an orbit: Triple Ellipse

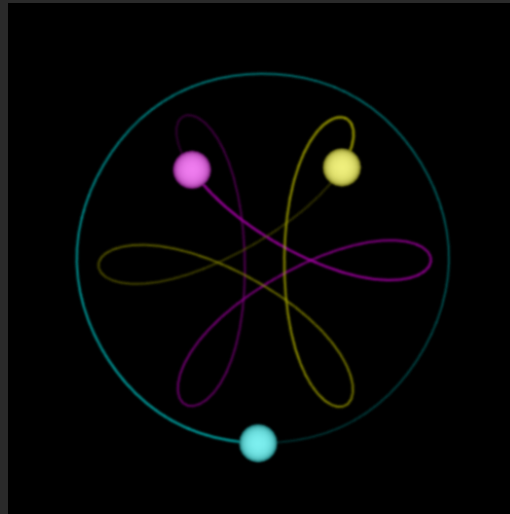


Reset Pause/Run Select an orbit: Quad Ellipse

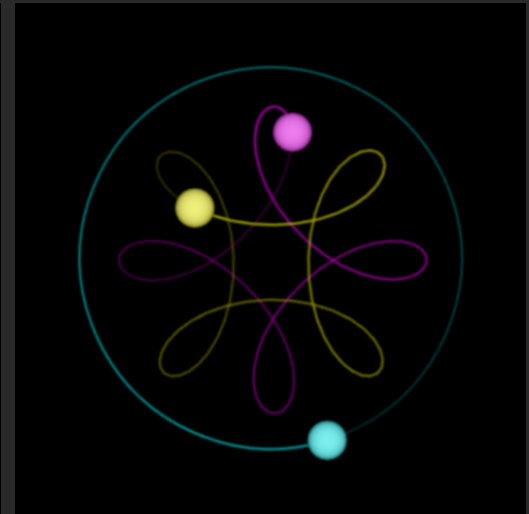
Hill-Type and Double/Doubles



Reset Pause/Run Select an orbit: Ducati3



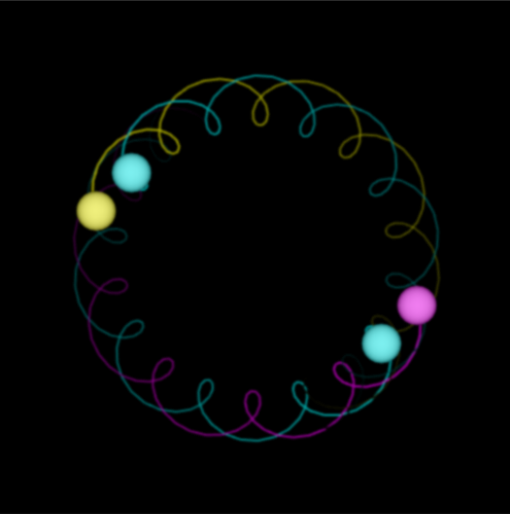
Reset Pause/Run Select an orbit: Hill 2 months/yr



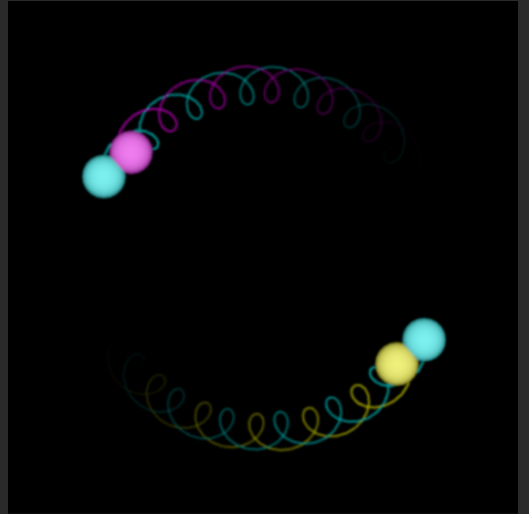
Reset Pause/Run Select an orbit: Hill 3 months/yr



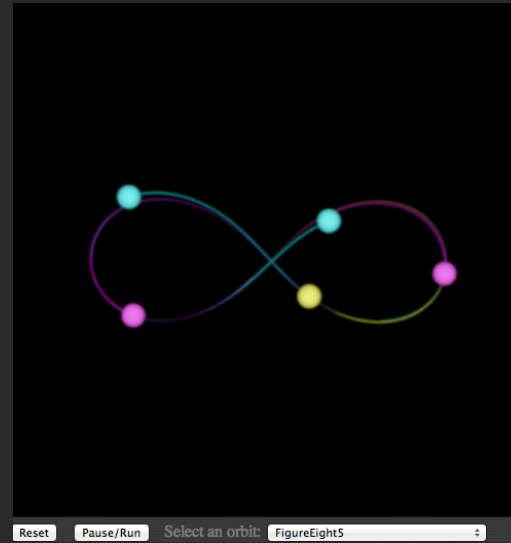
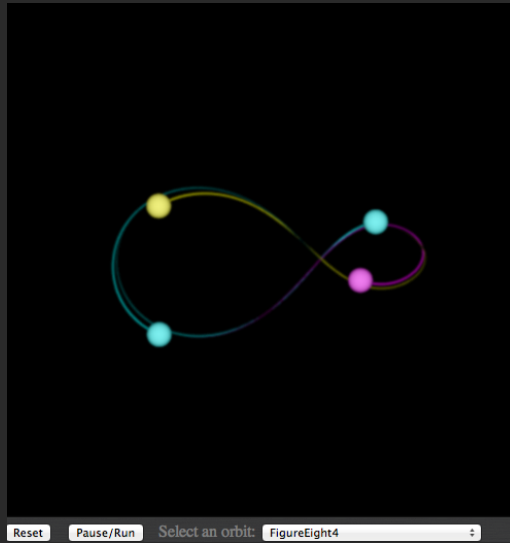
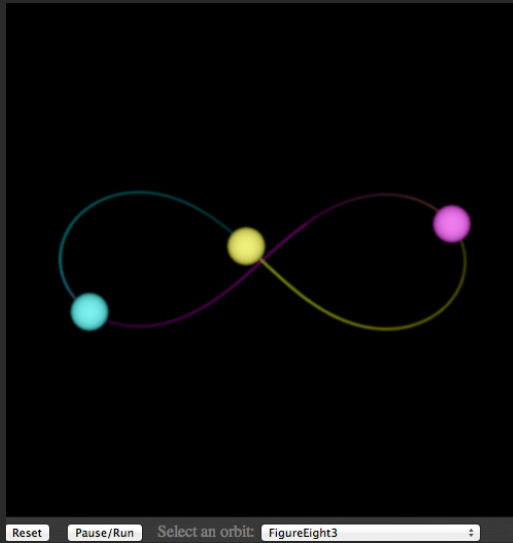
Reset Pause/Run Select an orbit: DoubleDouble5



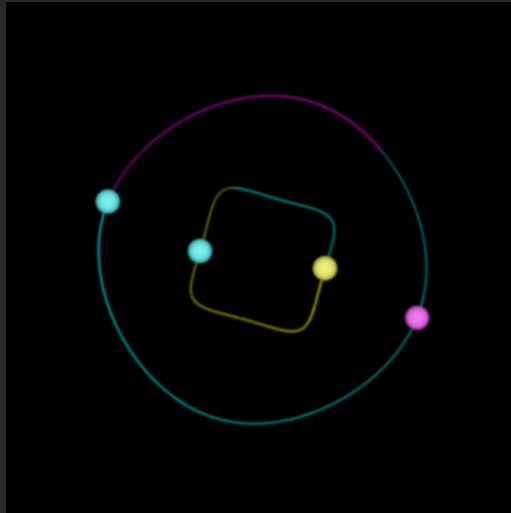
Reset Pause/Run Select an orbit: DoubleDouble10



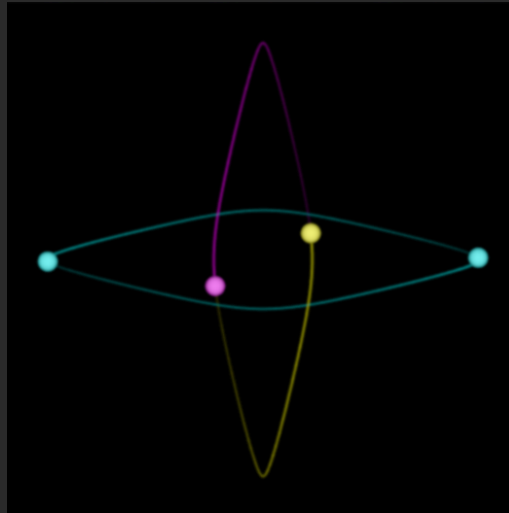
Reset Pause/Run Select an orbit: DoubleDouble20



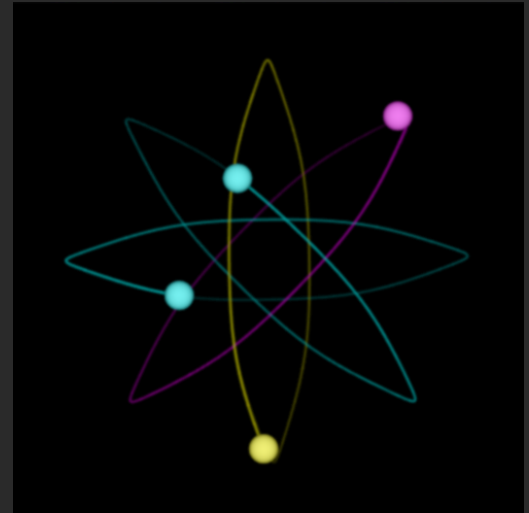
Exotic But Unstable



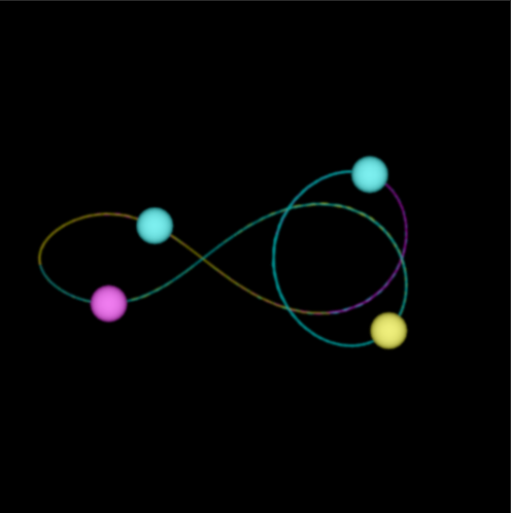
Reset Pause/Run Select an orbit: PlateSaucer4



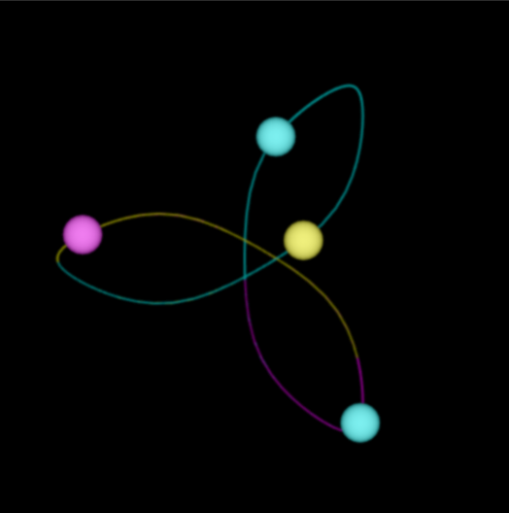
Reset Pause/Run Select an orbit: 1Month1Year



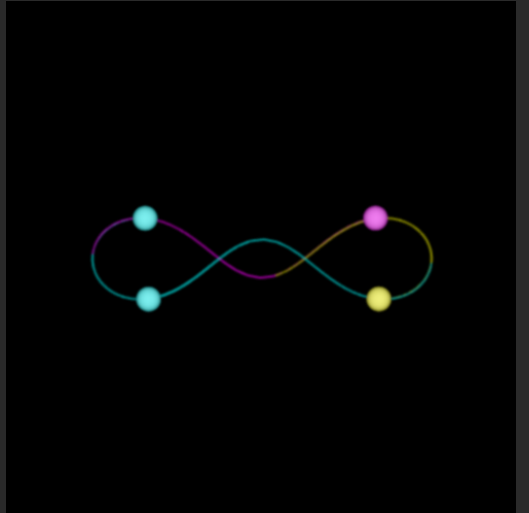
Reset Pause/Run Select an orbit: 1Month1YearAgain



Reset Pause/Run Select an orbit: FoldedTriLoop4

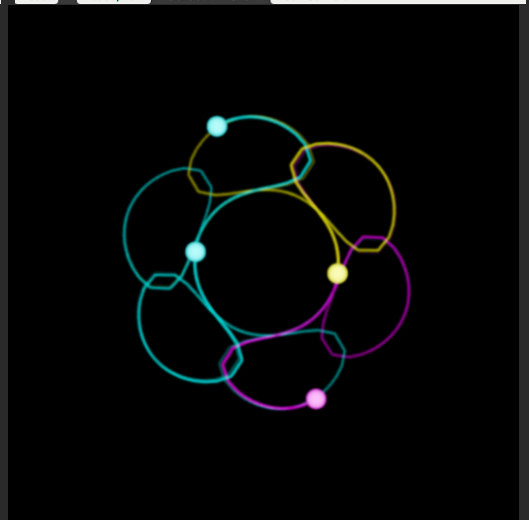
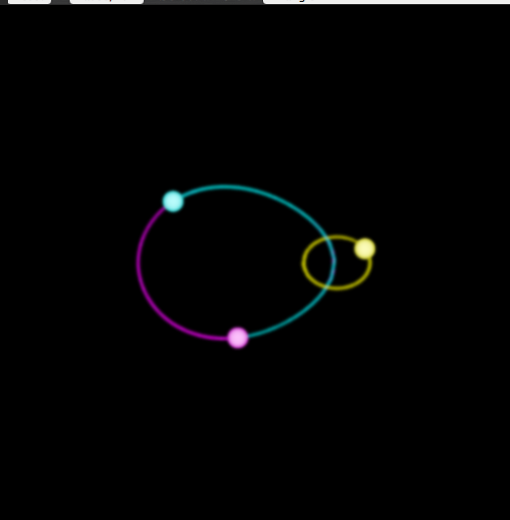
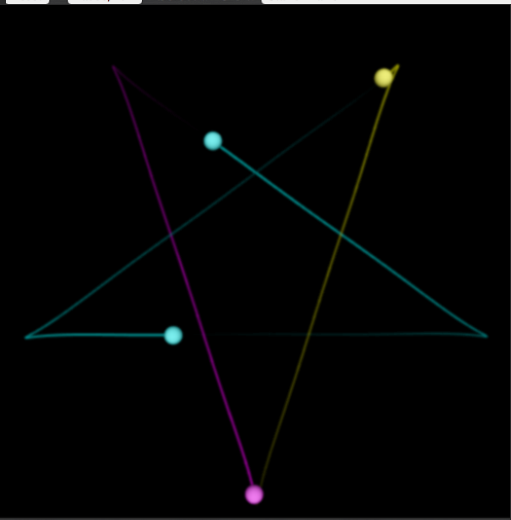
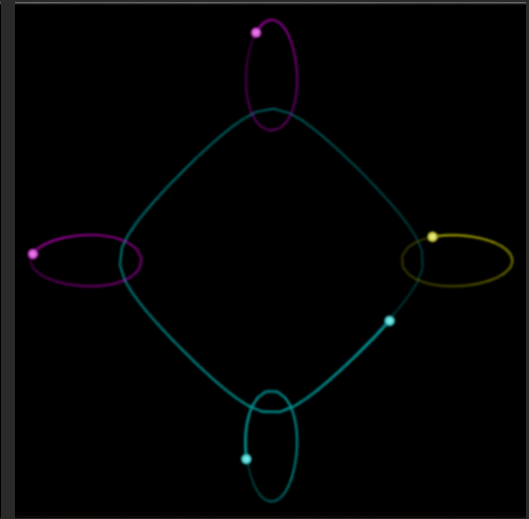
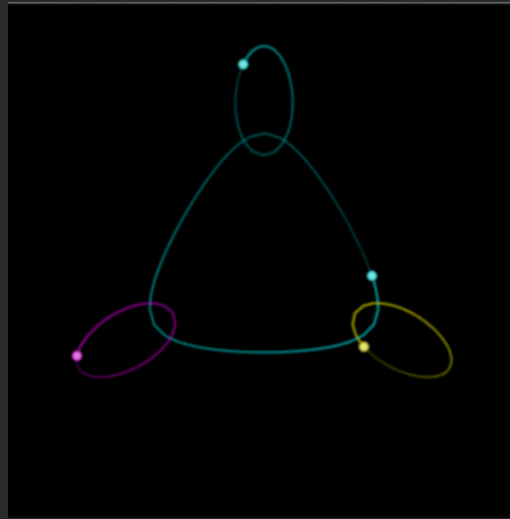
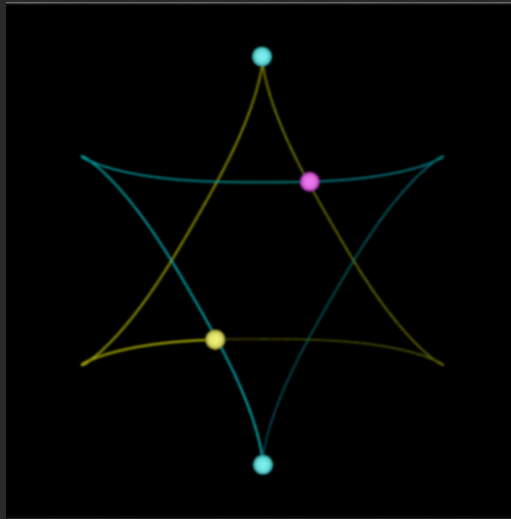


Reset Pause/Run Select an orbit: Trefoil4



Reset Pause/Run Select an orbit: Braid4

New Ones



Least Action Principle

Given: n bodies.

Let:

- m_j denote the mass and
- $z_j(t) = \begin{bmatrix} x_j(t) \\ y_j(t) \end{bmatrix}$ denote the position in \mathbb{R}^2 (or \mathbb{R}^3) of body j at time t .

Action Functional:

$$A = \int_0^{2\pi} \left(\sum_j \frac{1}{2} m_j \|\dot{z}_j\|^2 + \sum_{j,k:k < j} \frac{Gm_j m_k}{\|z_j - z_k\|} \right) dt.$$

Theorem: *Any critical point of the action functional is a solution of the n -body problem.*

Equation of Motion

First Variation:

$$\begin{aligned}\delta A &= \int_0^{2\pi} \left(\sum_j m_j \dot{z}_j^T \delta \dot{z}_j - \sum_{j,k:k<j} G m_j m_k \frac{(z_j - z_k)^T (\delta z_j - \delta z_k)}{\|z_j - z_k\|^3} \right) dt \\ &= - \int_0^{2\pi} \sum_j \left(m_j \ddot{z}_j + \sum_{k:k \neq j} G m_j m_k \frac{z_j - z_k}{\|z_j - z_k\|^3} \right)^T \delta z_j dt\end{aligned}$$

Setting first variation to zero, we get:

$$m_j \ddot{z}_j = - \sum_{k:k \neq j} G m_j m_k \frac{z_j - z_k}{\|z_j - z_k\|^3}, \quad j = 1, 2, \dots, n.$$

An AMPL Model

```
param n := 4;          # number of masses
param T := 40000;     # number of terms in numerical approx to integral over one period

param G := 1;

param period := 2*pi; # temporal length of the orbit segment
param dt := period / T;
set Time circular = setof {j in 0..T-1} j*dt;

var x {j in 1..n, t in Time} >= -5, <= 5;
var y {j in 1..n, t in Time} >= -5, <= 5;

var xdot {j in 1..n, t in Time} = (x[j,next(t)]-x[j,t])/dt;
var ydot {j in 1..n, t in Time} = (y[j,next(t)]-y[j,t])/dt;

var K {t in Time}
    = 0.5 * sum {j in 1..n} (xdot[j,t]^2 + ydot[j,t]^2);

var P {t in Time}
    = - sum {j in 1..n, k in 1..n: k<j} 1/sqrt((x[j,t]-x[k,t])^2 + (y[j,t]-y[k,t])^2);

minimize Action: sum {t in Time} (K[t] - P[t])*dt;
```

A Double/Double Initialization

```
param omega0 := 1;
param omega1 := 2;
param phi_p := -pi/2;
param phi_m := pi/2;
param r0 := (G/(2*omega0^2))^(1./3.);
param r1 := (G/(4*omega1^2))^(1./3.);

let {t in Time} x[0,t] := r0*cos(omega0*t) + r1*sin(-omega1*t + phi_p);
let {t in Time} y[0,t] := r0*sin(omega0*t) + r1*cos(-omega1*t + phi_p);

let {t in Time} x[1,t] := r0*cos(omega0*t) - r1*sin(-omega1*t + phi_p);
let {t in Time} y[1,t] := r0*sin(omega0*t) - r1*cos(-omega1*t + phi_p);

let {t in Time} x[2,t] := -r0*cos(omega0*t) + r1*cos(-omega1*t + phi_m);
let {t in Time} y[2,t] := -r0*sin(omega0*t) + r1*sin(-omega1*t + phi_m);

let {t in Time} x[3,t] := -r0*cos(omega0*t) - r1*cos(-omega1*t + phi_m);
let {t in Time} y[3,t] := -r0*sin(omega0*t) - r1*sin(-omega1*t + phi_m);

solve;
```

Limitations of the Model

- The physical problem is *infinite dimensional—a continuum*.
- Such problems can be really tough. Let's call it the *curse of the continuum*.
- *Discretize* the integral to a finite sum.
- Solutions can occur at *local maxima* and at *saddle points*.
Looking only for *local minima*, we miss these others.

Alternate Approach: Solve Equations of Motion

Minimizing the action functional is an unconstrained optimization problem...

$$\begin{aligned} &\text{minimize} && \int_0^{2\pi} \left(\sum_j \frac{1}{2} m_j \|\dot{z}_j\|^2 + \sum_{j,k:k<j} \frac{Gm_j m_k}{\|z_j - z_k\|} \right) dt, \\ &\text{subject to} && \text{no constraints,} \end{aligned}$$

Instead, we could simply look for trajectories that satisfy Newton's laws:

$$\begin{aligned} &\text{minimize} && 0, \\ &\text{subject to} && m_j \ddot{z}_j = - \sum_{k:k \neq j} Gm_j m_k \frac{z_j - z_k}{\|z_j - z_k\|^3}, \quad j = 1, 2, \dots, n, \quad 0 \leq t \leq 2\pi. \end{aligned}$$

(Note: the z_j 's are vectors in space so this is 2 (or 3) times n times a continuum.)

AMPL Model for the Equations of Motion

```
var x {i in 1..n, t in Time} >= -5, <= 5;
var y {i in 1..n, t in Time} >= -5, <= 5;

var xdot {i in 1..n, t in Time} = (x[i,next(t)]-x[i,t])/dt;
var ydot {i in 1..n, t in Time} = (y[i,next(t)]-y[i,t])/dt;

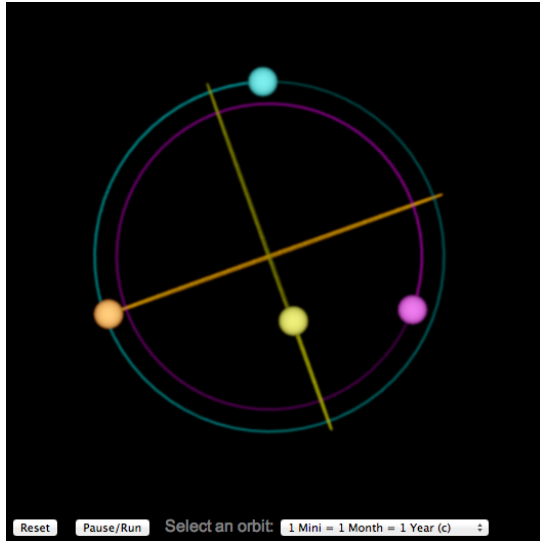
var xdot2 {i in 1..n, t in Time} = (xdot[i,t]-xdot[i,prev(t)])/dt;
var ydot2 {i in 1..n, t in Time} = (ydot[i,t]-ydot[i,prev(t)])/dt;

minimize zero: 0;

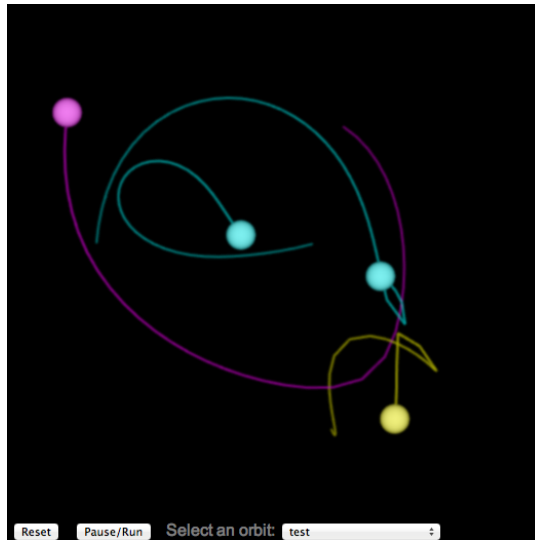
subject to F_equals_ma_x {i in 1..n, t in Time}:
    xdot2[i,t]
    = sum {j in 1..n: j != i}
        (x[j,t]-x[i,t]) / ((x[j,t]-x[i,t])^2+(y[j,t]-y[i,t])^2)^(3/2);

subject to F_equals_ma_y {i in 1..n, t in Time}:
    ydot2[i,t]
    = sum {j in 1..n: j != i}
        (y[j,t]-y[i,t]) / ((x[j,t]-x[i,t])^2+(y[j,t]-y[i,t])^2)^(3/2);
```

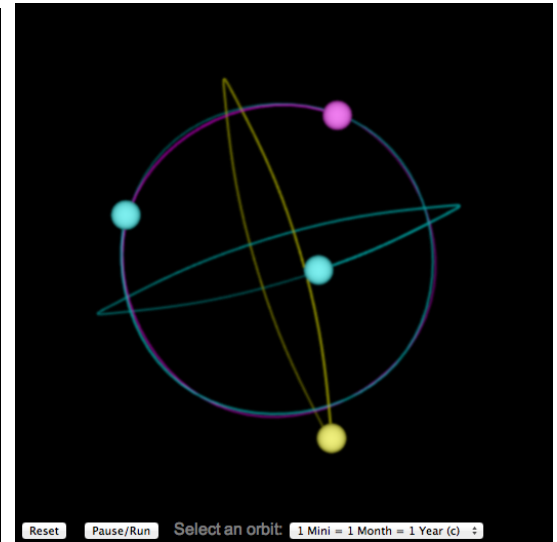
New Solution Via Equations of Motion



A Mini/Month/Year Design



Action Minimization



Equations of Motion

NOTE: Action minimization found an orbit. But, it is immediately unstable as the middle figure shows.

Making Stability an Objective

To the *Equations of Motion* model, add these perturbation parameters:

```
set Tim ordered = setof {j in -1..T} j*dt;
param dx := 1e-3;
param dy := 1e-3;
param dvx := 1e-3;
param dvy := 1e-3;
```

These new variables and objective function:

```
# perturbed trajectories
var xp {l in 1..n, k in 1..4, i in 1..n, t in Tim};
var yp {l in 1..n, k in 1..4, i in 1..n, t in Tim};
var xp_dot {l in 1..n, k in 1..4, i in 1..n, t in Tim: t != last(Tim)}
    = (xp[l,k,i,next(t)]-xp[l,k,i,t])/dt;
var yp_dot {l in 1..n, k in 1..4, i in 1..n, t in Tim: t != last(Tim)}
    = (yp[l,k,i,next(t)]-yp[l,k,i,t])/dt;
var xp_dot2 {l in 1..n, k in 1..4, i in 1..n, t in Tim: t in Time}
    = (xp_dot[l,k,i,t]-xp_dot[l,k,i,prev(t)])/dt;
var yp_dot2 {l in 1..n, k in 1..4, i in 1..n, t in Tim: t in Time}
    = (yp_dot[l,k,i,t]-yp_dot[l,k,i,prev(t)])/dt;
```

And...

Add this objective:

minimize instability:

$$\begin{aligned} & \text{sum } \{l \text{ in } 1..n, j \text{ in } 1..n: l \neq j\} (\text{xp}[l,1,j,\text{last}(\text{Time})] - \text{x}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{j \text{ in } 1..n\} (\text{xp}[j,1,j,\text{last}(\text{Time})] - \text{dx} - \text{x}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{l \text{ in } 1..n, j \text{ in } 1..n: l \neq j\} (\text{yp}[l,2,j,\text{last}(\text{Time})] - \text{y}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{j \text{ in } 1..n\} (\text{yp}[j,2,j,\text{last}(\text{Time})] - \text{dy} - \text{y}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{l \text{ in } 1..n, j \text{ in } 1..n: l \neq j\} (\text{xp_dot}[l,3,j,\text{last}(\text{Time})] - \text{xdot}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{j \text{ in } 1..n\} (\text{xp_dot}[j,3,j,\text{last}(\text{Time})] - \text{dvx} - \text{xdot}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{l \text{ in } 1..n, j \text{ in } 1..n: l \neq j\} (\text{yp_dot}[l,4,j,\text{last}(\text{Time})] - \text{ydot}[j,\text{last}(\text{Time})])^2 \\ & + \text{sum } \{j \text{ in } 1..n\} (\text{yp_dot}[j,4,j,\text{last}(\text{Time})] - \text{dvy} - \text{ydot}[j,\text{last}(\text{Time})])^2 \end{aligned}$$

And these constraints defining the perturbed trajectories:

subject to F_eq_ma_x_pert {l in 1..n, k in 1..4, i in 1..n, t in Time}:

$$\begin{aligned} & \text{xp_dot2}[l,k,i,t] \\ & = \text{sum } \{j \text{ in } 1..n: j \neq i\} \\ & \quad (\text{xp}[l,k,j,t] - \text{xp}[l,k,i,t]) / \\ & \quad ((\text{xp}[l,k,j,t] - \text{xp}[l,k,i,t])^2 + (\text{yp}[l,k,j,t] - \text{yp}[l,k,i,t])^2)^{(3/2)}; \end{aligned}$$

subject to F_eq_ma_y_pert {l in 1..n, k in 1..4, i in 1..n, t in Time}:

$$\begin{aligned} & \text{yp_dot2}[l,k,i,t] \\ & = \text{sum } \{j \text{ in } 1..n: j \neq i\} \\ & \quad (\text{yp}[l,k,j,t] - \text{yp}[l,k,i,t]) / \\ & \quad ((\text{xp}[l,k,j,t] - \text{xp}[l,k,i,t])^2 + (\text{yp}[l,k,j,t] - \text{yp}[l,k,i,t])^2)^{(3/2)}; \end{aligned}$$

subject to x_pert_init {i in 1..n}: xp[i,1,i,0] = x[i,0] + dx;

subject to y_pert_init {i in 1..n}: yp[i,2,i,0] = y[i,0] + dy;

subject to xdot_pert_init {i in 1..n}: xp_dot[i,3,i,0] = xdot[i,0] + dvx;

subject to ydot_pert_init {i in 1..n}: yp_dot[i,4,i,0] = ydot[i,0] + dvy;

Numerical Results

Name	Obj. Func. Value
Ducati3	1.4e-5
Star of David	1.6e-5
1Month1Year	1.4e-5
1Mini1Month1Year	5.5e-5
1Mini1Month1Year2	2.5e-1
1Mini1Month1Year(c)	4.2e-2
Five Point Star	8.9e-6

Thank You!

Backup Slides

Sensitivity Analysis

Let

$$\xi^*(t) = \begin{bmatrix} z^*(t) \\ v^*(t) \end{bmatrix}$$

be a solution to

$$\dot{\xi} = A(\xi)$$

where

$$A \left(\begin{bmatrix} z(t) \\ v(t) \end{bmatrix} \right) = \begin{bmatrix} v(t) \\ a(z(t)) \end{bmatrix}$$

and

$$a(z) = \begin{bmatrix} a_1(z) \\ \vdots \\ a_n(z) \end{bmatrix}$$

and

$$a_j(z) = - \sum_{k:k \neq j} \frac{z_j - z_k}{\|z_j - z_k\|^2}, \quad j = 1, 2, \dots, n.$$

Note: we've chosen units so that $G = 1$ and all masses are unit masses.

Sensitivity Analysis —Continued

Consider a nearby solution $\xi(t)$:

$$\begin{aligned}\dot{\xi}(t) &= A(\xi(t)) \\ &\approx A(\xi^*(t)) + A'(\xi^*(t)) (\xi(t) - \xi^*(t)) \\ &= \dot{\xi}^*(t) + A'(\xi^*(t)) (\xi(t) - \xi^*(t)).\end{aligned}$$

Put $\Delta\xi = \xi - \xi^*$. Then $\dot{\Delta\xi} = A'(\xi^*(t)) \Delta\xi$. A finite difference approximation yields

$$\begin{aligned}\Delta\xi(t+h) &= \Delta\xi(t) + \Delta t A'(\xi^*(t)) \Delta\xi(t) \\ &= (I + \Delta t A'(\xi^*(t))) \Delta\xi(t).\end{aligned}$$

Iterating around one period, we get:

$$\Delta\xi(T) = \left(\prod_{i=0}^{n-1} (I + \Delta t A'(\xi^*(t_i))) \right) \Delta\xi(0),$$

where $\Delta t = T/n$ and $t_i = i\Delta t$.

Linear Stability

Stability: all eigenvalues of

$$\Lambda = \prod_{i=0}^{n-1} (I + \Delta t A'(\xi^*(t_i)))$$

must be at most one in magnitude.

Numerical Results – Leap-Frog Stable Orbits

Name	$\max_i(\lambda_i(\Lambda))$
Lagrange2	1.0000
Double Ellipse	1.0001
Triple Ellipse	1.0000
Quad Ellipse	5.7921
Ducati3	1.0000
Broucke-Henon	1.0000
Hill 2 months/yr	1.2430
Hill 3 months/yr	1.4187
Star of David	7.1140
DoubleDouble5	18.8167
DoubleDouble10	8.8401
DoubleDouble20	1.0000
FigureEight3	1.0000
Five Point Star	1.0000

Numerical Results – Leap-Frog Unstable Orbits

Name	$\max_i(\lambda_i(\Lambda))$
Lagrange3	85.0138
Lagrange20	3.3644e+13
FigureEight4	168.38
FigureEight5	2126.0
PlateSaucer4	3658
1Month1Year	17.5830
1Month1YearAgain	1.7697
FoldedTriLoop4	7.4657e+04
Trefoil4	4.1726e+04
Braid4	7.7932e+03
Triangle	25.6724
Four Corners	269.0443
Binary Ellipse	1.4053
Hexagon	9.9829e+04

Leap-Frog Midpoint Integrator (using a Spring)

Differential equation:

$$\ddot{x} = -x$$

Given: $x(0), v(0)$

Compute:

$$a(0) = -x(0)$$

$$v\left(\frac{\Delta t}{2}\right) = v(0) + \frac{\Delta t}{2} a(0)$$

For $t = \Delta t, 2\Delta t, \dots$

$$a(t) = -x(t)$$

$$v\left(t + \frac{\Delta t}{2}\right) = v\left(t - \frac{\Delta t}{2}\right) + \Delta t a(t)$$

$$x(t + \Delta t) = x(t) + \Delta t v\left(t + \frac{\Delta t}{2}\right)$$

t	x	v	a
0.0	1.000	0.000	-1.000
		-0.050	
0.1	0.995		-0.995
		-0.150	
0.2	0.980		-0.980
		-0.248	
0.3	0.955		-0.955
		-0.343	
0.4	0.921		-0.921
		-0.435	
0.5	0.877		-0.877

The Midpoint Integrator

```
if (integrator == MIDPOINT) {
    for (j=0; j<n; j++) {
        p[j].x += p[j].vx * dt;
        p[j].y += p[j].vy * dt;
    }
    for (j=0; j<n; j++) {
        p[j].ax = 0; p[j].ay = 0;
        for (i=0; i<n; i++) {
            if (i != j) {
                double r3 = dist3(p[i], p[j]);
                if (r3<r03) r3=r03;
                p[j].ax -= G * p[i].m * (p[j].x - p[i].x)/r3;
                p[j].ay -= G * p[i].m * (p[j].y - p[i].y)/r3;
            }
        }
    }
    for (j=0; j<n; j++) {
        p[j].vx += p[j].ax * dt;
        p[j].vy += p[j].ay * dt;
    }
}
```