Celestial Mechanics Problem Set 2: The State Transition Matrix and Method of Differential Corrections

J.D. Mireles James

December 11, 2006

Contents

1 Introduction

In this set of notes the State Transition Matrix for the N-body problem is developed. This matrix is the derivative of the flow generated by the N-body vector field with respect to initial conditions. It, and it's inverse can be used in concert with the Newton Method in order to find/design orbits which begin at specified states and end in some specific configuration, such as a periodic orbit.

We develop the differential equation which the State Transition Matrix satisfies, as well as the numerical methods necessary to solve it. The practical application of the matrix is discussed at length, especially it's connection with

Newton Methods, which can be used to solve all kinds of boundary value problems. We then give several worked examples illustrating the use of these methods, including a method which allows one to compute Choreography orbits.

2 State Transition Matrix for the N Body

A state of the N bodies is a vector $\mathbf{x} \in \mathbb{R}^{6N}$ where

$$\mathbf{x} = \begin{bmatrix} \mathbf{r_1} \\ \mathbf{r_2} \\ \vdots \\ \mathbf{r_N} \\ \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_N} \end{bmatrix}$$

and each $\mathbf{r_1}, \ldots, \mathbf{v_N} \in \mathbb{R}^3$.

The equations of motion for the N body system can be written as

$$\dot{\mathbf{x}} = f(\mathbf{x})$$

with $f : \mathbb{R}^{6N} \to \mathbb{R}^{6N}$ given by

$$f(\mathbf{x}) = \begin{bmatrix} \mathbf{v_1} \\ \mathbf{v_2} \\ \vdots \\ \mathbf{v_N} \\ g_1(\mathbf{r_1}, \ldots, \mathbf{r_N}) \\ g_2(\mathbf{r_1}, \ldots, \mathbf{r_N}) \\ \vdots \\ g_N(\mathbf{r_1}, \ldots, \mathbf{r_N}) \end{bmatrix}$$

and

$$g_i(\mathbf{r_1}, \ldots, \mathbf{r_N}) = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{G\, m_j}{|r_{ij}|^3} \mathbf{r_{ij}}$$

The flow generated by this differential equation is a function $\phi : \mathbb{R} \times U \subset \mathbb{R} \times \mathbb{R}^{6N} \to \mathbb{R}^{6N}$ such that

1. $\phi(0, \mathbf{x}) = \mathbf{x}$ for all $\mathbf{x} \in U$.
2. $\phi(t + s, \mathbf{x}) = \phi(t, \phi(s, \mathbf{x}))$ for all $t, s \in \mathbb{R}$.

3. $\frac{d}{dt}\phi(t,\mathbf{x})|_{t=\tau} = f(\phi(\tau,\mathbf{x}))$ for all $x \in U$ and $\tau \in \mathbb{R}$

Here U is the set on which the N body problem has globally defined solutions, i.e. $\mathbb{R}^{6N}$ minus the set of initial conditions which lead to collisions or which escape to infinity in finite time.

Condition 1, and 2 express that the flow is a one parameter group acting on U, while 3 says simply that the curve $\phi(t,\mathbf{x_0})$ is the solution trajectory of the equation $\dot{\mathbf{x}} = f(\mathbf{x})$ with initial condition $\mathbf{x}_0 \in U$.

If f is a C^1 function then the flow ϕ is itself differentiable (Hersch and Smale), and the differential $D\phi$ is

$$D\phi = (D_t\,\phi, D_{\mathbf{x}}\phi) = \left(\frac{\partial\phi}{\partial t}, D_{\mathbf{x}}\phi\right)$$

Both of these partial derivatives can be computed along a given reference trajectory. If $\mathbf{x}_0 \in U$ is given then property 3 of the flow shows that $\frac{\partial\phi}{\partial t} = \frac{d\phi}{dt} = f(\phi(t,\mathbf{x_0})$ along the reference solution $\phi(t,\mathbf{x_0})$.

On the other hand if $\mathbf{x}_0 \in U$ and $t_0, \tau \in \mathbb{R}$ are fixed, let $\mathbf{x}^*(\tau) = \phi(\tau,\mathbf{x_0})$, and $\Phi(\tau,t_0) = D_{\mathbf{x_0}}\phi(\tau,\mathbf{x}_0)$, where t_0 is the time at which $\mathbf{x}^*(t_0) = \mathbf{x}_0$ Then one has by the chain rule and the equality of mixed partials, that

$$\begin{aligned}
\dot{\Phi} &= \frac{d}{dt}\Phi(t,t_0)|_{t=\tau} \\
&= \frac{d}{dt}\left[D_{x_0}\phi(t,\mathbf{x}_0)\right]_{t=\tau} \\
&= D_{x_0}\left[\frac{d}{dt}\phi(t,\mathbf{x}_0)|_{t=\tau}\right] \\
&= D_{x_0}f(\mathbf{x}^*(\tau)) \\
&= \left[D_{x^*(\tau)}f(\mathbf{x}*(\tau))\right]\circ\left[D_{x_0}\phi(\tau,\mathbf{x}_0)\right] \\
&= D_x f(\mathbf{x})|_{x=x*(\tau)}\circ\Phi(\tau,t_0)
\end{aligned}$$

Furthermore, evaluating the flow at $\tau = t_0$ gives

$$\Phi(t_0,t_0) = D_{x_0}\phi(t_0,\mathbf{x}_0) = D_{x_0}\mathbf{I} = \mathbf{I}$$

Then defining $F = D_x f(\mathbf{x})$ one has that Φ satisfies the linear nonautonomous matrix differential equation

$$\dot{\Phi}(t,t_0) = F\circ\Phi(t,t_0) \quad \Phi(t_0,t_0) = \mathbf{I}$$

F is called the state propagation matrix. We compute F for the N body problem;

$$F = D_x\ f(\mathbf{x})$$

$$= \quad D_x \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \\ g_1(\mathbf{r}_1, \ldots, \mathbf{r}_N) \\ \vdots \\ g_N(\mathbf{r}_1, \ldots, \mathbf{r}_N) \end{bmatrix}$$

$$= \begin{bmatrix} D_{r_1}\mathbf{v}_1 & \ldots & D_{r_N}\mathbf{v}_1 & D_{v_1}\mathbf{v}_1 & \ldots & D_{v_N}\mathbf{v}_1 \\ D_{r_1}\mathbf{v}_2 & \ldots & D_{r_N}\mathbf{v}_2 & D_{v_1}\mathbf{v}_2 & \ldots & D_{v_N}\mathbf{v}_2 \\ \vdots & \ddots & & & \ddots & \vdots \\ D_{r_1}\mathbf{v}_N & \ldots & D_{r_N}\mathbf{v}_N & D_{v_1}\mathbf{v}_N & \ldots & D_{v_N}\mathbf{v}_N \\ D_{r_1}g_1(\mathbf{r}) & \ldots & D_{r_N}g_1(\mathbf{r}) & D_{v_1}g_1(\mathbf{r}) & \ldots & D_{v_N}g_1(\mathbf{r}) \\ D_{r_1}g_2(\mathbf{r}) & \ldots & D_{r_N}g_2(\mathbf{r}) & D_{v_1}g_2(\mathbf{r}) & \ldots & D_{v_N}g_2(\mathbf{r}) \\ \vdots & \ddots & & & \ddots & \vdots \\ D_{r_1}g_N(\mathbf{r}) & \ldots & D_{r_N}g_N(\mathbf{r}) & D_{v_1}g_N(\mathbf{r}) & \ldots & D_{v_N}g_N(\mathbf{r}) \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ \mathbf{G} & \mathbf{0}_{3N\times 3N} \end{bmatrix}$$

Here $\mathbf{r} = \mathbf{r}_1, \ldots, \mathbf{r}_N$ so we see that $\mathbf{G}$ is a $3N \times 3N$ matrix function of $\mathbf{r}_1(t)$, $\ldots \mathbf{r}_N(t)$. The explicit form of $\mathbf{G}$ is the subject of the next section.

2.1 Derivation of G

From the previous computation it is clear that

$$\mathbf{G} = \begin{bmatrix} D_{r_1}g_1(\mathbf{r}) & \ldots & D_{r_N}g_1(\mathbf{r}) \\ D_{r_1}g_2(\mathbf{r}) & \ldots & D_{r_N}g_2(\mathbf{r}) \\ \vdots & \ddots & \vdots \\ D_{r_1}g_N(\mathbf{r}) & \ldots & D_{r_N}g_N(\mathbf{r}) \end{bmatrix} \tag{1}$$

The computation of these derivatives is aided by the following.

Lemma 1

$$D_{r_k}|\mathbf{r}_{ij}|^{-3} = \begin{cases} 3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}^T & k = i \\ -3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}^T & k = j \\ 0 & k \neq i, j \end{cases}$$

where all vectors are thought of as column vectors, hence a vector transposed is a row vector.

Proof First, by the chain rule

$$
\begin{aligned}
D_{r_k}|\mathbf{r}_{ij}|^{-3} &= D_{r_k}\left[(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij})^{1/2}\right]^{-3} && (2)\\
&= D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-3/2} && (3)\\
&= -\frac{3}{2}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) && (4)
\end{aligned}
$$

and the term$D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)$ depends on the values if i, j and k. Now, i and j will never be equal, so there are three cases; $k = i$ but $k \neq j$, $k = j$ but $k \neq i$, and $k \neq i, j$.

In case $k = i$

$$
\begin{aligned}
D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) &= D_{r_i}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)\\
&= D_{r_i}\left[(r_j^1 - r_i^1, r_j^2 - r_i^2, r_j^3 - r_i^3)\begin{bmatrix} r_j^1 - r_i^1 \\ r_j^2 - r_i^2 \\ r_j^3 - r_i^3 \end{bmatrix}\right]\\
&= D_{r_i}\left((r_j^1 - r_i^1)^2 + (r_j^2 - r_i^2)^2 + (r_j^3 - r_i^3)^2\right)\\
&= \begin{bmatrix} D_{r_i^1}\left((r_j^1 - r_i^1)^2 + (r_j^2 - r_i^2)^2 + (r_j^3 - r_i^3)^2\right) \\ D_{r_i^2}\left((r_j^1 - r_i^1)^2 + (r_j^2 - r_i^2)^2 + (r_j^3 - r_i^3)^2\right) \\ D_{r_i^3}\left((r_j^1 - r_i^1)^2 + (r_j^2 - r_i^2)^2 + (r_j^3 - r_i^3)^2\right) \end{bmatrix}^T\\
&= \begin{bmatrix} D_{r_i^1}\left(r_j^1 - r_i^1\right)^2 \\ D_{r_i^2}\left(r_j^2 - r_i^2\right)^2 \\ D_{r_i^3}\left(r_j^3 - r_i^3\right)^2 \end{bmatrix}^T\\
&= \begin{bmatrix} 2\left(r_j^1 - r_i^1\right) D_{r_i^1}\left(r_j^1 - r_i^1\right) \\ 2\left(r_j^2 - r_i^2\right) D_{r_i^2}\left(r_j^2 - r_i^2\right) \\ 2\left(r_j^3 - r_i^3\right) D_{r_i^3}\left(r_j^3 - r_i^3\right) \end{bmatrix}^T\\
&= \begin{bmatrix} -2\left(r_j^1 - r_i^1\right) \\ -2\left(r_j^2 - r_i^2\right) \\ -2\left(r_j^3 - r_i^3\right) \end{bmatrix}^T\\
&= -2\mathbf{r}_{ij}^T
\end{aligned}
$$

Substituting this into (4) gives

$$
\begin{aligned}
D_{r_k}|\mathbf{r}_{ij}|^{-3} &= -\frac{3}{2}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} D_{r_i}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)\\
&= -\frac{3}{2}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} - 2\mathbf{r}_{ij}^T\\
&= 3\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2}\mathbf{r}_{ij}^T\\
&= 3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}^T
\end{aligned}
$$

Similarly, if $k = j$ then the first six steps of the computation are identical and one has

$$\begin{aligned} D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) &= D_{r_j}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) \\ &= \begin{bmatrix} 2\left(r_j^1 - r_i^1\right) D_{r_j^1}\left(r_j^1 - r_i^1\right) \\ 2\left(r_j^2 - r_i^2\right) D_{r_j^2}\left(r_j^2 - r_i^2\right) \\ 2\left(r_j^3 - r_i^3\right) D_{r_j^3}\left(r_j^3 - r_i^3\right) \end{bmatrix}^T \\ &= \begin{bmatrix} 2\left(r_j^1 - r_i^1\right) \\ 2\left(r_j^2 - r_i^2\right) \\ 2\left(r_j^3 - r_i^3\right) \end{bmatrix}^T \\ &= 2\,\mathbf{r}_{ij}^T \end{aligned}$$

Substituting the previous line into (4) we have

$$\begin{aligned} D_{r_k}|\mathbf{r}_{ij}|^{-3} &= -\frac{3}{2}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} D_{r_j}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) \\ &= -\frac{3}{2}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} 2\mathbf{r}_{ij}^T \\ &= -3\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right)^{-5/2} \mathbf{r}_{ij}^T \\ &= -3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}^T \end{aligned}$$

Finally if $k \neq i, j$ then $D_{r_k}\left(\mathbf{r}_{ij}^T\ \mathbf{r}_{ij}\right) = 0$ as $\mathbf{r}_{ij}$ is independent of $\mathbf{r}_k$. Then in this case (4) is zero. Putting the three cases together gives the lemma.

The lemma aids in the speedy computation of the elements of $\mathbf{G}$. As in the previous section let $\mathbf{r} = \mathbf{r}_1, \ldots, \mathbf{r}_N$. Then

$$D_{r_k} g_i(\mathbf{r}) = D_{r_k} G \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{m_j}{|r_{ij}|^3}\mathbf{r}_{ik} \tag{5}$$

$$= G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \left(D_{r_k}|r_{ij}|^{-3}\mathbf{r}_{ik}\right) \tag{6}$$

(Here G is the gravitational constant). I this expression we have to deal with the terms $D_{r_k}|r_{ij}|^{-3}\mathbf{r_{ij}}$ in the indexed sum. Examining (1) shows that the two possible cases $k = i$ and $k \neq i$ both occur.

If $k \neq i$ then $k = j$ for exactly one of the summands, so that all but one of the terms is zero and we have

$$G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \left(D_{r_k} |r_{ij}|^{-3} \mathbf{r}_{ij} \right) = G\, m_k\, D_{r_k} |r_{ik}|^{-3} \mathbf{r}_{ik}$$

$$\begin{aligned} &= G\, m_k \left[\left(D_{r_k} |\mathbf{r}_{ik}|^{-3} \right)^T \mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3} D_{r_k} \mathbf{r}_{ik} \right] \\ &= G\, m_k \left[\left(-3 \frac{1}{|\mathbf{r}_{ik}|^5} \mathbf{r}_{ik}^T \right)^T \mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3} D_{r_k} (\mathbf{r}_k - \mathbf{r}_i) \right] \\ &= G\, m_k \left[-3 \frac{1}{|\mathbf{r}_{ik}|^5} \mathbf{r}_{ik} \mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3} \mathbf{I}_{3\times 3} \right] \end{aligned}$$

where we have used both the lemma and a vector product rule for derivatives, and it should be recalled that $\mathbf{r}_{ik}\mathbf{r}_{ik}^T$ is a column vector times a row vector. Explicitly, it is the 3×3 matrix

$$\mathbf{r}_{ik}\mathbf{r}_{ik}^T = (\mathbf{r}_k - \mathbf{r}_i)(\mathbf{r}_k - \mathbf{r}_i)^T$$

$$\begin{aligned} &= \begin{bmatrix} r_k^1 - r_i^1 \\ r_k^2 - r_i^2 \\ r_k^3 - r_i^3 \end{bmatrix} \quad (r_k^1 - r_i^1, r_k^2 - r_i^2, r_k^3 - r_i^3) \\ &= \begin{pmatrix} (r_k^1 - r_i^1) * (r_k^1 - r_i^1) & (r_k^2 - r_i^2) * (r_k^1 - r_i^1) & (r_k^3 - r_i^3) * (r_k^1 - r_i^1) \\ (r_k^1 - r_i^1) * (r_k^2 - r_i^2) & (r_k^2 - r_i^2) * (r_k^2 - r_i^2) & (r_k^3 - r_i^3) * (r_k^2 - r_i^2) \\ (r_k^1 - r_i^1) * (r_k^3 - r_i^3) & (r_k^2 - r_i^2) * (r_k^3 - r_i^3) & (r_k^3 - r_i^3) * (r_k^3 - r_i^3) \end{pmatrix} \end{aligned}$$

On the other hand if $k = i$ then none of the summands vanish as each depends on i, and we have

$$G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \left(D_{r_k} |r_{ij}|^{-3} \mathbf{r}_{ij} \right) = G \sum_{\substack{j=1 \\ j \neq i}}^{N} m_j \left(D_{r_i} |r_{ij}|^{-3} \mathbf{r}_{ij} \right)$$

$$
\begin{aligned}
&= G \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \left[\left(D_{r_i}|\mathbf{r}_{ij}|^{-3}\right)^T \mathbf{r}_{ij}^T + \frac{1}{|\mathbf{r}_{ij}|^3} D_{r_i}\mathbf{r}_{ij} \right] \\
&= G \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \left[\left(3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}^T\right)^T \mathbf{r}_{ij}^T + \frac{1}{|\mathbf{r}_{ij}|^3} D_{r_j}(\mathbf{r}_j - \mathbf{r}_i) \right] \\
&= G \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \left[3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}\mathbf{r}_{ij}^T - \frac{1}{|\mathbf{r}_{ij}|^3}\mathbf{I}_{3\times 3} \right]
\end{aligned}
$$

Then the 3×3 submatrices of $\mathbf{G}$ are

$$
(\mathbf{G})_{ik} = \begin{cases} G \sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \left[3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}\mathbf{r}_{ij}^T - \frac{1}{|\mathbf{r}_{ij}|^3}\mathbf{I}_{3\times 3} \right] & k = i \\ G\, m_k \left[-3\frac{1}{|\mathbf{r}_{ik}|^5}\mathbf{r}_{ik}\mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3}\mathbf{I}_{3\times 3} \right] & k \neq i \end{cases} \tag{7}
$$

where i and k each range from 1 to N. This gives $\mathbf{G}$ in a form that is easy to code.

2.2 Equal Masses and Symplectic Transition Matrices

The matrix $\Phi(t, t_0)$ is the derivative of the flow $\phi(t, \mathbf{x})$. Then, locally, it is the best linear approximation of the flow. So, qualitatively if $|\mathbf{x}_0 - \mathbf{x}|$ is small enough we can expect $|\Phi(t, t_0)\mathbf{x} - \phi(t, \mathbf{x})|$ to be small. In fact

$$
\dot{\mathbf{y}} = \Phi(t, t_0)\mathbf{y} \tag{8}
$$

is the linearized vector field near a reference solution of the fully nonlinear problem, and for small $\mathbf{y}_0 = \mathbf{x}_0 - \mathbf{x}$ the solution trajectory $y(t)$ of (8) with $\mathbf{y}(t_0) = \mathbf{y}_0$ is a good approximation at each time to the difference $\phi(t, \mathbf{x}_0) - \phi(t, \mathbf{x})$. In fact the degree to which this approximation is valid is a good measure of sensitivity of $\phi(t, \mathbf{x}_0)$ to initial conditions. For this reason (8) is called the variational equation.

A matrix $\mathbf{A}$ is said to be symplectic with respect to a second matrix $\mathbf{B}$ if and only if $\mathbf{A}^T\mathbf{B}\mathbf{A} = \mathbf{B}$. It is shown in [Meyer and Hall] that an even dimensional matrix A is symplectic with respect to $\mathbf{J}$, where

$$
\mathbf{J} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix}
$$

if and only if the system

$$\dot{\mathbf{z}} = \mathbf{A}\mathbf{z}$$

is a linear Hamiltonian dynamical system, i.e. if there is a quadratic form $H : \mathbb{R}^{2N} \to \mathbb{R}^{2N}$ so that

$$\dot{\mathbf{z}} = \mathbf{J}D_{\mathbf{z}}H$$

Therefore system (8) is a linear Hamiltonian system if and only if $\Phi(t, t_0)$ is symplectic (with respect to $\mathbf{J}$) in which case the linear flow preserves phase space volume. (Note that the phase space of ϕ is even dimensional as ϕ is generated by a Hamiltonian vector field. Then Φ is an even dimensional matrix). In this case $\Phi(t, t_0)$ is particularly easy to invert;

$$[\Phi(t, t_0)]^{-1} = -\mathbf{J}\Phi(t, t_0)\mathbf{J}$$

The inverse of Φ is the best linear approximation of ϕ^{-1} by the inverse function theorem. Then it is desirable to know when Φ is symplectic, if only to have the above formula for the inverse. We turn to necessary and sufficient conditions for this now.

First, Φ is symplectic with respect to $\mathbf{J}$ if and only if $\Phi^T\mathbf{J}\Phi = \mathbf{J}$. Differentiating this with respect to time gives

$$\begin{aligned}
\frac{d}{dt}[\Phi^T\mathbf{J}\Phi] &= \dot{\Phi}^T\mathbf{J}\Phi + \Phi^T\mathbf{J}\dot{\Phi} \\
&= [F\Phi]^T\mathbf{J}\Phi + \Phi^T\mathbf{J}F\Phi \\
&= \Phi^T F^T\mathbf{J}\Phi + \Phi^T\mathbf{J}F\Phi \\
&= \Phi^T[F^T\mathbf{J} + \mathbf{J}F]\Phi \\
&= 0
\end{aligned}$$

and since $\Phi(t_0, t_0) = \mathbf{I} \neq 0$ we have that $\Phi(t, t_0) \neq 0$ for all t (as we are assuming that the dynamical system is globally defined). Then Φ is symplectic if and only if

$$F^T\mathbf{J} + \mathbf{J}F = 0$$

or

$$F^T\mathbf{J} = -\mathbf{J}F$$

Using that for the N body problem

$$F = D_x f = \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ \mathbf{G} & \mathbf{0}_{3N\times 3N} \end{bmatrix}$$

we compute

$$
\begin{aligned}
F^T\mathbf{J} &= \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ \mathbf{G} & \mathbf{0}_{3N\times 3N} \end{bmatrix}^T \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ -\mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{G}^T \\ \mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ -\mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{I}_{3N\times 3N}\mathbf{G}^T & \mathbf{0}_{3N\times 3N} \\ \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N}\mathbf{I}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{G}^T & \mathbf{0}_{3N\times 3N} \\ \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \end{bmatrix}
\end{aligned}
$$

and

$$
\begin{aligned}
-\mathbf{J}F &= -\begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ -\mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \\ \mathbf{G} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{0}_{3N\times 3N} & -\mathbf{I}_{3N\times 3N} \\ \mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3N\times 3N} & \mathbf{G} \\ \mathbf{I}_{3N\times 3N} & \mathbf{0}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{I}_{3N\times 3N}\mathbf{G} & \mathbf{0}_{3N\times 3N} \\ \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N}\mathbf{I}_{3N\times 3N} \end{bmatrix} \\
&= \begin{bmatrix} -\mathbf{G} & \mathbf{0}_{3N\times 3N} \\ \mathbf{0}_{3N\times 3N} & \mathbf{I}_{3N\times 3N} \end{bmatrix}
\end{aligned}
$$

Then we have that $F^T\mathbf{J} = -\mathbf{J}F$ and hence Φ is symplectic if and only if $\mathbf{G} = \mathbf{G}^T$, so if $\mathbf{G}$ is symmetric.
Recall from the previous section that

$$
\mathbf{G} = \begin{bmatrix} D_{r_1}g_1(\mathbf{r}) & \dots & D_{r_N}g_1(\mathbf{r}) \\ D_{r_1}g_2(\mathbf{r}) & \dots & D_{r_N}g_2(\mathbf{r}) \\ \vdots & \ddots & \vdots \\ D_{r_1}g_N(\mathbf{r}) & \dots & D_{r_N}g_N(\mathbf{r}) \end{bmatrix}
$$

with

$$
D_{r_k}g_i(\mathbf{r}) = \begin{cases} G\sum_{\substack{j=1 \\ j\neq i}}^{N} m_j \left[3\frac{1}{|\mathbf{r}_{ij}|^5}\mathbf{r}_{ij}\mathbf{r}_{ij}^T - \frac{1}{|\mathbf{r}_{ij}|^3}\mathbf{I}_{3\times 3}\right] & k = i \\ G\,m_k\left[-3\frac{1}{|\mathbf{r}_{ik}|^5}\mathbf{r}_{ik}\mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3}\mathbf{I}_{3\times 3}\right] & k \neq i \end{cases}
$$

The question of when this is symmetric will be aided by the following.

Lemma 2

$$
D_{r_i}g_k = \frac{m_i}{m_k}D_{r_k}g_i
$$

Proof If $i = k$ then the statement is trivial as

$$D_{r_i} g_k = D_{r_i} g_i = \frac{m_i}{m_i} D_{r_i} g_i = \frac{m_i}{m_k} D_{r_k} g_i$$

Suppose then that $i \neq k$. Then since $|r_{ik}| = |r_{ki}|$ and $\mathbf{r}_{ik}\mathbf{r}_{ik}^T = \mathbf{r}_{ki}\mathbf{r}_{ki}^T$ we have that

$$\begin{aligned}
D_{r_k} g_i(\mathbf{r}) &= G\, m_k \left[-3 \frac{1}{|\mathbf{r}_{ik}|^5} \mathbf{r}_{ik}\mathbf{r}_{ik}^T + \frac{1}{|\mathbf{r}_{ik}|^3} \mathbf{I}_{3\times 3} \right] \\
&= G\, m_k \left[-3 \frac{1}{|\mathbf{r}_{ki}|^5} \mathbf{r}_{ki}\mathbf{r}_{ki}^T + \frac{1}{|\mathbf{r}_{ki}|^3} \mathbf{I}_{3\times 3} \right] \\
&= G m_k \frac{m_i}{m_i} \left[-3 \frac{1}{|\mathbf{r}_{ki}|^5} \mathbf{r}_{ki}\mathbf{r}_{ki}^T + \frac{1}{|\mathbf{r}_{ki}|^3} \mathbf{I}_{3\times 3} \right] \\
&= \frac{m_k}{m_i} G m_i \left[-3 \frac{1}{|\mathbf{r}_{ki}|^5} \mathbf{r}_{ki}\mathbf{r}_{ki}^T + \frac{1}{|\mathbf{r}_{ki}|^3} \mathbf{I}_{3\times 3} \right] \\
&= \frac{m_k}{m_i} D_{r_i} g_k(\mathbf{r})
\end{aligned}$$

Then, in fact

$$\begin{aligned}
\mathbf{G} &= \begin{bmatrix}
D_{r_1} g_1 & D_{r_2} g_1 & \dots & D_{r_{N-1}} g_1 & D_{r_N} g_1 \\
D_{r_1} g_2 & D_{r_2} g_2 & \dots & D_{r_{N-1}} g_2 & D_{r_N} g_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
D_{r_1} g_{N-1} & D_{r_2} g_{N-1} & \dots & D_{r_{N-1}} g_{N-1} & D_{r_N} g_{N-1} \\
D_{r_1} g_N & D_{r_2} g_N & \dots & D_{r_{N-1}} g_N & D_{r_N} g_N
\end{bmatrix} \\
&= \begin{bmatrix}
D_{r_1} g_1 & D_{r_2} g_1 & \dots & D_{r_{N-1}} g_1 & D_{r_N} g_1 \\
\frac{m_1}{m_2} D_{r_2} g_1 & D_{r_2} g_2 & \dots & D_{r_{N-1}} g_2 & D_{r_N} g_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\frac{m_1}{m_{N-1}} D_{r_{N-1}} g_1 & \frac{m_2}{m_{N-1}} D_{r_{N-1}} g_2 & \dots & D_{r_{N-1}} g_{N-1} & D_{r_N} g_{N-1} \\
\frac{m_1}{m_N} D_{r_N} g_1 & \frac{m_2}{m_N} D_{r_N} g_2 & \dots & \frac{m_{N-1}}{m_N} D_{r_N} g_{N-1} & D_{r_N} g_N
\end{bmatrix}
\end{aligned}$$

Then clearly $\mathbf{G}$ is symmetric if and only if $m_1 = m_2 = \ldots = m_N$.

Note however that even if $\mathbf{G}$ is not symmetric, the lemma still implies that the lower triangular entries differ from the upper triangular entries only by constant multiples. If this information is used in the computer implementation the expense of computing $\mathbf{G}$ is greatly reduced. (Instead of computing N^2 3×3 matrices, only $N + (N-1) + \ldots + 3 + 2 + 1 = (N^2 - N)/2$ matrices are needed. Even these contain many repeated terms so that the number of computations can be further reduced. See code in the appendix).

3 Computer Implementation of the First Variation Equation and Computation of the State Transition Matrix

The state transition matrix can be computed numerically by integrating the differential equation

$$\dot{\Phi}(t,t_0) = F \circ \Phi(t,t_0) \quad \Phi(t_0,t_0) = \mathbf{I}$$

This is a nonautonomous linear first order $6N \times 6N$ matrix differential equation. Since most numerical integration routines prefer input in the form of a system of first order scalar equations it's convenient to write the system in the form $\dot{\mathbf{x}} = f(t,\mathbf{x})$.

Begin by letting $K = 6N$, writing

$$\Phi(t,t_0) = \begin{bmatrix} a_{11} & \cdots & a_{1K} \\ \vdots & \ddots & \vdots \\ a_{K1} & \cdots & a_{KK} \end{bmatrix}$$

and defining a state vector $\mathbf{x} \in \mathbb{R}^{36N^2}$

$$\mathbf{x} = \begin{bmatrix} a_{11} \\ \vdots \\ a_{1K} \\ \vdots \\ a_{K1} \\ \vdots \\ a_{KK} \end{bmatrix}$$

With this notation consider the expression $F \circ \Phi(t,t_0)$. This becomes

$$\begin{aligned} F \circ \Phi(t,t_0) &= D_{\mathbf{x}} f(\mathbf{x}) \circ \Phi(t,t_0) \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \Phi(t,t_0) \\ &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1K} \\ \vdots & \ddots & \vdots \\ a_{K1} & \cdots & a_{KK} \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} a_{(K/2+1)1} & \cdots & a_{(K/2+1)K} \\ a_{(K/2+2)\,1} & \cdots & a_{(K/2+2)\,K} \\ \vdots & \ddots & \cdots \\ a_{K1} & \cdots & a_{KK} \\ \mathbf{G}\begin{bmatrix} a_{11} \\ \vdots \\ a_{(K/2)1} \end{bmatrix} & \cdots & \mathbf{G}\begin{bmatrix} a_{1K} \\ \vdots \\ a_{(K/2)K} \end{bmatrix} \end{bmatrix}$$

where each of the column vectors in the bottom half of the matrix can be computed using the know expressions for $\mathbf{G}$. Denote the i^{th} such column by $\mathbf{G}_i$. Note that since $\mathbf{G}$ is a $K/2 \times K/2$ matrix, the vector $\mathbf{G}_i$ has $K/2$ components. Then

$$\begin{aligned} \mathbf{G}_i &= \mathbf{G}\begin{bmatrix} a_{1i} \\ \vdots \\ a_{(K/2)i} \end{bmatrix} \\ &= \begin{bmatrix} D_{r_1}g_1(\mathbf{r}) & \ldots & D_{r_N}g_1(\mathbf{r}) \\ D_{r_1}g_2(\mathbf{r}) & \ldots & D_{r_N}g_2(\mathbf{r}) \\ \vdots & \ddots & \vdots \\ D_{r_1}g_N(\mathbf{r}) & \ldots & D_{r_N}g_N(\mathbf{r}) \end{bmatrix}\begin{bmatrix} a_{1i} \\ \vdots \\ a_{(K/2)i} \end{bmatrix} \\ &= \begin{bmatrix} D_{r_1}g_1(\mathbf{r})\,a_{1i} + & \ldots & + D_{r_N}g_1(\mathbf{r})\,a_{(K/2)i} \\ \vdots & \ddots & \vdots \\ D_{r_1}g_N(\mathbf{r})\,a_{1i} + & \ldots & + D_{r_N}g_N(\mathbf{r})\,a_{(K/2)i} \end{bmatrix} \end{aligned}$$

where it is recalled that expressions for the $D_{r_k}g_i$ terms are known explicitly. Then the scalar system of ODEs for the state transition matrix is

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{a}_{11} \\ \vdots \\ \dot{a}_{1K} \\ \vdots \\ \dot{a}_{(K/2)1} \\ \vdots \\ \dot{a}_{(K/2)K} \\ \dot{a}_{(K/2+1)\,1} \\ \vdots \\ \dot{a}_{(K/2+1)\,K} \\ \vdots \\ \dot{a}_{K1} \\ \vdots \\ \dot{a}_{KK} \end{bmatrix}_{36N^2} = \begin{bmatrix} a_{K/2\,1} \\ \vdots \\ a_{K/2\,K} \\ \vdots \\ a_{K\,1} \\ \vdots \\ a_{K\,K} \\ (\mathbf{G}_1)_1 \\ \vdots \\ (\mathbf{G}_1)_{K/2} \\ \vdots \\ (\mathbf{G}_1)_{K/2} \\ \vdots \\ (\mathbf{G}_{K/2})_{K/2} \end{bmatrix}_{36N^2}$$

where

$$(\mathbf{G}_i)_j = D_{r_1} g_j(\mathbf{r})\, a_{1i} \ + \ldots + \ D_{r_N} g_j(\mathbf{r})\, a_{(K/2)i}$$

is the j^{th} component of the i^{th} column vector $\mathbf{G}_i$, $i = 1, 2, \ldots, K = 6N$, $j = 1, 2, \ldots, K/2 = 3N$, and the initial condition is

$$\mathbf{x}_0 = \mathbf{x}(0) = \begin{bmatrix} a_{11}(0) \\ a_{12}(0) \\ \vdots \\ a_{1K}(0) \\ a_{21}(0) \\ a_{22}(0) \\ \vdots \\ a_{2K}(0) \\ \vdots \\ a_{K1}(0) \\ \vdots \\ a_{KK}(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

While this system is scalar and first order it is not autonomous as the functions

$$g_j(\mathbf{r}) = g_j(\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t))$$

clearly depend on time. The usual way to obtain an autonomous vector field from this system would be to append the scalar equation $\dot{t} = 1$ and integrate the resulting $36N^2 + 1$ equations.

However the present case is complicated by the fact that the functions $\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t)$ are not explicitly known, but are themselves components of a solution trajectory of the N body problem with initial conditions $\mathbf{r}_1(0), \ldots, \mathbf{r}_N(0)$ and $\mathbf{v}_1(0), \ldots, \mathbf{v}_N(0)$. One could numerically compute $\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t)$ and, using the resulting data, treat these as known function. If an adaptive step size integration routine were then used to integrate the state transition equations it would be necessary to interpolate this data repeatedly.

$\mathbf{r}_1(t), \ldots, \mathbf{r}_N(t)$ are in fact the position components of the reference solution mentioned above. The differential equations for the state transition matrix depend critically on this reference solution. Then another option is to incorporate this reference solution into an expanded system of equations.

To this end the N body equations of motion are appended to the state transition matrix equations. One obtains a new state vector $\mathbf{y} \in \mathbb{R}^{36N^2+6N}$ and the system of equations

$$\dot{y} = \begin{bmatrix} \dot{a}_{11} \\ \vdots \\ \dot{a}_{KK} \\ \dot{\mathbf{r}}_1 \\ \vdots \\ \dot{\mathbf{r}}_N \\ \dot{\mathbf{v}}_1 \\ \vdots \\ \dot{\mathbf{v}}_N \end{bmatrix}_{36N^2+6N} = \begin{bmatrix} a_{K/2\,1} \\ \vdots \\ (\mathbf{G}_{K/2})_{K/2} \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_N \\ g_1 \\ \vdots \\ g_3 \end{bmatrix}_{36N^2+N6}$$

with initial conditions

$$\mathbf{y}_0 = \mathbf{y}(0) = \begin{bmatrix} a_{11}(0) \\ \vdots \\ a_{KK}(0) \\ (\mathbf{r}_1)_0 \\ \vdots \\ (\mathbf{v}_N)_0 \end{bmatrix} = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \mathbf{r}_1(0) \\ \vdots \\ \mathbf{v}_N(0) \end{bmatrix}$$

This is an autonomous system of first order differential equations which any standard integration routine can compute. An example of such a program is given in the appendix.

4 Method of Differential Corrections

The method of differential corrections is a powerfull application of Newton's Method that employes the State Transition Matrix to solve various kinds of boundary value problems. The first subsection contains some motivation for the method, as well as it's specific form. The following section gives an example of it's use in solving a specific boundary value or targeting problem.

4.1 Newton's Method and the Differential of the Flow

Fix $\mathbf{x}_0^* \in U$ and let $\mathbf{x}^*(t) = \phi(t, \mathbf{x}_0^*)$ be the orbit which passes through $\mathbf{x}_0^*$ at time 0. $\mathbf{x}^*$ will be called a reference solution. Suppose that at a fixed time $t = \tau$ the reference trajectory passes near a target point $\mathbf{x}_f \in U$. Specifically, that for some $\epsilon > 0$, $|\mathbf{x}_f - \mathbf{x}^*(\tau)| < \epsilon$.

There are many practical situations where we may want to solve the following targeting problem; find an initial condition $\mathbf{x}_0$ near $\mathbf{x}_0^*$ and time T near τ so that $\phi(T, \mathbf{x}_0) = \mathbf{x}_f$, or so that at least $|\phi(T, \mathbf{x}_0) - \mathbf{x}_f| < \delta$ for some prescribed tolerance $\delta > 0$.

A first intuitive step in this direction can be made by utilizing the fact that the state transition matrix $\Phi(t, t_0)$ is the differential of the flow $\phi(t, \mathbf{x})$ with respect to the variable $\mathbf{x}$. Assume additionally that the state transition matrix is invertable on and near the reference solution, or at least in the open ball $B(\mathbf{x}^*(\tau), \epsilon)$. (It is enough to assume $\Phi(t, t_0)$ is invertable only at $\mathbf{x}^*(\tau)$ if ϵ is small enough).

Then, since $\phi(t, \cdot)$ is differentiable, by the inverse function theorem we have that $\phi^{-1}(t, \cdot)$ is defined and differentiable on $B(\mathbf{x}^*(\tau), \epsilon)$, and that

$$D_{\mathbf{y}}\phi(t, \mathbf{y})^{-1} = [D_x\phi(t, \mathbf{x})]^{-1}$$

where $\mathbf{y} \in B(\mathbf{x}^*(\tau), \epsilon)$ and $\mathbf{x} = \phi^{-1}(t, \mathbf{y})$. Define the vector $\delta\mathbf{x}_f \in \mathbb{R}^{6N}$ by

$$\delta\mathbf{x}_f = \mathbf{x}_f - \mathbf{x}^*(\tau)$$

We have that $\delta\mathbf{x}_f \in B(0, \epsilon)$, as $|\mathbf{x}_f - \mathbf{x}^*(\tau)| < \epsilon$.

$D_x\phi^{-1}(t, \mathbf{x}^*(\tau))$ is the best linear approximation of ϕ^{-1} at $\mathbf{x}^*(\tau)$ and ϵ is small. Then

$$\begin{aligned} \phi^{-1}(t, \mathbf{x}^*(\tau) + \delta\mathbf{x}_f) &\approx \mathbf{x}_0^* + [D_x\phi^{-1}(t, \mathbf{x}^*(\tau))]\delta\mathbf{x}_f & (9)\\ &= \mathbf{x}_0^* + [D_x\phi(t, \mathbf{x}^*(\tau))]^{-1}\delta\mathbf{x}_f & (10)\\ &= \mathbf{x}_0^* + [\Phi(t, t_0)]^{-1}\delta\mathbf{x}_f & (11) \end{aligned}$$

Define $\delta\mathbf{x}_0 = [\Phi(t, t_0)]^{-1}\delta\mathbf{x}_f$ and note that

$$\begin{aligned} \phi^{-1}(t, \mathbf{x}^*(\tau) + \delta\mathbf{x}_f) &= \phi^{-1}(t, \mathbf{x}_f) & (12)\\ &= \mathbf{x}_0 & (13) \end{aligned}$$

Where (12) and (13) are equal by definition of the targeting problem; $\mathbf{x}_0$ is an initial state such that $\phi(\tau, \mathbf{x}_0) = \mathbf{x}_f$. Applying $\phi(\tau, \cdot)$ to (12) and (13) gives exactly this.

Combining (9), (11), (13) and the definition of $\delta\mathbf{x}_0$ gives

$$\mathbf{x}_0 \approx \mathbf{x}_0^* + \delta\mathbf{x}_0$$

Now if we let $\mathbf{x}_1^* = \mathbf{x}_0^* + \delta\mathbf{x}_0$ then we expect that

$$|\phi(\tau, \mathbf{x}_1^*) - \mathbf{x}_f| < |\phi(\tau, \mathbf{x}_0^*) - \mathbf{x}_f|$$

This inequality has not been proved. But this is the intuition; that $\mathbf{x}_1^*$ is a good approximation of $\mathbf{x}_0$, so for small enough ϵ it should be a better approximation than the initial guess $\mathbf{x}_0^*$. By iterating this argument we might hope to produce a sequence of better and better approximations to the target $\mathbf{x}_0$, and might hope even that the sequence converges.

Newton's Method is a powerfull refinement of these ideas. Briefly, suppose we want to solve the equation $f(\mathbf{x}) = 0$ with f a differentiable map between Banach Spaces. Given an initial estimate $\mathbf{x}_0^*$ of the solution $\mathbf{x}_0$ with $|f(\mathbf{x}_\mathbf{0}^*)| < \epsilon$ small, the sequence

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [D_x f(\mathbf{x}_n)]^{-1} f(\mathbf{x}_n) \qquad \mathbf{x}_0 = \mathbf{x}_0^*$$

converges to a solution $\mathbf{x}_0$. A theorem of Kantorovich gives sufficient conditions under which quadratic convergence is guaranteed as well estimates of the size of the set on which the method converges (how good the initial guess must be). Good references are [Arbogast and Bonna] and [Cheney].

Here the point is that the targeting problem can be reformulated as follows. Define $f : U \to \mathbb{R}^{6N}$ by

$$f(\mathbf{x}) = \phi(\tau, \mathbf{x}) - \mathbf{x}_f$$

so that solving the targeting problem is equivalent to finding $\mathbf{x}_0$ so that

$$f(\mathbf{x}_0) = 0$$

The Newton method is applicable if $[D_x f]^{-1}$ exists (as a bounded linear operator). But assuming again that the conditions of the inverse function theorem are satisfied this is

$$\begin{aligned} [D_x f]^{-1} &= [D_x \phi(\tau, \mathbf{x}) - \mathbf{x}_f]^{-1} \\ &= [D_x \phi(\tau, \mathbf{x}) - 0]^{-1} \\ &= [\Phi(t, t_0)]^{-1} \end{aligned}$$

and a Newton algorithm can be implemented as

$$\begin{aligned} \mathbf{x}_{n+1} &= \mathbf{x}_n - [\Phi(\tau, t_0)|_{\mathbf{x}_n}]^{-1} f(\mathbf{x}_n) \\ &= \mathbf{x}_n - [\Phi(\tau, t_0)|_{\mathbf{x}_n}]^{-1} [\phi(\tau, \mathbf{x}_n) - \mathbf{x}_f] \quad \mathbf{x}_1 = \mathbf{x}_0^* \end{aligned}$$

The procedure described here is sometimes called the method of differential corrections or differential correction procedure.

4.2 Example; Targeting an Equilateral Triangle Configuration

In this example three bodies of unit mass in the plane have initial positions and velocities

$$\mathbf{x}_0^* = \begin{bmatrix} \mathbf{r}_0^* \\ \mathbf{v}_0^* \end{bmatrix} = \begin{bmatrix} r_1^1 \\ r_1^2 \\ r_2^1 \\ r_2^2 \\ r_3^1 \\ r_3^2 \\ v_1^1 \\ v_1^2 \\ v_2^1 \\ v_2^2 \\ v_3^1 \\ v_3^2 \end{bmatrix} = \begin{bmatrix} 5.434 \\ 5.290 \\ 4.531 \\ 5.231 \\ 5.033 \\ 4.478 \\ -0.594 \\ -1.924 \\ -0.402 \\ -0.186 \\ -2.003 \\ -0.888 \end{bmatrix}$$

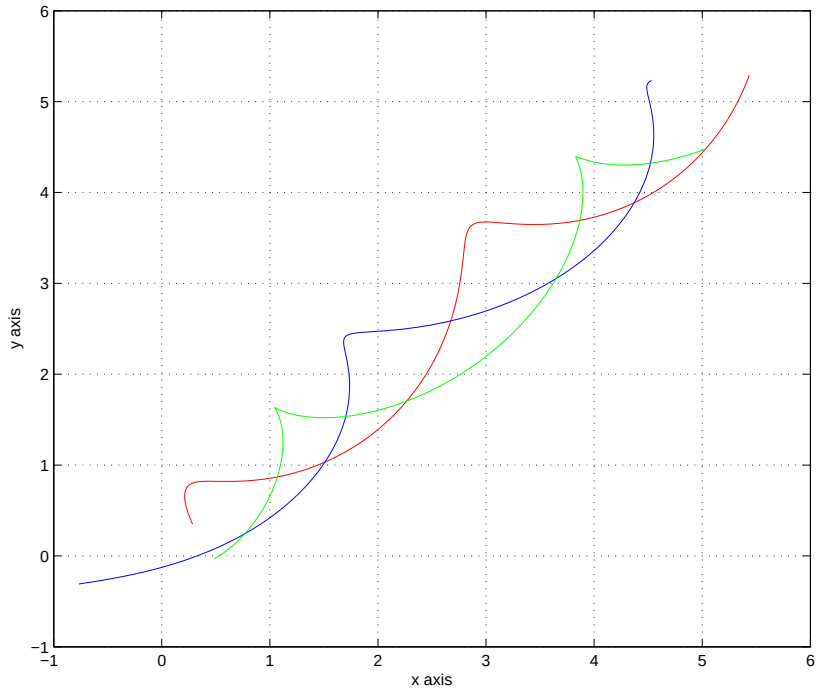


Figure 1: Initial Trajectory of Braid

which cause them to move in the braidlike orbit shown in figure (1) between time $t_0 = 0$ and $t_f = 5$. We compute $\Phi(t, 0)$ with this braided orbit as $\mathbf{x}^*(t)$.

Computing

$$[\Phi(t,0)]^{-1} + \mathbf{J}[\Phi(t,0)]^T\mathbf{J} \quad \text{and} \quad |\Phi(t,0)|$$

at 25 equally spaced time points t_i $i = 1, 2, \ldots, 25$ between $t = 0$ and $t = 10$ and then taking the norm of the result yields

$$|[\Phi(t_i,0)]^{-1} + \mathbf{J}[\Phi(t_i,0)]^T\mathbf{J}| = 10^4 \begin{bmatrix} 0.00000000000000 \\ 0.00000000000000 \\ 0.00000000000000 \\ 0.00000000000000 \\ 0.00000000000000 \\ 0.00000000000000 \\ 0.00000000000001 \\ 0.00000000000006 \\ 0.00000000000082 \\ 0.00000000000482 \\ 0.00000000011889 \\ 0.00000000047168 \\ 0.00000013910169 \\ 0.00000005198954 \\ 0.00000004077770 \\ 0.00000004886208 \\ 0.00000007512184 \\ 0.00000065826429 \\ 0.05350860773379 \\ 0.01195472846734 \\ 0.01669321943851 \\ 0.00943207572917 \\ 0.11736446468390 \\ 4.24183632901699 \\ 0.54824345987465 \end{bmatrix}$$

and

$$|\Phi(t_i,0)| \quad = \quad \begin{bmatrix} 1.00000000000000 \\ 0.99999999999998 \\ 0.99999999999999 \\ 0.99999999999993 \\ 0.99999999999997 \\ 0.99999999999980 \\ 1.00000000000137 \\ 0.99999999999811 \\ 1.00000000000181 \\ 0.99999999996121 \\ 1.00000000068733 \\ 0.99999999916063 \\ 0.99999996370961 \\ 0.99999998285752 \\ 0.99999997363241 \\ 0.99999996396527 \\ 0.99999996234473 \\ 1.00000007331777 \\ 0.99990474761275 \\ 0.99985938906079 \\ 0.99990047237883 \\ 0.99993582786588 \\ 0.99960621991467 \\ 1.00311403198487 \\ 1.00157520272770 \end{bmatrix} \tag{14}$$

which numerically verifies the properties

$$[\Phi(t,0)]^{-1} = -\mathbf{J}[\Phi(t,0)]^T\mathbf{J}$$

and

$$|\Phi(t,0)| = 1$$

for times well past $t = 5$. We note that the determinant property is preserved through the third decimal place through the experiment. Similarly if we single out the first twelve t_i then

$$|[\Phi(t_i,0)]^{-1} + \mathbf{J}[\Phi(t_i,0)]^T\mathbf{J}| \;=\; 10^{-5}\begin{bmatrix} 0.00000000114905 \\ 0.00000002360885 \\ 0.00000004617165 \\ 0.00000020291226 \\ 0.00000037912633 \\ 0.00000186135364 \\ 0.00000851617346 \\ 0.00005942789144 \\ 0.00082081707653 \\ 0.00481989605307 \\ 0.11889086929684 \\ 0.47167611124010 \end{bmatrix}$$

and the two methods of determining the inverse agree to more than five figures. Note that $t_{12} = 4.8$.

We are now ready to give an example of the use of these tools. Consider the following targeting problem. An orbit is sought which begins from the same initial positions, but whose positions at time $t_f = 5$ lie on the vertices of the equilateral triangle comprised of the points $(0.0, 0.577)$, $(-0.5, -0.288)$, and $(0.5, -0.288)$. We target

$$\mathbf{x}_f = \begin{bmatrix} r_1^1(t_f) \\ r_1^2(t_f) \\ r_2^1(t_f) \\ r_2^2(t_f) \\ r_3^1(t_f) \\ r_3^2(t_f) \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.577 \\ -0.5 \\ -0.288 \\ 0.5 \\ -0.288 \end{bmatrix}$$

with final velocities unconstrained.

To formulate the differential correction procedure we write

$$\begin{aligned} \phi(t, \mathbf{x}_\mathbf{0}^*) &= \phi(t, \mathbf{r}_\mathbf{0}^*, \mathbf{v}_\mathbf{0}^*) \\ &= \begin{bmatrix} \phi_r(t, \mathbf{r}_0^*, \mathbf{v}_0^*) \\ \phi_v(t, \mathbf{r}_0^*, \mathbf{v}_0^*) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{r}^*(t) \\ \mathbf{v}^*(t) \end{bmatrix} \end{aligned}$$

Holding $\mathbf{r}_0^*$ and t_f fixed define $f : \mathbb{R}^6 \to \mathbb{R}^6$ by

$$f(\mathbf{v}) = \phi_r(t_f, \mathbf{r}_0^*, \mathbf{v}) - \mathbf{r}_f$$

and note that this gives position at time t_f as a function of initial velocity, with initial position fixed. Further a $\mathbf{v}_0$ such that $f(\mathbf{v}_0) = 0$ solves the targeting

problem. However it it clear that we can have no hope of specifying the final velocities, (to do that we would need more freedoms, such as varying the initial positions).

The differentail correction procedure dictated by Newtons method is

$$\begin{aligned}
\mathbf{v}_{n+1} &= \mathbf{v}_n - [D_v f(\mathbf{v}_n)]^{-1} f(\mathbf{v}_n) \\
&= \mathbf{v}_n - [D_v(\phi_r(t_f, \mathbf{r}_0^*, \mathbf{v}) - \mathbf{r}_f)]^{-1}[\phi_r(t_f, \mathbf{r}_0^*, \mathbf{v}) - \mathbf{r}_f] \\
&= \mathbf{v}_n - [\Phi_{(1:6,7:12)}(t_f, 0)]^{-1}[\mathbf{r}^*_{\mathbf{v}_n}(t_f) - \mathbf{r}_f]
\end{aligned}$$

with $\mathbf{v}_1 = \mathbf{v}_0^*$. Here the notation $\Phi_{(1:6,7:12)}(t_f, 0)$ is the first 6 rows and the last 6 columns of $\Phi(t_f, 0)$, and $\mathbf{r}^*_{\mathbf{v}_n}(t_f)$ is the position at time t_f of trajectory with initial position $(\mathbf{r}_0^*, \mathbf{v}_n)$.

To see that the differential term is correct note that

$$\begin{aligned}
\Phi(t, t_0) &= D_x \phi(t, \mathbf{x}) && (15) \\
&= D_x \begin{bmatrix} \phi_r(t, \mathbf{x}) \\ \phi_v(t, \mathbf{x}) \end{bmatrix} && (16) \\
&= \begin{bmatrix} D_r\phi_r(t, \mathbf{r}, \mathbf{v}) & D_v\phi_r(t, \mathbf{r}, \mathbf{v}) \\ D_r\phi_v(t, \mathbf{r}, \mathbf{v}) & D_v\phi_v(t, \mathbf{r}, \mathbf{v}) \end{bmatrix} && (17)
\end{aligned}$$

Thus comparing the left hand side of (14) to (16) verifies the claim.

A MATLAB implementation of this procedure is given in the appendix. After 6 iterations we have

$$error = |f(\mathbf{v}_6)| = 2.3026e - 09$$

and we consider the sequence converged. The initial velocity $\mathbf{v}_0$ which will flow to the target state is

$$\mathbf{v}_0 = \begin{bmatrix} v_1^1(0) \\ v_1^2(0) \\ v_2^1(0) \\ v_2^2(0) \\ v_3^1(0) \\ v_3^2(0) \end{bmatrix} = \begin{bmatrix} -0.5946 \\ -1.9244 \\ -0.4022 \\ -0.1867 \\ -2.0032 \\ -0.8889 \end{bmatrix}$$

the two figures show the new trajectory and a close up of the triangle configuration of the masses at $t_f = 5$.

Finally the difference between the initial estimate $\mathbf{v}_0^*$ and the target velocity $\mathbf{v}_0$ is

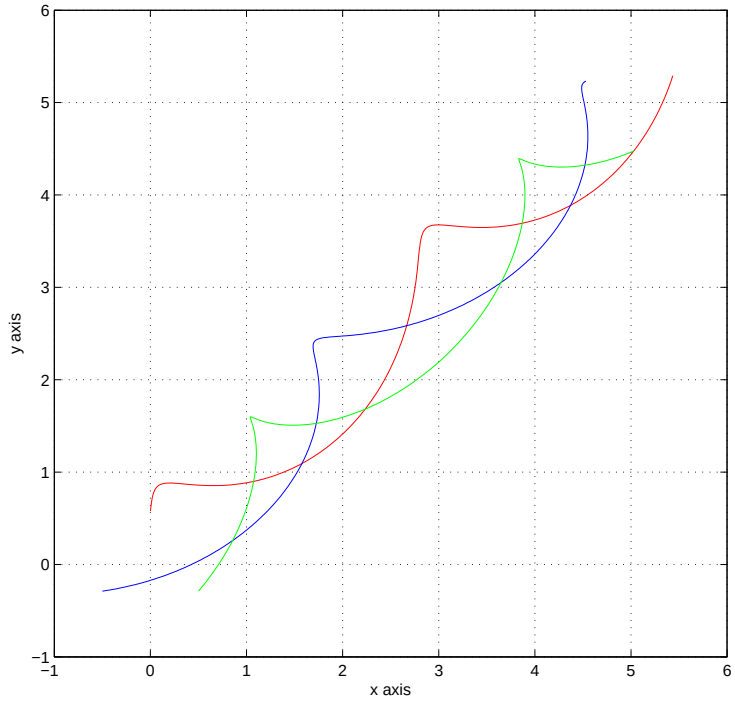


Figure 2: The Corrected Trajectory

$$\mathbf{v}_0 - \mathbf{v}_0^* \quad = \quad 10^{-3} \begin{bmatrix} 0.6065 \\ 0.3864 \\ 0.1548 \\ 0.7263 \\ 0.2387 \\ 0.8874 \end{bmatrix}$$

which shows this initial estimate was, in fact quite good.

5 A Choreography Orbit

A final example will demonstrate the flexibility of the differential correction procedure. The initial conditions

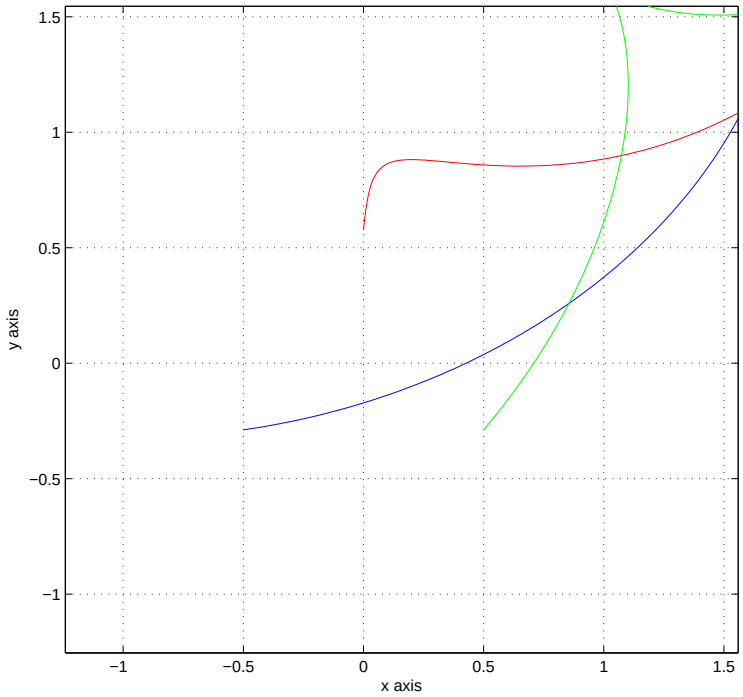


Figure 3: Equilateral Triangle Configuration

$$\begin{bmatrix} \mathbf{r}_0^* \\ \mathbf{v}_0^* \end{bmatrix} = \begin{bmatrix} r_1^1 \\ r_1^2 \\ r_2^1 \\ r_2^2 \\ r_3^1 \\ r_3^2 \\ v_1^1 \\ v_1^2 \\ v_2^1 \\ v_2^2 \\ v_3^1 \\ v_3^2 \end{bmatrix} = \begin{bmatrix} -0.7 \\ 0.36 \\ 1.1 \\ -0.07 \\ -0.4 \\ -0.3 \\ 0.99 \\ 0.078 \\ 0.1 \\ 0.47 \\ -1.1 \\ -0.53 \end{bmatrix}$$

give the very interesting trajectories shown in the next two figures. Note that each body seems to trace a different figure eight pattern. We would like to target

an initial state $\mathbf{x}_0$ such that the bodies follow eachother around the same figure eight pattern. Such an orbit is called a choreography and were first discovered only in the last $10 - 12$ years **??**.

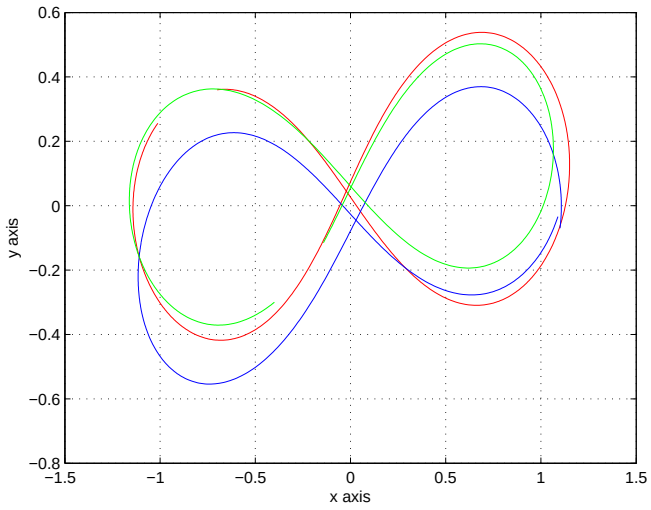


Figure 4: The Trajectories of the Initial Estimate

Such a trajectory would necessarily be periodic, with say period T. Further it the bodies follow one another around the curve then it seems reasonable that after one third of a period a body's position and velocity coincide with the initial position and velocity of the body leading it.

This can be expressed more precisely by requiring that $\mathbf{r}_1(T/3) = \mathbf{r}_2(0)$, $\mathbf{r}_2(T/3) = \mathbf{r}_3(0)$, $\mathbf{r}_3(T/3) = \mathbf{r}_1(0)$, $\mathbf{v}_1(T/3) = \mathbf{v}_2(0)$, $\mathbf{v}_2(T/3) = \mathbf{v}_3(0)$, and $\mathbf{v}_3(T/3) = \mathbf{v}_1(0)$.

Initial positions and velocities satisfying these conditions are sought. However we have no way of guessing the period of the figure 8 orbit, so the targeting procedure must find T as well (all the constraints above depend on T).

Further consideration reveals that there are other constraints the figure eight should satisfy. Bodies on such an orbit would certainly have trajectories which are bounded for all time. We also choose to constrain the orbit so that the figure does not drift, nor leave the plane. To meet this requirement the positons of the center of mass should be fixed for all time. Setting the initial position of the center arbitrarily at $(0, 0, 0)$ gives $\mathbf{r}_{cm} = 0$ and requiring $\mathbf{v}_{cm} = 0$ guaranteed that the center does not move.

The condition that the orbit lie in the xy plane combined with the requirement that it not rotate adds the constraint $h_z(t) = 0$.

Then the state vector $\mathbf{x} \in \mathbb{R}^{13}$ defined by

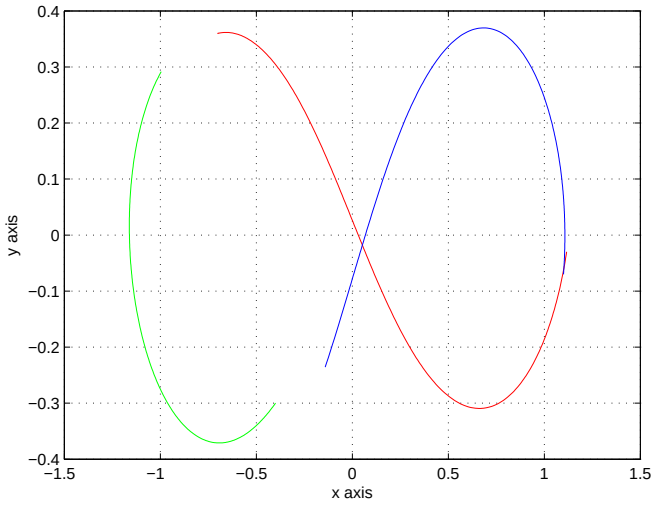


Figure 5: Time zero to initial tau: Ends Close but Not Matching

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \mathbf{v} \\ \tau \end{bmatrix}$$

where $\tau = T/3$ gives 13 scalar unknowns. However the constraints listed above give 17 scalar relations, so the system is surly overconstrained. (Note that in our notation this is the state space for the Newton Iteration, which is a dynamical system in it's own right, and not the state space of the original flow).

After a little experimentation one begins to suspect that these constraints are not independent. In fact if body one follows body two, body two follows body three, and in addition the bodies lie on a closed smooth curve in the plane, then surly body three must follow body one.

It would be nice to prove this, but for now we take it for granted, and remove the constraint $\mathbf{r}_3(T/3) = \mathbf{r}_1(0)$. (Uniqueness of solutions surly provides a quick proof). Similarly we guess that only two of the velocity constraints are independent and propose that the function $f : \mathbb{R}^{13} \to \mathbb{R}^{13}$ be defined by

$$f(\mathbf{x}) = f(\mathbf{r}, \mathbf{v}, \tau) = \begin{bmatrix} \mathbf{r}_1(T/3) - \mathbf{r}_2(0) \\ \mathbf{r}_2(T/3) - \mathbf{r}_3(0) \\ \mathbf{v}_1(T/3) - \mathbf{v}_2(0) \\ \mathbf{v}_2(T/3) - \mathbf{v}_3(0) \\ \mathbf{r}_{cm} \\ \mathbf{v}_{cm} \\ h_z(t) \end{bmatrix}$$

and note that if $f(\mathbf{x}) = 0$ then the constraints are satisfied.

5.1 The Derivative of the Constraint Function

To implement the correction procedure we must compute the derivative of f. This will be a 13×13 matrix. Let $M = m_1 + m_2 + m_3$ and write

$$f(\mathbf{x}) = f(\mathbf{r}, \mathbf{v}, \tau) = \begin{bmatrix} f_1(\mathbf{r}, \mathbf{v}, \tau) \\ f_2(\mathbf{r}, \mathbf{v}, \tau) \\ f_3(\mathbf{r}, \mathbf{v}, \tau) \\ f_4(\mathbf{r}, \mathbf{v}, \tau) \\ f_5(\mathbf{r}, \mathbf{v}, \tau) \\ f_6(\mathbf{r}, \mathbf{v}, \tau) \\ f_7(\mathbf{r}, \mathbf{v}, \tau) \end{bmatrix}$$

where

$$\begin{bmatrix} f_1(\mathbf{r}, \mathbf{v}, \tau) \\ f_2(\mathbf{r}, \mathbf{v}, \tau) \\ f_3(\mathbf{r}, \mathbf{v}, \tau) \\ f_4(\mathbf{r}, \mathbf{v}, \tau) \\ f_5(\mathbf{r}, \mathbf{v}, \tau) \\ f_6(\mathbf{r}, \mathbf{v}, \tau) \\ f_7(\mathbf{r}, \mathbf{v}, \tau) \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1(\tau) - \mathbf{r}_2(0) \\ \mathbf{r}_2(\tau) - \mathbf{r}_3(0) \\ \mathbf{v}_1(\tau) - \mathbf{v}_2(0) \\ \mathbf{v}_2(\tau) - \mathbf{v}_3(0) \\ \frac{1}{M}[m_1\mathbf{r}_1(0) + m_2\mathbf{r}_2(0) + m_3\mathbf{r}_3(0)] \\ \frac{1}{M}[m_1\mathbf{v}_1(0) + m_2\mathbf{v}_2(0) + m_3\mathbf{v}_3(0)] \\ \sum_{i=1}^{3}[r_i^1(0)v_i^2(0) - r_i^2(0)v_i^1(0)] \end{bmatrix}$$

Then

$$Df = \begin{bmatrix} D_{r_1}f_1 & \cdots & D_{v_3}f_1 & \frac{\partial}{\partial\tau}f_1 \\ \vdots & \ddots & \vdots & \vdots \\ D_{r_1}f_7 & \cdots & D_{v_3}f_7 & \frac{\partial}{\partial\tau}f_7 \end{bmatrix}$$

Then there are 49 separate terms to compute. There are however many similar terms. We will show several explicitly and leave the remaining to the readers imagination.

Consider the partials with respect to time. Only $f_1, \ldots, f_4$ depend on τ. In each case the needed partial comes down to either

$$\frac{\partial}{\partial\tau}\mathbf{r}_i(\tau) = \frac{\partial}{\partial\tau}\phi_{r_i}(\tau, \mathbf{x}_0)$$

or

$$\frac{\partial}{\partial\tau}\mathbf{v}_i(\tau) = \frac{\partial}{\partial\tau}\phi_{v_i}(\tau, \mathbf{x}_0)$$

But by property (3) of flows we have that these are $\dot{\mathbf{r}}_i(\tau) = \mathbf{v}_i(\tau)$ and $\dot{\mathbf{v}}_i(\tau) = g_i(\tau)$

It is the top left 8×12 matrix where terms which are related to $\Phi(\tau, 0)$ show up. We will call this matrix $U\Phi$ The computation of these goes like

$$D_{r_i} f_j(\mathbf{r}_0, \mathbf{v}_0) = D_{r_i} \mathbf{r}_j(\tau) = D_{r_i} \phi_{r_j}(\tau, \mathbf{r}_0, \mathbf{v}_0)$$

and we recall that this last term is a sub matrix of $\Phi(\tau, 0)$. Working through all the terms with τ dependence gives that

$$U\Phi = \begin{bmatrix} [\Phi(\tau,0)]_{(1:4,1:12)} - \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\ [\Phi(\tau,0)]_{(7:10,1:12)} - \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \end{bmatrix}$$

Here the $-\mathbf{I}$ terms come differentiating a term like $\mathbf{r}_j(\tau) - \mathbf{r}_{j+1}(0)$ with respect to $\mathbf{r}_{j+1}(0)$ as it is only in this case that the negative term has non zero derivative.

The terms remaining lie in the lower 5×13 matrix have no τ dependence. We are left to compute $D_{\mathbf{r}_i} \mathbf{r}_{cm}(0)$, $D_{\mathbf{v}_i} \mathbf{r}_{cm}(0)$, $D_{\mathbf{r}_i} \mathbf{v}_{cm}(0)$, $D_{\mathbf{v}_i} \mathbf{v}_{cm}(0)$, $D_{\mathbf{r}_i} h_z(0)$, and $D_{\mathbf{v}_i} h_z(0)$, and these are trivial. To get the flavor compute for example

$$D_{r_i} \mathbf{r}_{cm}(0) = D_{r_i} \frac{m_i}{M} \mathbf{r}_i(0) = \frac{m_i}{M} \mathbf{I}$$

and

$$\begin{aligned} D_{r_j} \sum_{i=1}^{3} m_i [r_i^1(0) v_i^2(0) - r_i^2(0) v_i^1(0)] &= D_{r_j} m_j [r_j^1(0) v_j^2(0) - r_j^2(0) v_j^1(0)] \\ &= m_j \begin{bmatrix} D_{r_j^1} [r_j^1(0) v_j^2(0) - r_j^2(0) v_j^1(0)] \\ D_{r_j^2} [r_j^1(0) v_j^2(0) - r_j^2(0) v_j^1(0)] \end{bmatrix}^T \\ &= m_j \begin{bmatrix} v_j^2(0) \\ -v_i^1(0) \end{bmatrix}^T \end{aligned}$$

Each of the 49 of the computations is similar to one we have done, and we can write down the entire differential. To save space denote the bottom left 1×12 matrix by

$$B = \begin{bmatrix} m_1 \begin{bmatrix} v_1^2(0) \\ -v_1^1(0) \end{bmatrix} \\ m_2 \begin{bmatrix} v_2^2(0) \\ -v_2^1(0) \end{bmatrix} \\ m_3 \begin{bmatrix} v_3^2(0) \\ -v_3^1(0) \end{bmatrix} \\ m_1 \begin{bmatrix} -r_1^2(0) \\ r_1^1(0) \end{bmatrix} \\ m_2 \begin{bmatrix} -r_2^2(0) \\ r_2^1(0) \end{bmatrix} \\ m_3 \begin{bmatrix} -r_3^2(0) \\ r_3^1(0) \end{bmatrix} \\ \end{bmatrix}^T$$

$$\begin{bmatrix} & & & & & \mathbf{v}_1(\tau) & \\ & U\Phi(\tau,0) & & & & \mathbf{v}_2(\tau) & \\ & & & & & g_1(\tau) & \\ & & & & & g_2(\tau) & \\ \frac{m_1}{M}\mathbf{I} & \frac{m_2}{M}\mathbf{I} & \frac{m_3}{M}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{m_1}{M}\mathbf{I} & \frac{m_2}{M}\mathbf{I} & \frac{m_3}{M}\mathbf{I} & \mathbf{0} \\ & & B & & & & 0 \end{bmatrix}$$

Now that we have the differential we can procede with the Newton Mathod exactly as before.

5.2 A Choreography Orbit

After running the differential correction procedure for 7 iterations the norm of the difference between the $\mathbf{x}_7$ and $\mathbf{x}_f$ is

$$\epsilon_{error} = |\mathbf{x}_7 - \mathbf{x}_f| = 6.777 * 10^{-13}$$

and we consider the sequence to have converged. The corrected initial conditions so obtained are

$$\mathbf{x}_0 = \begin{bmatrix} 0.9686 \\ 0.2802 \\ 1.3081 \\ 0.0939 \\ -0.3395 \\ -0.3741 \\ 0.7964 \\ 0.2701 \\ 0.0644 \\ 0.4377 \\ -0.8607 \\ -0.7078 \\ 2.8585 \end{bmatrix}$$

where the last entry is $\tau_7 = 2.8585$. Then the period of the choreography is roughly $T = 8.5755$, a little higher than the period of the known orbit.

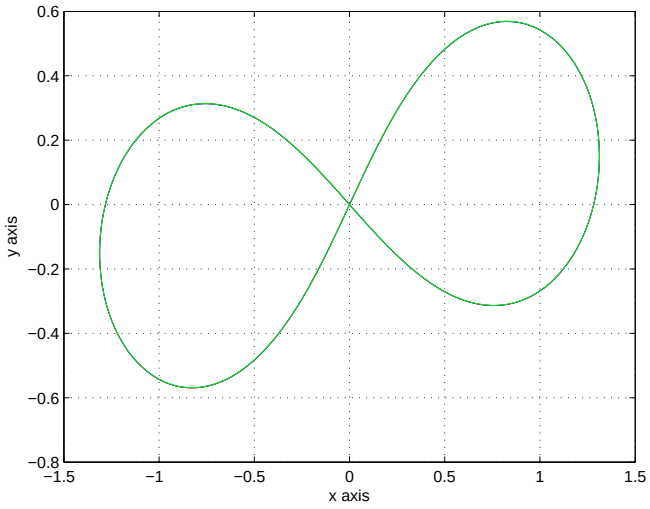


Figure 6: Corrected Orbits: A figure eight choreography

Integrating the N body problem with the first 12 components of $\mathbf{x}_0$ as the new initial conditions gives the trajectories shown in the figures ('Matched Boundary Conditions' and 'Corrected Orbits'), which both seem to show periodic trajectories with the bodies following one another around the a figure eight.

Once a periodic orbit is discovered the State Transition Matrix evaluated at any point on the orbit yields valuable geometric information. For example

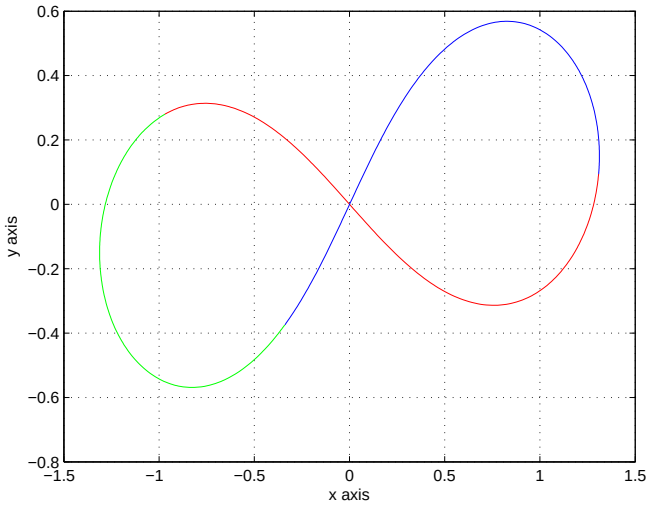


Figure 7: Matched Boundary Conditions

Robinsons Theorem 8.3 in chapter 5 guaranteed that the characteristic multipliers (eigenvalues of Φ that remain after ignoring the first (two) eigenvalues which are identically 1) are the eigenvalues of the derivative of the Poincare map evaluated at the periodic point, and restricted to the energy level of the orbit.

The forward dynamics of the periodic orbit are well understood be virtue of it being a periodic orbit. Since we are in a Hamiltonian system and examining a periodic orbit, the dynamics transverse to the energy surface are understood as well.

The theorem says that the remaining eigenvalues determine the dynamics near the periodic orbit in the directions transverse to the orbit and the energy surface. So eigenvalues with modulus less than one imply the existence of a local stable manifold for the orbit (similarly for eigenvalues larger than one and existence of an unstable manifold). Eigenvalues with modulus one (after ignoring the first one) imply the the existence of a local center manifold, where orbits are neither attracted to nor repelled by the periodic orbit. On this manifold we expect that recurrent dynamics may occur.

Computing the eigenvalues gives

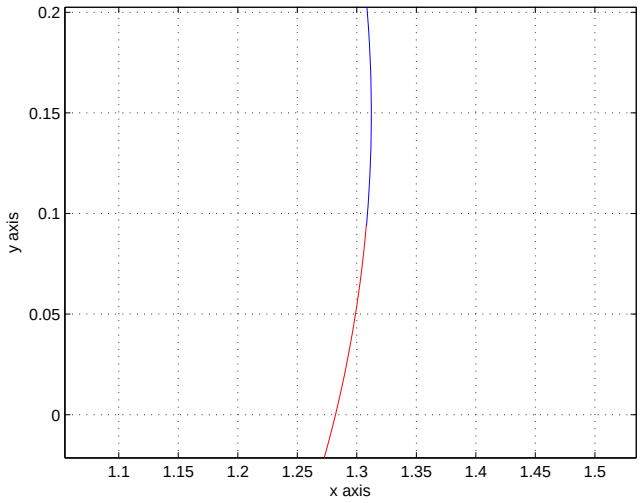


Figure 8: Close up of the boundary matching

$$[\Gamma] = \begin{bmatrix} -0.29759659585730 + 0.95469171261407i \\ -0.29759659585730 - 0.95469171261407i \\ 0.99859996433171 + 0.05289717607570i \\ 0.99859996433171 - 0.05289717607570i \\ 0.99999855937938 + 0.00169742136742i \\ 0.99999855937938 - 0.00169742136742i \\ 0.99999997027068 + 0.00024384093922i \\ 0.99999997027068 - 0.00024384093922i \\ 0.99999960785773 \\ 1.00000039214230 \\ 0.99999999999995 + 0.00000041249357i \\ 0.99999999999995 - 0.00000041249357i \end{bmatrix}$$

with moduli

$$|\Xi| = \begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}$$

The eigenvalues and their moduli tell the story of the dynamics near the orbit. We can clearly make out the two unity eigenvalues associated with the forward direction of flow, and the direction of movement transverse to the energy surface.

Then we note that all other eigenvalues are complex with modulus one. Then there is no stable or unstable manifold for this orbit, and all the dynamics is center manifold dynamics. Then the orbit is stable in the since of Lyapunov. Small perturbation of the orbit should remain near the choreography, and remain there for very long times (if nearby trajectories escape they must escape via center dynamics, which will not include exponential divergence in time).

A more dynamical systems prospective on this information is that, as mentioned above, the characteristic eigenvalues (the eigenvalues left when we disregard the two unity eigenvectors) of the manodromy matrix (the state transition matrix for a full period of a periodic orbit) are the eigenvectors of the linearization of the Poincare map near the orbit, restricted to the energy of the orbit. This map is a ten dimensional linear mapping, but neverthe less we know exactly what it must look like, at least near the orbit. The orbit would appear as an elliptic fixed point, with no hyperbolic directions at all. Then for the nonlinear map, orbits may diffuse (via Arnold diffusion) but only slowly.

We can also examine some of the critical observables associated with the N body problem for the orbit, and perhaps gain more insight into it's properties, or at least obtain independent conformation of some of what we have already obtained by other methods. The energy plot does not differ in any preceptable way from constant. In fact the plot of ΔE shows that the energy drifts on the order of 10^{-15} thoughout the integration. The suggests that the integration results can be trusted to roughly twelve or thirteen signifigant figures.

The plots of the moment of inertia I, as well as the plots of its first and second derivatives show the quantities oscillating in what looks like a slightly modulated sinusoidal fashion. The second derivative changes sign (not once but many times). This is good as it agrees with the eigenvalue data, which suggest that the orbit is Lyaponov Stable.

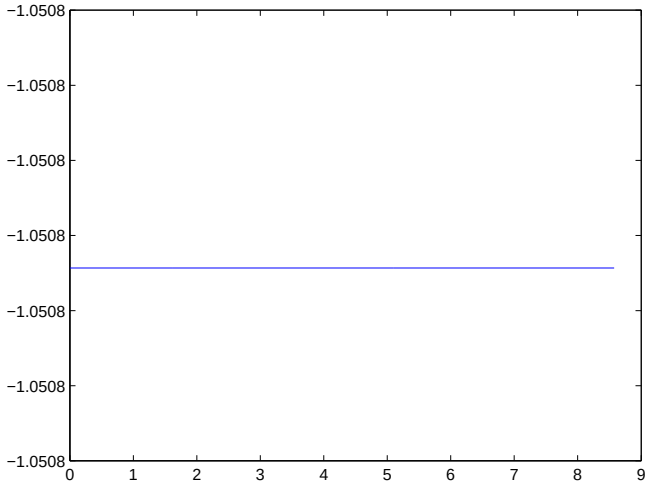


Figure 9: Energy

Finally, we remark that refinements of the methods used in these notes have been used by Kapela and Zgliczyski to give rigorous computer assisted proofs of the existence of choreography orbits [KZ].

A MatLab Code

The function 'stateTrans' is handed an initial time, an final time, an initial condition vector for the N body problem, the gravitational constant and three masses. The code works as written for three bodies in the plane.

'stateTrans.m' calls 'sysSolve.m' which in turn calls 'computeG.m'. None of the functions runs without the others.

```
function A=stateTrans(t0, tf, x,G,m1,m2,m3)

%Compute the state transition matrix at time tf for the
%path x(t) with x(t0)=x0

tspan=[t0,tf];                          %time span over which to run the integration

%--------------------------------------------------------------------------
%The state transition matrix is determined by a 12X12 matrix ODE
%This gives rise first to a system of 144 ODEs.  However the matrix
%ODE is nonautonomous and depends on a particular solution (trajectory)
%of the N-Body problem.  This trajectory is itself the solution of a system
```

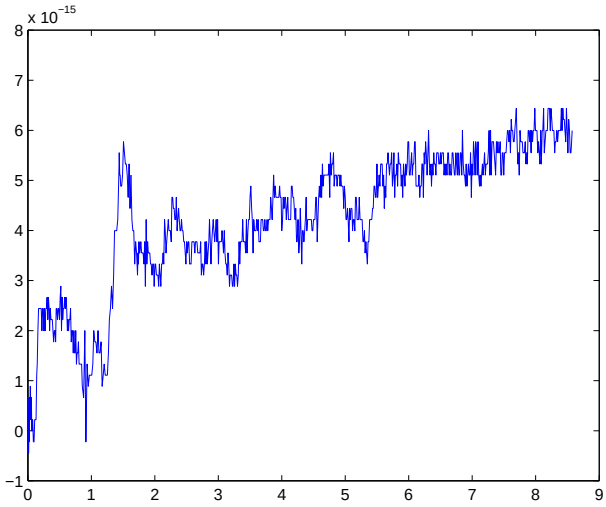
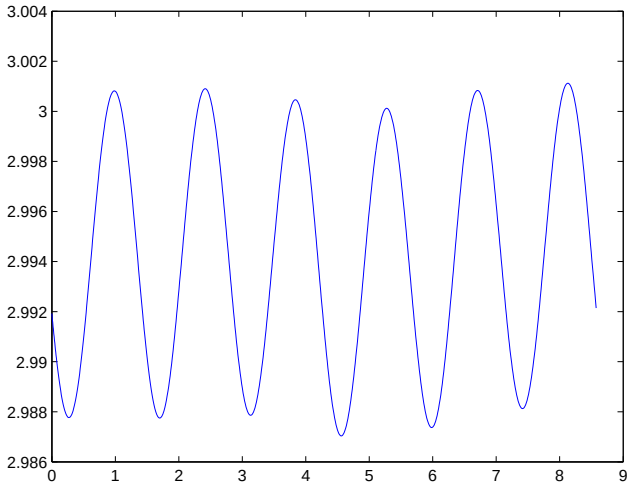


Figure 10: Drift in Energy

```
%of 12 ODEs.  The two systems are solved simultaneously, giving an
%autonomous system of 156 ODEs.
%-------------------------------------------------------------------------

%set up the initial condition, y0 for the 156 component system

y0=0;

%since the initial condition for the 12X12 matrix system is the 12X12
%identity matrix the initial condition vector is very sparse.  The first
%144 components are 0s and 1s, and the last 12 are the initial conditions
%from the N-Body problem.

 %Initialize a 12X12 identity matrix
I=eye(12);

%put the entries, row by row, into y0.
for i=1:12
   for j=1:12;
      y(12*(i-1)+j)=I(i,j);
   end
end

%the inital conditions for the particular orbit of the N-body problem have
```



of Inertia: I

```
%been passed in as 'x'. This to the end of y.

y(145:156)=x';

%Convert to a column vector and pass the initial conditions to the
%integrator
y0=y';                                                      %inital condition
options=odeset('RelTol',1e-14,'AbsTol',1e-22);              %set tolerences
[t,Y]=ode113('sysSolve',tspan,y0,options,[],G,m1,m2,m3);    %integrate the system

%[t,Y] is a huge matrix.  It is made up of a row for each time and 156
%columns.  But the desired data is the state transition matrix at the final
%time.  Then the first 144 entries of the last row of Y must be put into a
%12X12 matrix which will be passed back to the caller

%find out the row number of tf
b=size(Y);        %returns the number of rows and columns as a row vector
m=b(1,1);         %so 'm' is the number of rows
c=Y(m,1:144);     %so 'c' is a vector containing the first 144 entries of the last row of Y

%now c has to be made into a 12X12 matrix
d=0;
for i=1:12
    for j=1:12
        d(i,j)=c(12*(i-1)+j);
```

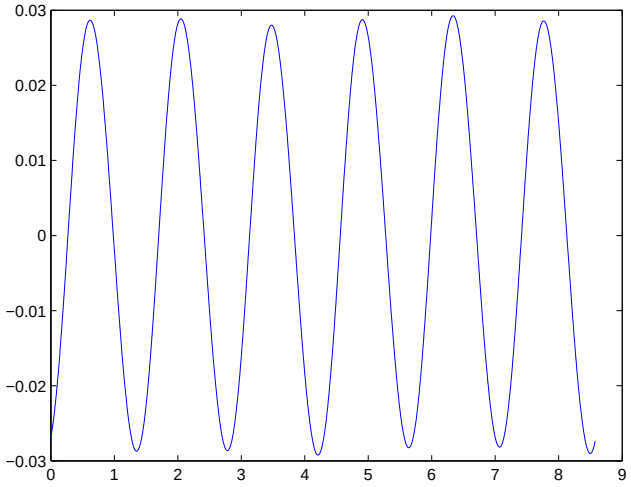
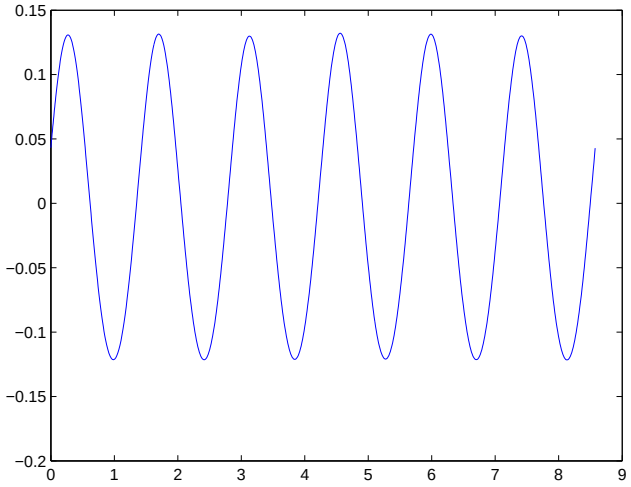


Figure 11: d/dt I

```
    end
end

%d is the state transition matrix at the final time and is passed back to
%the caller
A=d;
```

The program 'sysSolve.m' encodes the vector field for the state transition equations.

```
function ydot=sysSolve(t,y,options,flag,G,m1,m2,m3)

%This file contains the right hand side for the system which defines the
%state transition matrix for the three body problem in the plane.  So ydot
%is a vector with 156 components

%-------------------------------------------------------------------------
%The structure of the system is
%
```




```
%          A'=Df*A, x'=f(x)
%
%Where A,Df, are 12X12 matrices and A' is the derivative of A.
%Df is the derivative of the N-body vector field f and has the form
%
%                     |  0   I  |
%               Df = |         |
%                     | G    0  |
%
%where all submatrices are 6X6 and G has depends only on the position
%vectors of the bodies (all be it in a complicated way), and x'=f(x)
% is the 3-body problem in the plane.  All of this is combined onto one
%system of 156 1st order ODEs. The right hand side for the system is coded
%in the remainder of the file
%-------------------------------------------------------------------------


%FIRST
%The matrix G is computed
%This matrix depends on the positions of the N-body problem.
%These positions are contained in the 6 entries y(145:150).

%put the positions aside
x(1:6)=y(145:150);
```

```
%Now compute the matrix G.  Since 'G' already denotes the gravatational
%constant call the matrix G 'GMatrix'.
%This is done by calling 'computeG.m'
GMatrix=computeG(x,G, m1, m2, m3);

%SECOND
%The right hand side for the state transition system is computed
%To do this construct a matrices 'A' 'I' and 'O', where 'A' contains the
%variables and 'O' is a 6X6 matrix of zeros and 'I' is the 6X6 identity.
O=zeros(6);
I=eye(6);

%Now the complete Jacobian of f is assembled
Df=0;
Df=[O,I;
    GMatrix,O];

%Make A
for i=1:12
    for j=1:12
        A(i,j)=y(12*(i-1)+j);
    end
end

%Then compute the 12X12 matrix Df*A, is named DfA.
DfA=0;
DfA=Df*A;

%This has to be put into vector format. We temporaly place the results in
%the 144-vector 'a'.  Later this will be the first 144 components of ydot.
a=0;
for i=1:12
    for j=1:12
        a(12*(i-1)+j)=DfA(i,j);
    end
end

%THIRD
%The last 12 entries are the vector field for the 3-Body problem,
%restricted to the plane.  These are stored in 'c'.



%-------------------------------------------------------------------------
%--------The following code is essentialy the code from------------------
%-------'Nbody.m' copied here and adjusted for 2 dimensions----------------
```

```
%-------------------------------------------------------------------------
N=3;                  %Number of bodies
M=m1+m2+m3;           %Total mass
Mass=[m1 m2 m3];      %Vector of Masses

%Initialize the variables
acc=0;
s1=0;
s2=0;
s3=0;

%Compute the elements of the n-body vector field
for i=1:N
    for j=1:N
            Rij=(y(144+2*i-1)-y(144+2*j-1))^2+(y(144+2*i)-y(144+2*j))^2;
            %compute the three components of acceleration i
            if j~=i
               s1=s1+(Mass(1,j)/(sqrt(Rij))^3)*(y(144+2*j-1)-y(144+2*i-1));
               s2=s2+(Mass(1,j)/(sqrt(Rij))^3)*(y(144+2*j)-y(144+2*i));
            else
               s1=s1+0;
               s2=s2+0;
            end
    end
    acc(2*i-1)=G*s1;
    acc(2*i)=G*s2;
    s1=0;
    s2=0;
end

%Store the accelerations
accelerations=0;
accelerations=acc;

%enter the velociety components
velocities=0;
for i=1:(2*N)
velocities(i)=y(144+2*N+i);
end

%constructs a vector whose first 3*N entries are the velocities and whose
%last 3*N entries are the accelerations
c=0;
c(1:2*N,1)=velocities;
c(2*N+1:4*N,1)=accelerations;
```

```
%-------------------------------------------------------------------------
%-----------------------'End of Nbody.m'----------------------------------
%-------------------------------------------------------------------------

%Put it all toghether and pass back to integrator

ydot=[a';c];
```

'computeG' does exactly that.

```
function GMatrix=computeG(x,G, m1, m2, m3)

%function returns the matrix 'G' for the three body problem in the
%plane with parameters as specified.

%Collection the x data into the r_i vectors.
r1=x(1:2);
r2=x(3:4);
r3=x(5:6);


%The distances.  Rij stands for |r_j-r_i| where here r_i and
%r_j are vectors.  (two scripts is a scalar and one script is a vector.)
R12=sqrt((r2-r1)*(r2-r1)');
R21=R12;
R13=sqrt((r3-r1)*(r3-r1)');
R31=R13;
R23=sqrt((r3-r2)*(r3-r2)');
R32=R23;

%Only three terms, which are denoted by A12, A13, and A23 and are each 2X2
%matrices are necessary tocompute G.  All the elements of G are composed
%of these.

A12=((3/(R12)^5)*(r2-r1)'*(r2-r1)-(1/(R12)^3)*eye(2));
A13=((3/(R13)^5)*(r3-r1)'*(r3-r1)-(1/(R13)^3)*eye(2));
A23=((3/(R23)^5)*(r3-r2)'*(r3-r2)-(1/(R23)^3)*eye(2));


%These are combined to make G;
GMatrix=[G*(m2*A12+m3*A13), -G*m2*A12, -G*m3*A13;
    -G*m1*A12, G*(m1*A12+m3*A23), -G*m3*A23;
    -G*m1*A13, -G*m2*A23, G*(m1*A13+m2*A23)];
```

Here is an example program which calls repeatedly calls 'stateTrans.m' during a differential correction scheme. This prigram finds the coreography orbit from problem 4. It also calls the program 'plamarNbody.m' which is included below.

```
%Program produces results for problem 4 of homework two
%This is the Choreography problem

%This is the code to cary out problem 3c and 3d in homework 2

G=1;          %gravatational constant
N=3;          %Number of bodies

%Masses
m1=1;
m2=1;
m3=1;

%Sum of masses
M=m1+m2+m3;

%mass vector
mass=[m1,m2,m3];

%time parameters
t0=0.0;     %initial time

%Initial positions
r11=-0.755;
r12=0.355;
r21=1.155;
r22=-0.0755;
r31=-0.4055;
r32=-0.3055;

%Initial velocities
v11=0.9955;
v12=0.07855;
v21=0.1055;
v22=0.4755;
v31=-1.1055;
v32=-0.5355;
```

```
%Initial state
xStar0=[r11; r12;
   r21; r22
   r31; r32;
   v11; v12
   v21; v22
   v31; v32];

%Aux matrices
I=eye(2);
O=zeros(2);
J=[O,I; -I,O];

%-------------------------------------------------------
%The program will search for a coreography orbit
%Using a Newton method.  An equation of the form
%f(x)=0, where f:R^13->R^13 is the constraint
%vector. Then the difference equation has the form
%
%    x_{n+1} = x_n - [Df(x_n)]^-1 * f(x_n)
%
%To begin an initial condition is necessary.  It
%will be called x_n, and depends on xStar0, the
%final time at each step.
%-------------------------------------------------------
%Set the initial guess for the final time;
tau=2.15;   %This should be roughly 1/3 the desired period

%Number of itterations
K=8;
%Itteration loop
for i=1:K
  i
  %x_n  and f(x_n) must be computed.  They depend on the initial
  %positions and velocities of the reference orbit,
  %the final time tau, and the final positions and
  %velocities.  So, first a reference orbit must be computed;
  tspan=[t0, tau];
  options=odeset('RelTol',1e-13,'AbsTol',1e-22);
  [t,xStar]=ode113('planarNbody',tspan,xStar0',options,[],G,m1,m2,m3,N);
  %xStar_tau is now extracted.  This is the positions and
  %velocities at time tau.
  b=size(xStar);
```

```
m=b(1,1);
xStar_tau=xStar(m,:)';
%Then the current state depends on the following variables;
r1_tau=xStar_tau(1:2);
r2_tau=xStar_tau(3:4);
r3_tau=xStar_tau(5:6);
r1_0=xStar0(1:2);
r2_0=xStar0(3:4);
r3_0=xStar0(5:6);
v1_tau=xStar_tau(7:8);
v2_tau=xStar_tau(9:10);
v1_0=xStar0(7:8);
v2_0=xStar0(9:10);
v3_0=xStar0(11:12);
rcm_0=(1/M)*(m1*r1_0 + m2*r2_0 + m3*r3_0);
vcm_0=(1/M)*(m1*v1_0 + m2*v2_0 + m3*v3_0);
hz_01=m1*(r1_0(1)*v1_0(2) - r1_0(2)*v1_0(1));
hz_02=m2*(r2_0(1)*v2_0(2) - r2_0(2)*v2_0(1));
hz_03=m3*(r3_0(1)*v3_0(2) - r3_0(2)*v3_0(1));
hz_0=hz_01+hz_02+hz_03;
%The current state is;
x_n=[r1_0;
     r2_0;
     r3_0;
     v1_0;
     v2_0;
     v3_0;
     tau];
%While the current constraint vector is
fx_n=[r1_tau-r2_0;
        r2_tau-r3_0;
        v1_tau-v2_0;
        v2_tau-v3_0
        rcm_0;
        vcm_0;
        hz_0];
error=norm(fx_n)
%-------------------------------------------
%To execute the Newton algorithm the derivative
%of the constraint vector is needed.  This
%13X13 matrix is put together as follows;
%First the 2X2 blocks are computed
%-------------------------------------------
%The upper right 8X12 submatrix 'UR' of Df
%depends on the state transition matrix
X=stateTrans(t0,tau,xStar0,G,m1,m2,m3);
```

```
  %This submatrix is made of two 4X12 pieces UR1 and UR2;
  I1=[O, I, O, O, O, O;
        O, O, I, O, O, O];
  I2=[O, O, O, O, I, O;
      O, O, O, O, O, I];
  X1=X(1:4,:);
  X2=X(7:10,:);
  UR1=X1-I1;
  UR2=X2-I2;
  UR=[UR1; UR2];
  %Df also contains two of the N-body field terms
  R12=sqrt((r2_tau - r1_tau)'*(r2_tau - r1_tau));
  R13=sqrt((r3_tau - r1_tau)'*(r3_tau - r1_tau));
  R23=sqrt((r3_tau - r2_tau)'*(r3_tau - r2_tau));
  g1=G*(m2*((r2_tau - r1_tau)/(R12)^3) + m3*((r3_tau - r1_tau)/(R13)^3));
  g2=G*(m1*((r1_tau - r2_tau)/(R12)^3) + m3*((r3_tau - r2_tau)/(R23)^3));
  %For typographic reasons the following three terms are defined
  a=m1/M;
  b=m2/M;
  c=m3/M;
  %The partial derivatives of hz_0 with respect to
  %the initial positions and velocities are;
  d1h=m1*[v1_0(2),  -v1_0(1)];
  d2h=m2*[v2_0(2),  -v2_0(1)];
  d3h=m3*[v3_0(2),  -v3_0(1)];
  d4h=m1*[-r1_0(2), r1_0(1)];
  d5h=m2*[-r2_0(2), r2_0(1)];
  d6h=m3*[-r3_0(2), r3_0(1)];
  %The 8X13 top half 'TH' of Df is
  TH=[UR, [v1_tau; v2_tau; g1; g2]];
  %The bottom 5X13 submatrix 'BH' is
  BH= [a*I, b*I, c*I,    O,    O,   O,   [0;0];
         O,   O,   O,  a*I,  b*I, c*I,   [0;0];
       d1h, d2h, d3h,  d4h,  d5h, d6h,      0];
  %These compose Df which is now computed to be;
  Df=[TH; BH];
  %This must now be inverted
  DfInv=inv(Df);
  %Now the Newton step can be maid;
  x_n= x_n - DfInv*fx_n;
  %Then prepair for the next itteration
  tau=x_n(13);
  xStar0=x_n(1:12);
end

%-------------------------------------------------------------------------
```

```
%---------------------------------Results---------------------------------
%-------------------------------------------------------------------------

tau_target=tau
T=3*tau
xStar0_target=xStar0
RoughError=norm(fx_n)

%Compute the state transition matrix for the target trajectory
X=stateTrans(t0,T,xStar0_target,G,m1,m2,m3);
%Test Determinant
detTest=det(X)
%Compute eigenvalues and eigenvectors
[eigenVectors,eigenvalues]=eig(X);
outputEigenVect=eigenVectors
outputeigenvalues=eigenvalues

%moduli of eigenvalues
moduli=[norm(eigenvalues(1,1));
        norm(eigenvalues(2,2));
        norm(eigenvalues(3,3));
        norm(eigenvalues(4,4));
        norm(eigenvalues(5,5));
        norm(eigenvalues(6,6));
        norm(eigenvalues(7,7));
        norm(eigenvalues(8,8));
        norm(eigenvalues(9,9));
        norm(eigenvalues(10,10));
        norm(eigenvalues(11,11));
        norm(eigenvalues(12,12))]

%real parts of eigenvalues
realParts=[real(eigenvalues(1,1));
        real(eigenvalues(2,2));
        real(eigenvalues(3,3));
        real(eigenvalues(4,4));
        real(eigenvalues(5,5));
        real(eigenvalues(6,6));
        real(eigenvalues(7,7));
        real(eigenvalues(8,8));
        real(eigenvalues(9,9));
        real(eigenvalues(10,10));
        real(eigenvalues(11,11));
        real(eigenvalues(12,12))]
```

```
%Integrate the system with the target conditions;
tspan=[t0, 6*tau_target];
options=odeset('RelTol',1e-13,'AbsTol',1e-22);
[t,xCoreography]=ode113('planarNbody',tspan,xStar0_target',options,[],G,m1,m2,m3,N);


%_____________________________________________________________________________
%-------------------------Constants of Motion-----------------------------
%_____________________________________________________________________________



%The functions below were writen for the N body in three dimensions.
%Rather than rewriting them just make 'y' the system which is equal
%to xCoreography in the xy plane and equal to zeros in the z direction

A=size(xCoreography)                %vector containing (rows,columns)
timeMax=A(1,1)            %number of rows of y is number of time steps
Mass=mass;

y(1:timeMax,1:2)=xCoreography(1:timeMax,1:2);
y(1:timeMax,3)=zeros(1:timeMax,1);
y(1:timeMax,4:5)=xCoreography(1:timeMax,3:4);
y(1:timeMax,6)=zeros(1:timeMax,1);
y(1:timeMax,7:8)=xCoreography(1:timeMax,5:6);
y(1:timeMax,9)=zeros(1:timeMax,1);
y(1:timeMax,10:11)=xCoreography(1:timeMax,7:8);
y(1:timeMax,12)=zeros(1:timeMax,1);
y(1:timeMax,13:14)=xCoreography(1:timeMax,9:10);
y(1:timeMax,15)=zeros(1:timeMax,1);
y(1:timeMax,16:17)=xCoreography(1:timeMax,11:12);
y(1:timeMax,18)=zeros(1:timeMax,1);


%Energy: (T-U)(t)

%Potential Energy:  U(t)
U=0;
for i=1:timeMax
    S=0;              %Initialize S
        for j=1:N
            s=0;
            for k=1:N
                Rjk=(y(i,3*j-2)-y(i,3*k-2))^2+(y(i,3*j-1)-y(i,3*k-1))^2+(y(i,3*j)-y(i,3*k))
                if k~=j
```

```
                    s=s+Mass(1,k)/[sqrt(Rjk)];
                else
                   s=s+0;
                end
            end
            S=S+Mass(1,j)*s;
        end
    U(i)=(G/2)*S;
end


%Kenetic Energy;  T(t)
T=0;
for i=1:timeMax
    S=0;              %Initialize S
    for j=1:N
        S=S+Mass(1,j)*[(y(i,3*N+(3*j-2)))^2+(y(i,3*N+(3*j-1)))^2+(y(i,3*N+(3*j)))^2];
    end
    T(i)=S/2;
end

%Tenth integral; Energy   E(t)
E=0;
for i=1:timeMax
    E(i)=T(i)-U(i);
end


%Aux Quanties


%I or moment of inerita
I=0;
for i=1:timeMax
    S=0;              %Initialize S
    for j=1:N
        S=S+Mass(1,j)*[(y(i,3*j-2))^2+(y(i,3*j-1))^2+(y(i,3*j))^2];
    end
    I(i)=S;
end

%d/dt(I)
Idot=0;
for i=1:timeMax
    S=0;              %Initialize S
    for j=1:N
```

```
        S=S+Mass(1,j)*[y(i,3*N+(3*j-2))*y(i,3*j-2)+y(i,3*N+(3*j-1))*y(i,3*j-1)+y(i,3*N+(3*j
    end
    Idot(i)=S*2;
end

%d^2/(dt)^2(I)
Idotdot=0;
for i=1:timeMax
    Idotdot(i)=2*T(i)+2*E(i);
end

%Change in Energy.  If integration is perfect then this is always zero.
%Its deviation from zero measures the solutions deviation from the real
%one.

%deltaE
deltaE=0;
deltaE(1)=0;
for i=2:timeMax
    deltaE(i)=E(i)-E(1);
end

%---------------------------------Plotting---------------------------------

%Plotting
 plot(xCoreography(:,1), xCoreography(:,2),'r', xCoreography(:,3), xCoreography(:,4),'b', xC
 grid on
 xlabel('x axis'), ylabel('y axis')


%Plotting the integrals

%plot(t(1:timeMax),I(1:timeMax))
```

This program is necessary only to run the coreography program and is included only for completeness.

```
function ydot=planarNdody(t,y,options,flag,G,m1,m2,m3,N)

%code solves the planar N-Body problem

M=m1+m2+m3;          %Total mass
Mass=[m1 m2 m3];     %Vector of Masses
```

```
%Initialize the variables
acc=0;
s1=0;
s2=0;
s3=0;

%Compute the elements of the n-body vector field
for i=1:N
    for j=1:N
            Rij=(y(2*i-1)-y(2*j-1))^2+(y(2*i)-y(2*j))^2;
            %compute the three components of acceleration i
            if j~=i
               s1=s1+(Mass(1,j)/(sqrt(Rij))^3)*(y(2*j-1)-y(2*i-1));
               s2=s2+(Mass(1,j)/(sqrt(Rij))^3)*(y(2*j)-y(2*i));
            else
               s1=s1+0;
               s2=s2+0;
            end
    end
    acc(2*i-1)=G*s1;
    acc(2*i)=G*s2;
    s1=0;
    s2=0;
end

%Store the accelerations
accelerations=0;
accelerations=acc;

%enter the velociety components
velocities=0;
for i=1:(2*N)
velocities(i)=y(2*N+i);
end

%constructs a vector whose first 3*N entries are the velocities and whose
%last 3*N entries are the accelerations
c=0;
c(1:2*N,1)=velocities;
c(2*N+1:4*N,1)=accelerations;

ydot=c;
```

References

[A] Methods of Applied Mathematics, Unpublished Notes: www.math.utexas.edu/ arbogast/

[C] Alain Chenciner, Richard Montgomery, " A Remarkable periodic solution of the three-body problem in the case of equal masses", Annals of Mathematics, 152 (2000), 881-901

[CH] Ward Cheney, Analysis for Applied Mathematics, Springer

[GR] Marian Gidea, Clark Robinson, "Symbolic Dynamics for Transition Tori-II", Qualitative Theory of Dynamical Systems 1, 1-14 (2000) Article No.1

[HS] Morris W. Hirsch, Stephen Smale, Differential Equations, Dynamical Systems, and Linear Algebra, Academic Press

[MH] K.R. Meyer, G.R. Hall, Introduction to Hamiltonian Dynamical Systems and the N-body Problem, Springer-Verlag

[M] Richard Montgomery, "A New Solution to the Three-Body Problem", Notices of the AMS, May 2001, 471-481

[R] Clark Robinson, Dynamical Systems: Stability, Symbolic Dynamics, and Chaos, Second Edition, CRC Press, Boca Raton Florida, 1999

[KZ] Tomasz Kapela, Piotr Zgliczynski, Preprint