Some tricks with Maple to make it easier to work with many variables and equations

What I really meant to do in class yesterday was to talk about a few tricks with Maple that will make it simpler to work with the many variables and functions you will have in the astronomy project. The general idea is this: let's assume we only had to track a single body in its way through the solar system, then it will have x-, y-, and z-coordinates. We could have individual functions, say $x(t), y(t), z(t)$ for this, but it would be awkward to work with this because we continuously have to have them all in a single formula. For example, the gravity term has the distance between bodies cubed in the denominator. So we would have to write $\mathrm{sqrt}(x(t)^2 + y(t)^2 + z(t)^2)$ for the distance. It would be much nicer if we could somehow do this through a sum.

The way to do this is not to assign different symbols to the three components, but to say that $x[1](t), x[2](t), x[3](t)$ are the three functions that describe the x-, y- and z-coordinates of the body at time t. Then, the distance can be computed as

$\mathrm{sqrt}(\mathit{sum}(x[i](t)^2, i=1..3));$

$$\sqrt{x_1(t)^2 + x_2(t)^2 + x_3(t)^2} \tag{1}$$

Note that I didn't have to write out the sum, but that I rather summed over all indices I needed. Note also that I'm careful to carry around the fact that each of these $x[i]$ is a function of time.

So let's move beyond this: let's assume I have N bodies in my solar system. Then I would say that $x[i,1](t)$ is the x-coordinate of the ith body, and similarly $x[i,2](t), x[i,3](t)$ for the y- and z-coordinates. In my formulas I then need the distance between bodies i and j. Let me introduce a function for this:

$\mathit{distance} := (i,j) \rightarrow \mathrm{sqrt}(\mathit{sum}((x[i,k](t) - x[j,k](t))^2, k=1..3));$

$$(i,j) \rightarrow \sqrt{\sum_{k=1}^{3} \left(x_{i,k}(t) - x_{j,k}(t)\right)^2} \tag{2}$$

This is a function that takes 2 arguments, namely the numbers of the two bodies, and returns their (scalar) distance. For example, to get the distance between bodies 1 and 3, I would call

$\mathit{distance}(1,3);$

$$\sqrt{\left(x_{1,1}(t) - x_{3,1}(t)\right)^2 + \left(x_{1,2}(t) - x_{3,2}(t)\right)^2 + \left(x_{1,3}(t) - x_{3,3}(t)\right)^2} \tag{3}$$

Similarly, I mighth want to define the vector that points from the position of body i to the position of body j:

$\mathit{distancevector} := (i,j) \rightarrow [x[j,1](t) - x[i,1](t), x[j,2](t) - x[i,2](t), x[j,3](t) - x[i,3](t)];$

$$(i,j) \rightarrow \left[x_{j,1}(t) - x_{i,1}(t), x_{j,2}(t) - x_{i,2}(t), x_{j,3}(t) - x_{i,3}(t)\right] \tag{4}$$

$\mathit{distancevector}(1,3);$

$$\left[x_{3,1}(t) - x_{1,1}(t), x_{3,2}(t) - x_{1,2}(t), x_{3,3}(t) - x_{1,3}(t)\right] \tag{5}$$

In the function that we defined above, the function takes two arguments, i and j and returns a vector with three elements. If we called it with arguments $i=1, j=3$ as in the line immediately above, we get the three-components of the vector that separates bodies 1 and 3. We could extract the x-component of this vector like this:

$\mathit{distancevector}(1,3)[1];$

$$x_{3,1}(t) - x_{1,1}(t) \tag{6}$$

The problem with the definition of the $\mathit{distancevector}$ function is that we had to write out the elements of the vector by hand. That's tedious, because the three elements of the vector the function returns only differ by the respective second index of each term. Fortunately, there's a better way to do that: Maple's seq function just repeats an expression several times, each time replacing an index by a different value.

For example, we could call

$seq(x[j,k](t)-x[i,k](t), k=1..3);$

$$x_{j,1}(t)-x_{i,1}(t), x_{j,2}(t)-x_{i,2}(t), x_{j,3}(t)-x_{i,3}(t) \tag{7}$$

And, yes indeed, we get the expression three times, with k replaced by 1, 2, and 3 respectively. So to make a vector of this, we just have to enclose the call of the seq function by square brackets and we can simplify the definition of the distancevector function as follows:

$distancevector := (i,j) \rightarrow [seq(x[j,k](t)-x[i,k](t), k=1..3)];$

$$(i,j) \rightarrow \left[seq\left(x_{j,k}(t)-x_{i,k}(t), k=1..3\right)\right] \tag{8}$$

$distancevector(1,3);$

$$\left[x_{3,1}(t)-x_{1,1}(t), x_{3,2}(t)-x_{1,2}(t), x_{3,3}(t)-x_{1,3}(t)\right] \tag{9}$$

$distancevector(1,3)[1];$

$$x_{3,1}(t)-x_{1,1}(t) \tag{10}$$

So we get the same result with less typing and hopefully fewer errors.

So what can you do with this?

Let's assume you have $N=5$ bodies, numbered $0 \ldots N-1$, in such a way that each of them is attached by a rubber band to the previous and to the next one, with a rubber band also between body 1 and body 5 (i. e. they are linked in a circle). What this means is that body i is connected to body $(i+1) \bmod N$ (its right neighbor) and to body $(i-1) \bmod N$. Rubberbands produce a force that is proportional to the distance between the bodies and in the negative direction of the distance vector. Let's assume the proportionality constant is 1, then the force on body i is given by the following sum over the two bodies it is attached to:

$force := i \rightarrow -distancevector((i+1) \bmod N, i) - distancevector((i-1) \bmod N, i);$

$$i \rightarrow -distancevector((i+1) \bmod N, i) - distancevector((i-1) \bmod N, i) \tag{11}$$

For example, let's say there are $N=5$ bodies, then the x-component of the force on body 1 is given by

$N := 5 : force(1)[1];$

$$-2\,x_{1,1}(t)+x_{0,1}(t)+x_{2,1}(t) \tag{12}$$

That looks about right. There are no other forces, so you can imagine this to be five bodies that are mutually attached to each other through rubber bands but otherwise are freefloating somewhere out in space.

So now we need differential equations. We need them for all 5 bodies and all 3 components of the vector position. Let's start with the x-component of body zero, using Newton's law and assuming that the mass of all the bodies is one:

$diff(x[0,1](t), t, t) = force(0)[1];$

$$\frac{d^2}{dt^2} x_{0,1}(t) = -2\,x_{0,1}(t)+x_{4,1}(t)+x_{1,1}(t) \tag{13}$$

We could repeat this for all 15 equations we need, but that would be boring. Let's start slow and use the seq function again only for the three components of the ith body and respective force (note how we call $D(D(f))(t)$ to get the second time derivative of $f(t)$):

$seq(D(D(x[i,1]))(t) = force(i)[k], k=1..3);$

$$D^{(2)}(x_{i,1})(t) = -2\,x_{i,1}(t)+x_{i+4,1}(t)+x_{i+1,1}(t), D^{(2)}(x_{i,1})(t) = -2\,x_{i,2}(t)+x_{i+4,2}(t)+x_{i+1,2}(t), D^{(2)}(x_{i,1})(t) = -2\,x_{i,3}(t)+x_{i+4,3}(t)+x_{i+1,3}(t) \tag{14}$$

So this gives us the three differential equations for body i. Now let's wrap this whole thing into one more seq call to get the differential equations for all bodies $i = 0 ..N - 1$ (which we will all put into the variable ode):

$ode := seq(seq(\mathrm{D}(\mathrm{D}(x[i, k]))(t) = force(i)[k], k = 1 ..3), i = 0 ..N - 1);$

$$\mathrm{D}^{(2)}(x_{0,1})(t) = -2\,x_{0,1}(t) + x_{4,1}(t) + x_{1,1}(t), \mathrm{D}^{(2)}(x_{0,2})(t) = -2\,x_{0,2}(t) + x_{4,2}(t) + x_{1,2}(t), \mathrm{D}^{(2)}(x_{0,3})(t) = -2\,x_{0,3}(t) + x_{4,3}(t) + x_{1,3}(t), \mathrm{D}^{(2)}(x_{1,1})(t) = -2\,x_{1,1}(t) + x_{0,1}(t) + x_{2,1}(t), \mathrm{D}^{(2)}(x_{1,2})(t) = -2\,x_{1,2}(t) + x_{0,2}(t) + x_{2,2}(t), \mathrm{D}^{(2)}(x_{1,3})(t) = -2\,x_{1,3}(t) + x_{0,3}(t) + x_{2,3}(t), \mathrm{D}^{(2)}(x_{2,1})(t) = x_{1,1}(t) - 2\,x_{2,1}(t) + x_{3,1}(t), \mathrm{D}^{(2)}(x_{2,2})(t) = x_{1,2}(t) - 2\,x_{2,2}(t) + x_{3,2}(t), \mathrm{D}^{(2)}(x_{2,3})(t) = x_{1,3}(t) - 2\,x_{2,3}(t) + x_{3,3}(t), \mathrm{D}^{(2)}(x_{3,1})(t) = x_{2,1}(t) - 2\,x_{3,1}(t) + x_{4,1}(t), \mathrm{D}^{(2)}(x_{3,2})(t) = x_{2,2}(t) - 2\,x_{3,2}(t) + x_{4,2}(t), \mathrm{D}^{(2)}(x_{3,3})(t) = x_{2,3}(t) - 2\,x_{3,3}(t) + x_{4,3}(t), \mathrm{D}^{(2)}(x_{4,1})(t) = x_{3,1}(t) - 2\,x_{4,1}(t) + x_{0,1}(t), \mathrm{D}^{(2)}(x_{4,2})(t) = x_{3,2}(t) - 2\,x_{4,2}(t) + x_{0,2}(t), \mathrm{D}^{(2)}(x_{4,3})(t) = x_{3,3}(t) - 2\,x_{4,3}(t) + x_{0,3}(t) \quad \textbf{(15)}$$

Ah, 15 differential equations generated in a single command! Since we have a second order differential equation (or, rather, 15 such equations), we have to have 2 initial conditions for each variable: position and velocity. So we need an N-by-3 matrix of initial positions and an N-by-3 matrix of initial velocities. Since this is just a toy problem, let us generate them randomly netween zero and one (we have to specify the range as floating point numbers to get floating point positions rather than integers -- if we had given the range as $-1 ..1$, Maple would have drawn each element of the random matrix as either zero or one, never in between):

$initialPositions := LinearAlgebra[RandomMatrix](N, 3, generator = -1.0 ..1.0);$

$$\begin{bmatrix} -0.857109070798715234 & 0.000943248309686062214 & -0.942651695071787810 \\ -0.915137724998516644 & 0.425388943357828175 & 0.773023866152202643 \\ 0.363943808298126337 & 0.957361299282317502 & -0.520135978862565240 \\ -0.880762264840721620 & -0.664145708635486498 & -0.634155061170172063 \\ -0.0578232509161213937 & -0.0201972229755522648 & 0.153443031229370197 \end{bmatrix} \quad \textbf{(16)}$$

$initialVelocities := LinearAlgebra[RandomMatrix](N, 3, generator = -1.0 ..1.0);$

$$\begin{bmatrix} -0.833060370282171947 & 0.297982985424712243 & 0.444879184733684596 \\ 0.650627590804091271 & 0.945949109527725041 & 0.635094184158572661 \\ -0.135216992433076610 & 0.0371898850210763410 & 0.636297107719249411 \\ -0.0924045825461610182 & 0.319210505816614364 & -0.806539948438266041 \\ 0.600661150704802971 & -0.700269115044066304 & 0.0432996849285673946 \end{bmatrix} \quad \textbf{(17)}$$

Next, we have to generate the initial conditions in a way that Maple's dsolve function can understand them. We could again write them out like so:

$x[0, 1](0) = initialPositions[1, 1];$

$$x_{0,1}(0) = -0.857109070798715234 \quad \textbf{(18)}$$

(Note that we have numbered bodies starting at zero while the matrix entries are numbered starting at 1!) . But that would be cumbersome, since we would have to write 30 such equations (initial positions + velocities for each of the 3 components of each of the 5 bodies). Let's just use the seq command again:

$ic1 := seq(seq(x[i,k](0) = initialPositions[i+1,k], k = 1..3), i = 0..N-1);$

$$x_{0,1}(0) = -0.857109070798715234, x_{0,2}(0) = 0.000943248309686062214, x_{0,3}(0) = -0.942651695071787810, x_{1,1}(0) = -0.915137724998516644, x_{1,2}(0) = 0.425388943357828175, x_{1,3}(0) = 0.773023866152202643, x_{2,1}(0) = 0.363943808298126337, x_{2,2}(0) = 0.957361299282317502, x_{2,3}(0) = -0.520135978862565240, x_{3,1}(0) = -0.880762264840721620, x_{3,2}(0) = -0.664145708635486498, x_{3,3}(0) = -0.634155061170172063, x_{4,1}(0) = -0.0578232509161213937, x_{4,2}(0) = -0.0201972229755522648, x_{4,3}(0) = 0.153443031229370197 \tag{19}$$

$ic2 := seq(seq(D(x[i,k])(0) = initialVelocities[i+1,k], k = 1..3), i = 0..N-1)$

$$D(x_{0,1})(0) = -0.833060370282171947, D(x_{0,2})(0) = 0.297982985424712243, D(x_{0,3})(0) = 0.444879184733684596, D(x_{1,1})(0) = 0.650627590804091271, D(x_{1,2})(0) = 0.945949109527725041, D(x_{1,3})(0) = 0.635094184158572661, D(x_{2,1})(0) = -0.135216992433076610, D(x_{2,2})(0) = 0.0371898850210763410, D(x_{2,3})(0) = 0.636297107719249411, D(x_{3,1})(0) = -0.0924045825461610182, D(x_{3,2})(0) = 0.319210505816614364, D(x_{3,3})(0) = -0.806539948438266041, D(x_{4,1})(0) = 0.600661150704802971, D(x_{4,2})(0) = -0.700269115044066304, D(x_{4,3})(0) = 0.0432996849285673946 \tag{20}$$

So let's put all this together into one big list that we can then feed to *dsolve*:

$equations := \{ode, ic1, ic2\};$

$$\{x_{0,1}(0) = -0.857109070798715234, x_{0,2}(0) = 0.000943248309686062214, x_{0,3}(0) = -0.942651695071787810, x_{1,1}(0) = -0.915137724998516644, x_{1,2}(0) = 0.425388943357828175, x_{1,3}(0) = 0.773023866152202643, x_{2,1}(0) = 0.363943808298126337, x_{2,2}(0) = 0.957361299282317502, x_{2,3}(0) = -0.520135978862565240, x_{3,1}(0) = -0.880762264840721620, x_{3,2}(0) = -0.664145708635486498, x_{3,3}(0) = -0.634155061170172063, x_{4,1}(0) = -0.0578232509161213937, x_{4,2}(0) = -0.0201972229755522648, x_{4,3}(0) = 0.153443031229370197, D(x_{0,1})(0) = -0.833060370282171947, D(x_{0,2})(0) = 0.297982985424712243, D(x_{0,3})(0) = 0.444879184733684596, D(x_{1,1})(0) = 0.650627590804091271, D(x_{1,2})(0) = 0.945949109527725041, D(x_{1,3})(0) = 0.635094184158572661, D(x_{2,1})(0) = -0.135216992433076610, D(x_{2,2})(0) = 0.0371898850210763410, D(x_{2,3})(0) = 0.636297107719249411, D(x_{3,1})(0) = -0.0924045825461610182, D(x_{3,2})(0) = 0.319210505816614364, D(x_{3,3})(0) = \tag{21}$$

$$-0.806539948438266041, \mathrm{D}(x_{4,1})(0) = 0.600661150704802971, \mathrm{D}(x_{4,2})(0) = -0.700269115044066304, \mathrm{D}(x_{4,3})(0) = 0.0432996849285673946, \mathrm{D}^{(2)}(x_{0,1})(t) = -2\,x_{0,1}(t) + x_{4,1}(t) + x_{1,1}(t), \mathrm{D}^{(2)}(x_{0,2})(t) = -2\,x_{0,2}(t) + x_{4,2}(t) + x_{1,2}(t), \mathrm{D}^{(2)}(x_{0,3})(t) = -2\,x_{0,3}(t) + x_{4,3}(t) + x_{1,3}(t), \mathrm{D}^{(2)}(x_{1,1})(t) = -2\,x_{1,1}(t) + x_{0,1}(t) + x_{2,1}(t), \mathrm{D}^{(2)}(x_{1,2})(t) = -2\,x_{1,2}(t) + x_{0,2}(t) + x_{2,2}(t), \mathrm{D}^{(2)}(x_{1,3})(t) = -2\,x_{1,3}(t) + x_{0,3}(t) + x_{2,3}(t), \mathrm{D}^{(2)}(x_{2,1})(t) = x_{1,1}(t) - 2\,x_{2,1}(t) + x_{3,1}(t), \mathrm{D}^{(2)}(x_{2,2})(t) = x_{1,2}(t) - 2\,x_{2,2}(t) + x_{3,2}(t), \mathrm{D}^{(2)}(x_{2,3})(t) = x_{1,3}(t) - 2\,x_{2,3}(t) + x_{3,3}(t), \mathrm{D}^{(2)}(x_{3,1})(t) = x_{2,1}(t) - 2\,x_{3,1}(t) + x_{4,1}(t), \mathrm{D}^{(2)}(x_{3,2})(t) = x_{2,2}(t) - 2\,x_{3,2}(t) + x_{4,2}(t), \mathrm{D}^{(2)}(x_{3,3})(t) = x_{2,3}(t) - 2\,x_{3,3}(t) + x_{4,3}(t), \mathrm{D}^{(2)}(x_{4,1})(t) = x_{3,1}(t) - 2\,x_{4,1}(t) + x_{0,1}(t), \mathrm{D}^{(2)}(x_{4,2})(t) = x_{3,2}(t) - 2\,x_{4,2}(t) + x_{0,2}(t), \mathrm{D}^{(2)}(x_{4,3})(t) = x_{3,3}(t) - 2\,x_{4,3}(t) + x_{0,3}(t)\}$$

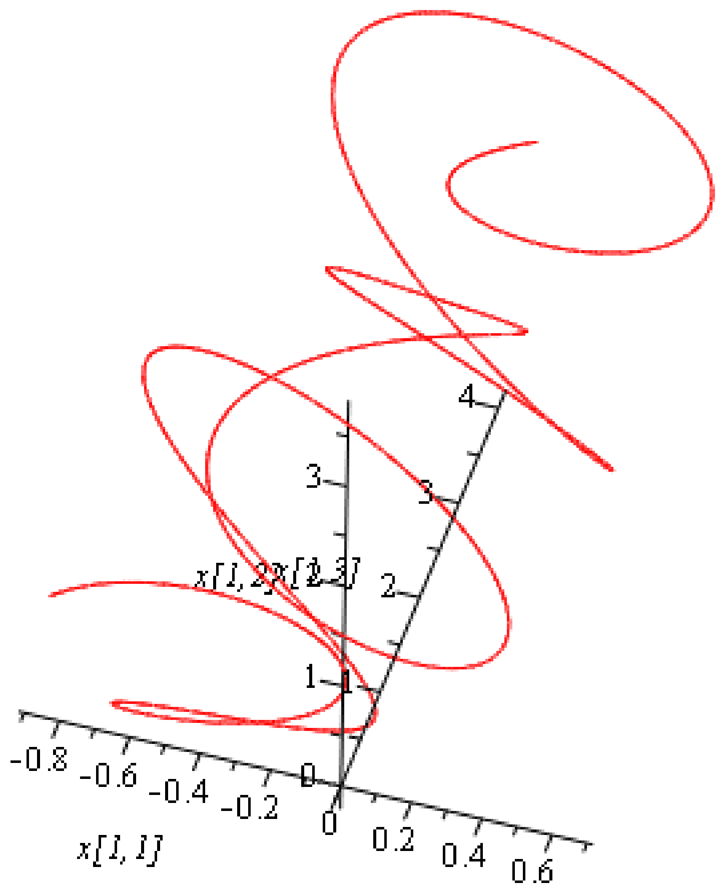
We can now give this to *dsolve*. Maybe surprisingly, Maple can solve this set of equations analytically, giving us an incredibly long expression for the 15 time dependent functions. Let's not go there, we are, after all, not interested in the exact solution. Rather, we want to plot the result, so let's get ourselves a numerical solution and assign it to a variable:

$sol := dsolve(equations, numeric);$

$$\mathbf{proc}(x\_rkf45)\ \ldots\ \mathbf{end\ proc} \tag{22}$$

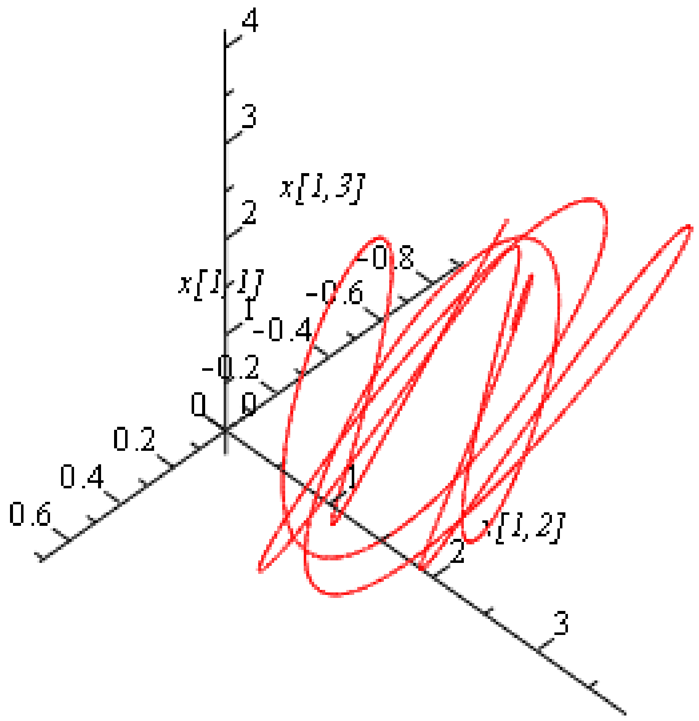
As a final step, we can plot it the solution. If we're only interested in the trajectory of the first body, here it is (where I have rotated the picture for a better view):

$plots[odeplot](sol, [x[1, 1](t), x[1, 2](t), x[1, 3](t)], t = 0..20, numpoints = 1000, axes = normal);$



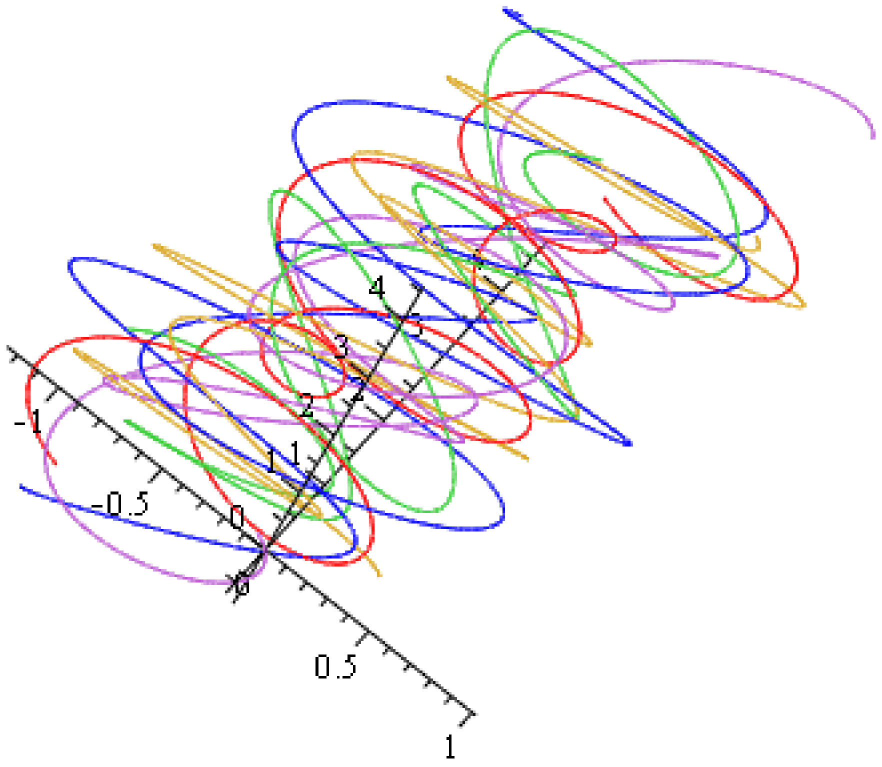
OK, so that's already pretty cool. It was a bit cumbersome to write out the three variables again, but we could have used the *seq* command once more (in the un-rotated image):

plots[*odeplot*](*sol*, [*seq*(*x*[1, *k*](*t*), *k* = 1 ..3)], *t* = 0 ..20, *numpoints* = 1000, *axes* = *normal*);



Of course, what we'd really like to do is see how the set of all five bodies behaves. So out comes the *seq* command one last time:

$plots[odeplot](sol, [seq([seq(x[i, k](t), k=1..3)], i=0..N-1)], t=0..20, numpoints=1000, axes=normal);$



Lovely, isn't it!? Now if you wanted to play around with this a bit, increase the number of bodies N -- and notice that you don't have to change any of the equations at all as they should all do the right thing for *any* number of bodies N!