Newton's method for ODEs!

PHYS 250 (Autumn 2024) – Lecture 10

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

October 31, 2018

Outline

Reminders from last time

Looked at one of two primary and exemplary methods for root finding, which is part of the foundation of optimization and differential equation solving.

Fundamental root finding methods

- **Bisection method (aka "incremental search"):** (Tuesday)
  - **PROs:** exceptionally simple and requires no knowledge of the function whose roots are sought
  - **CONs:** doesn't use the potentially very useful knowledge of the roots that are sought
- **Newton's Method:** (today)
  - **PROs:** converges much faster than bisection
  - **CONs:** requires a calculation or estimation of the first derivative of the function

Today, we will expand on Newton's method and go several steps further.

Newton's method (I)

The fundamental form of the method alluded to in the last lecture is **Newton's method**. The core of this approach is to use information about the function in order to inform how **big** of a step size to take with each iteration.

Suppose a function $F(x)$ has a root $x = x_0$ and you are within δ of that root at $x = x_0 - \delta$. The question we want to answer is:

How far away from the root are we actually? What is the value of δ?

Using a Taylor expansion (as Wah told us!)

$$F(x_0) = F(x+\delta) \approx F(x) + \delta F'(x) + \frac{1}{2}\delta^2 F''(x) + \mathcal{O}(\delta^3) \tag{1}$$

By setting $F(x_0) = 0$ by assumption, we can make the linear approximation to $\delta \approx \Delta$ as

$$\Delta = -\frac{F(x)}{F'(x)} \tag{2}$$

Newton's method (II)

Therefore, we can choose a starting point x_i and then update x according to Newton

$$x_{i+1} = x_i - \frac{F(x_i)}{F'(x_i)} \tag{3}$$

The iteration stops after j iterations when

$$|x - x_j| \leq \epsilon \tag{4}$$

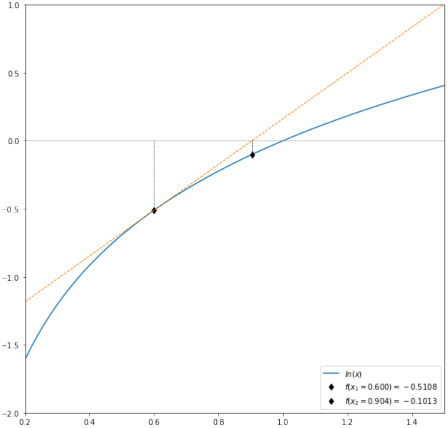
Precision of Newton's method

For any estimate x_i of the method, the error, E_i is the difference between the true root x and the estimate:

$$E_i = x - x_i \tag{5}$$

Merely by inspecting the design of Newton's method, you can see that the precision of the estimate for a subsequent iteration will be given by

$$\begin{aligned} E_{i+1} &= E_i + \frac{F(x)}{F'(x)} && (6) \\ &= -\frac{F''(x)}{2F'(x)} E_i^2 && (7) \end{aligned}$$

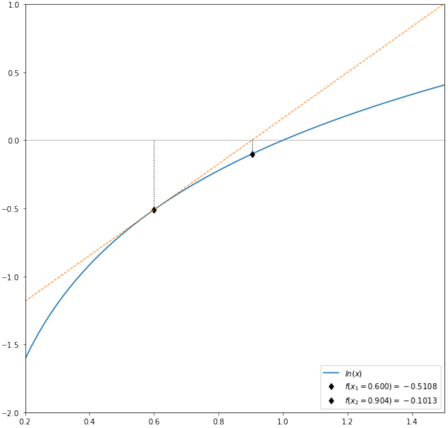


Convergence of Newton's method

Consequently, Newton's method **converges quadratically**

- the error is the square of the error in the previous step)
- the number of significant figures is roughly doubled in every iteration, provided that x_i is **sufficiently** close to the root.

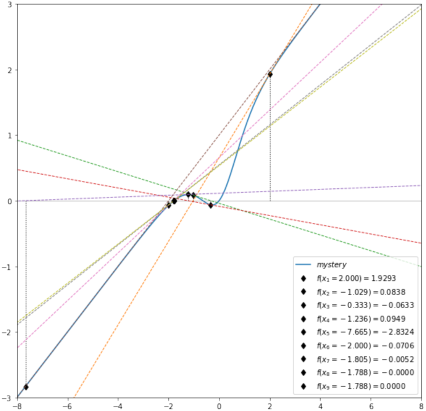
However, a critical assumption is that $F'(x) \neq 0$; for all $x \in I$, where I is the interval $[x-r, x+r]$ for some $r \geq |x - x_0|$ and x is the true root and x_0 was the starting point.



Pathologies and divergent scenarios

That is definitely not always the case. Let's look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

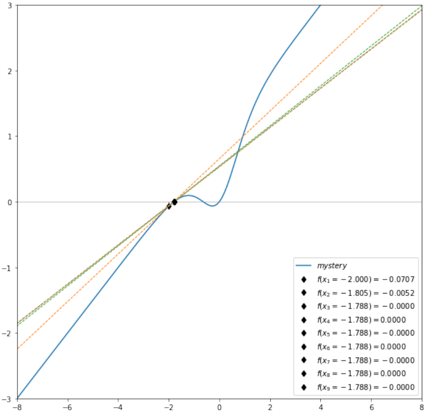
- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification: $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Pathologies and divergent scenarios

That is definitely not always the case. Let's look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

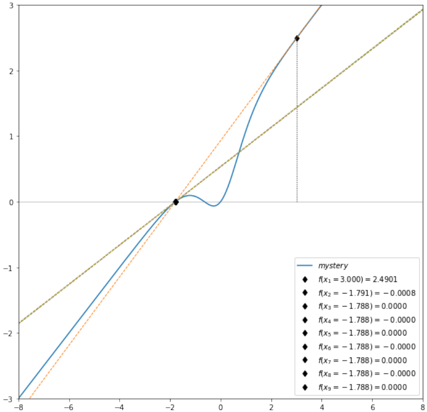
- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification: $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Pathologies and divergent scenarios

That is definitely not always the case. Let’s look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

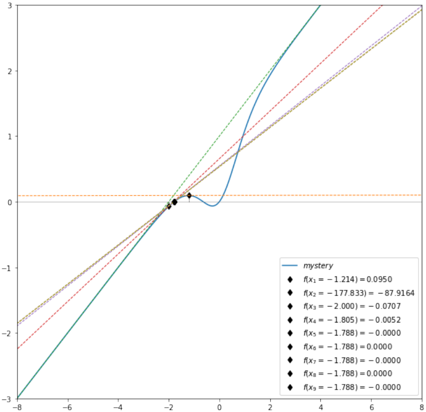
- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification: $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Pathologies and divergent scenarios

That is definitely not always the case. Let's look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

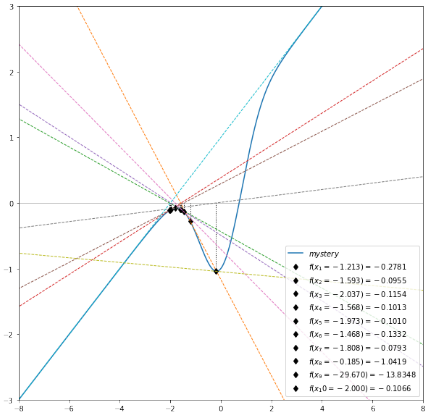
- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification: $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Pathologies and divergent scenarios

That is definitely not always the case. Let's look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

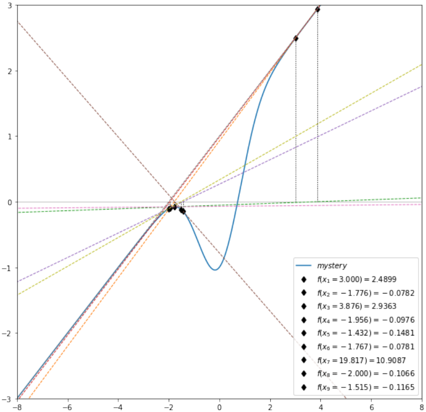
- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification:
  $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Pathologies and divergent scenarios

That is definitely not always the case. Let's look at a pathological example. Here is a fun mystery function that I cooked up (since you need to do something similar on your homework):

- $x_0 = 2.000$, **7 iterations**
- $x_0 = -2.000$, **2 iterations**
- $x_0 = 3.000$, **2 iterations**
- $x_0 = -1.214$, **4 iterations**
- Slight modification: $x_0 = -1.213$, **no convergence**
- Slight modification: $x_0 = 3.000$, **no convergence**



Backtracking

In the last examples above we have a case where the search falls into the pathology of a situation where the initial guess was not **sufficiently close** to the root. an "infinite" loop without ever getting there.

A solution to this problem is called **backtracking**.

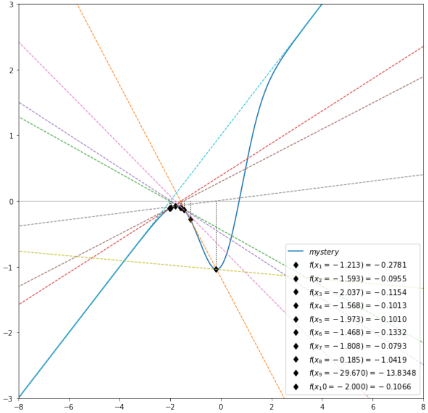
Backtracking

In cases where the new guess $x_0 + \Delta x$ leads to an increase in the magnitude of the function, $|f(x_0 + \Delta x)|^2 > |f(x_0)|^2$, you should backtrack somewhat and try a smaller guess, say, $x_0 + \Delta x/2$. If the magnitude of f still increases, then you just need to backtrack some more, say, by trying $x_0 + \Delta x/4$ as your next guess, and so forth.

Pathological case fixed with backtracking

Fixing the pathological example with backtracking:

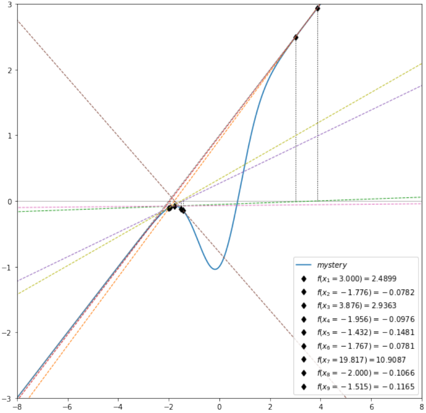
- $x_0 = -1.213$, **no convergence**
- $x_0 = 3.000$, **no convergence**
- $x_0 = 1.500$, **3 iterations**



Pathological case fixed with backtracking

Fixing the pathological example with backtracking:

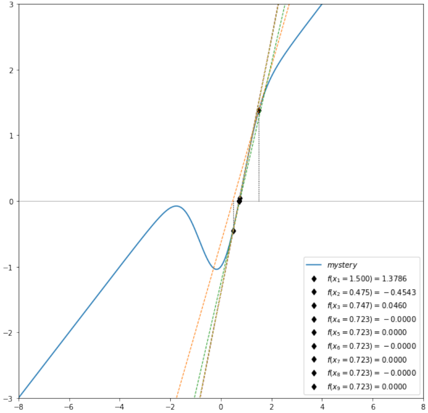
- $x_0 = -1.213$, **no convergence**
- $x_0 = 3.000$, **no convergence**
- $x_0 = 1.500$, **3 iterations**



Pathological case fixed with backtracking

Fixing the pathological example with backtracking:

- $x_0 = -1.213$, **no convergence**
- $x_0 = 3.000$, **no convergence**
- $x_0 = 1.500$, **3 iterations**



Multidimensional problems

Up to this point, we have confined our attention to solving the single equation $F(x) = 0$. Let us now consider the n-dimensional version of the same problem, namely

$$\vec{F}(\vec{x}) = 0 \tag{8}$$

where we allow for a vector of functions $\vec{F} = \{f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})\}$, and $\vec{x} = \{x_1, x_2, ..., x_n\}$.

The solution of n simultaneous, nonlinear equations is a much more formidable task than finding the root of a single equation. The trouble is the lack of a reliable method for bracketing the solution vector $\vec{x}$. Therefore, we cannot always provide the solution algorithm with a good starting value of x, unless such a value is suggested by the physics of the problem.

Newton's method is the workhorse here!

Reminder of the general problem

Start by considering each one of the n functions, $f_n(x)$, separately:

$$\begin{aligned} f_i(\vec{x}) &= f_i(\vec{a}) + \sum_j^n \frac{\partial f_i}{\partial x_j}|_{\vec{x}}\Delta x_j + \frac{1}{2!}\sum_j^n\sum_k^n \frac{\partial^2 f_i}{\partial_j \partial_k}|_{\vec{x}}\Delta x_j \Delta x_k &\quad (9) \\ &= f_i(\vec{a}) + \vec{\nabla} f_i(\vec{a}) \cdot \vec{\Delta x} + \frac{1}{2!}\vec{\Delta x}^{\mathrm{T}}\mathbf{H}(\vec{a})\vec{\Delta x} &\quad (10) \end{aligned}$$

where $\mathbf{H}$ is the **Hessian matrix**, describing the **curvature** of $f_i(\vec{x})$ by

$$\mathbf{H}_{j,k} = \frac{\partial^2 f(\vec{a})}{\partial x_j \partial x_k} \quad (11)$$

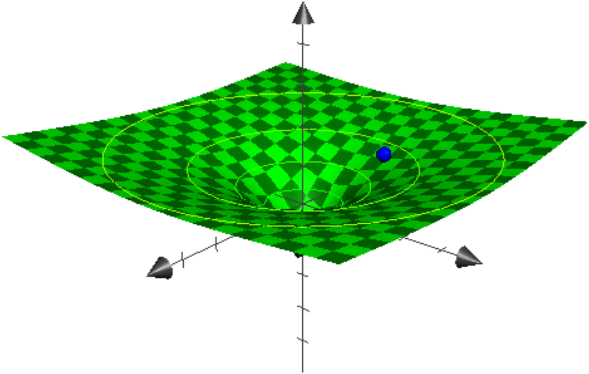
(The determinant of $\mathbf{H}$ is also sometimes referred to as **the Hessian**) and $\vec{\Delta x}$ is a vector that describes the distance from the point about which we are expanding the function $f_i(\vec{x})$.

Specific example (I)

Let’s take a very explicit example of a function:

$$f(\vec{x}) = f(x,y) = \ln\sqrt{x^2+y^2} \tag{12}$$

and compute the gradient and Hessian matrix at $x=-2, y=1$ (see point on graph below).



Specific example (II)

First, we calculate the gradient:

$$\frac{\partial f}{\partial x} = \frac{1}{\sqrt{x^2+y^2}}\left(\frac{1}{2}\frac{2x}{\sqrt{x^2+y^2}}\right) \tag{13}$$

$$= \frac{x}{x^2+y^2} \tag{14}$$

$$\frac{\partial f}{\partial y} = \frac{y}{x^2+y^2} \tag{15}$$

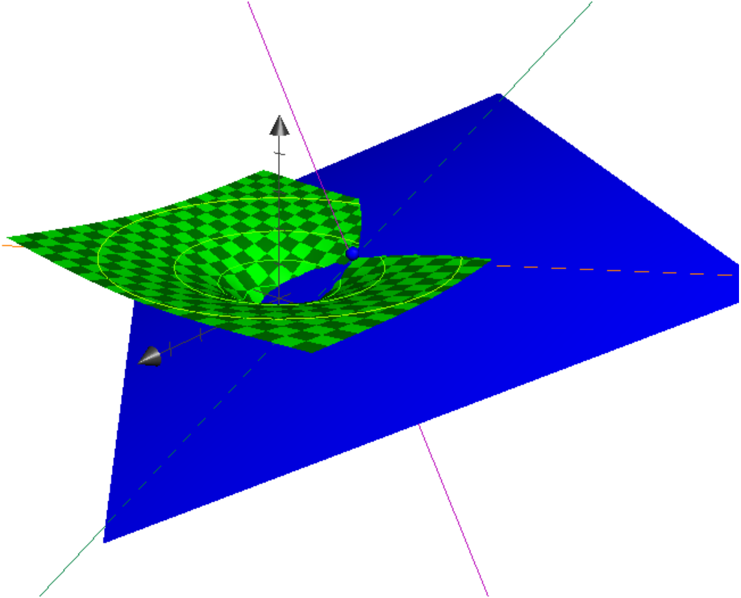
Therefore

$$\vec{\nabla} f(x,y) = (\frac{x}{x^2+y^2}, \frac{y}{x^2+y^2}) \tag{16}$$

Or, in matrix notation

$$\vec{\nabla} f(x,y) = \begin{bmatrix} \frac{x}{x^2+y^2} \\ \frac{y}{x^2+y^2} \end{bmatrix}, \vec{\nabla} f(-2,1) = \begin{bmatrix} -0.4 \\ 0.2 \end{bmatrix} \tag{17}$$

Specific example (III)



Specific example (IV)

Next, we calculate the Hessian matrix:

$$\frac{\partial^2 f}{\partial x^2} = \frac{(x^2+y^2)-x(2x)}{(x^2+y^2)^2} \tag{18}$$

$$= \frac{-x^2+y^2}{(x^2+y^2)^2} \tag{19}$$

$$\frac{\partial^2 f}{\partial y^2} = \frac{x^2-y^2}{(x^2+y^2)^2} \tag{20}$$

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial x \partial y} = \frac{-2xy}{(x^2+y^2)^2} \tag{21}$$

Therefore

$$\mathbf{H} = \begin{bmatrix} -x^2+y^2 & -2xy \\ -2xy & x^2-y^2 \end{bmatrix} \frac{1}{(x^2+y^2)^2}, \mathbf{H} = \begin{bmatrix} -0.12 & 0.16 \\ 0.16 & 0.12 \end{bmatrix} \tag{22}$$

Jacobian matrix

In the case that the function F is a vector of functions, $\vec{F} = \{f_1(\vec{x}), f_2(\vec{x}), ..., f_n(\vec{x})\}$ (i.e. a system of equations) then the gradient **also has a name: Jacobian**.

$$\mathbf{J} = \left[\frac{\partial \vec{f}}{\partial x_1} \cdots \frac{\partial \vec{f}}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{23}$$

$$\mathbf{J}_{ij} = \frac{\partial f_i}{\partial x_j} \tag{24}$$

The Jacobian matrix is important because if the function $\vec{f}$ is differentiable at a point $\vec{x}$, then $\mathbf{J}$ defines a linear map which is the best (pointwise) linear approximation of the function $\vec{f}$ near the point $\vec{x}$.

That's exactly what we want!

Newton's method for a system of equations

The following steps constitute Newton's method for simultaneous, nonlinear equations:

1. Estimate the solution vector $\vec{x}$
2. Evaluate $\vec{f}(\vec{x})$
3. Compute the Jacobian matrix $\mathbf{J}$ (J_{ij})
4. Setup the simultaneous equations $\mathbf{J}(\vec{x})\vec{\Delta x} = -\vec{f}(\vec{x})$ and solve for $\vec{x}$
5. Let $\vec{x} \leftarrow \vec{x} + \vec{\Delta x}$ and repeat steps 2-5

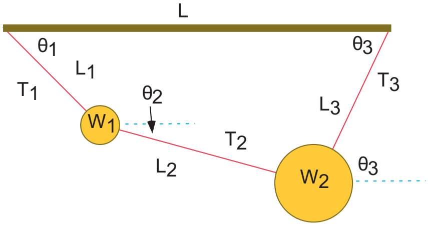
Comments on matrix computing and manipulations

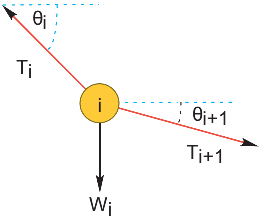
Physical systems are often modeled by systems of simultaneous equations, and it is very convenient to write these matrix form. In fact, several physical systems can be expressed this way very conveniently.

- Physical optics (lenses, mirrors, etc)
- Electromagnetic waves propagating near boundaries and in matter
- Quantum mechanical systems
- Classical mechanical dynamical systems
- etc, etc

As the models are made more realistic, the matrices often become large, and comput- ers become an excellent tool for solving such problems.

Example: weights and strings (I)





Two weights $(W_1, W_2) = (10, 20)$ are hung from three pieces of string with lengths $(L_1, L_2, L_3) = (3, 4, 4)$ and a horizontal bar of length $L = 8$.

We want to find the angles assumed by the strings and the tensions exerted by the strings.

Example: weights and strings (II)

The first 5 equations implement the geometric constraints that the horizontal length of the structure is L and that the strings begin and end at the same height.

$$\begin{aligned}
L_1 \cos\theta_1 + L_2 \cos\theta_2 + L_3 \cos\theta_3 &= L && (25)\\
L_1 \sin\theta_1 + L_2 \sin\theta_2 - L_3 \sin\theta_3 &= 0 && (26)\\
\sin^2\theta_1 + \cos^2\theta_1 &= 1 && (27)\\
\sin^2\theta_2 + \cos^2\theta_2 &= 1 && (28)\\
\sin^2\theta_3 + \cos^2\theta_3 &= 1 && (29)
\end{aligned}$$

The last three equations include trigonometric identities as independent equations because we are treating $\sin\theta$ and $\cos\theta$ as independent variables; this makes the search procedure easier to implement.

Example: weights and strings (III)

The basics physics says that since there are no accelerations, the sum of the forces in the horizontal and vertical directions must equal zero.

$$\begin{aligned}
T_1 \sin\theta_1 - T_2 \sin\theta_2 - W_1 &= 0 \qquad (30)\\
T_1 \cos\theta_1 - T_2 \cos\theta_2 &= 0 \qquad (31)\\
T_2 \sin\theta_2 + T_3 \sin\theta_3 - W_2 &= 0 \qquad (32)\\
T_2 \cos\theta_2 - T_3 \cos\theta_3 &= 0 \qquad (33)
\end{aligned}$$

Here W_i is the weight of mass i and T_i is the tension in string i. Note that since we do not have a rigid structure, we cannot assume the equilibrium of torques.

These equations represent nine simultaneous nonlinear equations. While linear equations can be solved directly, nonlinear equations cannot.

Example: weights and strings (IV)

We apply to our set the same Newton's method algorithm as used to solve a single equation by renaming the nine unknown angles and tensions as the components of our vector $\vec{x}$ and placing the variables together as a vector:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} \sin\theta_1 \\ \sin\theta_2 \\ \sin\theta_3 \\ \cos\theta_1 \\ \cos\theta_2 \\ \cos\theta_3 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

Example: weights and strings (V)

The nine equations to be solved are written in a general form with zeros on the right-hand sides and placed in a vector:

$$\vec{f}(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ f_3(\vec{x}) \\ f_4(\vec{x}) \\ f_5(\vec{x}) \\ f_6(\vec{x}) \\ f_7(\vec{x}) \\ f_8(\vec{x}) \\ f_9(\vec{x}) \end{bmatrix} = \begin{bmatrix} L_1x_4 + L_2x_5 + L_3x_6 - 8 \\ L_1x_1 + L_2x_2 - L_3x_3 \\ x_7x_1 - x_8x_2 - W_1 \\ x_7x_4 - x_8x_5 \\ x_8x_2 + x_9x_3 - W_2 \\ x_8x_5 - x_9x_6 \\ x_1^2 + x_4^2 - 1 \\ x_2^2 + x_5^2 - 1 \\ x_3^2 + x_6^2 \end{bmatrix} = \begin{bmatrix} 3x_4 + 4x_5 + 4x_6 - 8 \\ 3x_1 + 4x_2 - 4x_3 \\ x_7x_1 - x_8x_2 - 10 \\ x_7x_4 - x_8x_5 \\ x_8x_2 + x_9x_3 - 20 \\ x_8x_5 - x_9x_6 \\ x_1^2 + x_4^2 - 1 \\ x_2^2 + x_5^2 - 1 \\ x_3^2 + x_6^2 \end{bmatrix}$$

And this is then solved by computing the Jacobian and using Newton's method as usual!