Computational lattice design

Numerical methods I

Dr Robert Apsimon
r.apsimon@lancaster.ac.uk

Plan for today

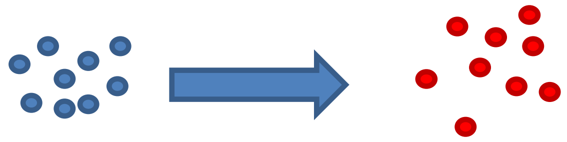
- Lecture 1:
  - Conceptual overview of numerical methods for accelerator design and simulation codes.
- Lecture 2:
  - Detailed look at the structure of tracking codes
- Lecture 3:
  - Practical tutorial to writing your own basic tracking code
    - If you don't have a laptop with you, or don't have some coding language (matlab, Python, C/C++…) then please pair up with someone.

What are computational methods used for?

- Tracking codes:
  - A generated particle distribution is tracked along a beam line with the distribution output at certain points to allow for analysis.
- Single-particle tracking is used for accurate modelling of trajectories.
  - Usually works with small step sizes or higher order integrators.
  - Computationally intensive, so only used for small number of particles.
  - Most commonly used for longitudinal dynamics where transverse effects are less important.
- Multi-particle tracking used for global exploration of phase space
  - Less computationally intensive, at cost of slight reduction in accuracy, allowing for more particles
  - Good for long-term or nonlinear effects (bunching, dynamic aperture studies…)

Single-particle tracking

multi-particle tracking



What are computational methods used for?

- Beam line design and optimisation:
  - For example, MAD, ELEGANT
  - Primarily focused on matching beam and/or lattice parameters.

$$\begin{pmatrix} x_1 \\ x_1' \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{pmatrix} \begin{pmatrix} x_0 \\ x_0' \end{pmatrix}$$

$$\begin{pmatrix} \beta_1 \\ \alpha_1 \\ \gamma_1 \end{pmatrix} = \begin{pmatrix} R_{11}^2 & -2R_{11}R_{12} & R_{12}^2 \\ -R_{11}R_{21} & R_{11}R_{22} + R_{12}R_{21} & -R_{12}R_{22} \\ R_{21}^2 & -2R_{21}R_{22} & R_{22}^2 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \alpha_0 \\ \gamma_0 \end{pmatrix}$$

What types of methods are used for accelerator codes?

- Tracking codes:
  - Numerical integrators:
    - There are many different options on integration that we will look into.
    - This is the main focus of today's lectures
- Lattice design and optimisation
  - Numerical optimisers
    - Again, many different options with pros/cons depending on application
    - Will touch on this, but won't go into too much depth

Why do we need computational methods in lattice design?

- In simple cases, we can solve Hill's Equation ($X'' + K(s)X = 0$), or use other techniques to solve or optimise analytically (e.g., transfer matrices)
  - E.g., thin-lens FODO can be completely solved analytically
  - However, this is arduous, and the complexity of the equations grows rapidly if we add more elements.

- In many cases, as we shall see, the equations cannot be solved analytically.
- We must either solve numerically, or use approximations (e.g., thin-lens)
  - Optimisation of lattice and beam parameters can be difficult even in simple cases
    - E.g., minimising the beam size in a thin-lens FODO cell
  - Nonlinear elements such as sextupoles must be solved numerically.

Example 1: trajectory through a quadrupole

- For a quadrupole, we know the magnetic field varies as:

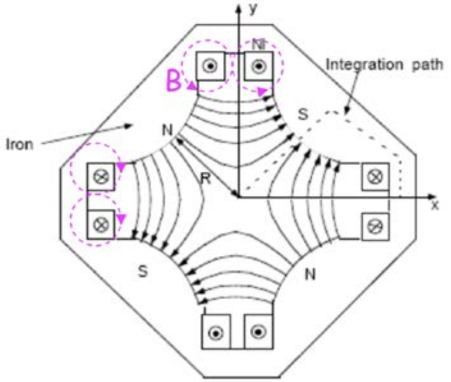
$$B_y = x\frac{dB_y}{dx}$$

  – Recall: $k = \frac{e}{p}\frac{dB_y}{dx}$

- The force on a moving charged particle due to a magnetic field is:

$$F = ev \times B$$

$$\Rightarrow F_x = ev_zB_y = v_ze\left(x\frac{dB_y}{dx}\right) = kv_zpx$$



- Writing this as a differential equation:

$$\frac{d^2x}{dt^2} = \frac{kv_zp}{\gamma m}x$$

Example 1: trajectory through a quadrupole

- Next, we want to convert from time to z position:

$$z = v_z t$$

- So our differential equation changes as:

$$\frac{d^2x}{dt^2} = \frac{kv_z p}{\gamma m}x \Rightarrow v_z^2\frac{d^2x}{dz^2} = \frac{kpv_z}{\gamma m}x$$

- Rearranging gives us:

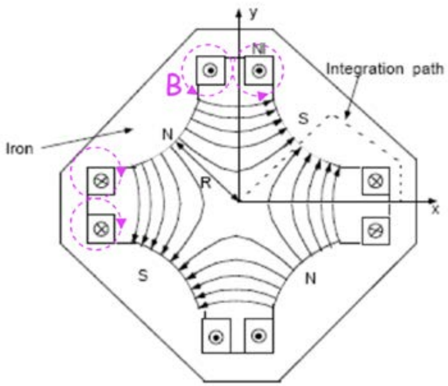
$$\frac{d^2x}{dz^2} = k\frac{p}{\gamma m v_z}x = kx$$

  – Recall: $p = \beta\gamma mc = \gamma m v_z$

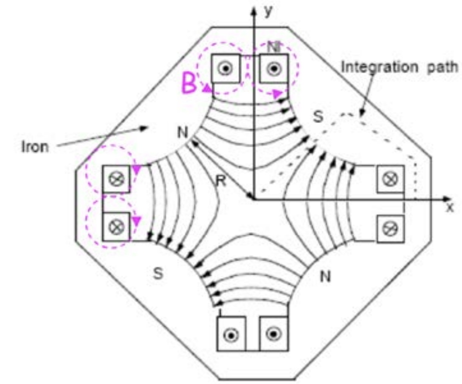
- Therefore:

$$\frac{d^2x}{dz^2} = kx$$

  – This is the Hill's equation for a quadrupole



Example 1: trajectory through a quadrupole



$$\frac{d^2x}{dz^2} = kx$$

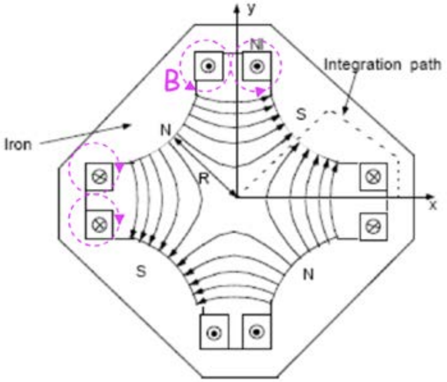
- The general solution for this is:

$$x = \begin{cases} A\cos\left(\sqrt{k}z\right) + B\sin\left(\sqrt{k}z\right), & k < 0 \\ A\cosh\left(\sqrt{k}z\right) + B\sinh\left(\sqrt{k}z\right), & k \geq 0 \end{cases}$$

Differentiating, we get:

$$x' = \begin{cases} -\sqrt{k}A\sin\left(\sqrt{k}z\right) + \sqrt{k}B\cos\left(\sqrt{k}z\right), & k < 0 \\ \sqrt{k}A\sinh\left(\sqrt{k}z\right) + \sqrt{k}B\cosh\left(\sqrt{k}z\right), & k \geq 0 \end{cases}$$

Example 1: trajectory through a quadrupole



- If we furthermore apply the initial conditions:

$$x(0) = x_0$$
$$x'(0) = x'_0$$

- We get:

$$x = \begin{cases} x_0 \cos\left(\sqrt{k}z\right) + \dfrac{x'_0}{\sqrt{k}} \sin\left(\sqrt{k}z\right), & k < 0 \\ x_0 \cosh\left(\sqrt{k}z\right) + \dfrac{x'_0}{\sqrt{k}} \sinh\left(\sqrt{k}z\right), & k \geq 0 \end{cases}$$

$$x' = \begin{cases} -x_0\sqrt{k} \sin\left(\sqrt{k}z\right) + x'_0 \cos\left(\sqrt{k}z\right), & k < 0 \\ x_0\sqrt{k} \sinh\left(\sqrt{k}z\right) + x'_0 \cosh\left(\sqrt{k}z\right), & k \geq 0 \end{cases}$$

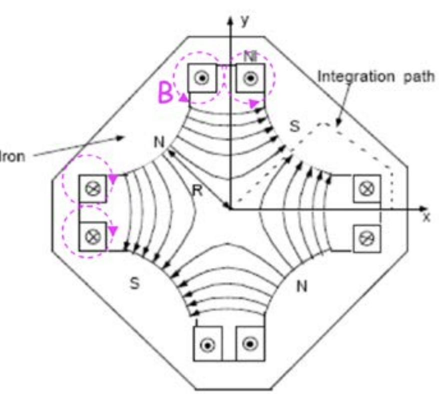
Example 1: trajectory through a quadrupole

- And we can put this all together into a familiar transfer matrix form:
- Focusing quad ($k < 0$):

$$\begin{pmatrix} x_1 \\ x_1' \end{pmatrix} = \begin{pmatrix} \cos\left(\sqrt{k}z\right) & \dfrac{\sin\left(\sqrt{k}z\right)}{\sqrt{k}} \\ -\sqrt{k}\sin\left(\sqrt{k}z\right) & \cos\left(\sqrt{k}z\right) \end{pmatrix} \begin{pmatrix} x_0 \\ x_0' \end{pmatrix}$$

- Defocusing quad ($k > 0$):

$$\begin{pmatrix} x_1 \\ x_1' \end{pmatrix} = \begin{pmatrix} \cosh\left(\sqrt{k}z\right) & \dfrac{\sinh\left(\sqrt{k}z\right)}{\sqrt{k}} \\ \sqrt{k}\sinh\left(\sqrt{k}z\right) & \cosh\left(\sqrt{k}z\right) \end{pmatrix} \begin{pmatrix} x_0 \\ x_0' \end{pmatrix}$$



Example 2: trajectory through a sextupole

- What about a sextupole? In this case, we can write the magnetic field due to a transverse offset as:

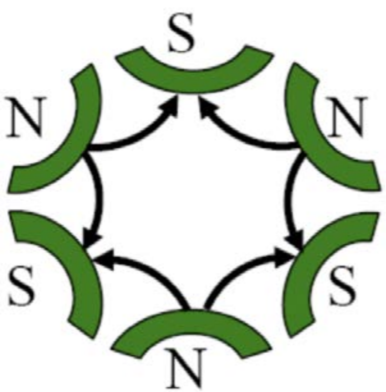
$$B_y = \frac{k_3 p}{e} x^2$$

- From the force and converting $t \rightarrow z$, we get the differential equation:

$$\frac{d^2 x}{dz^2} = k_3 x^2$$

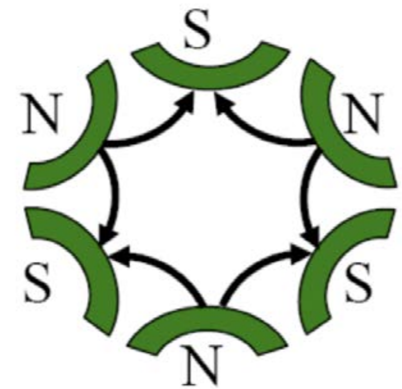
- To solve this, we will use a different approach and assume the solution is:

$$x = \sum_{n=0}^{\infty} a_n z^n$$



Example 2: trajectory through a sextupole

$$x = \sum_{n=0}^{\infty} a_n z^n \rightarrow \frac{d^2x}{dz^2} = k_3 x^2$$



- Plugging our assumed solution into our equation, we get:

$$\sum_{n=2}^{\infty} (n-1) n a_n z^{(n-2)} = k_3 \left(\sum_{n=0}^{\infty} a_n z^n \right)^2$$

- Writing out the first few terms on each side, we get:
$2a_2 + 6a_3 z + 12a_4 z^2 + \cdots = k_3 a_0^2 + 2k_3 a_0 a_1 z + k_3\left(2a_0 a_2 + a_1^2\right) z^2 + \cdots$

- Now we can solve coefficients by comparing like terms:
$a_2 = \frac{k_3 a_0^2}{2}$; $a_3 = \frac{k_3 a_0 a_1}{3}$; $a_4 = \frac{k_3^2 a_0^3 + a_1^2}{12}$

Example 2: trajectory through a sextupole

- Applying the same initial conditions as before, we find that our solution is:

$$x = k_3 x_0^2 + 2k_3 x_0 x_0' z + \left(k_3^2 x_0^3 + k_3 {x_0'}^2\right) z^2 + \cdots$$

$$x' = 2k_3 x_0 x_0' + 2\left(k_3^2 x_0^3 + k_3 {x_0'}^2\right) z + \cdots$$

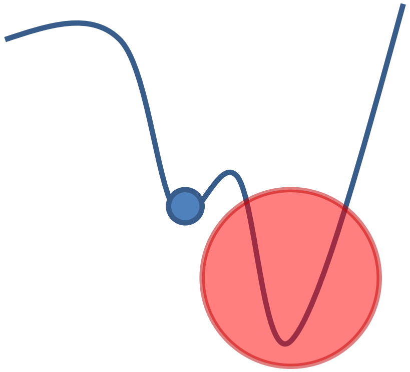
- Key points about this:
  - For sextupoles and other nonlinear elements, there is no closed form solution for the trajectory.
  - The trajectory has a nonlinear dependence on initial conditions and sextupole strength, making them difficult to model analytically.
  - By assuming a series expansion solution, we can solve similar problems like this to any order.
    - But need to note this will always be an approximation.

Lattice design & optimisation: numerical optimisers

- As mentioned at the beginning of the lecture, lattice design and optimisation is focused on matching beam and lattice parameters.
  - Require numerical optimisers to do this, but what type?

- Global optimisers
  - Good at finding globally optimal region of parameter space
  - Generally poor (or slow) at converging to the globally optimal solution.

- Local optimisers
  - Will rapidly converge to an optimal solution
  - Prone to getting stuck in a local minimum

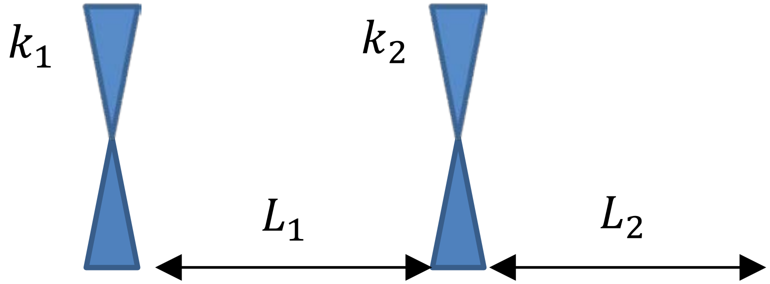


- Quite common to use a global optimiser then a local one to find solutions

Lattice design and optimisation

- As with integration, optimisation problems can get very complicated very quickly!
  - E.g., Consider a beam line of 2 quads and 2 drift lengths, where we want to define the quad strengths and drift lengths in order to match the initial and final beam parameters:
  - Unknowns: L_1, L_2, k_1, k_2
  - Knowns: Initial and final $\beta_x, \beta_y, \alpha_x, \alpha_y$

- This seems like quite an easy problem until you look at the equations to solve…



Constraint equations for a 2-quad matching cell

$$
\begin{aligned}
\beta_x^1 = {} & \Biggl(\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sin(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right)\Biggr)^2 \beta_x^0 \\
& - 2\Biggl(\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sin(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right)\Biggr)\Biggl(\cos(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right) \\
& + \left(\frac{\sin(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cos(\sqrt{k_1}l_q)\right)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr)\Biggr)\alpha_x^0 + \Biggl(\cos(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right) + \left(\frac{\sin(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cos(\sqrt{k_1}l_q)\right)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr)\Biggr)^2 \frac{1+(\alpha_x^0)^2}{\beta_x^0}
\end{aligned}
$$

$$
\begin{aligned}
\beta_y^1 = {} & \Biggl(\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sinh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right)\Biggr)^2 \beta_y^0 \\
& - 2\Biggl(\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sinh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right)\Biggr)\Biggl(\cosh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right) \\
& + \left(\frac{\sinh(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cosh(\sqrt{k_1}l_q)\right)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr)\Biggr)\alpha_y^0 \\
& + \Biggl(\cosh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right) + \left(\frac{\sinh(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cosh(\sqrt{k_1}l_q)\right)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr)\Biggr)^2 \frac{1+(\alpha_y^0)^2}{\beta_y^0}
\end{aligned}
$$

$$
\begin{aligned}
\alpha_x^1 = {} & \Biggl(\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sin(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right)\Biggr)\Bigl(\sqrt{k_1}\sin(\sqrt{k_1}l_q)\cos(\sqrt{k_2}l_q) + \sqrt{k_2}\sin(\sqrt{k_2}l_q)\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\Bigr)\beta_x^0 \\
& + \Biggl(\Biggl(\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sin(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right)\Biggr)\Biggl(\cos(\sqrt{k_1}l_q)\cos(\sqrt{k_2}l_q) - \sqrt{k_2}\sin(\sqrt{k_2}l_q)\left(\frac{\sin(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cos(\sqrt{k_1}l_q)\right)\Biggr) \\
& - \Biggl(\cos(\sqrt{k_1}l_q)\left(\frac{\sin(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cos(\sqrt{k_2}l_q)\right) + \left(\frac{\sin(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cos(\sqrt{k_1}l_q)\right)\bigl(\cos(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sin(\sqrt{k_2}l_q)\bigr)\Biggr)\Bigl(\sqrt{k_1}\sin(\sqrt{k_1}l_q)\cos(\sqrt{k_2}l_q) + \sqrt{k_2}\sin(\sqrt{k_2}l_q)\bigl(\cos(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sin(\sqrt{k_1}l_q)\bigr)\Bigr)\Biggr)\alpha_x^0 \\
& + \Biggl(\cos(\sqrt{k_1}l_q)\cos(\sqrt{k_2}l_q) - \sqrt{k_2}\sin(\sqrt{k_2}l_q)\left(\frac{\sin(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cos(\sqrt{k_1}l_q)\right)\Biggr)^2 \frac{1+(\alpha_x^0)^2}{\beta_x^0}
\end{aligned}
$$

$$
\begin{aligned}
\alpha_y^1 = {} & \Biggl(\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sinh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right)\Biggr)\Bigl(\sqrt{k_1}\sinh(\sqrt{k_1}l_q)\cosh(\sqrt{k_2}l_q) + \sqrt{k_2}\sinh(\sqrt{k_2}l_q)\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\Bigr)\beta_y^0 \\
& + \Biggl(\Biggl(\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr) - \sqrt{k_1}\sinh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right)\Biggr)\Biggl(\cosh(\sqrt{k_1}l_q)\cosh(\sqrt{k_2}l_q) \\
& - \sqrt{k_2}\sinh(\sqrt{k_2}l_q)\left(\frac{\sinh(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cosh(\sqrt{k_1}l_q)\right)\Biggr) \\
& - \Biggl(\cosh(\sqrt{k_1}l_q)\left(\frac{\sinh(\sqrt{k_2}l_q)}{\sqrt{k_2}} + L_2\cosh(\sqrt{k_2}l_q)\right) + \left(\frac{\sinh(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cosh(\sqrt{k_1}l_q)\right)\bigl(\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}L_2\sinh(\sqrt{k_2}l_q)\bigr)\Biggr)\Bigl(\sqrt{k_1}\sinh(\sqrt{k_1}l_q)\cosh(\sqrt{k_2}l_q) \\
& + \sqrt{k_2}\sinh(\sqrt{k_2}l_q)\bigl(\cosh(\sqrt{k_1}l_q) - \sqrt{k_1}L_1\sinh(\sqrt{k_1}l_q)\bigr)\Bigr)\Biggr)\alpha_y^0 + \Biggl(\cosh(\sqrt{k_1}l_q)\cosh(\sqrt{k_2}l_q) - \sqrt{k_2}\sinh(\sqrt{k_2}l_q)\left(\frac{\sinh(\sqrt{k_1}l_q)}{\sqrt{k_1}} + L_1\cosh(\sqrt{k_1}l_q)\right)\Biggr)^2 \frac{1+(\alpha_y^0)^2}{\beta_y^0}
\end{aligned}
$$

- Not analytically solvable
- Parameter space riddled with local minima
- Even numerical optimisers struggle with this

My preferred approach to solving difficult optimisation problems

1. Simplify the problem to a thin-lens approximation
   - This reduces the constraints to polynomials, so we know there will only be a finite number of solutions (at most)
2. Solve this simplified problem numerically
   - Numerical optimisers will spit out a list of solutions
   - If there are no solutions, it’s likely that there are no solutions in the thick-lens case (though not certain…)
3. Discard unphysical solutions from your list and select the “best” solution
   - Discard cases with negative lengths or complex values
   - “best” solution is usually obvious, such as smallest length, or lowest magnet strengths etc.
4. Use your “best” solution as the starting point for the thick-lens problem
   - Much more likely to find the best solution without being stuck in local minima.