Portfolio Optimisation

What is a portfolio?
A collection of investments held by some body (investment company, hedge fund, an individual etc.)

Simplified Problem.
In our simplified problem we just wish to minimise the risk of the portfolio (variance of portfolio) w.r.t a budget constraint.

Measuring Risk in Noisy Estimates.
The toy example is as follows: we pretend we know the true covariance matrix of returns. We sample N time series of length T according to some "true" covariance matrix and then estimate a covariance matrix from this time series data. $\sigma^{(est)}$ is computed based on this time series data so it and q_0 are random variables.
$q_0 \geq 1$.

The reason for the estimated covariance matrix becoming singular is due to its construction, involving XX^T, an $N \times N$ matrix and $rank(X) = \min(N, T)$.
In fact it is the expectation of q_0 that satisfies the $\frac{1}{\sqrt{1-r}}$ relation. The est error diverges as $r \to 1$.

Experimental Framework.
This framework is advantageous over empirical studies is that we know the true covariance matrix.

In-built Solvers.
We were just considering a simple quadratic optimisation problem with a closed solution. In general this is not easy to solve for so you'll have to use some optimisation solver. These could either be a general quadratic solver for our situation, some more general function minimising solver, etc.

We know that there should be nothing allowed to happen beyond $r = 1$, but many solvers do allow this.

BLACKBOX. When people blindly follow solutions from such methods disastrous conclusions can be drawn - in this case the graphs seem to suggest that after $N = T$ if we dramatically increase N our risk actually diminishes, counter intuitive and dangerous.

Many fields wish to minimise/maximise such problems. Most of them will not be statistically astute or even when they are the problem could arise in a subtle manner.

Why?
Though it appears that the solvers themselves have something in the code to manually change the problem so it can be solved.
There are examples of solvers that do not do it: cvxopt in Python, quadprog in R, qpsolve in Scilab.

How?
Even the documentation does not warn this sort of behaviour and most in-built convergence checkers also provide incorrect information.
It's worth also mentioning that Mathematica seems to do something even stranger than these two methods
Regularisation is the main one.

Regularisation.
When models are measuring too many variables (e.g. trying to predict cancer survival rates and using silly things like hair colour, etc.) they become too sensitive to input data. To counteract this you penalise the measurement of too many variables by adding a multiple of the norm of your parameters. In our case I just used to 2-norm as it is easier to deal with for differentiability reasons. η has to be chosen for each model, a popular technique to do this is with cross validation.
η was chose randomly here, no cross validation or anything. Just to get a feel for what a regulariser does in this case.
In general regularisation pushes solutions towards being uniform.

Broader Context
Non-reproducibility is a problem in science of replicating results. This issue is compounded when software also acts against you.
I particularly looked into dimensionality towards the end of my project - random problems often have solutions dependent on dimension alone.
Even if you are statistically astute - unless you have a good knowledge of your system - you may not even notice such issues.