ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Международный институт экономики и финансов

Дергачев Кирилл Олегович

СРАВНЕНИЕ АЛГОРИТМОВ ОПТИМИЗАЦИИ ДЛЯ ОЦЕНКИ КРИВОЙ ДОХОДНОСТИ (COMPARISON OF OPTIMIZATION ALGORITHMS IN TERM STRUCTURE ESTIMATION)

Выпускная квалификационная работа - БАКАЛАВРСКАЯ РАБОТА
по направлению подготовки 38.03.01 «Экономика»
образовательная программа «**Программа двух дипломов по экономике НИУ ВШЭ и Лондонского университета**»

Рецензент
д-р …. наук, проф.
_____
И.О. Фамилия

Научный руководитель
к.ф.-м.н.
Лапшин В.А
И.О. Фамилия

Москва 2021

Contents

1 Introduction **3**

2 Academic literature analysis **5**

2.1 Term structure and its uses 5

2.2 Estimation methods 5

2.3 Fitting the models 7

2.3.1 Past works 7

2.3.2 Algorithms 8

3 Model and framework **11**

3.1 Bond math 11

3.2 NS, NSS models 13

3.3 Methodology 15

3.3.1 Data 15

3.3.2 Optimization 18

3.3.3 Performance measures 24

4 Empirical tests **26**

4.1 In sample results 26

4.2 Out of sample results 26

4.3 Overall 27

4.4 Additions 27

4.5 Comparison 28

5 Conclusion **32**

Appendix A (Algorithms) **37**

Appendix B (Tables) **42**

Abstract

Fitting the term structure is associated with multiple possible issues irrespective of the method used: some restrict the problem too much, some are far from being theoretically based and some involve nonlinear optimization with no guarantee of returning a global optimum and possibility of taking a long time to compute without certainty about quality of results. In this paper I use various algorithms, starting points generation methods and two stopping rules to find optimal parameters of Nelson-Siegel and Nelson-Siegel-Svensson models and compare performance of models and algorithms in terms of accuracy and optimization time. In short the results are that for chosen algorithms starting points choice mostly doesn't matter though one of the used method used a good guess, additionally all algorithms perform with similar accuracy and a good stopping rule greatly increases speed by sacrificing accuracy a little.

1 Introduction

Term structure of interest rates is the relationship between interest rates or yields and respective maturities or tenors. Some overview can be seen in (Shiller and McCulloch 1990). Term structure can be applied to different types of debt instruments and generally the securities for each term structure model have to be of similar risk so that risk premia differences are not incorporated in the discount factor analysis but often by term structure one means the one derived from treasury debt instruments showing "risk-free" rates. Interst rates can be both indicators of state of the economy and one of the tools used to control it. Because of this term structure is widely used by practitioners academics and regulators in their works.

Term structure comes from bond cash flows discounting and this imposes some restrictions on its estimation. For instance there should be some long term rate to which interest rates tend as tenor goes to infinity. These properties of term structure make use of simple techniques such as using a polynomial regression over whole data's domain not viable. Initially polynomial spline models were used. These divided domain into multiple subsets and estimated polynomial models in these subsets ensuring the combined function over the whole domain is differentiable. Then it was proposed to use exponential splines as they had

properties better for term structure estimation. Then a model was proposed based on solution of a difference equation which supposedly was a good assumption about the term structure. The latter has proven to be a good model of the term structure, but it is more difficult to fit it given empirical data due to non-linearity. It is clear that in estimating model one expects to get the ”true” parameters - the ones that are implied by data available thus the ones that fit data the best and it is beneficial to obtain them as fast as possible in most cases. This becomes a problem with nonlinear methods as solutions based on calculus are not viable, and recently the nonlinear models such as Nelson-Siegel are becoming more widely used in practice which calls for investigation of optimization techniques and algorithms to improve the procedure.

In this paper I use the extended Nelson-Siegel-Svensson model with restrictions on coefficients that avoid multicollinearity problem with different stopping strategies and starting point generation. By doing this I aim to investigate what algorithm and starting point generation technique is suited better for such a problem and additionally see if one can implement a rule that speeds up the optimization without sacrificing much accuracy. The results should be helpful to improve the procedure of estimating the term structure which is widely used as will be discussed below and parts of the methodology if might even be of use in some other optimization problems that frequently arise in economics if they prove to be working for chosen problem well enough.

2 Academic literature analysis

2.1 Term structure and its uses

Term structure has many various practical applications. It's shape is often considered to be a good predictor of future economic growth. Positive slope of term structure often means increase in levels of economic activity and negative might mean that the economy is soon to enter recession. More precisely it is shown to signal future changes in investment and consumption as well as being related to multiple indicators. Moreover it often outperforms various other forecasts. The main part of explanation for that is that term structure incorporates expectations of forward rates and these are related to changes in GDP. (Estrella and Hardouvelis 1991) and (Plosser and Geert Rouwenhorst 1994) show that term structure is a useful tool in predicting economic growth and that even foreign term structures can include some information about domestic economic activity. (Fama and Robert R Bliss 1987) and (Mishkin 1990) show that information about future path of inflation and short term interest rates can be extracted from yield curve. (Stock and Watson 1989) provide evidence that slope of the yield curve is a good leading indicator.
With all that term structure is an important instrument widely applied in multiple circumstances and it's predictive power is especially useful in central bank's operation. (Estrella and Mishkin 1997) and (Csajbok et al. 1998) research how it can be useful for monetary policymaking and how relevant information can be extracted from it. Additionally in (Bank For International Settlements 2005) it can be seen that most banks use it to some extent in decision making and evaluating policies' effects. These are some of the uses for term structure and as they need the shape of it investigating what functional forms work better and how to improve the process of estimating it is beneficial.

2.2 Estimation methods

Initially when term structure was estimated yields on individual bonds were used rather than interest rates as they can be easily calculated and then the problem is fitting a function using yield-tenor sample while interest rates cannot be directly calculated from bond prices and coupons resulting in complexity in solving

possible optimization problems as discussed later.

The first attempts at displaying term structure relationship simply involved placing an arbitrary smooth curve through points on tenor-yield plane such as in (Durand 1942). This method is clearly not robust enough as no statistical model is used to derive the formulaic relationship but it was one of the few methods available at the time and one can still see that the author even managed to capture common term structure shapes known now. Then as term structure research gained more interest and availability of using computers for optimization more robust estimation methods appeared. The second most prominent way used to capture the relationship between maturity and yield is regression splines as used in (McCulloch 1971) or more recent (Fernández-Rodrıguez 2006) where the domain is split into subsamples on which a polynomial model is fitted, usually order two or three of the polynomial is chosen. In (Vasicek and Fong 1982) exponential splines were tried as well. These models allow for fitting of various shapes of yield curve quite well but at the same time do not require polynomials or other functions of high order thus resulting in good fit and lower risk of overfitting, also as stated in these papers exponential splines take shapes more suitable for term structure than polynomial ones. Moreover if yields are used in the procedure the estimation problem is linear and computers of the time, which were not good at nonlinear optimization, were capable of solving it in reasonable time. The spline methods mentioned provide good results and are still used nowadays, but there are still some unfavorable properties of them.

This and need for a more parsimonious model to use in term structure estimation resulted in Nelson-Siegel model (NS) (Nelson and Siegel 1987). They point out that most models used previously were not designed for term structure estimation and because of that have properties unlikely for it, such as not tending to some long term interest rate.Their idea was to get the model as a solution to a differential equation that follows from expectations theory of interest rates. The model derived is quite parsimonious, having five parameters and applicable over the whole domain, but at the same time allows for various shapes often observed in the term structure. With some restrictions on parameters the model also has properties expected from term structure and the summands can be interpreted as short term, medium term and long term interest rates. Later in (Svensson 1994) a fourth term was added, same as the hump component but

with different parameter to allow for greater flexibility. This model (NSS for later use) is not as theoretically justified as Nelson-Siegel one, but it allows to capture some irregularities in data and does not add any unwanted properties and it encompasses the NS model so if there is no second hump in the data one could expect the coefficient to be close to zero resulting in the same model as NS.
Some empirical results show that listed models are each one an improvement over previous one with exponential spline models doing moderately well and parsimonious Nelson-Siegel(-Svensson) perforimng better than the other ones. One example of such tests is (Ioannides 2003), some other models are tested and mentioned but it seems that the NS(S) form are better suited for the task.
Additionally it is worth noting that there is still research into what model would work best and though these are outside the scope of this paper it is worth mentioning. For instance in (Diebold and Li 2006) to forecast yields the authors assume that factors follow an AR(1) process and in (Christensen, Diebold, and Rudebusch 2011) derive an arbitrage free model that approximates Nelson-Siegel-Svensson and show that it performs well, improves predictive performance and as was intended initially is arbitrage-free.

2.3 Fitting the models

2.3.1 Past works

When it comes to the latter Nelson-Siegel(-Svensson) models fitting becomes non-trivial. Initially the NS was applied to zero coupon rates as in the original work. This allowed the estimation to be carried out at the time but at cost of reducing quality of the model. One of the examples of restricting the model to allow for easier fitting is in (Diebold and Li 2006) where the authors decided to avoid the problem of nonlinear optimization by first equating the decay parameters of the NS model and fitting a linear model with them set as $1/0.0609$ which is the value that maximizes the loading on the medium term factor at 30 months. Another part of the problem where the researcher has a choice is the cost function. Generally one punishes for deviations of estimated values from observed ones, but it can be tweaked to suit the problem better. For the term structure model as shown in (Robert Russell Bliss 1997) down weighing longer maturity/duration bond errors provide better out of sample fit. While

modern computers are capable of fitting it in reasonable time with rare failures to converge improving on the process is still helpful. So more recently more research started focusing on finding a way to improve upon the optimization presented by these models. Among such research is (Ahi, Akgiray, and Sener 2018) which shows that choice of optimizing algorithm greatly affects the results of optimization - namely that single point methods do not find global optima and rather get trapped in local optima and that algorithms like PSO and especially their own modification of it - Hybrid PSO perform better on usual data and on perturbed one. The results confirm similar findings from (Manousopoulos and Michalopoulos 2009) in which multiple algorithms were tested applied to the similar problem of estimating term structure with Nelson-Siegel-Svensson model. In that paper it was also shown that gradient based algorithms perform worse than other classes of optimization algorithms.

2.3.2 Algorithms

For this paper I used 6 algorithms to estimate term structure. These are: L-BFGS-B, Dual Annealing, Differential Evolution, Particle Swarm Optimization, COBYLA, Nelder-Mead[1].

L-BFGS-B (Liu and Nocedal 1989) is a modification of BFGS (Fletcher 1970). Original method uses a second degree Taylor approximation to the objective function to find a search direction and then using an appropriate step size moves the point to the new position, supposedly a better one. The modification limits the amount of past point position changes and gradient changes to decrease load on memory. This a reduced version of Hessian approximation used in the original method as full version requires keeping all of these values and using them each iteration. Additionally the "-B" version bounds the algorithm to provided region. Generally single point algorithms such as BFGS are bad at ill behaved functions as they usually when they find a local minimum they stop thus not exploring the whole domain in search of a better solution. Multiple initialization from different starting points might help solve the problem.

Dual Annealing (Tsallis and Stariolo 1996) is a generalized version of Simulated Annealing (Černý 1985) which was inspired by annealing process in metallurgy. The idea is to move generated point around and using a probability depending

[1] Simplified procedures for these algorithms and specifics of implementation are described in appendix section

on function values and temperature parameter accept that new position or keep the previous one. Visiting distribution depends on temperature parameter as well so that lower temperature means new candidates will be generally closer to the current position and probability that the point stays at the new position decreases. Over iterations temperature is decreased resulting is equilibrium where said point is at or close to global optimum. Dual Annealing uses a form of visiting distribution that generalises the two most common visiting distribution choices - Boltzmann Machine of Classical Simulated Annealing and Cauchy Machine of Fast Simulated Annealing. By changing parameters it can replicate both of them and other variations and is shown to do so faster than the other two mentioned.
Differential Evolution introduced in (Storn and Price 1997) is an optimization method that optimizes by iteratively changing components of points based on other points in population and objective function of these points. Changes are somewhat heuristic and generally imply some probabilistic exchange of vector components i.e. used in this paper best2bin variation that combines the best points of the population and two randomly selected points with some weight and randomly assigns some of components of the new vector to a chosen point. In the original paper through testing it in different applications it was shown that it was faster than most algorithms at the time, often found global optima of the problems. Moreover it requires few parameters and is doable in parallel.
Particle Swarm Optimization (Kennedy and Eberhart 1995) sets up a swarm of particles which with each iteration move in some direction mostly towards the best known positions. The most basic version of it defines particle velocity as weighted average of its past velocity, distance to own best position and distance to swarm's best position with some uniform variables scaling some of the components up or down. In other works it was shown to perform well, but I did not manage to replicate such results, quite possibly due to sub-optimal parameters or an error in implementation.
COBYLA - Constrained Optimization BY Linear Approximations (Powell 1994) was initially designed to be an algorithm that can accept constraints to the problem without any change in cost/energy function. Overall procedure is complex but the idea is to construct a simplex in the search space and solve problem using linear approximations inside some given trust region, sometimes changing the simplex to found solution, sometimes changing trust region or

tolerance to bound violation or instead of searching for the optimal solution tweak the simplex to be a better one for further search. This algorithm is less widely used in practice, but as will be seen later it performs quite well in the given problem.

Nelder-Mead (Nelder and Mead 1965) is an algorithm that uses a simplex of the search space to find optima. The idea is to rank points of the simplex by their objective function values and trying to mirror the worst around the center of opposite face in different ways or shrinking the simplex. The best point is never moved and only replaced by the better point so that these actions cannot decrease quality of the solution. Its benefits are that it has to evaluate 3 points of the simplex and the new points each iteration and does not require derivative evaluations or approximations making it quite efficient.

3 Model and framework

3.1 Bond math

The main idea in the underlying cost function for the optimization is cash flow discounting. Price of a bond can be determined as a sum of cash flows discounted by respective interest rates. The rates are affected by multiple factors but generally one assumes some mapping from term to interest rates, that is term structure for some given level of risk that the bond bears. So when risk is fixed $r(t)$ is used to discount $c(t)$ where $r(t)$ is the rate, $c(t)$ is the payment for the same time (note that $c(T)$ will include principal and coupon):

$$P = \sum_{t=1}^{T} \frac{c(t)}{(1+r(t))^t} \tag{1}$$

In some instances continuous discounting is used where change in time between payments is assumed to approach 0:

$$P = \int_0^T c(t)e^{-r(t)t}dt \tag{2}$$

As $e^{-r(t)t} \approx 1/(1+r(t))^t$ for small $r(t)$ the discount factors are used interchangeably. In cases where exponentiation is used the rate is considered to be log-return and where fractions are used the return is arithmetic one:

$$r_{normal}(t) = \frac{P_t}{P_{t-1}} - 1 \tag{3}$$

$$r_{log}(t) = \log(\frac{P_t}{P_{t-1}}) \tag{4}$$

For term structure estimation coupons and prices are observed and shape of $r(t)$ needs to be found. Thus one assumes some $r(t,\Theta)$ relationship where Θ is a vector of parameters of the $r(t)$ function. This turns previous equations into:

$$\hat{P}_i(\Theta) = \sum_{t=1}^{T} \frac{c_i(t)}{(1+r(t|\Theta))^t} \tag{5}$$

$$\hat{P}_i(\Theta) = \int_0^T c(t)e^{-r(t|\Theta)t}dt \tag{6}$$

In addition there are some other choices for the variable in optimization. For example yield can be chosen rather than price. Yield to maturity is total or average return anticipated throughout the bonds life. Intuitively it is the interest rate for the bond if term structure was flat:

$$P_i = \sum_{t=1}^{T} \frac{c_i(t)}{(1+y_i)^t} \tag{7}$$

Here y_i is the yield associated with bond i. When yield is used instead of interest rates the tenor of calculated yield is time to maturity of bond it is associated with. If the sample of bonds is big enough yields against respective tenors should be similar to interest rates, but not exactly - with horizontal term structure they will coincide, if term structure is an increasing function yields will underestimate it and vice-versa.

Where indices are added as there is a whole sample of bonds for which $r(t)$ is assumed the same. From this sample and given functional forms one finds $\hat{r}(t) = r(t|\hat{\Theta})$. Doing so through prices involves non-linear optimization thus minimum cannot be strictly found and one has to resort to using non-linear optimization techniques which might have convergence problems, get stuck in local minima rather than global one or take a long time to find a result.

Additionally an important part of term structure analysis are forward rates. Forward rate relationship $f(t,t+\varepsilon)$ shows rates from period t to the period $t+\varepsilon$ and when this interval approaches 0 one gets instantaneous forward rate functon:

$$\lim_{\varepsilon \to 0} f(t,t+\varepsilon) = \tilde{f}(t) \tag{8}$$

From this function other non-instantaneous forward and interest rates can be found:

$$f(t_0,t_1) = \frac{1}{t_1 - t_0} \int_{t_0}^{t_1} \tilde{f}(\tau) d\tau \tag{9}$$

If one plugs $t_0 = 0$ this becomes the term structure relationship. Going back from interest rates to forward rates which are now implied forward rates as they are calculated from term structure rather than directly simply involves finding the difference in earnings from current period to forward end date to forward start date:

$$f(t_0,t_1) = \frac{t_1 r(t_1) - t_0 r(t_0)}{t_1 - t_0} \tag{10}$$

3.2 NS, NSS models

The functional forms this paper focuses on are the ones used in (Nelson and Siegel 1987) and (Svensson 1994). The NS model was derived as a solution to a differential equation and is of the form:

$$r_{NS}(t) = \beta_0 + \beta_1 g(t|\tau_1) + \beta_2 h(t|\tau_2) \tag{11}$$

To save space $g(t|\tau)$ and $h(t|\tau)$ are defined as:

$$g(t|\tau) = \frac{1-\exp(-t/\tau)}{t/\tau} \tag{12}$$

$$h(t|\tau) = \frac{1-\exp(-t/\tau)}{t/\tau} - \exp(-t/\tau) \tag{13}$$

In the same notation as used in previous section the function is $r(t|\Theta)$, $\Theta = \{\beta_0, \beta_1, \tau_1, \beta_2, \tau_2\}$

$$r_{NSS}(t) = \beta_0 + \beta_1 g(t|\tau_1) + \beta_2 h(t|\tau_2) + \beta_3 h(t|\tau_3) \tag{14}$$

Here $\Theta = \{\beta_0, \beta_1, \tau_1, \beta_2, \tau_2, \beta_3, \tau_3\}$
Talking here about the constraints used in the optimization all taus are defined on the set $(0, \infty)$ (though unlikely to be into hundreds) and from economic interpretation β_0 is the long term rate - so it is unlikely to be below 0 or significantly higher than 1 if the model's theoretical basis is adequate. Additionally from the assumption that interest rates cannot be below 0 a constraint $\beta_0 + \beta_1 \geq 0$ can be formulated.
As can be seen on figures (2) and (1) these five parameters allow for various shapes of the term structure and a capable of producing the common ones as outlined before. Often a restriction $\tau_1 = \tau_2$ is imposed, especially in NSS version as often a multicollinearity problem arises from a completely free estimation. In NSS model then $\tau_3 = \tau_1 + 1$ for the same reasons and $\beta_3 = \beta_2 + 1$. So the models most usually fitted are of the form:

$$r_{NSr}(t) = \beta_0 + \beta_1 g(t|\tau) + \beta_2 h(t|\tau) \tag{15}$$

$$r_{NSSr}(t) = \beta_0 + \beta_1 g(t|\tau) + \beta_2 h(t|\tau) + (\beta_2 + 1) h(t|\tau + 1) \tag{16}$$

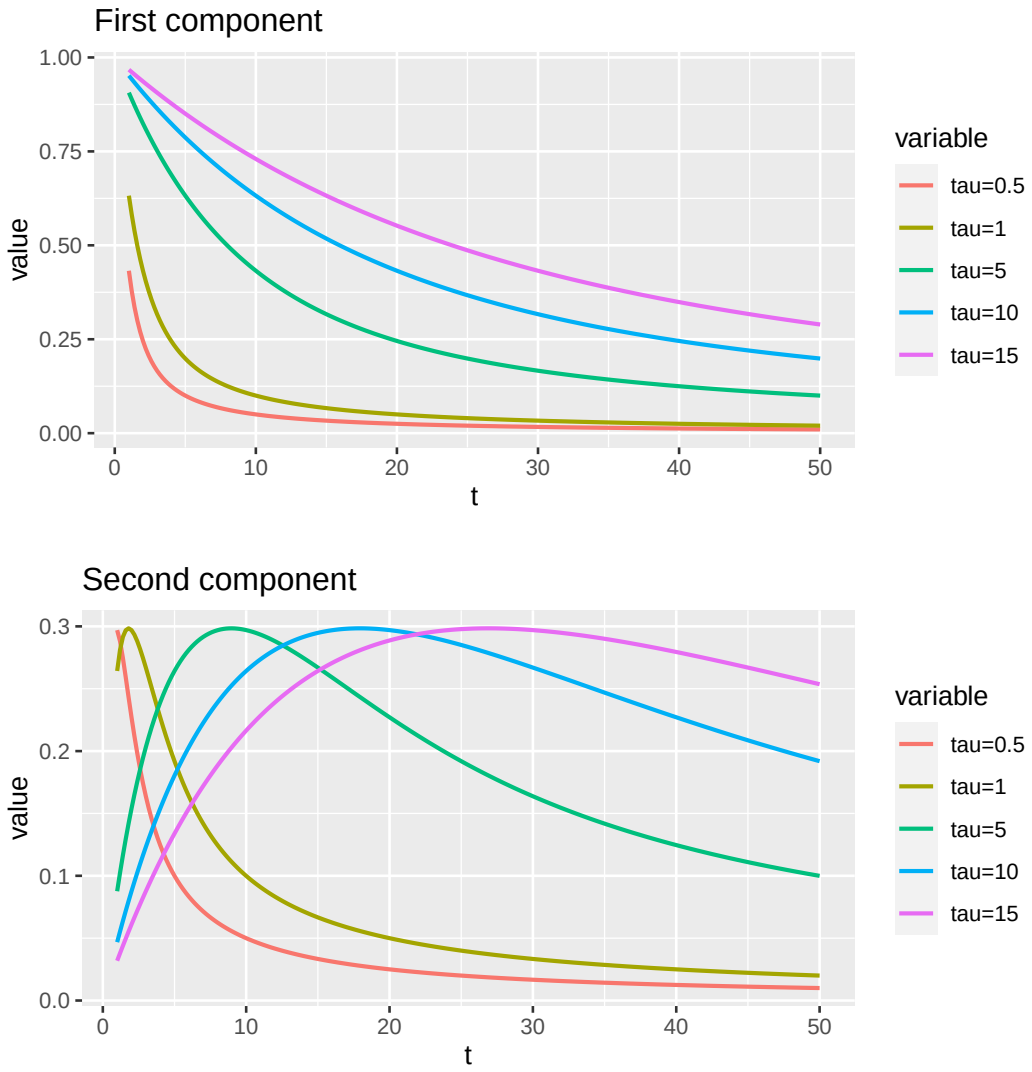

Figure 1: $g(t|tau)$ and $h(t|tau)$ components of NS and NSS for different values of tau

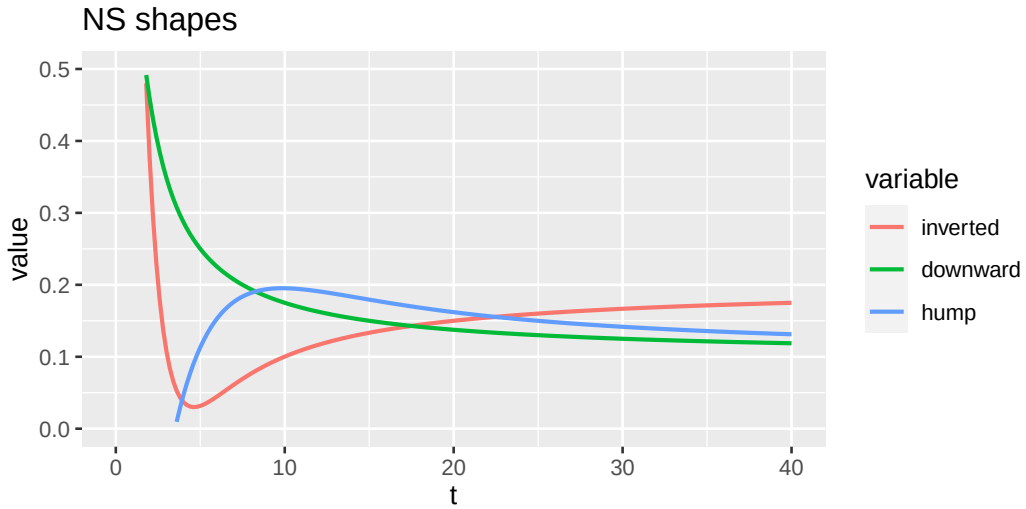

Figure 2: Some of the shapes NS functional form can take

$$\Theta_{NSr} = \Theta_{NSSr} = \{\beta_0, \beta_1, \tau, \beta_2\}$$

3.3 Methodology

3.3.1 Data

Generally to construct cost function for term structure estimation one needs a sample of data for bonds - in this paper US treasury notes, bonds and bills, over given days. To calculate estimated price given term structure function one needs coupon payments data and when these coupons are paid out. Additionally having price, maturity date and date the data was sampled is useful for fitting models and other analysis such as calculating yields is helpful.

For my work data was obtained using Bloomberg terminal. There using SRCH function for a given day I chose Asset Classes: Government, Sources: All Securities, Country/Region of Incorporation: United States of America, Ticker: (T, B). This SRCH parameters capture all traded US treasury securities for that day excluding TIPS including which would have been complicated due to price adjustments with inflation. For these securities data fields added to standard ones are Ask Price, Bid Price, Maturity, Coupon, Minimum Piece, Issue Date, Par Amount. Minimum Piece and Par Amount being the same thing for such a SRCH but some entries are empty for one of these two fields while both were never empty in given dataset. The tables obtained through such search for all Fridays from 4 Jan 2019 through 25 Dec 2020 were downloaded with names encoding the day the data was for to later process and input in various calculations.

The result is data for 104 days of bonds (30yrs, 20yrs), notes (10yrs, 7yrs, 5yrs, 3yrs, 2yrs) and bills (52w, 26w, 13w, 8w, 4w) necessary for further calculations. Such a dataset is also good in the respect that it contains a period of crisis caused by COVID-19 so methods as well as algorithms stand a robustness check on data in which shocks are expected.

All the bonds included pay semiannual coupons and notes pay no coupons. To get the initial coupon payment dates one starts at maturity date and decrements month by six until issue date keeping month end days if months have different lengths. Then as stated in (Federal Register 2021) ”If any principal or interest payment date is a Saturday, Sunday, or other day on which the Federal Reserve

System is not open for business, we will make the payment (without additional interest) on the next business day." so the dates obtained before should be shifted forward until business day is reached.
The quotations obtained in Bloomberg Terminal also have to be adjusted to get the correct data to be used in the modelling. First coupon payment is given as a percentage annualized coupon, as par amount was always 100 in the data the number is just divided by two to get semiannual coupon. For bills the quotes are given as a discount on face value. To do so I used the formula:

$$P = FV\left(1 - \frac{quote}{100} \cdot t \cdot adj\right) \tag{17}$$

where P - bid or ask price, FV - face value, $quote$ - bid or ask quote, t - fraction of the year left until maturity in actual/actual days accounting, adj - 365/360 or 366/360 depending on whether the year is leap or not is a factor adjusting t to show actual/actual day from actual/360 used for treasury bills quotes; the quote is divided by 100 to get it from percent to fraction.
For bonds and notes quotes have to be converted to dirty price as well. With assumption that all the coupon payments are semi-annual (which was double checked to be true in the code) quotes were calculated by the formula:

$$Q_{dirty} = Q_{clean} + (0.5 - t_0) \cdot C \tag{18}$$

(t_0 is time to next coupon payment in years, C is coupon to be paid and Q is the quote).
Price used in all calculations is the midpoint of bid and ask prices. In most calculations such as time in years until next payment actual/actual convention was used throughout the work.
The data actually used in optimizations is a reduced version of what was obtained because even though all the securities are supposed to be explained by equation 1 there are some problems with using the whole dataset. Yields on all the papers vary a lot depending on how long ago the paper was issued. For example yield of a 30 bond with one year left to maturity date will be much different from the one of 52 week bill issued yesterday or a 20 year old bill for instance. One of the explanations of such effect is the Price Pressure Hypothesis (Scholes 1972) investigated further in (Babbel et al. 2004). Thus one can conclude that

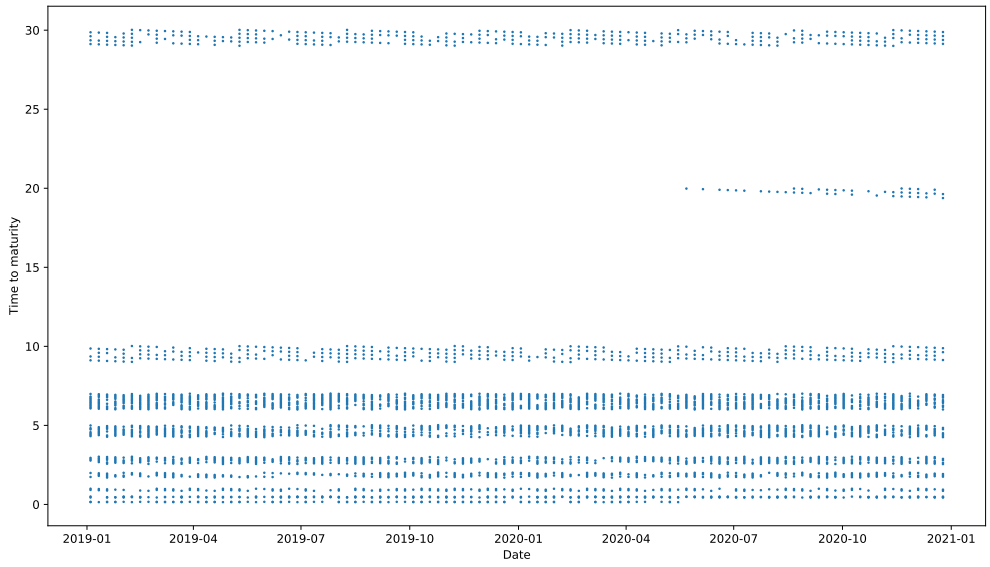

Figure 3: Maturities of bonds used in the training set for run with past points used as best guesses. 20yr bonds were recently introduced so first 2/3 of the dataset misses them as they did not exist at the time.

there is some effect similar to market segmentation and so securities with all identical parameters, except for how close they are to being on the run will have different term structures possibly caused by liquidity. In such case to model term structure for a particular day one has to use more recently issued instruments. My method to get such a sample from all the obtained instruments was to introduce a parameter $Life$ that shows fraction of time passed since issue date of instruments overall lifespan: so for example a 3-year note 365 days after its issue would have the parameter be $Life = 365/(365*3) = 1/3$ (assuming no leap years for simplicity of the example, but they are accounted for). But this relative measure would collect some relatively old instruments still when applied to 20-30 year ones so together with this measure there was introduced a limit on time in years since issue. The resulting condition looks like $(Life < 0.15) \wedge (Age < 1)$ (Age being time in years since issue). This condition provided dataset of around 45 securities including all or most of the maturities traded for that day. The data was then randomly split into $0.8 - 0.2$ training and test subsamples. Test subsamples then included around 10 points but with a moderately big dataset I hope the amount of observations is still big enough to look at test set losses at least aggregated over all days.

3.3.2 Optimization

Now that the data collection and preparation is outlined optimization aspects are to be discussed. First the model used is the restricted for multicollineratity version of NSS as presented in (16). There are also some other possibilities but based on evidence from (De Pooter 2007) including the third loading does not cause overfitting and even performs better on out-of-sample data if one model is to be chosen for testing it probably should be the NSS one. Pricing equation is the one with log-returns in descrete setting:

$$P = \sum_{t=1}^{T} c(t) \exp(-t \cdot r(t)) \tag{19}$$

Optimization was done with respect to price rather than yields due to yields and interest rates not coinsiding in most cases with cost function of the form:

$$L = \sum_{i \in \mathcal{I}} w_i (P_i - \hat{P}_i)^2, \; w_i = \frac{1/D_i}{\sum 1/D_i} \tag{20}$$

where $D_i = (\sum t c_i(t) e^{-t y_i})/P_i$ is Macaulay duration - higher duration errors are down-weighted as there are significantly fewer of them and errors in them might cause worse fit for other maturities. The yields of bonds in data preparation were found with the same formula as equation (19) but rather than searching for parameters of a functional from $r(t)$ a constant y is assumed, root was found with MINPACKS hybr (Burton S. Garbow 1980) method which is modified Powell hybrid method (Powell 1964).

For all runs performed it is worth noting that in practice a candidate may turn out to be so bad that calculating loss raised OverflowError meaning result was greater than $1.8 * 10^{308}$ the loss was set to be $1.797 * 10^{100}$ which is just an arbitrary big number, but far enough from maximum limit to not cause overflows later. Thus loss function was defined as

$$L(X) = \begin{cases} L(X), \; L(X) < 2^{1024} \\ 1.797 \cdot 10^{100}, \; \text{otherwise} \end{cases} \tag{21}$$

Used algorithms take various different inputs so can not be exactly standardized for comparison - some use swarms of points, some are bounded, etc. By adjusting

arguments I tried to somewhat even the playing field.

Some used algorithms allow for bound specification (PSO, L-BFGS-B, Differential Evolution) others don't (COBYLA, Nelder-Mead and Dual Annealing). Similarly not all algorithms allow for constraints (Nelder-Mead, L-BFGS-B, Dual Annealing). Fully bounding unconstrained optimization algorithms feels too restrictive so only the more important constraints were introduced: $\beta_0 + \beta_1 \geq 0$ so the model does not give negative interest as result, $\tau_1 > 0$ so $h(t|\tau)$ and $g(t|\tau)$ are of the appropriate form and $\beta_0 >= 0$ so long term interest rate is non-negative. One has to provide closed interval in the software used so > 0 was changed to ≥ 0.001. For COBYLA which allows constraints bounds were wrapped in constraints as mentioned:

$$l_i + 0.0002 < \Theta_i, \; i \in \mathcal{T} \tag{22}$$

where Θ_i is an element of theta, l_i are lower bounds of that element respectively and $\mathcal{T}$ is the set of positions of these "important" constraints. $+0.0002$ was added as the algorithm allows for deviation from constraints by 0.0002 so to keep it strictly bounded bounds have to be adjusted. For other algorithms missing one of the two inputs loss function was modified to[2]:

$$L^*(x) = L(x) + \sum_{i \in \mathcal{T}} (l_i - x_i)^+ + (-(\beta_0 + \beta_1))^+ \tag{23}$$

($(x)^+ := \max(0, x)$). Such a modification discourages algorithms from moving outside bounds or constraints, but there is still a chance that an unbounded algorithm will move there. Multiplying it by some large constant for some reason significantly decreases performance of Nelder-Mead algorithm. Luckily the ones that tend to are not really swarm optimization ones and this is accounted for by simply checking for "important" constraints when new best result is achieved.

Out of implemented optimizers Nelder-Mead, L-BFGS-B and COBYLA are not optimizing using one big swarm of points so I decided to run them multiple times. Based on first runs finalized version does it as explained: m points are sampled from supplied starting points and iteratively used as best guess for the

[2] In Dual Annealing 1000 was added to cost if $\beta_0 + \beta_1 > 0$ was not satisfied as without it some of the results violated this constraint

optimizer. The best result out of m runs is then saved. m for different algorithms are Nelder-Mead: $\lceil n/k \rceil$, COBYLA: $0.66n$, L-BFGS-B: $\lceil n/2.5 \rceil$. These were set to not take too much time if stopping conditions are not being met. Swarm algorithms were simply supplied with $n = 400$ in my runs.

For all the optimizations maximum amount of iterations was set to 200 with the exception of Dual Annealing - it proved to be much faster so it was set to iterate at most $200 * 14 = 2800$ times. The optimization was set to return two sets of results: one with the standard stopping conditions - change in objective function, maximum iterations reached, etc. (complete optimization) and one up to some function value with time to obtain this value and results recorded as well (fast/threshold optimization). So while the standard complete optimization was being conducted on each iteration the best value found so far was checked and if it is below the threshold result and time to obtain it were recorded, while the main optimization went on.

Minimum of loss function was set to $0.2 * \delta * N$ where N is the size of training set. For runs where squared errors were not weighted ($\delta = 1$) this gave RMSE of 0.447 or lower which seemed good enough on given data, but with weighting targeting upper bound of average RMSE became harder so, through trial and error, correction of the harmonic mean times 1.25 seemed to preform just as desired (possibly using geometric mean would work without any additional corrections)[3]:

$$\delta = \frac{1.25n}{\sum_{j \in \mathcal{I}} \frac{\sum_{i \in \mathcal{I}} 1/D_i}{1/D_j}}, \text{ for weighted cases} \tag{24}$$

So in some sense (if the target is achieved) comparison can be made as for how fast the algorithms found a "good enough" result and how good it is exactly given it's better than target. Even though some might argue that time to optimize is not much of an issue e.g. (Gilli and Schumann 2010) saying that the process is fast and accuracy is much more important, but still there are some parameters of task that could make time an important issue to keep in mind. For example I would assume that one of the reasons their optimization was so much faster is that zero-coupon bonds were used and there were less of them; in any case it can be seen that with parameters I used optimization took 100-250 seconds on average per one day. Together for each starting point generation method about 20 hours were needed to get all the results (though optimization was repeated 5 times for

[3] Initially I forgot to convert to dirty prices and got slightly worse results, when this was changed this coefficient stayed the same. With better results of the overall optimization one could try to scale this coefficient down a bit to get better results in the threshold optimization

each day) which is quite a long time, it might be not relevant for practitioners, but for some research I could imagine such a dataset being required and if the data is not provided one will have to wait for the results to be calculated.
Starting points generation was implemented in three ways. First was a simple (uniform) random generation, then the second one assumed that best theta of the previous estimation is a good guess of current theta so more starting points should be generated around it thirdly Diebold-Li regression was used to get an estimation of parameters around which starting points were generated similar to the second points generation method. For simple random generation of points for first a $k \times n$ matrix was created where each element is distributed as Uniform(0, 1), then these k points were transformed from [0, 1] interval to [l, u] through a simple linear transformation[4]:

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & \dots & l_k \\ l_1 & l_2 & \dots & l_k \\ \vdots & \vdots & \ddots & \vdots \\ l_1 & l_2 & \dots & l_k \end{bmatrix} + \begin{bmatrix} U_{11} & U_{12} & \dots & U_{1k} \\ U_{21} & U_{22} & \dots & U_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ U_{n1} & U_{n2} & \dots & U_{nk} \end{bmatrix} \circ \begin{bmatrix} d_1 & d_2 & \dots & d_k \\ d_1 & d_2 & \dots & d_k \\ \vdots & \vdots & \ddots & \vdots \\ d_1 & d_2 & \dots & d_k \end{bmatrix} \tag{25}$$

Where $[x_{i1}, x_{i2}, \dots x_{ik}]$ is one of n resulting starting points, $[l_1, l_2, \dots l_k]$ is lower bound vector, $[d_1, d_2, \dots d_k]$ is upper bound less lower bound vector and $U_{ij} \sim U(0,1)$. This procedure allows to get n uniformly distributed k-dimensional vectors to be used as starting points.
For case when best guess is formed the procedure is somewhat similar, but instead of uniform distribution truncated normal is used, centered at above mentioned best result:

$$\begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & \dots & l_k \\ l_1 & l_2 & \dots & l_k \\ \vdots & \vdots & \ddots & \vdots \\ l_1 & l_2 & \dots & l_k \end{bmatrix} + \begin{bmatrix} n_{11} & n_{12} & \dots & n_{1k} \\ n_{21} & n_{22} & \dots & n_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ n_{n1} & n_{n2} & \dots & n_{nk} \end{bmatrix} \circ \begin{bmatrix} d_1 & d_2 & \dots & d_k \\ d_1 & d_2 & \dots & d_k \\ \vdots & \vdots & \ddots & \vdots \\ d_1 & d_2 & \dots & d_k \end{bmatrix} \tag{26}$$

here notation is the same, but $n_{ij} \sim \mathcal{N}_{tr}(\tilde{x}_j,\ \sigma(\tilde{x}_j),\ 0,\ 1)$ is used rather than U_{ij}. $\sigma(\tilde{x}_j)$ was defined as

$$\sigma(\tilde{x}_j) = \frac{0.4 * s}{1 - |\tilde{x}_j - 0.5|} \tag{27}$$

in which s is the scale changing parameter (for all samples was set to 1, but can be changed to suit the user) and $\tilde{x}_j$ is position of the best guess of Θ_j mapped from $[l_j, u_j] \cap \mathbb{R}$ to $\mathbb{R} \cap [0,1]$: $\tilde{x}_j = (\hat{\Theta}_j - l_j)/d_j$. Graph of this function with $s = 1$ is presented in figure (4) What this all does is rescales the proposed

[4] $\circ$ denotes Hadamard product here

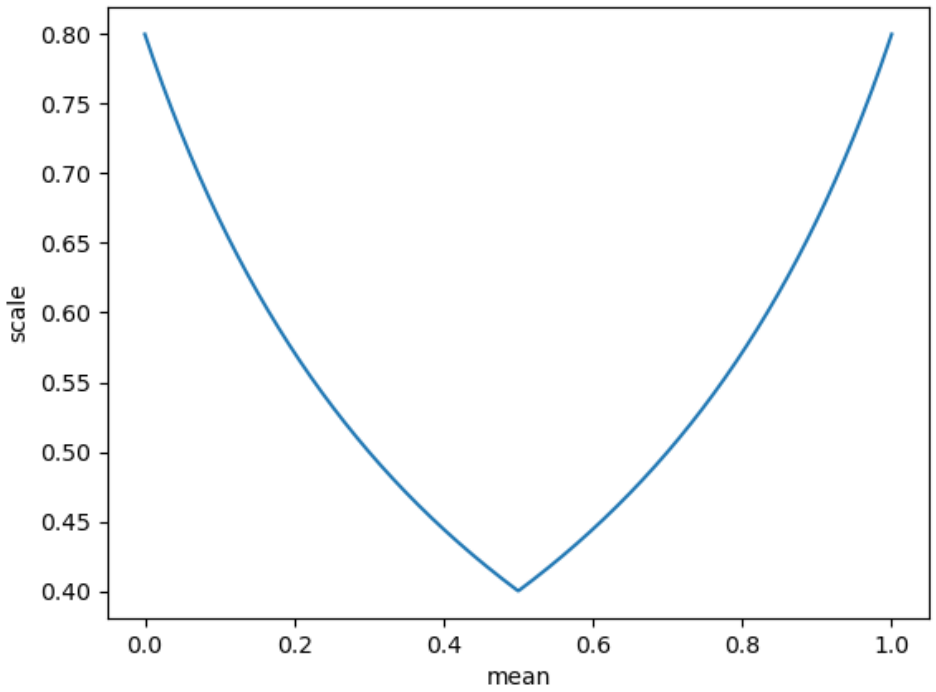

Figure 4: Graph of function $\sigma(\tilde{x}_j)$ for $\tilde{x}_j \in (0,1)$

center point to $[0,1]$ and samples normal distribution truncated from 0 to 1 with mean of $\tilde{x}_j$ and then rescales all the points back to their locations through equation (26). $\sigma(\tilde{x}_j)$ form was chosen arbitrarily by generating multiple different possible distributions and looking at their histograms to account for cases where distribution mean is too close to bounds thus resulting in more flat distribution than desired. Namely if rescaled mean is 0.5 scale of the distribution is 0.4, but as mean approaches bounds scale decreases so the distribution does not become significantly flatter. Some examples of such a sampling method can be seen in figure (5).

The Diebold-Li point generation was as follows: for points in training dataset a regression

$$y_t = \hat{\beta}_0 + \hat{\beta}_1 g(t|1/0.0609) + \hat{\beta}_2 h(t|1/0.0609) \tag{28}$$

The best guess was then set as $\hat{\Theta} = \{\hat{\beta}_0, \hat{\beta}_1, 1/0.0609, \hat{\beta}_2\}$.

The first point in each set of starting points was the best guess (a random point for randomly generated points). This point is supplied to Dual Annealing as the starting one because it is a global optimizer and requires only one point as a starting value. Starting it multiple times from different points like COBYLA or

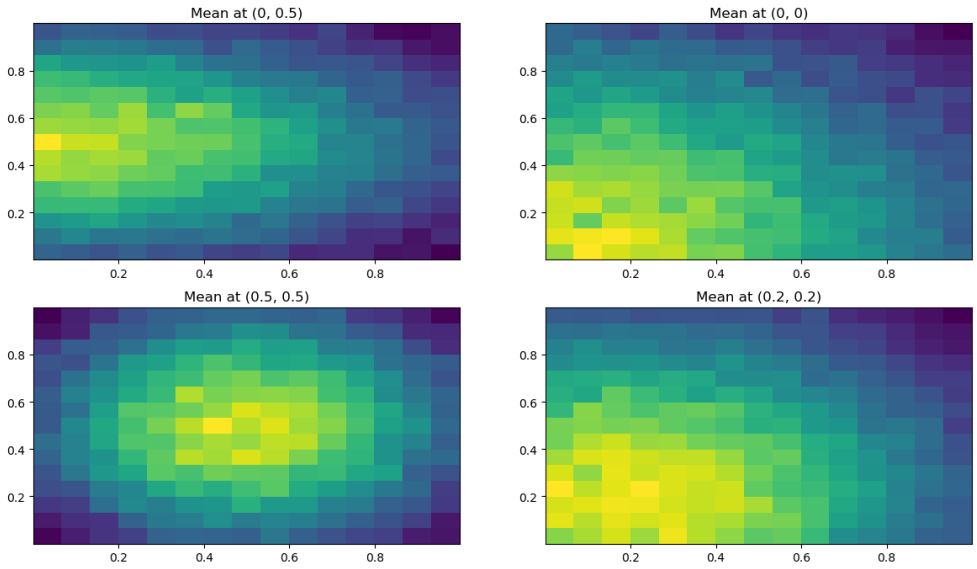

Figure 5: Heatmaps of 2-dimensional variables sampled from truncated normal distribution with different means. Anything above 0.5 for any of the components should look symmetrical around line through (0, 1) and (1, 0).

L-BFGS-B felt like defeating the purpose of the algorithm.

In all cases after initial generation points not satisfying constraints are then resampled by the same rules until all the points are suitable: $[l_1, l_2, \ldots, l_k] \geq [x_{i1}, x_{i2}, \ldots, x_{ik}] \leq [u_1, u_2, \ldots, u_k], \quad \forall i.$ After such a correction the shapes of starting points distribution is changed but it is impossible to generate points like that with constraints so these procedures should suffice.

The double rescaling allows to set scale by one function that gives desired shape independent of the shape of the region provided.

For algorithms taking bounds they were set as:

$$\begin{bmatrix} 0 \\ -5 \\ 0.001 \\ -20 \end{bmatrix} \geq \begin{bmatrix} \beta_0 \\ \beta_1 \\ \tau_1 \\ \beta_2 \end{bmatrix} \geq \begin{bmatrix} 10 \\ 35 \\ 35 \\ 35 \end{bmatrix} \tag{29}$$

Sometimes unbounded algorithms returned results outside bounds but satisfying the ”important” constraints. These were still taken as results but for next starting points where such an occurrence happened the bound that was violated was set as center for the starting points. The fact that some of the solutions were outside of

bounds should not be big of a downside because the more important constraints are satisfied and violating others does not ruin the economic interpretation as much as $\tau \leq 0$ or $\beta_0 + \beta_1 < 0$.

3.3.3 Performance measures

For the performance measures time to optimize, RMSE, bias and R^2 were chosen. RMSE and R^2 were calculated for both prices and yields while bias was only for price.

Though R^2 has some flaws it seemed suitable for the case as what matters more is comparison of different algorithms for any given day and given they use the same sample sizes for train and test sets there is no difference in sample sizes (and parameter amount) so it should be perfectly viable to use them to see how good the model predicts the prices or yields even though they are not the main targets. As I noticed in first attempts there is sometimes bias when the algorithm cannot increase β_0 any more due to constraint so to properly calculate the metric one has to remember about it in the calculations:

$$R^2(\hat{X}) = 1 - \frac{\sum_{i=1}^{N}(X_i - \hat{X}_i - \overline{(X_i - \hat{X}_i)})^2}{\sum_{i=1}^{N}(X_i - \bar{X}_i)^2} \tag{30}$$

$$RMSE(\hat{X}) = \sqrt{\frac{\sum_{i=1}^{N}(X_i - \hat{X}_i)^2}{N}} = \sqrt{\frac{L(\hat{X})}{N}} \tag{31}$$

The variation in prices is not really the final target but a measure of how good term structure was fitted and also generally do not have a big variance so if the method does provide some moderately high $R^2(P)$ measure it was likely fitted well. Additionally average bias was calculated for all results, but in later more refined implementations it turned out to be negligible.

For each subset of data RMSE measures were separated into short term (no more than 3 years to maturity), medium (from 3 to 10 years to maturity) and long term (no less than 10 years to maturity).

All work was done on Python 3.8.8; the packages and their versions used are listed in the repository[5]. One thing I'd like to note here is that the packages

[5] https://github.com/kdergachev/term_struct_diploma

with Particle Swarm Optimization, Differential Evolution and Dual Annealing algorithms are modified by me to give required results, the modified versions are in the repository as well. The additions are mostly to implement the stoping rule as discussed below. They add one comparison of two values per iteration so I would assume this addition should not increase time considering full optimization as will be show later takes about 200 seconds.

4 Empirical tests

In this section I will refer to tables (2), (3) and (4) from appendix B. There the main results of the runs are presented, grouped by starting points selection over training, testing and overall subsets of the data obtained from initial tables. To better reflect results for each parameter chosen 0, 25, 50, 75 and 100 percentiles are shown together with mean and standard deviation of said parameter. The main ones shown are residual mean sum of squares for all three discussed terms of bonds and time taken to get the solution for both complete optimization and the threshold one together with R^2 of the combined sample produced from complete optimization results. There are some other parameters calculated which can be seen in the repository, but not presented here. As expected due to the loss function used accuracy in all models decreases with duration and thus term to maturity increase so for short bonds RMSE is significantly smaller than for long ones.

4.1 In sample results

For all starting points generation techniques short bond errors are very similar between complete and threshold runs. The algorithms performed similarly as well with differential evolution and COBYLA doing slightly worse in the fast runs. Medium bond errors have some more variation to them but still the RMSE is not too high, though the difference between complete and fast is more noticeable. And as for long bonds the errors are the highest with greater difference between fast and complete runs. Differential Evolution and COBYLA perform worse in fast optimization while still being somewhat competitive in complete runs. Nelder-Mead seems to perform slightly worse than other algorithms in complete runs but works moderately well in the threshold optimization. Dual Annealing and L-BFGS-B perform well in all the runs and both variations of stopping rule.

4.2 Out of sample results

As for out of sample performance the relative results between algorithms and stopping rules are close to the ones observed in the previous subsection - fast

runs' accuracy is marginally worse than the complete ones. And between algorithms the results are very similar with long bonds errors being the most different due to weighting scheme and the fact that there are fewer long bonds in used data. Once again L-BFGS-B and Dual Annealing could be considered the best two with one being better than the other in some cases and other way around in others.

4.3 Overall

Independent of starting points choice time to reach the result in fast optimization is orders of magnitude faster than the complete ones. While complete optimization time is not exactly the best measure to compare algorithms with as more often the optimization was stopped by maximum iterations conditions rather than any other one so a part of it is due to my choice of parameters of optimization it at least should be kept in mind when looking at results and to some extent the difference was reduced by parameter choice - means are no more than twice as big as mean of any other algorithm. Generally COBYLA, Dual Annealing and L-BFGS-B were the faster ones with Nelder-Mead slightly slower and differential evolution the slowest even in the case where it speed improved significantly. In complete runs Nelder-Mead and COBYLA are the more stable in time it takes to get the result and Dual Annealing is the least with the parameters used. In all cases the degree to which the price deviations were explained by estimated term structure model is surprisingly high with most of the R^2 values being greater than 0.995 and bias being close to 0 in most cases.

4.4 Additions

As was stated above the massive improvement in time for Differential Evolution led me to believe that past starting point generation is significantly better than the other ones. So here I will provide some evidence for that claim. To test it I used the values of best guess thetas for each day and the best result out of all three starting point specification runs obtained from complete optimization for that day excluding the ones where they did not match e.g. the first day for past point generation does not have a guess. On these points I then ran a regression of the Mincer-Zarnowitz (Mincer and Zarnowitz 1969) form. With Diebold-Li τ it

is cannot include the constant term as the τ itself is a constant so this regression was without an intercept. The results as provided in (1) support the claim that out of the two guesses of current optimal values the past points are significantly better than Diebold-Li results. From these results it seems that there was some error in calculating Diebold-Li coefficients because only τ works well while the other ones are very bad guesses. Still if it is so this is an opportunity to see how well optimization is done if most points are grouped around a bad guess of the optimum. Best past theta on the other hand performs moderately well. As can be observed τ and β_1 are both close to 1 and very significant with insignificant constants while the other two components are significant but not as close to 1 and with more significant intercepts. Random points can also be considered a bad guess so the intuition derived from Differential Evolution optimization time is likely correct. Such a difference implies that given some very good guess it will even faster converge to a moderately good result after which it will seek this marginal improvement as seen in difference between fast and full optimization performance.

Additionally the fact that results are good independent of how good starting points are leads me to believe that something about the distribution of starting points could be excessive for the task (number of points, number of iterations, decision to use truncated normal distribution, etc) so one could try to decrease some of the parameters of optimizers and see which one works better then.

Now I would like to mention robustness to shocks caused by crises. If they significantly decrease performance one would expect higher time and/or RMSE from around the start of 2020 to summer of the same year. As can be seen on (6) and (7) results are mostly stable during the period with only March 20 2020 having an RMSE outlier in both full and fast runs. From this I would conclude that optimization did not suffer significantly as throughout 15 different runs no good solution was obtained so this is more likely greater variance in prices for that day or a shape of yield curve that could not be modeled by used specifications under chosen restrictions.

4.5 Comparison

Overall it seems that though using past parameters as starting points for current optimization is a better choice the improvement is negligible and may be even

Past points

	β_0		β_1		τ		β_2	
	c	*b*	*c*	*b*	*c*	*b*	*c*	*b*
v	0.001	0.327	0.002	0.837	3.43	0.758	-0.196	0.586
p	0.015	0.	0.001	0.	0.	0.	0.	0.

Diebold-Li points

	β_0		β_1		τ		β_2	
	c	*b*	*c*	*b*	*c*	*b*	*c*	*b*
v	0.001	0.027	0.004	-0.269	-	0.974	-0.474	0.093
p	0.366	0.487	0.012	0.	-	0.	0.	0.

Table 1: Coefficients of regression of realized best result for each day on best guess used to generate starting points. For each parameter a separate regression was run. *c* stands for intercept, *b* - for coefficient at best guess, v is value of given coefficient, p is p-value of this coefficient. Results are rounded to 3 decimal places. Diebold-Li τ has no intercept in the regression as it is a constant itself.

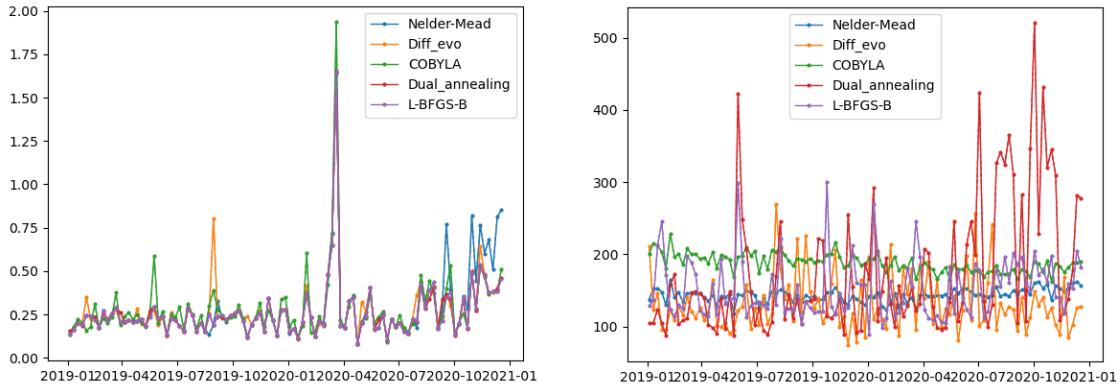

Figure 6: RMSE (left) and time (in seconds) to optimize (right) for test subsample in past points complete runs over all used algorithms

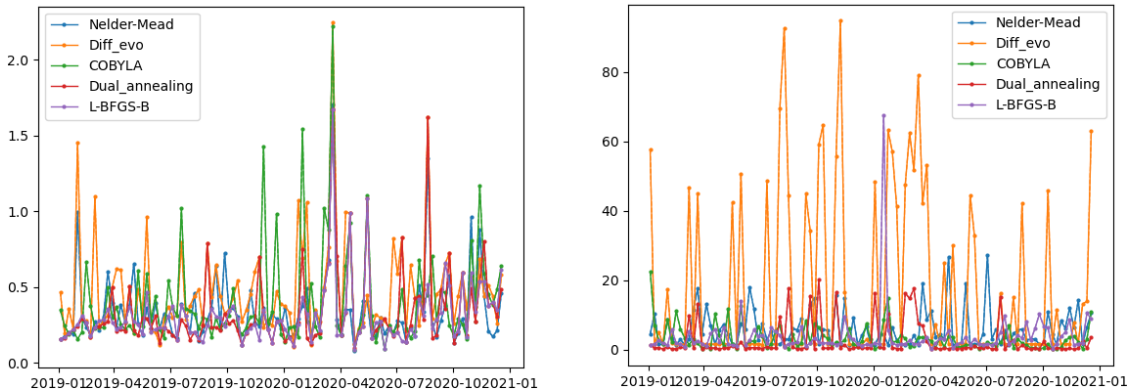

Figure 7: RMSE (left) and time (in seconds) to optimize (right) for test subsample in past points threshold runs over all used algorithms

unnoticeable if not for the differential evolution result. This is similar to findings in (Gilli and Schumann 2010) where they also pointed out that together with all guesses of optimal solution to the problem being very similar in resulting accuracy random points with restarts did not do worse than these starting points. As was investigated above andom points, a good guess or a bad guess did not make much of a difference in final result in complete optimization so from this one can conclude either points and to some extent optimization algorithm choice does not matter for the problem and/or that all algorithms are robust enough to achieve global optimum (or get arbitrarily close to it) given parameters supplied in my tests independent of starting points generation rule.
The threshold runs show that with a correctly specified stopping rule one can obtain results that are slightly worse than the optimal ones but much faster, additionally Differential Evolution is the only algorithm that was significantly affected by starting points selection and it was mostly the time to reach the solution.
Similarly all algorithms seem to perform reasonably well with some (Dual Annealing and L-BFGS-B) performing marginally better and faster/with similar speed. It is surprising that a gradient descent algorithm performed so well as in past works other authors such as (Ahi, Akgiray, and Sener 2018) and (Manousopoulos and Michalopoulos 2009) concluded that they perform poorly or that they should perform poorly due to the problem specifics. In the same vein it is not expected that Differential Evolution, while not being definitively worse than other algorithms chosen, was similar or worse while at the same time being slower. The results show that under a proper stopping rule one can achieve results with similar accuracy to the ones achieved under complete optimization with significantly less time. With non-weighted loss RMSE targeting becomes much easier as weight do not have to be corrected, but one could try to target loss rather than RMSE. The problem with such approach is knowing what is a good target as here it was chosen heuristically based on results of previous runs on the same data. The simplest way to do it without any prior knowledge would be to set some value that is slightly higher than the one from the previous optimization, especially if interval between them is short, still this is just a guess and there may be periods with different volatilities and this guess will not be optimal. In any case the threshold for the main chunk of work in this

paper was too high as only in one of them there were two instances where it was not breached during the complete optimization. and the results are still quite close to the ones in the complete runs. The most significant change is time to optimize up to the threshold for differential evolution - it performed the worst in the random starting point generation marginally better in Diebold-Li and the best by far in past one. Unless there was an oversight my guess is that the result was somewhere around the best past point quite often and the fact that there is greater density of points around it differential evolution benefited greatly from it as the new point in this algorithm is generated as some combination of the best so far and two other randomly chosen points. Thus if the area around the solution has more points the solution will improve much faster.

5 Conclusion

Term structure is a tool widely used in multiple areas of economics and finance and with the latest models and attempts to fit them more directly - to prices rather than yields bring about some newer problems even though theoretically this functional forms if fit with prices are good ways to model term structure. Most of these problems are due to the fact that the problem becomes non-linear or too restricted.

In this paper I used the less restricted Nelson-Siegel-Svensson model with two restrictions on parameters that avoid multicollinearity in fitting and other two that bound it for better economic interpretation. This model was fit with various parameters to find the more efficient ways to estimate parameters of functional form.

My results demonstrate that difference between algorithms is negligible, but if I were to recommend L-BFGS-B with restarts or Dual Annealing. As for starting points choice the conclusion is that past points generation is the better of the two and in fact a good guess of the next optimal solution at leas if data frequency is weekly, but this advantage is completely removed during the optimization in all algorithms except Differential Evolution. Once again it shows that the procedures are robust to quality of starting points or that parameters were good enough to facilitate global search irrespective of starting points provided. Additionally the proposed stopping rule proved to be moderately good at finding results close to the optimal ones in significantly less time with again an exception being Differential Evolution which as it seems cannot escape high density of points around a bad prediction.

References

Ahi, Emrah, Vedat Akgiray, and Emrah Sener (2018). "Robust term structure estimation in developed and emerging markets". In: *Annals of Operations Research* 260.1, pp. 23–49.

Babbel, David F et al. (2004). "The effect of transaction size on off-the-run Treasury prices". In: *Journal of Financial and Quantitative Analysis*, pp. 595–611.

Bank For International Settlements (2005). *Zero-coupon yield curves: technical documentation*. Tech. rep. 25. Monetary and Economic Department.

Bliss, Robert Russell (1997). "Testing term structure estimation methods". In: *Advances in Futures and Options research* 9, pp. 197–232.

Burton S. Garbow Kenneth E. Hillstrom, Jorge J. More (1980). "Documentation for MINPACK subroutine HYBRD". In:

Černy, Vladimır (1985). "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm". In: *Journal of optimization theory and applications* 45.1, pp. 41–51.

Christensen, Jens HE, Francis X Diebold, and Glenn D Rudebusch (2011). "The affine arbitrage-free class of Nelson–Siegel term structure models". In: *Journal of Econometrics* 164.1, pp. 4–20.

Csajbok, Attila et al. (1998). *Zero-coupon yield curve estimation from a central bank perspective*. Tech. rep. Magyar Nemzeti Bank (Central Bank of Hungary).

De Pooter, Michiel (2007). "Examining the Nelson-Siegel class of term structure models: In-sample fit versus out-of-sample forecasting performance". In: *Available at SSRN 992748*.

Diebold, Francis X and Canlin Li (2006). "Forecasting the term structure of government bond yields". In: *Journal of econometrics* 130.2, pp. 337–364.

Durand, David (1942). "Basic yields of corporate bonds, 1900-1942". In: *Basic Yields of Corporate Bonds, 1900-1942*. NBER, pp. 1–40.

Estrella, Arturo and Gikas A. Hardouvelis (1991). "The Term Structure as a Predictor of Real Economic Activity". In: *The Journal of Finance* 46.2, pp. 555–576. ISSN: 00221082, 15406261. URL: http://www.jstor.org/stable/2328836.

Estrella, Arturo and Frederic S Mishkin (1997). “The predictive power of the term structure of interest rates in Europe and the United States: Implications for the European Central Bank”. In: *European economic review* 41.7, pp. 1375–1401.

Fama, Eugene F and Robert R Bliss (1987). “The information in long-maturity forward rates”. In: *The American Economic Review*, pp. 680–692.

Federal Register (2021). *Title 31 - Money and Finance: Treasury; Subtitle B - Regulations Relating to Money and Finance; CHAPTER II - FISCAL SERVICE, DEPARTMENT OF THE TREASURY; SUBCHAPTER A - BUREAU OF THE FISCAL SERVICE PART 356 - SALE AND ISSUE OF MARKETABLE BOOK-ENTRY TREASURY BILLS, NOTES, AND BONDS (DEPARTMENT OF THE TREASURY CIRCULAR, FISCAL SERVICE SERIES NO. 1-93); Subpart D - Miscellaneous Provisions*.

Fernández-Rodrıguez, Fernando (2006). “Interest Rate Term Structure Modeling Using Free-Knot Splines”. In: *The Journal of Business* 79.6, pp. 3083–3099.

Fletcher, Roger (1970). “A new approach to variable metric algorithms”. In: *The computer journal* 13.3, pp. 317–322.

Gilli, Manfred and Enrico Schumann (2010). “A note on ‘good starting values’ in numerical optimisation”. In: *Available at SSRN 1620083*.

Ioannides, Michalis (2003). “A comparison of yield curve estimation techniques using UK data”. In: *Journal of Banking & Finance* 27.1, pp. 1–26.

Kennedy, James and Russell Eberhart (1995). “Particle swarm optimization”. In: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE, pp. 1942–1948.

Liu, Dong C and Jorge Nocedal (1989). “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1, pp. 503–528.

Manousopoulos, Polychronis and Michalis Michalopoulos (2009). “Comparison of non-linear optimization algorithms for yield curve estimation”. In: *European Journal of Operational Research* 192.2, pp. 594–602.

McCulloch, J Huston (1971). “Measuring the term structure of interest rates”. In: *The Journal of Business* 44.1, pp. 19–31.

Mincer, Jacob A and Victor Zarnowitz (1969). "The evaluation of economic forecasts". In: *Economic forecasts and expectations: Analysis of forecasting behavior and performance*. NBER, pp. 3–46.

Mishkin, Frederic S (1990). "What does the term structure tell us about future inflation?" In: *Journal of monetary economics* 25.1, pp. 77–95.

Nelder, John A and Roger Mead (1965). "A simplex method for function minimization". In: *The computer journal* 7.4, pp. 308–313.

Nelson, Charles R and Andrew F Siegel (1987). "Parsimonious modeling of yield curves". In: *Journal of business*, pp. 473–489.

Plosser, Charles I. and K. Geert Rouwenhorst (1994). "International term structures and real economic growth". In: *Journal of Monetary Economics* 33.1, pp. 133–155. ISSN: 0304-3932. DOI: https://doi.org/10.1016/0304-3932(94)90017-5. URL: https://www.sciencedirect.com/science/article/pii/0304393294900175.

Powell, Michael JD (1964). "An efficient method for finding the minimum of a function of several variables without calculating derivatives". In: *The computer journal* 7.2, pp. 155–162.

— (1994). "A direct search optimization method that models the objective and constraint functions by linear interpolation". In: *Advances in optimization and numerical analysis*. Springer, pp. 51–67.

Scholes, Myron S (1972). "The market for securities: Substitution versus price pressure and the effects of information on share prices". In: *The Journal of Business* 45.2, pp. 179–211.

Shiller, Robert J and J Huston McCulloch (1990). "The term structure of interest rates". In: *Handbook of monetary economics* 1, pp. 627–722.

Stock, James H and Mark W Watson (1989). "New indexes of coincident and leading economic indicators". In: *NBER macroeconomics annual* 4, pp. 351–394.

Storn, Rainer and Kenneth Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4, pp. 341–359.

Svensson, Lars EO (1994). *Estimating and interpreting forward interest rates: Sweden 1992-1994*. Tech. rep. National bureau of economic research.

Tsallis, Constantino and Daniel A Stariolo (1996). “Generalized simulated annealing”. In: *Physica A: Statistical Mechanics and its Applications* 233.1-2, pp. 395–406.

Vasicek, Oldrich A and H Gifford Fong (1982). “Term structure modeling using exponential splines”. In: *The Journal of Finance* 37.2, pp. 339–348.

Appendix A (Algorithms)

Optimization algorithms

Nelder-Mead

Algorithm uses a simplex - convex polytope with $n+1$ vertices in n dimensional space. It moves points based on their and other points evaluations

Algorithm

For $f(x) : \mathbb{R}^n \to \mathbb{R}$

1. Initialize $n+1$ points $\{x_1, x_2, ..., x_{n+1}\}$ and calculate function value of each one of them: $f_i = f(x_i)$. For notation say $\{x_{(1)}, x_{(2)}, ..., x_{(n+1)}\}$ are such that $f(x_{(1)}) \leq f(x_{(2)}) \leq ... \leq f(x_{(n+1)})$
2. Calculate centroid of points excluding the worst one $x_c = \left(\frac{\sum_{k=1}^{n} x_{(k)}}{n} \right)$
3. Reflect the worst point about the x_c with some parameter α: $x_r = x_c + \alpha(x_c - x_{(n+1)})$.
    - if $f(x_{(1)}) \leq f(x_r) < f(x_{(n)})$: replace $x_{(n+1)}$ with x_r to obtain a new simplex.
    - if $f(x_r) < f(x_{(1)})$: $x_e = x_c + \gamma(x_r - x_c)$. Replace $x_{(n+1)}$ with the best out of the two (x_r, x_e)
    - if $f(x_{(n)}) \leq f(x_r) \leq f(x_{(n+1)})$ Compute $x_s = x_c + \rho(x_{(n+1)} - x_c)$. If $f(x_c) < f(x_{(n+1)})$ replace $x_{(n+1)}$ with x_c.
    - if none of the above changed the simplex change all but $x_{(1)}$ with $x_i = x_i + \sigma(x_i - x_{(1)})$
4. Go to step 2 unless stopping condition is satisfied.

Parameters used in this paper are: $\alpha = 1$, $\gamma = 2$, $\rho = 0.5$, $\sigma = 0.5$.

Particle swarm optimization

Algorithm

1. Initialize k points of n dimensions, their speeds uniformly distributed (v_{low}, v_{high}), where $v_{low} = -u + l$, $v_{high} = u - l$
2. Set points' best positions (p_i) as their current positions and get swarm's best position (g).
3. Update speed: $v_i^{new} = \omega v_i + U_1 \phi_p (p_i - x_i) + U_2 \phi_g (g - x_i)$, U_1, U_2 both uniformly distributed from $(0,1)$.
4. Get new positions: $x_i^{new} = x_i + v_i^{new}$ and correct for bounds. In the algorithm used the correction is for any point exceeding bounds return coordinates that do back to the bound they exceeded.
5. Evaluate new points and update particles' and swarm's best positions.
6. Return to step 3

Parameters used in this paper are: $\omega = 0.5$, $\phi_p = 0.5$, $\phi_g = 0.5$, $k = 400$

Differential evolution

Algorithm

1. Population of k points of n dimensions is initialized.
2. Find the best point of the population g
3. Obtain vector $t_i = g + M(x_{r1} - x_{r2})$ where M is a constant and x_r are randomly chosen points from the population excluding x_i.
4. Preform a recombination: for each component of the n dimensional vector x_i^{new} sample a random variable $r_i^j = U(0,1)$
   - if $r_i^j < C$ then j'th component of x_i^{new} is assigned j'th component of t_i
   - otherwise j'th component of x_i is assigned.

- The last component is the last component of t_i independent of r_i^j or C

5. x_i^{new} is evaluated. If x_i^{new} is better than x_i it replaces x_i, if it is better than the best solution yet it replaces it as well.

6. Return to step 2

Dual annealing

Algorithm

1. Initialize point x_1 of n dimensions.

2. Get a point x^* by applying some transformation to x_t. The version of the algorithm used in this paper obtains it using distorted Cauchy-Lorentz visiting distribution "variance" of which is controlled by the visiting parameter and depends on temperature $T_{q_v}(t)$.

3. Evaluate x^* and based on some acceptance probability function (here $p(f(x^*), f(x_t), q_a)$) move the point to x^* or leave it at x_t.

4. Return to step 2

Parameters used in temperature, acceptance probability function and visiting distribution are the same as in the original paper (Tsallis and Stariolo 1996) (Appendix, page 405). $T_{q_a}(1) = 5230$, $q_v = 2.62$, $q_a = -5$. This implementation also restarts the process each time temperature reaches initial $temperature * restart\_temp\_ratio$ which was set at $2e-05$

L-BFGS-B

Algorithm

1. Initialize a starting point and initial Hessian matrix H_0

2. Get the direction of search as $d_k = -H_k^{-1}\nabla f(x_k)$

3. With line search find appropriate step size α_k

4. Move x: $x_{k+1} = x_k + \alpha_k * d_k$

5. Update inverse Hessian (as the inverse one is used in direction formula) some $f(H_0, \{s_k\}, \{y_k\}, d)$ derived from recurrent relation:

$$H_{k+1}^{-1} = \left(I - \frac{s_k y_k^t}{y_k^t s_k}\right) H_k^{-1} \left(I - \frac{y_k s_k^t}{y_k^t s_k}\right) + \frac{s_k s_k^t}{y_k^t s_k}$$

Where $s_k = x_k - x_{k-1}$, $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$

6. Return to step 2

In L-BFGS version only m past s_k and y_k are used for greater memory efficiency. L-BFGS-B additionally does it inside provided bounds.
Here m was set to 5, maximum number of line search steps to 20 and step used to estimate gradient set to $1e-08$.

COBYLA

Algorithm is complicated and explaining it briefly is sure to omit some details so here I will just describe the main actions while for better understanding would suggest looking into the original work (Powell 1994) [pp54-59].
Actions mentioned explained more in depth:

- $\Phi(x) = F(x) + \mu(\max(-c_i(x)))^+$, hatted symbols represent linear approximations.
- Acceptable simplex is the one satisfying:

$$\begin{cases} \sigma^{(j)} \geq \alpha\rho \\ \eta^{(j)} \leq \beta\rho \end{cases}$$

- Revise ρ

$$\rho_{new} = \begin{cases} 0.5\rho, & \rho > 3\rho_{end} \\ \rho_{end}, & \text{otherwise} \end{cases}$$

- Revise μ

$$\mu_{new} = \begin{cases} \mu, \ \mu \geq 1.5\bar{\mu} \\ 2\bar{\mu}, \ \text{otherwise} \end{cases}$$

  $\bar{\mu}$ being $\arg\min \hat{\Phi}(x^{(*)})$ s.t. $\hat{\Phi}(x^{(*)}) < \hat{\Phi}(x^{(0)})$

- $x^{(\Delta)} = x^{(0)} \pm 0.5\rho v^{(l)}$, $v^{(l)}$ - normal to the face opposite $x^{(l)}$, l is chosen based on acceptability of points in the simplex

1. Initialize the starting simplex, set ρ_{start}, ρ_{end} and μ
2. Ensure the simplex is acceptable
3. Get $x^{(*)}$ inside trust region $||x^{(0)} - x^*||_2 < \rho$ that is a minimum of linear approximation of the problem or if constraints are incompatible minimizes the greatest constraint violation.
4. If $x^{(*)}$ turned out to be too close to the center of trust region ($||x^{(0)} - x^*||_2 < 0.5\rho$) revise ρ and μ.
5. Revise μ if needed and if simplex is still correctly specified
6. If $x^{(*)}$ is an improvement revise the simplex, check if approximation of improvement is good enough. Return to 2

In some cases calculate $x^{(\Delta)}$ instead of $x^{(*)}$ to make simplex acceptable rather than to improve results.
The implementation used translates Fortran code into Python one and due to not knowing Fortran I cannot really say which modifications are done and which parameters were changed so I would assume it is the same as in the original paper or not significantly different.

Appendix B (Tables)

Train sample

	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ
	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.05	0.10	0.12	0.14	0.29	0.13	0.04	0.15	0.20	0.22	0.25	0.64	0.25	0.09	0.00	0.23	0.38	0.58	2.61	0.52	0.48
Diff_evo	0.04	0.09	0.11	0.13	0.43	0.11	0.05	0.15	0.20	0.21	0.23	0.77	0.23	0.08	0.00	0.19	0.29	0.52	2.98	0.38	0.33
COBYLA	0.05	0.09	0.12	0.15	0.25	0.12	0.04	0.16	0.20	0.22	0.25	0.47	0.24	0.06	0.03	0.26	0.37	0.58	0.94	0.44	0.22
Dual_annealing	0.05	0.09	0.11	0.14	0.39	0.12	0.04	0.15	0.20	0.22	0.25	0.56	0.24	0.06	0.00	0.24	0.38	0.55	1.61	0.44	0.27
L-BFGS-B	0.04	0.09	0.11	0.13	0.19	0.11	0.03	0.15	0.19	0.21	0.23	0.45	0.22	0.05	0.00	0.20	0.29	0.51	1.00	0.36	0.22
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.06	0.10	0.13	0.16	0.27	0.13	0.04	0.14	0.22	0.26	0.34	0.58	0.29	0.11	0.03	0.33	0.55	0.77	2.53	0.67	0.51
Diff_evo	0.04	0.13	0.16	0.20	0.34	0.17	0.06	0.17	0.31	0.36	0.44	0.66	0.38	0.11	0.25	0.75	1.14	1.55	2.68	1.16	0.54
COBYLA	0.07	0.12	0.18	0.26	0.37	0.19	0.08	0.16	0.23	0.27	0.33	0.57	0.29	0.08	0.09	0.51	0.87	1.23	4.10	0.97	0.65
Dual_annealing	0.05	0.09	0.12	0.14	0.29	0.12	0.04	0.15	0.20	0.23	0.26	0.48	0.24	0.06	0.01	0.26	0.40	0.67	3.45	0.50	0.41
L-BFGS-B	0.06	0.09	0.12	0.15	0.29	0.12	0.04	0.15	0.20	0.23	0.26	0.45	0.24	0.06	0.01	0.28	0.40	0.65	3.45	0.51	0.41

Test sample

	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.01	0.07	0.11	0.16	0.31	0.12	0.08	0.08	0.19	0.25	0.30	0.62	0.26	0.10	0.01	0.25	0.54	1.09	2.80	0.74	0.63
Diff_evo	0.00	0.06	0.10	0.15	0.31	0.11	0.08	0.05	0.18	0.22	0.27	0.49	0.23	0.07	0.06	0.24	0.46	0.76	2.94	0.57	0.50
COBYLA	0.00	0.06	0.09	0.16	0.39	0.12	0.09	0.03	0.18	0.22	0.26	0.45	0.23	0.08	0.06	0.33	0.49	0.73	3.10	0.59	0.44
Dual_annealing	0.00	0.06	0.10	0.15	0.74	0.12	0.10	0.06	0.18	0.23	0.29	0.68	0.24	0.09	0.06	0.26	0.58	0.83	2.76	0.63	0.48
L-BFGS-B	0.00	0.05	0.10	0.15	0.31	0.11	0.08	0.06	0.18	0.22	0.27	0.43	0.23	0.07	0.06	0.24	0.41	0.76	2.76	0.54	0.44
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.00	0.08	0.12	0.17	0.42	0.14	0.08	0.08	0.21	0.27	0.35	1.15	0.31	0.16	0.02	0.35	0.71	1.33	3.33	0.93	0.73
Diff_evo	0.01	0.10	0.15	0.22	0.58	0.17	0.11	0.14	0.27	0.38	0.48	1.30	0.41	0.20	0.06	0.76	1.29	1.70	3.87	1.34	0.81
COBYLA	0.01	0.09	0.15	0.26	0.67	0.18	0.12	0.06	0.20	0.25	0.35	0.86	0.29	0.13	0.02	0.55	0.96	1.23	4.35	1.09	0.87
Dual_annealing	0.00	0.06	0.11	0.15	0.32	0.12	0.08	0.06	0.18	0.23	0.30	0.52	0.25	0.09	0.04	0.29	0.57	0.84	3.73	0.70	0.62
L-BFGS-B	0.01	0.06	0.11	0.15	0.33	0.12	0.08	0.08	0.19	0.23	0.30	0.52	0.25	0.09	0.02	0.33	0.60	0.87	3.73	0.70	0.61

Combined

	Time							Time_F							R^2						
Nelder-Mead	117.88	131.70	138.09	144.42	161.36	138.30	9.95	1.14	2.84	5.86	10.41	145.78	10.26	17.34	0.91	0.99	1.0	1.0	1.0	0.99	0.02
Diff_evo	147.06	207.20	227.45	242.16	282.33	221.15	29.84	50.44	69.43	77.46	88.11	134.64	80.09	15.88	0.96	0.99	1.0	1.0	1.0	0.99	0.01
COBYLA	161.23	176.69	187.16	195.80	222.55	187.33	13.45	0.45	1.47	3.20	6.18	24.92	4.90	4.77	0.96	0.99	1.0	1.0	1.0	0.99	0.01
Dual_annealing	82.91	98.65	125.89	204.05	811.27	164.95	105.66	0.91	1.64	2.55	5.94	208.03	8.49	25.20	0.84	0.99	1.0	1.0	1.0	0.99	0.02
L-BFGS-B	100.30	132.26	160.30	191.18	307.64	168.69	45.58	1.00	1.73	2.45	3.71	36.41	3.83	5.04	0.97	0.99	1.0	1.0	1.0	0.99	0.01

Table 2: RMSE, time in seconds to optimize and R^2 for runs with randomly generated starting points. "_S", "_M" and "_L" mean short, medium and long term bonds respectively; "_F" means the statistic is for the fast(threshold run). Time is independent of test or train sample and R^2 is shown only for test and train combined with parameters from complete optimization. Everything is rounded to 2 decimal places. Each parameter is divided into 0, 25, 50, 75, 100 percentiles, mean and standard deviation.

Train sample

	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ
	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.05	0.09	0.11	0.15	0.25	0.12	0.04	0.13	0.19	0.21	0.23	0.61	0.23	0.08	0.01	0.20	0.29	0.55	1.65	0.43	0.35
Diff_evo	0.04	0.09	0.11	0.14	0.18	0.11	0.03	0.13	0.19	0.21	0.23	0.45	0.23	0.06	0.03	0.21	0.29	0.54	1.62	0.39	0.26
COBYLA	0.06	0.09	0.11	0.14	0.19	0.12	0.04	0.14	0.20	0.22	0.24	0.45	0.23	0.05	0.04	0.30	0.47	0.58	1.41	0.47	0.22
Dual_annealing	0.05	0.08	0.11	0.13	0.18	0.11	0.03	0.15	0.19	0.21	0.23	0.45	0.22	0.05	0.01	0.20	0.28	0.53	0.98	0.36	0.21
L-BFGS-B	0.05	0.09	0.11	0.13	0.18	0.11	0.03	0.13	0.19	0.21	0.22	0.46	0.22	0.05	0.01	0.21	0.29	0.53	0.98	0.36	0.21
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.05	0.10	0.13	0.16	0.24	0.13	0.04	0.15	0.21	0.23	0.29	0.58	0.27	0.09	0.00	0.29	0.47	0.69	2.04	0.53	0.35
Diff_evo	0.04	0.09	0.13	0.16	0.29	0.13	0.05	0.16	0.24	0.32	0.41	0.67	0.33	0.11	0.15	0.55	1.00	1.53	2.59	1.07	0.66
COBYLA	0.05	0.11	0.15	0.23	0.37	0.17	0.08	0.15	0.22	0.26	0.31	0.49	0.27	0.07	0.18	0.50	0.70	1.07	3.50	0.86	0.57
Dual_annealing	0.05	0.09	0.11	0.15	0.25	0.12	0.04	0.16	0.20	0.22	0.27	0.49	0.25	0.08	0.02	0.22	0.36	0.78	2.43	0.60	0.56
L-BFGS-B	0.06	0.10	0.12	0.15	0.20	0.12	0.04	0.14	0.20	0.24	0.27	0.48	0.24	0.06	0.07	0.26	0.40	0.59	2.32	0.51	0.39

Test sample

	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.01	0.08	0.11	0.15	0.29	0.12	0.06	0.07	0.19	0.23	0.28	0.63	0.25	0.10	0.03	0.21	0.38	0.85	2.78	0.61	0.59
Diff_evo	0.01	0.06	0.11	0.15	0.31	0.11	0.06	0.10	0.19	0.23	0.30	0.53	0.24	0.08	0.00	0.22	0.43	0.74	2.81	0.54	0.47
COBYLA	0.01	0.07	0.11	0.15	0.29	0.12	0.06	0.08	0.19	0.23	0.29	0.47	0.24	0.08	0.04	0.23	0.45	0.74	3.30	0.57	0.51
Dual_annealing	0.01	0.06	0.11	0.14	0.27	0.11	0.06	0.08	0.19	0.23	0.28	0.47	0.23	0.07	0.01	0.21	0.41	0.74	2.81	0.52	0.45
L-BFGS-B	0.01	0.06	0.11	0.14	0.27	0.11	0.06	0.08	0.19	0.22	0.27	0.48	0.23	0.07	0.01	0.21	0.39	0.74	2.81	0.51	0.45
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.02	0.07	0.12	0.16	0.44	0.13	0.07	0.07	0.19	0.24	0.32	0.75	0.28	0.13	0.03	0.34	0.63	0.92	2.89	0.72	0.56
Diff_evo	0.01	0.07	0.11	0.18	0.40	0.13	0.08	0.08	0.24	0.33	0.42	0.74	0.34	0.14	0.01	0.66	0.95	1.66	3.84	1.24	0.83
COBYLA	0.00	0.08	0.13	0.20	0.49	0.15	0.10	0.08	0.21	0.27	0.37	0.56	0.28	0.10	0.09	0.37	0.70	1.41	3.80	1.02	0.90
Dual_annealing	0.01	0.07	0.12	0.16	0.27	0.12	0.06	0.08	0.19	0.25	0.31	0.53	0.26	0.09	0.02	0.24	0.44	1.14	3.14	0.79	0.78
L-BFGS-B	0.01	0.07	0.11	0.15	0.29	0.12	0.06	0.10	0.19	0.24	0.31	0.49	0.25	0.09	0.00	0.21	0.53	0.82	3.14	0.68	0.66

Combined

	Time							Time_F							R^2						
Nelder-Mead	118.56	139.47	143.73	148.04	165.00	143.49	8.85	0.84	1.53	4.38	7.29	27.28	5.79	5.44	0.93	0.99	1.0	1.0	1.0	0.99	0.01
Diff_evo	73.69	103.77	122.88	145.72	269.84	131.87	39.96	1.28	1.58	1.67	42.15	94.69	18.45	24.79	0.98	0.99	1.0	1.0	1.0	0.99	0.01
COBYLA	162.56	179.39	187.75	196.67	228.61	188.24	12.61	0.19	0.90	2.16	3.87	22.52	3.27	3.61	0.97	0.99	1.0	1.0	1.0	0.99	0.01
Dual_annealing	87.03	107.84	138.50	210.54	520.19	173.86	91.68	0.08	0.30	0.47	0.95	20.19	2.51	4.94	0.98	0.99	1.0	1.0	1.0	0.99	0.01
L-BFGS-B	88.28	121.29	139.33	178.42	300.08	153.17	41.74	0.06	1.58	1.98	4.22	67.58	3.82	6.85	0.98	0.99	1.0	1.0	1.0	0.99	0.01

Table 3: RMSE, time in seconds to optimize and R^2 for runs with starting points generated from past best results. "_S", "_M" and "_L" mean short, medium and long term bonds respectively; "_F" means the statistic is for the fast(threshold run). Time is independent of test or train sample and R^2 is shown only for test and train combined with parameters from complete optimization. Everything is rounded to 2 decimal places. Each parameter is divided into 0, 25, 50, 75, 100 percentiles, mean and standard deviation.

Train sample

	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ	min	0.25	median	0.75	max	μ	σ
	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.04	0.09	0.12	0.15	0.30	0.13	0.04	0.16	0.20	0.22	0.25	0.55	0.24	0.08	0.01	0.23	0.33	0.55	1.77	0.48	0.39
Diff_evo	0.04	0.09	0.11	0.14	0.18	0.11	0.03	0.16	0.19	0.21	0.23	0.47	0.22	0.05	0.00	0.20	0.27	0.50	1.20	0.36	0.23
COBYLA	0.04	0.09	0.12	0.15	0.22	0.12	0.04	0.16	0.20	0.23	0.26	0.46	0.24	0.06	0.06	0.25	0.37	0.63	1.36	0.46	0.26
Dual_annealing	0.04	0.10	0.12	0.15	0.19	0.12	0.04	0.16	0.20	0.23	0.26	0.48	0.24	0.06	0.00	0.25	0.37	0.56	1.20	0.43	0.24
L-BFGS-B	0.04	0.09	0.11	0.14	0.18	0.11	0.03	0.15	0.19	0.21	0.23	0.48	0.22	0.06	0.00	0.21	0.27	0.50	1.20	0.36	0.23
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.04	0.10	0.14	0.18	0.29	0.15	0.05	0.17	0.23	0.26	0.33	0.59	0.29	0.08	0.02	0.33	0.50	0.85	3.59	0.65	0.51
Diff_evo	0.05	0.12	0.17	0.22	0.37	0.17	0.06	0.20	0.30	0.36	0.45	0.70	0.38	0.11	0.17	0.77	1.14	1.57	3.79	1.22	0.63
COBYLA	0.04	0.11	0.17	0.26	0.36	0.18	0.08	0.18	0.24	0.27	0.33	0.56	0.29	0.08	0.17	0.44	0.77	1.21	3.01	0.88	0.59
Dual_annealing	0.05	0.10	0.12	0.15	0.19	0.12	0.04	0.17	0.20	0.24	0.27	0.49	0.24	0.06	0.02	0.27	0.39	0.62	2.28	0.49	0.34
L-BFGS-B	0.04	0.10	0.12	0.15	0.29	0.12	0.04	0.16	0.20	0.23	0.26	0.50	0.24	0.06	0.02	0.27	0.37	0.59	3.19	0.49	0.39

Test sample

	RMSE_S							RMSE_M							RMSE_L						
Nelder-Mead	0.00	0.08	0.12	0.16	0.36	0.12	0.06	0.08	0.19	0.23	0.30	0.71	0.26	0.11	0.03	0.22	0.42	0.90	2.13	0.64	0.53
Diff_evo	0.00	0.06	0.10	0.14	0.35	0.11	0.06	0.04	0.17	0.23	0.27	0.58	0.23	0.09	0.06	0.23	0.37	0.68	1.28	0.49	0.33
COBYLA	0.01	0.07	0.11	0.15	0.37	0.12	0.07	0.07	0.18	0.23	0.29	0.55	0.24	0.09	0.01	0.27	0.45	0.80	1.79	0.58	0.39
Dual_annealing	0.01	0.08	0.11	0.15	0.39	0.12	0.07	0.09	0.19	0.24	0.29	0.58	0.25	0.10	0.05	0.23	0.51	0.82	1.44	0.56	0.38
L-BFGS-B	0.00	0.06	0.10	0.14	0.35	0.11	0.06	0.04	0.17	0.22	0.27	0.59	0.23	0.09	0.05	0.24	0.38	0.67	1.28	0.50	0.33
	RMSE_S_F							RMSE_M_F							RMSE_L_F						
Nelder-Mead	0.01	0.10	0.13	0.18	0.39	0.14	0.07	0.08	0.22	0.28	0.35	0.95	0.31	0.14	0.02	0.29	0.62	1.01	3.65	0.78	0.70
Diff_evo	0.01	0.09	0.15	0.21	0.44	0.16	0.10	0.07	0.25	0.36	0.49	1.17	0.38	0.18	0.16	0.75	1.29	1.74	3.13	1.32	0.71
COBYLA	0.01	0.10	0.15	0.26	0.50	0.18	0.11	0.00	0.20	0.27	0.36	0.78	0.30	0.14	0.00	0.41	0.87	1.29	2.97	0.98	0.69
Dual_annealing	0.01	0.08	0.11	0.15	0.39	0.12	0.07	0.08	0.20	0.24	0.32	0.64	0.27	0.10	0.02	0.20	0.53	0.88	2.02	0.60	0.45
L-BFGS-B	0.00	0.08	0.11	0.15	0.58	0.12	0.08	0.10	0.18	0.24	0.31	0.63	0.26	0.10	0.05	0.22	0.53	0.84	3.43	0.63	0.53

Combined

	Time							Time_F							R^2						
Nelder-Mead	117.52	130.62	137.28	142.35	156.77	137.19	9.27	0.98	1.54	4.38	10.24	73.75	7.79	10.16	0.93	0.99	1.0	1.0	1.0	0.99	0.01
Diff_evo	136.69	194.68	218.98	238.17	275.34	214.88	30.86	10.80	63.03	72.03	80.30	155.06	71.33	17.27	0.97	0.99	1.0	1.0	1.0	0.99	0.01
COBYLA	149.39	173.37	182.41	190.71	216.41	183.11	13.33	0.23	0.91	2.27	6.38	20.59	3.92	4.01	0.97	0.99	1.0	1.0	1.0	0.99	0.01
Dual_annealing	83.28	96.11	126.80	157.30	813.66	155.00	105.09	1.56	6.23	9.83	12.94	40.64	9.75	5.62	0.96	0.99	1.0	1.0	1.0	0.99	0.01
L-BFGS-B	86.77	107.77	135.98	166.60	256.31	142.71	39.95	0.91	1.63	2.62	4.75	19.31	3.66	2.96	0.97	0.99	1.0	1.0	1.0	0.99	0.01

Table 4: RMSE, time in seconds to optimize and R^2 for runs with starting points generated around Diebold-Li ones. "_S", "_M" and "_L" mean short, medium and long term bonds respectively; "_F" means the statistic is for the fast(threshold run). Time is independent of test or train sample and R^2 is shown only for test and train combined with parameters from complete optimization. Everything is rounded to 2 decimal places. Each parameter is divided into 0, 25, 50, 75, 100 percentiles, mean and standard deviation.