COMISEF Computational Optimization Methods in Statistics, Econometrics and Finance

- Marie Curie Research and Training Network funded by the EU Commission through MRTN-CT-2006-034270 -

COMISEF WORKING PAPERS SERIES

WPS-031 30/03/2010

Calibrating the Nelson–Siegel–Svensson model

M. Gilli
Stefan Große
E. Schumann

Calibrating the Nelson–Siegel–Svensson model

Manfred Gilli[*], Stefan Große[†] and Enrico Schumann[*]

March 30, 2010

Abstract

The Nelson–Siegel–Svensson model is widely-used for modelling the yield curve, yet many authors have reported 'numerical difficulties' when calibrating the model. We argue that the problem is twofold: firstly, the optimisation problem is not convex and has multiple local optima. Hence standard methods that are readily available in statistical packages are not appropriate. We implement and test an optimisation heuristic, Differential Evolution, and show that it is capable of reliably solving the model. Secondly, we also stress that in certain ranges of the parameters, the model is badly conditioned, thus estimated parameters are unstable given small perturbations of the data. We discuss to what extent these difficulties affect applications of the model.

1 Introduction

The model of Nelson and Siegel (1987) and its extension by Svensson (1994) are widely used by central banks and other market participants as a model for the term structure of interest rates (Gimenoa and Nave, 2009; BIS, 2005). Academic studies have provided evidence that the model can also be a valuable tool for forecasting the term structure, see for instance Diebold and Li (2006). Model calibration, ie, obtaining parameter values such that model yields accord with market yields, is difficult; many authors have reported 'numerical difficulties' when working with the model (for instance, Bolder and Stréliski, 1999; Gurkaynak et al., 2006; De Pooter, 2007) . In this paper we analyse the calibration of the model in more detail. We argue that the problem is twofold: firstly, the optimisation problem is not convex and has multiple local optima. Hence methods that are readily available in statistical packages – in particular methods based on derivatives of the objective function – are not appropriate to obtain parameter values. We implement and test an optimisation heuristic, Differential Evolution, to obtain parameters. We find that Differential Evolution gives solutions that fit the data very well. Secondly, we also stress that in certain ranges of the parameters, the model is badly conditioned, thus estimated parameters are unstable given small perturbations of the data. We discuss to what extent these difficulties affect applications of the model.

[*]University of Geneva, Switzerland. [†]NordLB, Hanover, Germany. Corresponding author: `enrico.schumann AT unige.ch`. M. Gilli and E. Schumann gratefully acknowledge financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF.

The paper is structured as follows: Section 2 introduces the Nelson–Siegel and Nelson–Siegel–Svensson models and presents an estimation experiment. Section 3 explains the collinearity problem; Section 4 compares results for alternative estimation techniques. Section 5 concludes.

2 Models and estimation

We look into the two main variants of the model, namely the original formulation of Nelson and Siegel (1987), and the extension of Svensson (1994). De Pooter (2007) gives an overview of other variants.

Nelson and Siegel (1987) suggested to model the yield curve at a point in time as follows: let $y(\tau)$ be the zero rate for maturity τ, then

$$y(\tau) = \beta_1 + \beta_2 \left[\frac{1-\exp(-\tau/\lambda)}{\tau/\lambda}\right] + \beta_3 \left[\frac{1-\exp(-\tau/\lambda)}{\tau/\lambda} - \exp(-\tau/\lambda)\right] . \tag{1}$$

Thus, for given a given cross-section of yields, we need to estimate four parameters: β_1, β_2, β_3, and λ. For m observed yields with different maturities $\tau_1, \dots, \tau_m$, we have m equations. There is a simple strategy to obtain parameters for this model: fix a λ, and then estimate the β-values with Least Squares (Nelson and Siegel, 1987, p. 478); see also below. We do not assume that the model's parameters are constant, but they can change over time. To simplify notation, we do not add subscripts for the time period.

In the Nelson–Siegel (NS) model, the yield y for a particular maturity is hence the sum of several components. β_1 is independent of time to maturity, and so it is often interpreted as the long-run yield level. β_2 is weighted by a function of time to maturity. This function is unity for $\tau = 0$ and exponentially decays to zero as τ grows, hence the influence of β_2 is only felt at the short end of the curve. β_3 is also weighted by a function of τ, but this function is zero for $\tau = 0$, increases, and then decreases back to zero as τ grows. It thus adds a hump to the curve. The parameter λ affects the weight functions for β_2 and β_3; in particular does it determine the position of the hump. An example is shown in Figures 1 to 3. The parameters of the model thus have, to some extent, a direct (observable) interpretation, which brings about the constraints

$$\beta_1 > 0\,, \qquad \beta_1 + \beta_2 > 0\,.$$

We also need to have $\lambda > 0$.

The Nelson–Siegel–Svensson (NSS) model adds a second hump term (see Figure 3) to the NS model. Let again $y(\tau)$ be the zero rate for maturity τ, then

$$\begin{aligned} y(\tau) = \beta_1 + \beta_2 \left[\frac{1-\exp(-\tau/\lambda_1)}{\tau/\lambda_1}\right] + \\ \beta_3 \left[\frac{1-\exp(-\tau/\lambda_1)}{\tau/\lambda_1} - \exp(-\tau/\lambda_1)\right] + \beta_4 \left[\frac{1-\exp(-\tau/\lambda_2)}{\tau/\lambda_2} - \exp(-\tau/\lambda_2)\right] . \end{aligned} \tag{2}$$

Here we need to estimate six parameters: β_1, β_2, β_3, β_4, λ_1 and λ_2. The constraints remain

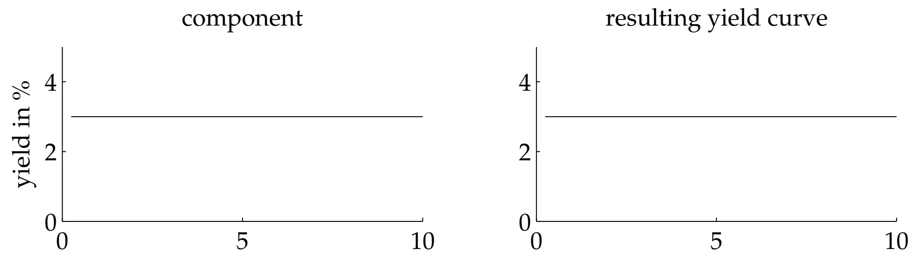


Figure 1: Level. The left panel shows $y(\tau) = \beta_1 = 3$. The right panel shows the corresponding yield curve, in this case also $y(\tau) = \beta_1 = 3$. The influence of β_1 is constant for all τ.

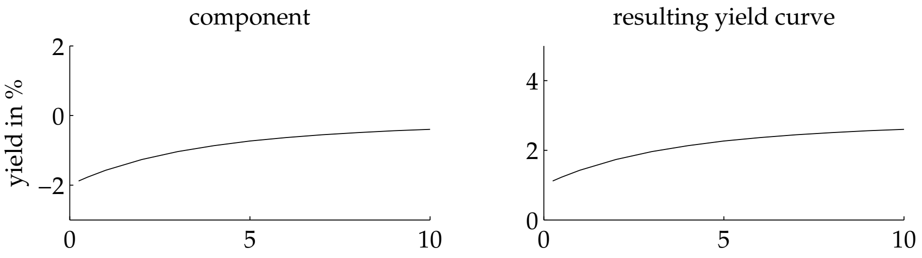


Figure 2: Short-end shift. The left panel shows $y(\tau) = \beta_2 \left[\frac{1-\exp(-\tau/\lambda)}{\tau/\lambda}\right]$ for $\beta_2 = -2$. The right panel shows the yield curve resulting from the effects of β_1 and β_2, ie, $y(\tau) = \beta_1 + \beta_2 \left[\frac{1-\exp(-\tau/\lambda)}{\tau/\lambda}\right]$ for $\beta_1 = 3$, $\beta_2 = -2$. The short-end is shifted down by 2%, but then curve grows back to the long-run level of 3%.

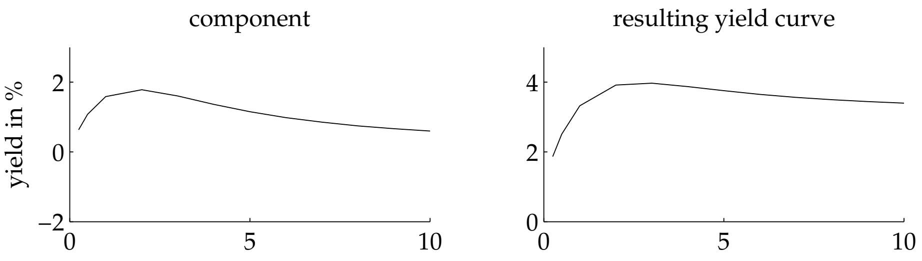


Figure 3: Hump. The left panel shows $\beta_3 \left[\frac{1-\exp(-\tau/\lambda)}{\tau/\lambda} - \exp(-\tau/\lambda)\right]$ for $\beta_3 = 6$. The right panel shows the yield curve resulting from all three components. In all panels, λ is 2.

the same, but we also have $\lambda_{1,2} > 0$. Like for NS, we could fix the λ-values – ie, use a grid of different values –, and then run a Least Squares algorithm to obtain parameter estimates, even though we cannot handle inequality constraints with a standard Least Squares solver.

More generally, the parameters of the models can be estimated by minimising the difference between the model rates y, and observed rates y^M where the superscript stands for 'market'. An optimisation problem can be stated as

$$\min_{\beta,\lambda} \sum \left(y - y^M\right)^2 \tag{3}$$

subject to the constraints given above. (If the model were correctly specified, the parameters would have a clear economic interpretation, and constraints should not be necessary. Practically, they should be included, since we are not guaranteed to obtain reasonable values from a numeric procedure.)

There are many variants of this objective function: we could use absolute values instead of squares, or a more robust function of scale. Likewise, we could use bond prices instead of rates, and so on. (If we use the Least-Squares-with-fixed-λ-values strategy, this automatically sets our objective function.) This paper deals with the numerical aspects of the calibration, the decision on which specification to use should rather follow from empirical tests which are beyond the scope of this study, so we work with specification (3). Below we will apply an optimisation method that is capable of estimating all parameters in one step for different variants of the objective function, and under different constraints.

An estimation example

On each business day Germany's central bank, the Bundesbank, fits an NSS model to the yields of German government bonds (Deutsche Bundesbank, 1997). On 15 September 2009, the fitted curve looked like the one in Figure 4. The corresponding NSS parameters were

$$\begin{aligned}\beta_1 &= 2.05, & \beta_2 &= -1.82, & \beta_3 &= -2.03,\\ \beta_4 &= 8.25, & \lambda_1 &= 0.87, & \lambda_2 &= 14.38\,.\end{aligned}$$

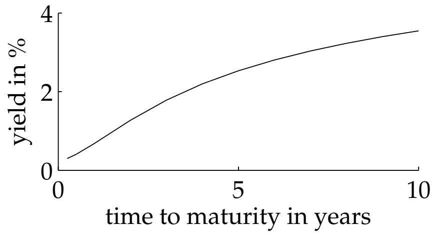


Figure 4: NSS yield curve for German government bond yields, as of 15 September 2009.

Inserting these parameters and various maturities into Equation (2), we obtain the data in Table 1. For convenience, we write yields and model parameters in terms of percentage points, so for instance five percent are written as 5.0 instead of 0.05. In practice, parameter estimates are often computed by running a standard optimisation technique, like a steepest descent (or more generally methods based on the derivatives of the objective function) or direct search. So next we try to fit the NSS model with such methods; in principle we should be able to back out the parameters exactly. We try to find NSS parameters that solve problem (3) for the 'market data' from Table 1. We use the function `nlminb` from R's `stats` package; this function is also used in the `termstrc` package (Ferstl and Hayden, 2009).

maturity in years	1/4	1/2	1	2	3	4	5	6
spot rate (in %)	0.30	0.40	0.68	1.27	1.78	2.20	2.53	2.80
maturity in years	7	8	9	10	15	20	25	30
spot rate (in %)	3.03	3.23	3.40	3.54	4.04	4.28	4.38	4.38

Table 1: NSS yields on 15 September 2009.

We run the algorithm 500 times; for each restart we randomly choose a different starting value. We set box constraints as follows: $0 < \beta_1 < 15$, $-15 < \beta_2 < 30$, $-30 < \beta_3 < 30$, $-30 < \beta_4 < 30$, $0 < \lambda_1 < 30$, $0 < \lambda_2 < 30$; the choice of these values should become apparent from Figure 6. The starting values are randomly drawn from these ranges. Histograms of parameter estimates are shown in Figure 5. We repeated the procedure with other algorithms like Matlab's `fminsearch`, which implements a Nelder–Mead direct search, and `fminunc`; we always obtained similar results.

The estimated parameter values are mostly quite different from their true counterparts. They do not even seem centred around the true values (indicated by the vertical lines). We seem not to be the only ones to have problems with the estimation: Figure 6 shows the daily estimates of parameters supplied by the Bundesbank. These data can be obtained from `http://www.bundesbank.de/statistik/statistik_zinsen.php`; the relevant series are WT3201, WT3202, WT3203, and WT3205 for the β estimates; and WT3204, WT3206 for the λ. The estimates vary widely from one day to the next, with parameters often at their boundaries (apparently, the Bundesbank used box constraints, which is why we chose constraints in the same fashion). Given that some of the parameters have economic meaning, such wild variations seem strange. Consider for instance the path of β_1, the long-run level of interest. From one day to the next, estimates jump by several percentage points.

So we have difficulties in recovering the parameters, but what if our aim had been to fit yield curves? Figure 7 shows the maximum absolute differences between the estimated and the true yield curve for our 500 runs. The yield fit is much better than the parameter fit; most errors are of a magnitude of about 10 basis points. But again, by construction, a perfect fit would have been possible; and in some cases there remain substantial deviations. In particular, and not visible from the histogram, we sometimes have estimates that are way off the true parameters, with a maximum error in the yield curve of sometimes more than one percentage point. In our runs, the maximum absolute error had been 1.8%.

To sum up this example: NSS works well to interpolate observed yields; but when estimating the model with a standard optimisation method, several restarts with different starting values are imperative. Yet if our aim is to identify the parameters, for instance to model their evolution over time (like in Diebold and Li, 2006), we need to be more careful how we set up the optimisation problem.

There are two reasons for these estimation difficulties. Firstly, the objective function is not convex and exhibits several local minima. This is particularly true for the NSS model. Figure 8 shows the value of the objective function (sum of squares) when we vary two parameters (here β_1 and λ_2) while keeping the remaining parameters fixed. This is relevant

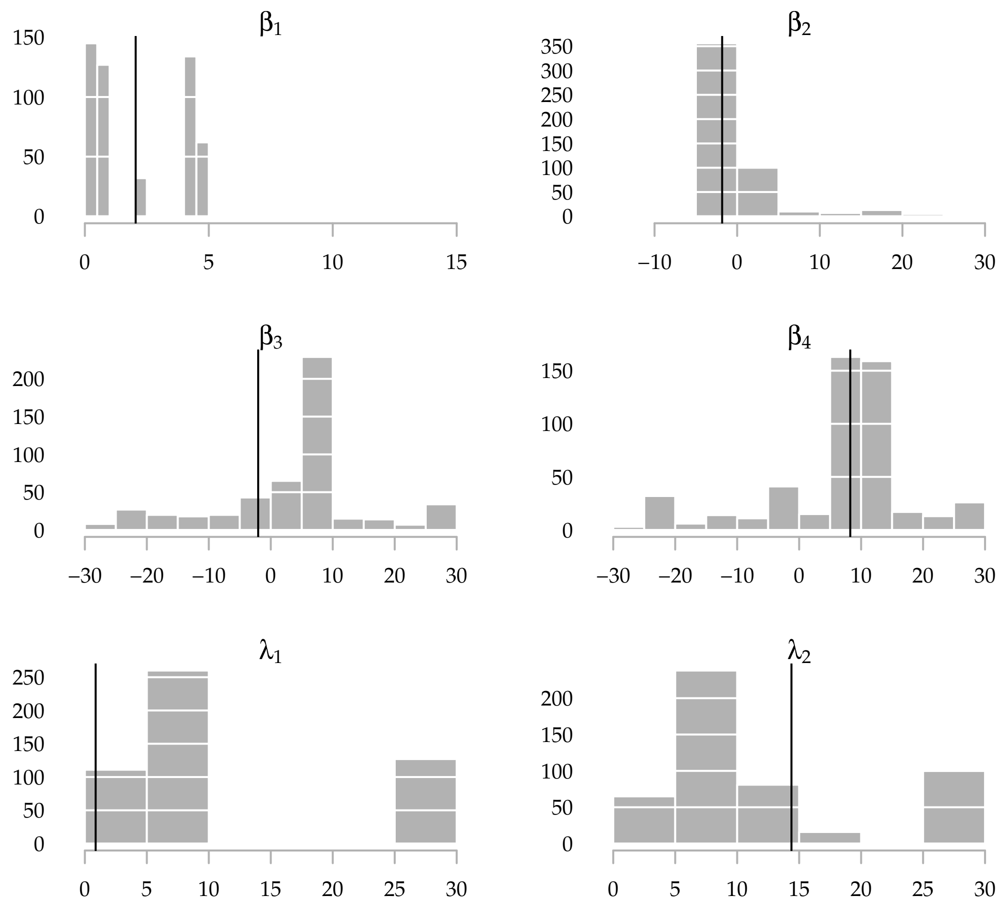


Figure 5: Parameter estimates from a gradient search.

when we wish to obtain model yields close to observed yields; better methods may improve the fit. We will refer to this as the *optimisation problem*. A second problem is *collinearity*, and this stems from the model specification, not from an inappropriate optimisation method. We will discuss both problems in the next two sections.

3 The collinearity problem

Both NS and NSS can be interpreted as factor models (Diebold and Li, 2006): the β-coefficients are the factor realisations; the factor loadings are the weight functions of these parameters. For the NS-model the loadings for a maturity τ are thus given by

$$\left[1 \qquad \frac{1-\exp(-\tau/\lambda)}{\tau/\lambda} \qquad \frac{1-\exp(-\tau/\lambda)}{\tau/\lambda} - \exp(-\tau/\lambda)\right]' . \tag{4}$$

So by setting λ, we impose fixed factor loadings on a given maturity. The three factors are then interpreted as level (β_1), steepness (β_2), and curvature (β_3). Explaining the yield curve through these three factors is empirically well-supported, see for instance Litterman and Scheinkman (1991). We can also specify these factors in a model-free way: the level

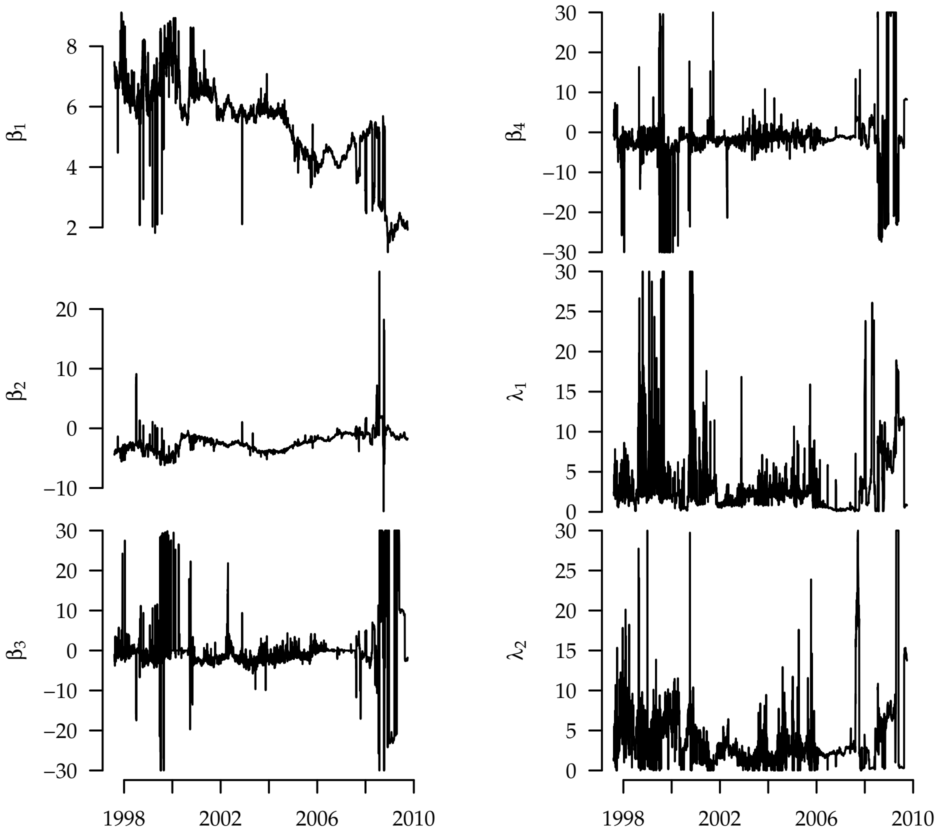


Figure 6: The Bundesbank's daily estimates for NSS parameters.

of yields could be the average yield, steepness can be measured by the yield difference between long-dated and short-dated bonds. Diebold and Li (2006) for instance suggest to define the level as the ten-year rate $y^M(10)$; steepness as the ten-year rate minus the three-month rate $y^M(10) - y^M(1/4)$; and curvature as $2y^M(2) - y^M(1/4) - y^M(10)$. We can, for each cross-section, compute these factors and observe their evolution over time. Empirically, the three factors are found to be only mildly correlated. This is also corroborated by studies that build on principal component analysis or assume zero correlations in factor analysis, yet still arrive at these factors, like Litterman and Scheinkman (1991).

In the NS-case with m different maturities $\tau_1, \ldots, \tau_m$ and with a fixed λ-value, we have

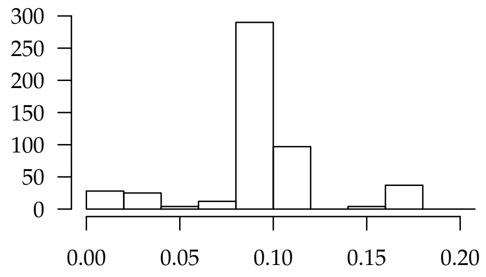


Figure 7: Maximum absolute deviations between model and true yield curves.

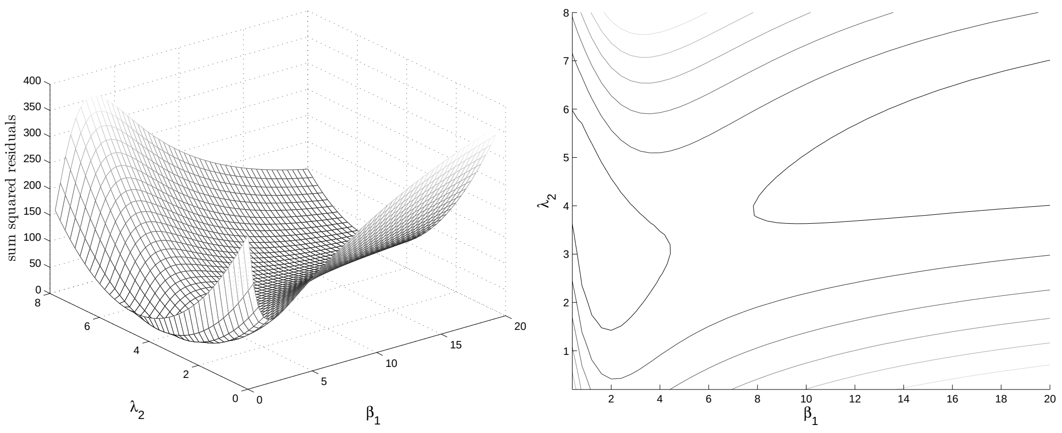


Figure 8: An example search space for the NSS model.

m linear equations from which to estimate three parameters. So we need to solve

$$\begin{pmatrix} 1 & \frac{1-\exp(-\tau_1/\lambda)}{\tau_1/\lambda} & \frac{1-\exp(-\tau_1/\lambda)}{\tau_1/\lambda} - \exp(-\tau_1/\lambda) \\ 1 & \frac{1-\exp(-\tau_2/\lambda)}{\tau_2/\lambda} & \frac{1-\exp(-\tau_2/\lambda)}{\tau_2/\lambda} - \exp(-\tau_2/\lambda) \\ 1 & \frac{1-\exp(-\tau_3/\lambda)}{\tau_3/\lambda} & \frac{1-\exp(-\tau_3/\lambda)}{\tau_3/\lambda} - \exp(-\tau_3/\lambda) \\ \vdots & \vdots & \vdots \\ 1 & \frac{1-\exp(-\tau_m/\lambda)}{\tau_m/\lambda} & \frac{1-\exp(-\tau_m/\lambda)}{\tau_m/\lambda} - \exp(-\tau_m/\lambda) \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} y^M(\tau_1) \\ y^M(\tau_2) \\ y^M(\tau_3) \\ \vdots \\ \vdots \\ y^M(\tau_m) \end{pmatrix} \tag{5}$$

for β. We can interpret the NSS-model analogously, just now we have to fix two parameters, λ_1 and λ_2. Then we have a fourth loading

$$\left[\dots \frac{1-\exp(-\tau_i/\lambda_2)}{\tau_i/\lambda_2} - \exp(-\tau_i/\lambda_2)\right]', \tag{6}$$

ie, a fourth regressor, and can proceed as before. This system of equations is overidentified for the practical case $m > 3$ (or $m > 4$ for NSS), so we need to minimise a norm of the residuals. With Least Squares, we use the 2-norm.

A well-known result from numerical analysis is that the size of the minimised residual is not necessarily influenced by the conditioning of the equations. This also holds if we do not fix the λ-values, and thus have to solve non-linear equations. Even for badly-conditioned problems, we may obtain small residuals (ie, a good fit), but we cannot accurately compute the parameters any more. In other words, many different parameter values give similarly-good fits.

This is the problem here. For many values of λ, the factor loadings are highly correlated, and hence Equations (5) are badly conditioned; we have an identification problem. (Note that we use the word 'badly-conditioned' in a loose sense here. Numerically, the problem can usually be solved. But economic interpretation should stop well before a problem cannot be numerically solved any more.) The most obvious case occurs in the NSS model if λ_1 and λ_2 are roughly equal: then β_3 and β_4 will have the same factor loading, we have two perfectly collinear regressors. This extreme case was noticed before, for instance in Xiao (2001) or De Pooter (2007). But the correlation becomes a problem well before the λ-values are equal, and it is a problem in the simpler NS model, too.

In the NS model, for many values of the λ-parameter, the correlation between the second and the third loading is high, thus the attribution of a particular yield curve shape to the specific factor becomes difficult. Figure 9 shows the correlation between the factor loadings for different values of λ. We see that the correlation is 1 at a λ of zero, and rapidly decays to

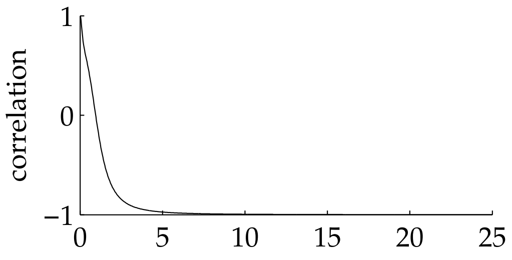


Figure 9: NS: Correlations between factor loadings of β_2 and β_3 for different λ.

-1 as λ grows. Outside a range from 0.1 to 4 or so, an optimisation procedure can easily trade off changes in one variable against changes in the other. This is troublesome for numeric procedures since they then lack a clear indication into which direction to move. We obtain the same results for the NSS-model. Figure 10 shows the correlation between the second and the third, the second and the fourth, and the third and the fourth factor loading, respectively. Again, we see that for λ-values greater than about 5 or so, the correlation rapidly reaches either 1 or -1. The correlations here were computed for maturities up to 10 years. The collinearity is mitigated when we include long-term yields, but only somewhat: even when we have, for instance, yields for maturities up to 20 years, correlation is below -0.9 for $\lambda > 6$ in the NS-model. So we have the expect large estimation errors; estimation results will be sensitive to small changes in the data.

If we only want to obtain a tractable approximation to the current yield curve, for instance for pricing purposes, we need not care much. True, many different parameter values give similar fits; but we are not interested in the parameters, only in interpolation. How

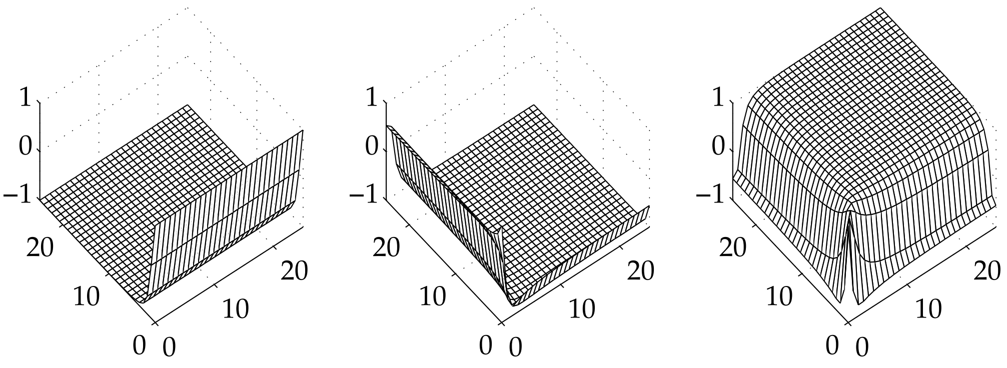


Figure 10: NSS: Correlations between factor loadings for different λ.

about forecasting? Correlated regressors are not necessarily a problem in forecasting. We are often not interested in disentangling the effects of two single factors as long as we can assess their combined effect. The problem changes if we want to predict the regression coefficients themselves. Diebold and Li (2006) for instance use the NS-model to forecast interest rates. They first fix λ at 1.4, and then estimate the β-values by Least Squares as explained above. That is, for each cross-section of yields, they run a regression, and so obtain a time series of β-values. They then model the β-values as AR(1)-processes, and use these to predict future β-values and hence yield curves. Note that according to our analysis above, their λ value is well-chosen, as it implies only a weak negative correlation between the factor loadings.

To demonstrate the effects of high correlation, we replicate some of the results of Diebold and Li (2006). Their data set comprises monthly zero rates for maturities 1/12, 3/12, 6/12, 9/12, 1, 2, . . . , 10 years. Altogether, there are 372 yield curves, from January 1970 to December 2000. These data are available from `http://www.ssc.upenn.edu/~fdiebold/YieldCurve.html`. We first set the value of λ to 1.4 as in Diebold and Li (2006). (We measure time to maturity in years. Diebold and Li (2006) use months and also define the model slightly differently; their stated λ is 0.0609, which translates into $1/(12 \cdot 0.0609) \simeq 1.4$ in our formulation.) Then we run regressions to obtain β time series; these are shown in Figure 11. We have slightly rescaled the series to make them correspond to Figure 7 in Diebold and Li (2006, p. 350): for β_2 we switch the sign, β_3 is multiplied by 0.3. We would like to remark here that it was a pleasing experience to be able to really replicate these results from Diebold's and Li's paper.

Next, we run regressions with λ at 10. Figure 9 suggest that then the weight functions of β_2 and β_3 will be strongly negatively correlated, ie, we cannot accurately estimate β_2 and β_3 any more. Figure 12 shows the obtained time series in black lines. Note that the y-scales have changed; the grey lines are the time series that were depicted in Figure 11. We see that the series are much more volatile, and thus are very likely more difficult to model. We can also see that the β-series look very similar; the correlation between β_2 and β_3 is 0.98, very different from the empirically low correlation between the factors. Correlation between the regressors affects the sampling correlation of the coefficients (not just their variances). In a linear regression model like (5), the correlation between parameter estimates is inversely

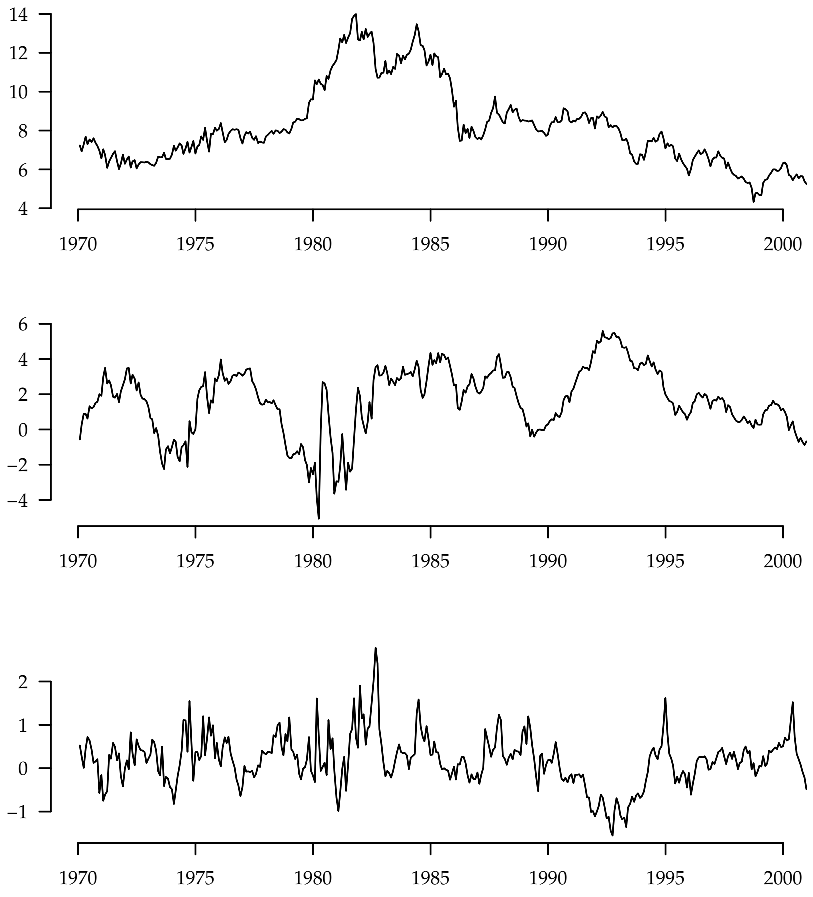


Figure 11: Time series of β_1, -β_2, and 0.3β_3. The rescaling corresponds to Diebold and Li (2006, p. 350, Figure 7). λ is set to 1.4.

related to the correlation between the regressors. If the regressors are negatively correlated (as they are for a λ of 10), the estimated β-values will be positively correlated. The constant β_1 is also affected since it is closely connected to β_2. (β_1 represents the level-factor, β_2 the difference between the curve's short end and the level. When β_1 increases, β_2 needs to increase as well to fit the short maturities.) So in sum, if we aim to meaningfully estimate parameters for the NS or the NSS model, we need to restrict the λ-values to ranges where practical identification is still possible. For both the NS and the NSS-case, this means a λ up to about 4 or 5; for the NSS-model we should make sure that the λ-values do not become too similar. The exact correlations for a particular case can readily be calculated by fixing the maturities with which to work, inserting them into Equations 5, and computing the correlations between the columns.

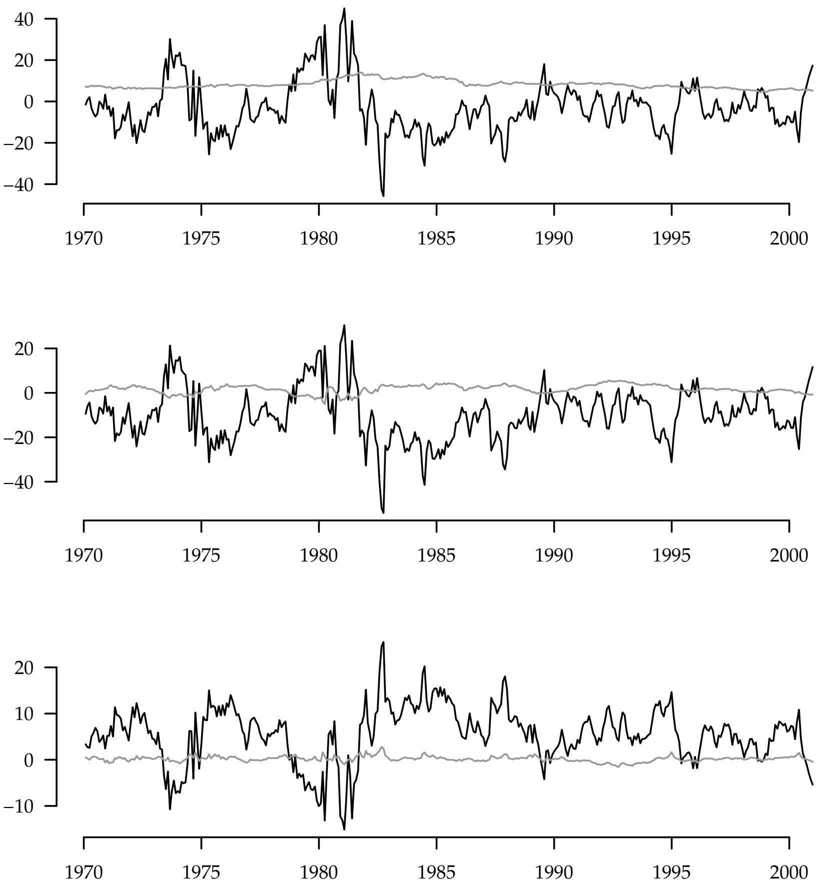


Figure 12: Time series of β_1, -β_2, and 0.3β_3. The rescaling corresponds to Diebold and Li (2006, p. 350, Figure 7). λ is set to 10. The grey lines are the time series from Figure 11.

4 The optimisation problem

In this section, we will deal with the optimisation problem, so our aim will be to solve model (3). The problem is not convex, so we will use appropriate procedures: optimisation heuristics (Gilli and Winker, 2009). More specifically, we will apply Differential Evolution (DE; Storn and Price, 1997). We will not discuss the algorithm in detail here; the implementation follows the pseudocode given in Gilli and Schumann (2010 (in press). R-code is given in the appendix and can be downloaded from `http://comisef.eu`. We parameterise DE as follows: `F` is 0.5, `CR` is 0.99. We use a population of 200 solutions and stop the algorithm after 600 generations. (For a discussion of the meaning of these parameters, see Gilli and Schumann, 2010 (in press.) To handle constraints in DE, we include a penalty function that adds a scalar (proportional to the violation of a given constraint) to a solution. We include the following restrictions:

$$0 \leq \beta_1 \leq 15, \quad -15 \leq \beta_2 \leq 30, \quad -30 \leq \beta_3 \leq 30, \quad -30 \leq \beta_4 \leq 30,$$
$$0 \leq \lambda_1 \leq 2.5, \quad 2.5 \leq \lambda_2 \leq 5.5\,.$$

The chosen values for λ_1 and λ_2 should result in acceptable correlations. The computing time for a run of DE in R 2.8.1 on an Intel P8700 (single core) at 2.53GHz with 2GB RAM is less than 10 seconds.

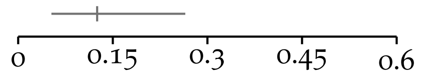
The Experiment

We use again the data set from Diebold and Li (2006), monthly zero rates for maturities 1/12 to 10 years; see the previous section. We have 372 cross-sections of yields. For each cross-section, we fit an NSS model by running ten times a gradient-based search (`nlminb` from R's `stats` package), and DE. The starting values for `nlminb` are drawn randomly from the allowed ranges of the parameters; in the same way we set up the initial population of DE.

For each cross-section of yields (ie, each month) we have ten solutions obtained from `nlminb`, and ten from DE. For each solution, we compute the root-mean-squared (rms) error in percentage points, ie,

$$\sqrt{\frac{1}{m}\sum_{i=1}^{m}\Big(y^M(\tau_i) - y(\tau_i)\Big)^2}\,. \tag{7}$$

Since the problem is not convex, we should not expect to obtain the same error for different restarts, not even – or rather, in particular not – for `nlminb`, given that we use different starting values for each restart. Thus, for each month for which we calibrate the model, we normally have ten different solutions for a given method. We compute the worst of these solutions in terms of fit as given by Equation (7), the best solution, and the median. We plot the results in Figures 13 and 14. The grey bars show the range from minimum error to maximum error for a given month; the vertical ticks indicate the median fit. An example: for May 1984, gradient search's best solution resulted in a rms error of 5.3 basis points (bp), its worst solution had an error of 26.5 bp, the median error was 13.6 bp. Thus we get:



As a third column ('△ of medians'), we plot the difference between the median solution for DE, and the median solution for gradient search. If this quantity is negative, DE returned better solutions; if it is positive, gradient search gave better results. We see from Figures 13 and 14 that with very few exceptions, DE had better median solutions.

If we just compare the average error for gradient search and DE, we find them not too different. The median-median rms error is 5.4 bp for DE, compared with 8.1 bp for gradient search. ('median-median' means: for each month, we have ten results for each method, of which we compute the median; then we take median over all these median values, ie, we average over all months.) This underlines the point made in Section 2 that we can find acceptable fits even with a gradient-based method, even though it is actually not an appropriate method. We stress again, however, that several restarts are required.

But the impression that Figures 13 and 14 give is that while repeated runs of DE also result in different solutions, these solutions are much more stable than for `nlminb`. The median range over all months for DE is exactly zero, the mean range is 0.2 bp; in 97% of all cases was the range for DE smaller than one basis point! In contrast, gradient search has a median range of 6.1 bp (mean range 8.1 bp); a range smaller than one basis point was only achieved in 8% of cases.

Given that DE found solutions at least as good and mostly better than gradient search, and more reliably so, and that computation time is not prohibitive, we conclude that for this particular problem, DE is more appropriate than a traditional optimisation technique based on the gradient.

5 Conclusion

In this paper we have analysed the calibration of the Nelson–Siegel and Nelson–Siegel–Svensson model. Both models are widely used, yet it is rarely discussed that fitting the models to market rates often causes problems. We have shown that these difficulties can possibly be reduced by using alternative optimisation techniques. Differential Evolution, which we tested, gave results that were reliably better than those obtained by a traditional method based on the derivatives of the objective function. But these improvements concern the fit, that is, the discrepancy between market rates and model rates. We also showed that parameter identification is only possible when specific parameters are restricted to certain ranges; unconstrained optimisation runs the risk of moving into parameter ranges where single parameters cannot be accurately computed any more.

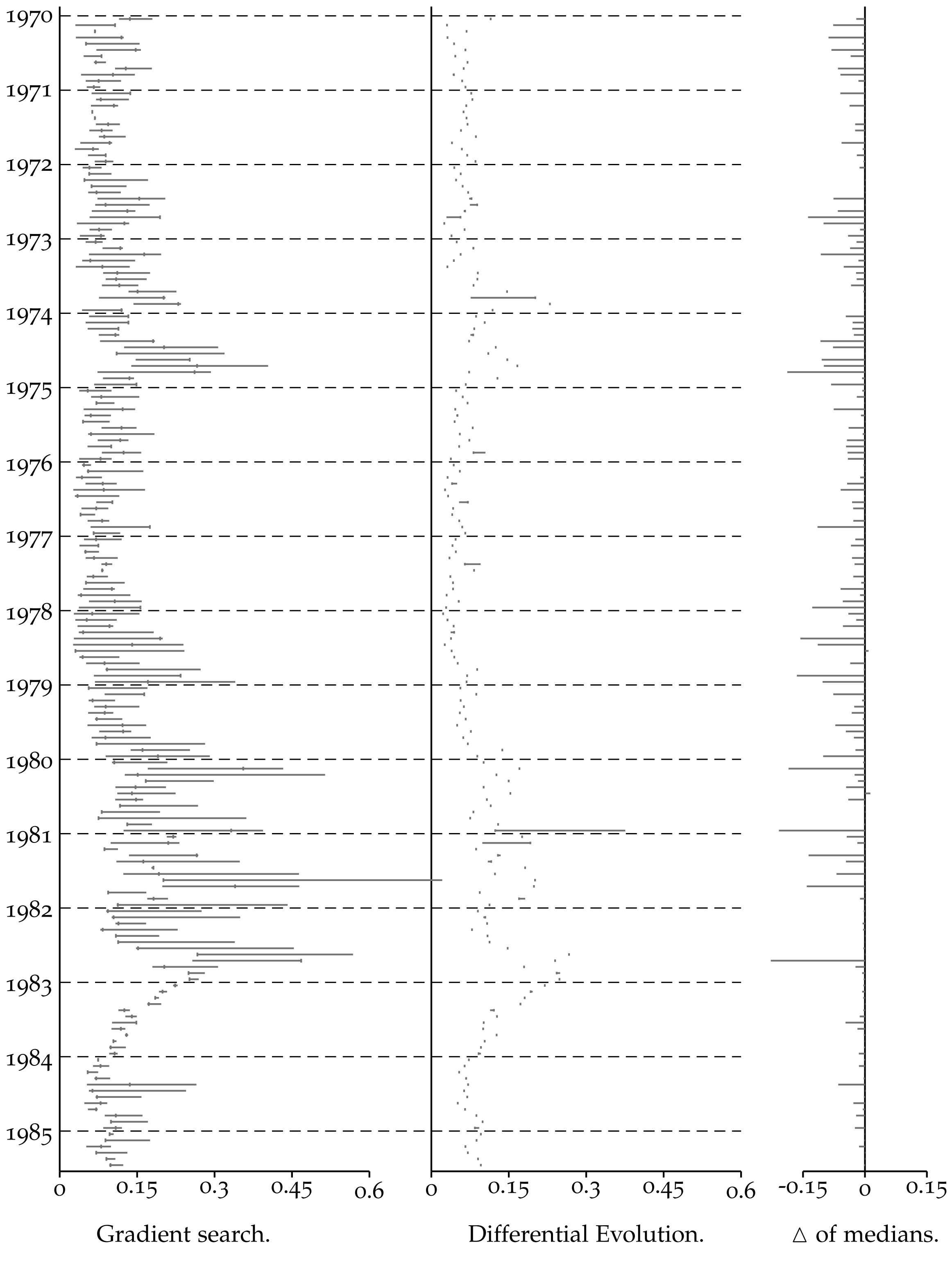


Figure 13: Errors distributions and differences in median solutions.

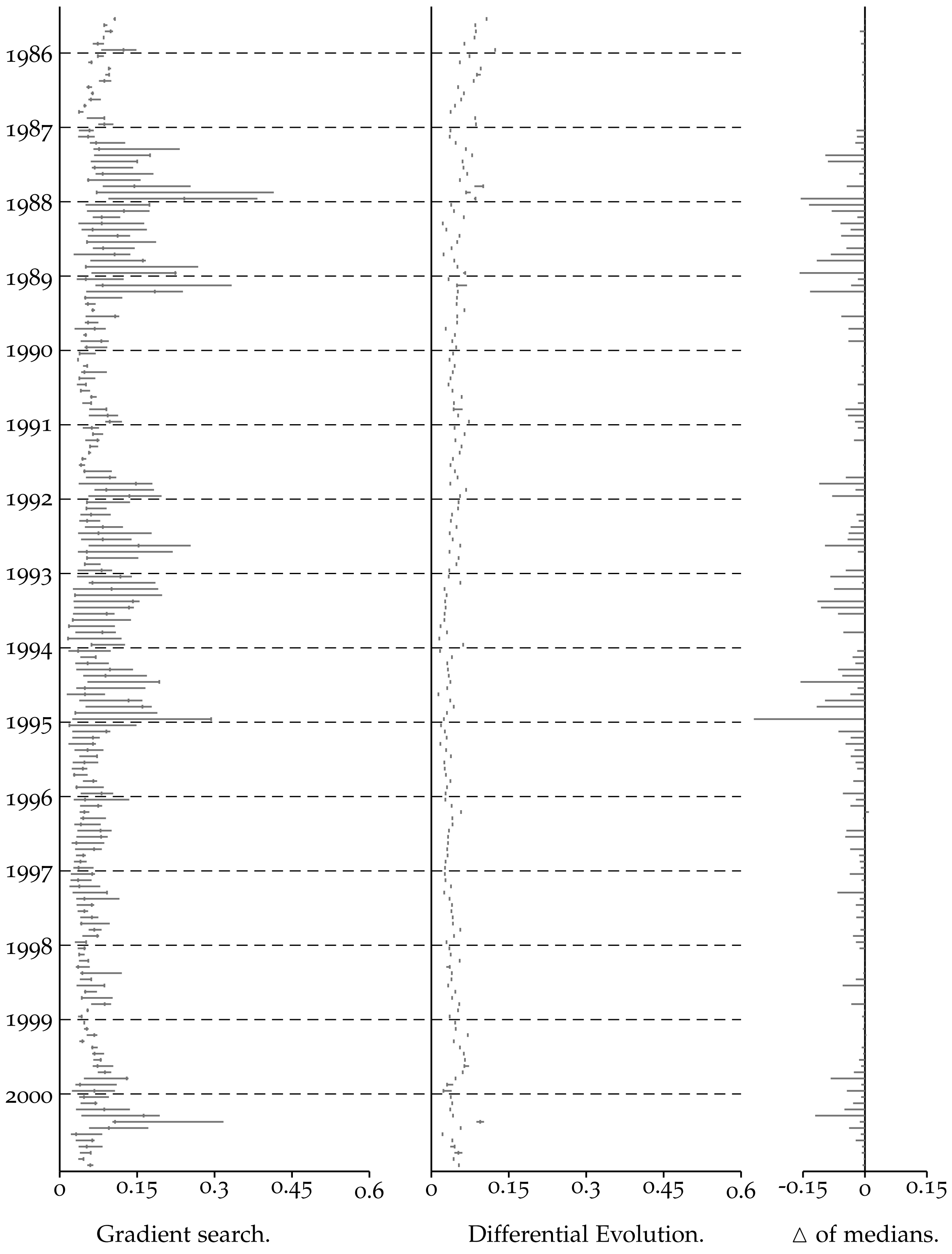


Figure 14: Errors distributions and differences in median solutions.

References

BIS. Zero-Coupon Yield Curves: Technical Documentation. BIS Papers 25, Bank for International Settlements, 2005.

David Bolder and David Stréliski. Yield Curve Modelling at the Bank of Canada. *Bank of Canada Technical Report*, 84, 1999.

Michiel De Pooter. Examining the Nelson–Siegel Class of Term Structure Models. *Tinbergen Institute Discussion Paper 2007-043/4*, 2007.

Deutsche Bundesbank. Schätzung von Zinsstrukturkurven. *Monatsbericht*, Oktober 1997.

Francis X. Diebold and Canlin Li. Forecasting the Term Structure of Government Bond Yields. *Journal of Econometrics*, 130(2):337–364, 2006.

Robert Ferstl and Josef Hayden. `termstrc`: Zero-coupon Yield Curve Estimation. 2009. URL `http://CRAN.R-project.org/package=termstrc`. R package version 1.1.1.

Manfred Gilli and Enrico Schumann. Robust Regression with Optimisation Heuristics. In Anthony Brabazon, Michael O'Neill, and Dietmar Maringer, editors, *Natural Computing in Computational Finance*, volume 3. Springer, 2010 (in press).

Manfred Gilli and Peter Winker. Heuristic optimization methods in econometrics. In David A. Belsley and Erricos Kontoghiorghes, editors, *Handbook of Computational Econometrics*. Wiley, 2009.

Ricardo Gimenoa and Juan M. Nave. A Genetic Algorithm Estimation of the Term Structure of Interest Rates. *Computational Statistics & Data Analysis*, 53:2236–2250, 2009.

Refet S. Gurkaynak, Brian Sack, and Jonathan H. Wright. The U.S. Treasury Yield Curve: 1961 to the Present. *Federal Reserve Board Finance and Economics Discussion Series*, 2006-28, 2006.

Robert Litterman and José Scheinkman. Common Factors Affecting Bond Returns. *Journal of Fixed Income*, 1(1):54–61, 1991.

Charles R. Nelson and Andrew F. Siegel. Parsimonious Modeling of Yield Curves. *Journal of Business*, 60(4):473–489, 1987.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Rainer M. Storn and Kenneth V. Price. Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

Lars E.O. Svensson. Estimating and Interpreting Forward Interest Rates: Sweden 1992–1994. *IMF Working Paper 94/114*, 1994.

Jerry Yi Xiao. Term Structure Estimation for U.S. Corporate Bond Yields. *RiskMetrics Journal*, 2(1):19–34, 2001.

A R-code: Example

To use the example, first run the script `de.r` (`source("de.r")`).

```r
# ----------------------------------------------------------------
# example 1: NS
# ----------------------------------------------------------------

# set up yield curve (in this example: artificial data), and plot it
mats <- c(1,3,6,9,12,15,18,21,24,30,36,48,60,72,84,96,108,120)/12
betaTRUE <- c(6,3,8,1); yM <- NS(betaTRUE,mats)
plot(mats,yM,xlab="maturities in years",ylab="yields in %")

# collect all data in dataList
dataList <- list(yM = yM, mats = mats, model = NS)

# set parameters for de
de <- list(
                min      = c( 0,-15,-30,0),
                max      = c(15, 30, 30,10),
                d        = 4,
                nP       = 200,
                nG       = 600,
                ww       = 0.1,
                F        = 0.50,
                CR       = 0.99,
                R        = 0.00 # random term (added to change)
)

system.time(sol <- DE(de = de,dataList = dataList,OF = OF))
# maximum error
max(abs(dataList$model(sol$beta,mats)-dataList$model(betaTRUE,mats)))
# value of objective function
sqrt(sol$OFvalue)
lines(mats,dataList$model(sol$beta,mats), col="blue")

s0 <- de$min + (de$max - de$min) * runif(de$d)
system.time(sol2 <- nlminb(s0,OF,data=dataList,lower = de$min, upper = de$max, control
     = list(eval.max=50000,iter.max=50000)))
# maximum error
max(abs(dataList$model(sol2$par,mats)-dataList$model(betaTRUE,mats)))
# value of objective function
sqrt(sol2$objective)
lines(mats,dataList$model(sol2$par,mats), col="green", lty=2)

legend(x  = "bottom", legend = c("true yields","DE","nlminb"),
        col = c("black","blue","green"),
        pch = c(1,NA,NA), lty=c(0,1,2))




# ----------------------------------------------------------------
# example 2: NSS
# ----------------------------------------------------------------

# set up yield curve (in this example: artificial data), and plot it
mats <- c(1,3,6,9,12,15,18,21,24,30,36,48,60,72,84,96,108,120)/12
betaTRUE <- c(5,-2,5,-5,1,3); yM <- NSS2(betaTRUE,mats)
plot(mats,yM,xlab="maturities in years",ylab="yields in %")

# collect all in dataList
```

```r
dataList <- list(yM = yM, mats = mats, model = NSS2)

# set parameters for de
de <- list(
                 min     = c( 0,-15,-30,-30,0  ,2.5),
                 max     = c(15, 30, 30, 30,2.5,5  ),
                 d       = 6,
                 nP      = 200,
                 nG      = 600,
                 ww      = 0.1,
                 F       = 0.50,
                 CR      = 0.99,
                 R       = 0.00 # random term (added to change)
)

system.time(sol <- DE(de = de,dataList = dataList,OF = OF))
# maximum error
max(abs(dataList$model(sol$beta,mats)-dataList$model(betaTRUE,mats)))
# value of objective function
sqrt(sol$OFvalue)
lines(mats,dataList$model(sol$beta,mats), col="blue")

s0 <- de$min + (de$max - de$min) * runif(de$d)
system.time(sol2 <- nlminb(s0,OF,data=dataList,lower = de$min, upper = de$max, control
     = list(eval.max=50000,iter.max=50000)))
# maximum error
max(abs(dataList$model(sol2$par,mats)-dataList$model(betaTRUE,mats)))
# value of objective function
sqrt(sol2$objective)
lines(mats,dataList$model(sol2$par,mats), col="green", lty=2)

legend(x  = "bottom", legend = c("true yields","DE","nlminb"),
                 col = c("black","blue","green"),
                 pch = c(1,NA,NA), lty=c(0,1,2))
```

B R-code: Differential Evolution

```r
# differential evolution
# enrico schumann, version 2010-03-27
# ---------------------------------------------------------------



# ---------------------------------------------------------------
# optimisation function
# ---------------------------------------------------------------

DE <- function(de,dataList,OF)
{
# auxiliary functions
# ---------------------------------------------------------------
# random numbers: like rand(m,n)/randn(m,n) in Matlab
        mRU <- function(m,n){
                return(array(runif(m*n), dim = c(m,n)))
        }
        mRN <- function(m,n){
                return(array(rnorm(m*n), dim = c(m,n)))
        }
        shift <- function(x)
        {
                rr <- length(x)
                return(c(x[rr],x[1:(rr-1)]))
        }
        # penalty
        pen <- function(mP,pso,vF)
        {
                minV <- pso$min
                maxV <- pso$max
                ww <- pso$ww

                # max constraint: if larger than maxV, element in A is positiv
                A <- mP - as.vector(maxV)
                A <- A + abs(A)

                # max constraint: if smaller than minV, element in B is positiv
                B <- as.vector(minV) - mP
                B <- B + abs(B)

                # beta 1 + beta2 > 0
                C <- ww*((mP[1,]+mP[2,])-abs(mP[1,]+mP[2,]))
                A <- ww * colSums(A + B)*vF - C
                return(A)
        }

        # main algorithm
        # -----------------------------------------------------------------
        # set up initial population
        mP <- de$min + diag(de$max - de$min) %*% mRU(de$d,de$nP)
        # include extremes
        mP[,1:de$d] <- diag(de$max)
        mP[,(de$d+1):(2*de$d)] <- diag(de$min)

        # evaluate initial population
        vF <- apply(mP,2,OF,data = dataList)

        # constraints
        vP <- pen(mP,de,vF)
        vF <- vF + vP

        # keep track of OF
        Fmat <- array(NaN,c(de$nG,de$nP))

        for (g in 1:de$nG){

                # update population
                vI <- sample(1:de$nP,de$nP)
                R1 <- shift(vI)
                R2 <- shift(R1)
                R3 <- shift(R2)

                # prelim. update
                mPv = mP[,R1] + de$F * (mP[,R2] - mP[,R3])
                if(de$R > 0){mPv <- mPv + de$R * mRN(de$d,de$nP)}

                mI <- mRU(de$d,de$nP) > de$CR
                mPv[mI] <- mP[mI]

                # evaluate updated population
                vFv <- apply(mPv,2,OF,data = dataList)
                # constraints
                vPv <- pen(mPv,de,vF)
                vFv <- vFv + vPv
                vFv[!(is.finite(vFv))] <- 1000000


                # find improvements
                logik <- vFv < vF
                mP[,logik] <- mPv[,logik]
                vF[logik] <- vFv[logik]
                Fmat[g,] <- vF

        } # g in 1:nG
        sGbest <- min(vF)
        sgbest <- which.min(vF)[1]

        # return best solution
        return(list(beta = mP[,sgbest], OFvalue = sGbest, popF = vF, Fmat = Fmat))
}



# ------------------------------------------------------------
# define functions
# ------------------------------------------------------------

# nelson--siegel
NS <- function(betaV,mats)
{
        # betaV = beta1-3, lambda1
        gam <- mats / betaV[4]
        y = betaV[1] + betaV[2] * ((1 - exp(-gam)) / (gam)) + betaV[3] * (((1 - exp(-
            gam)) / (gam)) - exp(-gam))
        return(y)
}
# nelson--siegel--svensson 1
NSS <- function(betaV,mats)
{
        # betaV = beta1-4, lambda1-2
        gam1 <- mats / betaV[5]
        gam2 <- mats / betaV[6]
        y <- betaV[1] + betaV[2] * ((1 - exp(-gam1)) / (gam1)) +
                        betaV[3] * (((1 - exp(-gam1)) / (gam1)) - exp(-gam1)) +
                        betaV[4] * (((1 - exp(-gam2)) / (gam2)) - exp(-gam2))
        return(y)
}
# nelson--siegel--svensson 2 (a bit faster)
NSS2 <- function(betaV,mats)
{
        # betaV = beta1-4, lambda1-2
        gam1 <- mats / betaV[5]
        gam2 <- mats / betaV[6]
        aux1 <- 1 - exp(-gam1)
        aux2 <- 1 - exp(-gam2)
        y       <- betaV[1] + betaV[2] * (aux1 / gam1) +
                        betaV[3] * (aux1 / gam1 + aux1 - 1) +
                        betaV[4] * (aux2 / gam2 + aux2 - 1)
        return(y)
}
# generic objective function
OF <- function(betaV,data)
{
        mats            <- data$mats
        yM              <- data$yM
        model            <- data$model
        y                <- model(betaV,mats)
        aux              <- y - yM
        aux              <- crossprod(aux)
        return(aux)
}
```