Special topics: Regularized Continuous Time SEM

In large parts the following content is based on:

Orzek, J. H., & Voelkle, M. C. (2023). Regularized continuous time structural equation models: A network perspective. *Psychological Methods, 28*(6), 1286–1320

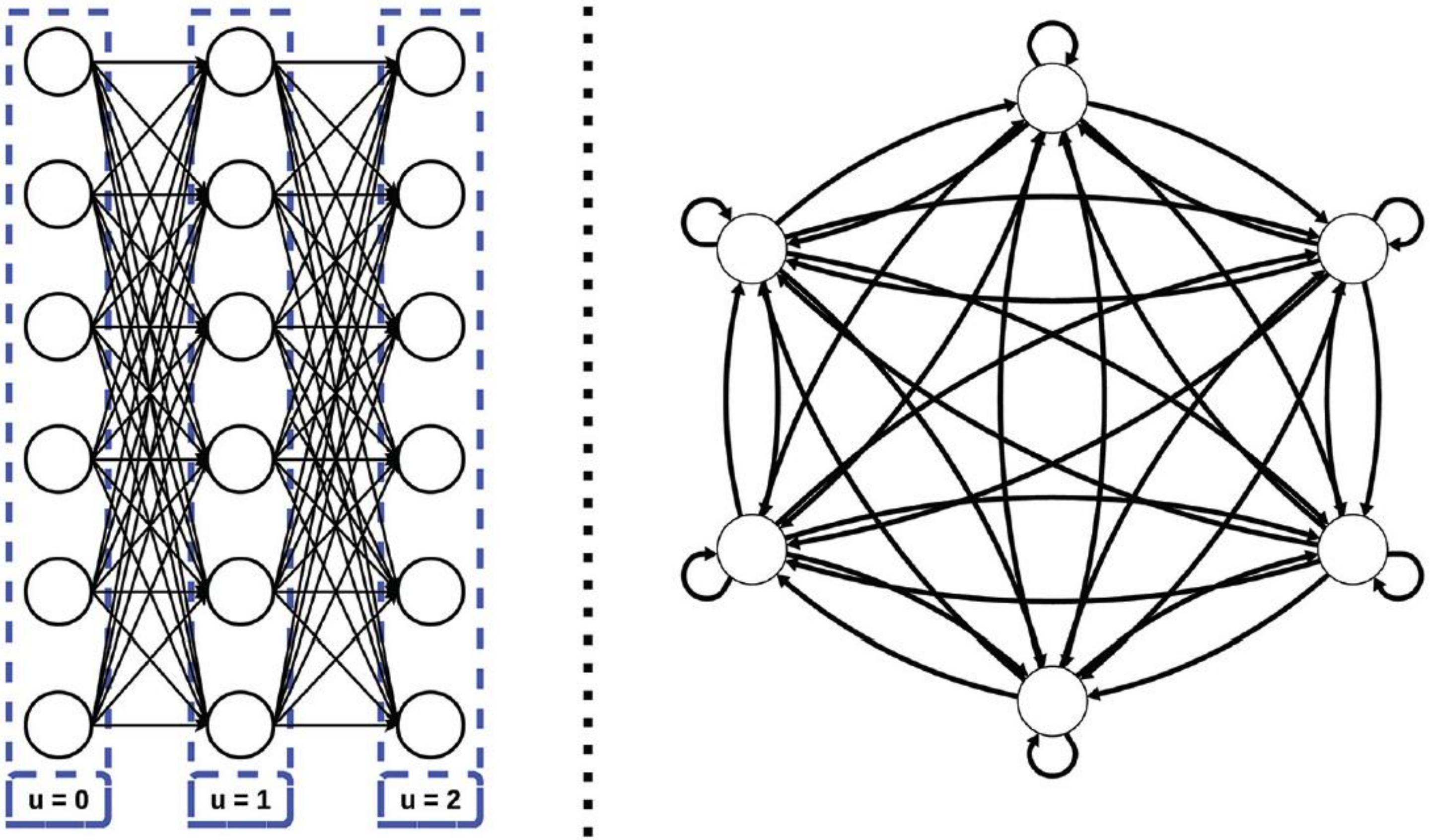
Outline

- The curse of dimensionality: Complex continuous time models
- An introduction to regularization
- Regularization in continuous time models
    - Standardization
    - Optimization
- An introduction to regCtsem

The curse of dimensionality: Complex continuous time models

- Especially in case of multiple processes (e.g., network models; in contrast to multiple indicators), the number of parameters increases quickly.

- For example, for just six variables the drift matrix contains 36 unique parameters.



- Estimation problems (especially in case of small *N* and *T*)
- Interpretation problems
- Overfitting (poor out of sample performance)

An introduction to regularization

- The goal of regularization is to overcome these problems (i.e., to find a parsimonious model and to prevent overfitting).

- In the following we will focus on LASSO regularization (least absolute shrinkage and selection operator).

- LASSO was originally proposed in the context of linear regression (Tibshirani, 1996) but got adapted to many different models, including SEM (e.g., Jacobucci et al. 2016; Huang et al. 2017).

- LASSO regularization is essentially a two-step process:

    1. **Generating sparse models**: Many different models are generated, which differ in their sparseness.

    2. **Model selection**: The “best” model is selected.

An introduction to regularization

1. Generating sparse models:

- Sparsity is induced by adding a penalty term to the likelihood function.

vector of all parameters

-2 log likelihood

selected parameter

$$f_{\mathrm{L}}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \boxed{\lambda N \sum_{j \in J} |\, \theta_j \,|}$$

$|\cdot|$ denotes the absolute value

lasso loss function

tuning parameter

sample size

J = set of indicators of parameters to be regularized with *j* indicating one element in *J*.

An introduction to regularization

1. Generating sparse models:

- The tuning parameter $\lambda \geq 0$ weighs -2 log likelihood against the penalty term.
- For $\lambda = 0$ the loss function is equivalent to the -2 log likelihood function.
- For $\lambda > 0$ the tuning parameter will shrink parameters towards zero. For large λ values parameters will be zeroed.

$$f_{\mathrm{L}}(\boldsymbol{\theta}) = \underbrace{L(\boldsymbol{\theta})} + \underbrace{\boxed{\lambda} N \sum_{j \in J} |\, \theta_j \,|}$$

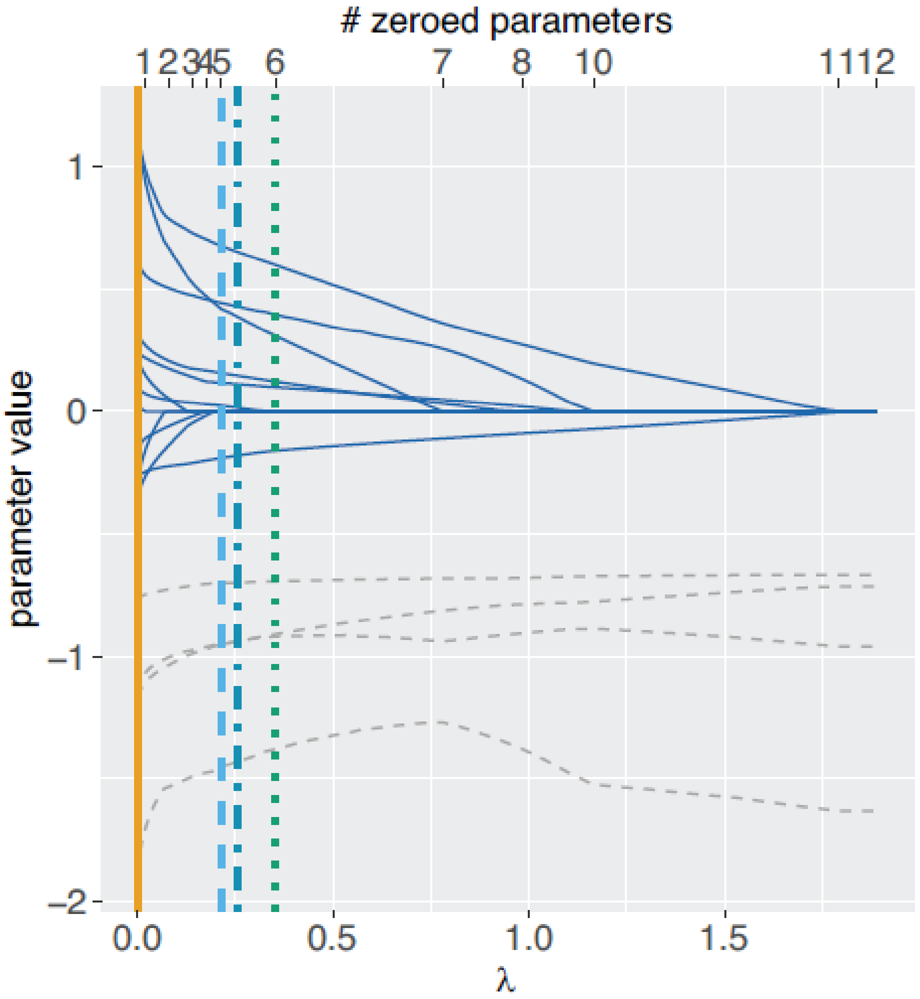
An introduction to regularization

1. Generating sparse models:

- The tuning parameter λ must be chosen by the researcher.
- Usually, a set of different λ-values is chosen (e.g., λ = 0, 0.01, 0.02, λ_{max}).
- For each λ-value a separate model is estimated, resulting in many different models with increasing penalty on selected parameters.

An introduction to regularization

1. Generating sparse models:



- ➢ The regularization paths illustrate the increasing shrinkage and the zeroing of parameters.
- ➢ The generation of increasingly sparse models concludes step 1…
- ➢ …and brings us to the question how to select the „best“ of these model (step 2).

2. Model selection:

- The final model is selected by choosing the "best" λ value.
- Two common criteria of determining what is "best" are (1) information criteria and (2) cross-validation.
- **Regarding (1)** information criteria, usually the AIC or BIC are used:

$$\mathrm{AIC} = L(\boldsymbol{\theta}) + 2p \qquad \mathrm{BIC} = L(\boldsymbol{\theta}) + \ln(N)p$$

- The model with the smallest AIC (BIC) is selected.
- Information criteria reward sparsity by penalizing the number of free parameters (*p*).
- In contrast to cross-validation (next) they are computationally cheap and do not/less suffer from convergence problems particularly in small samples.
- For $N \geq 8$, BIC imposes a higher penalty and tends to select sparser models.

2. Model selection:

- **Regarding (2)** cross-validation, usually *k*-fold cross-validation, is used:
- By using cross-validation, the model (i.e., λ value) that provides the best average out-of-sample generalization is selected.
- classical *k*-fold cross-validation proceeds as follows:

1. The total sample is split into *k* independent subsamples (e.g., *k* = 20 subsamples).
2. A model is fitted based on *k*-1 subsamples (the so-called training set)
3. The out-of-sample fit is determined by fixing the model parameters to the values obtained in step 2 and fitting this (constrained) model to the remaining subsample *s* (the so-called test set). The fit ($L(\boldsymbol{\theta})$) of this model is recorded.
4. Steps 1 to 3 are repeated for each subsample (i.e., *k* times) and the average fit (average $L(\boldsymbol{\theta})$) is computed.

An introduction to regularization

2. Model selection:

- To determine the λ-value that provides the best average out-of-sample generalization, steps 1 to 4 are repeated for each λ-value. E.g., for *k* = 20 and 50 different λ-values, 1000 models are estimated (yes, that takes a while, which is the main problem with cross-validation…).

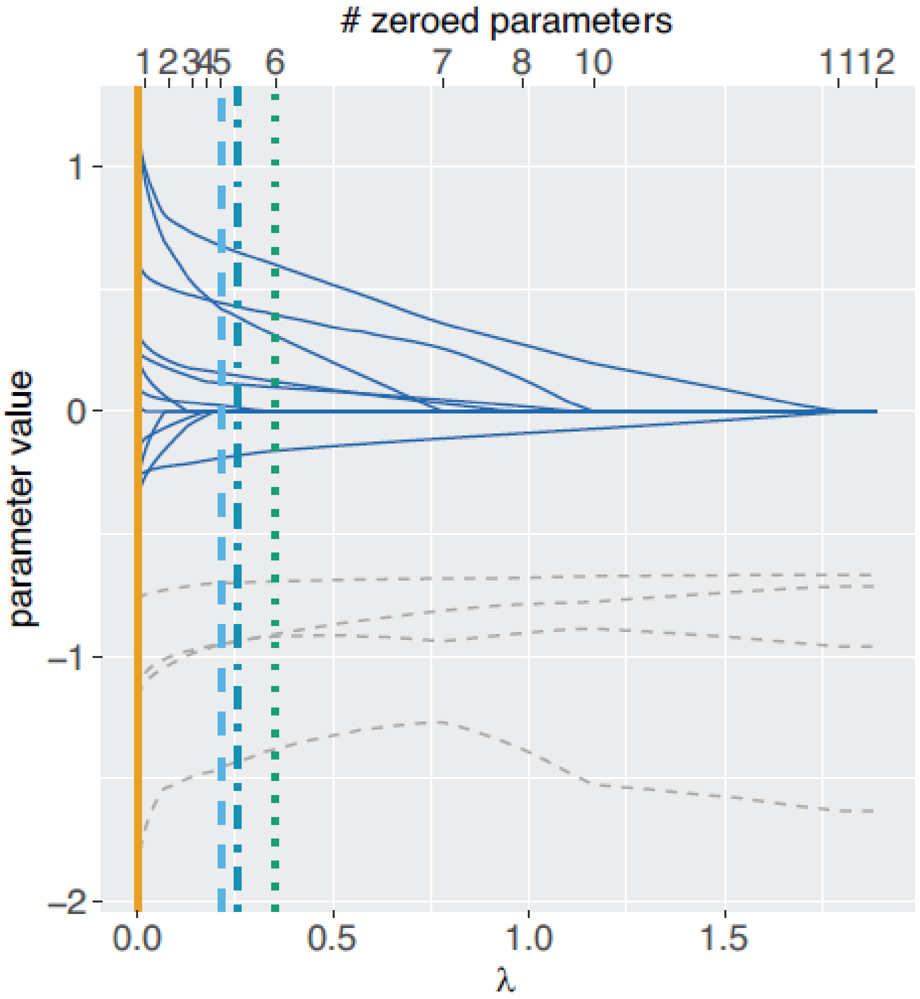
- The λ-value that results in the best average out-of-sample fit (i.e., the minimal average $L(\widehat{\boldsymbol{\theta}}_{\lambda})$) is chosen.

- For the final parameter estimates the model is fitted once again *using the entire sample* with the previously selected, optimal, λ value.

- While in panel data the approach works well (as long as individuals are independent), the situation is trickier in time-series analysis (observations are not independent). Here, it is common to use blocked cross-validation, where the time series is partitioned in *k* consecutive blocks of equal size (e.g., see Bulteel et al., 2018; Loossens, 2021).

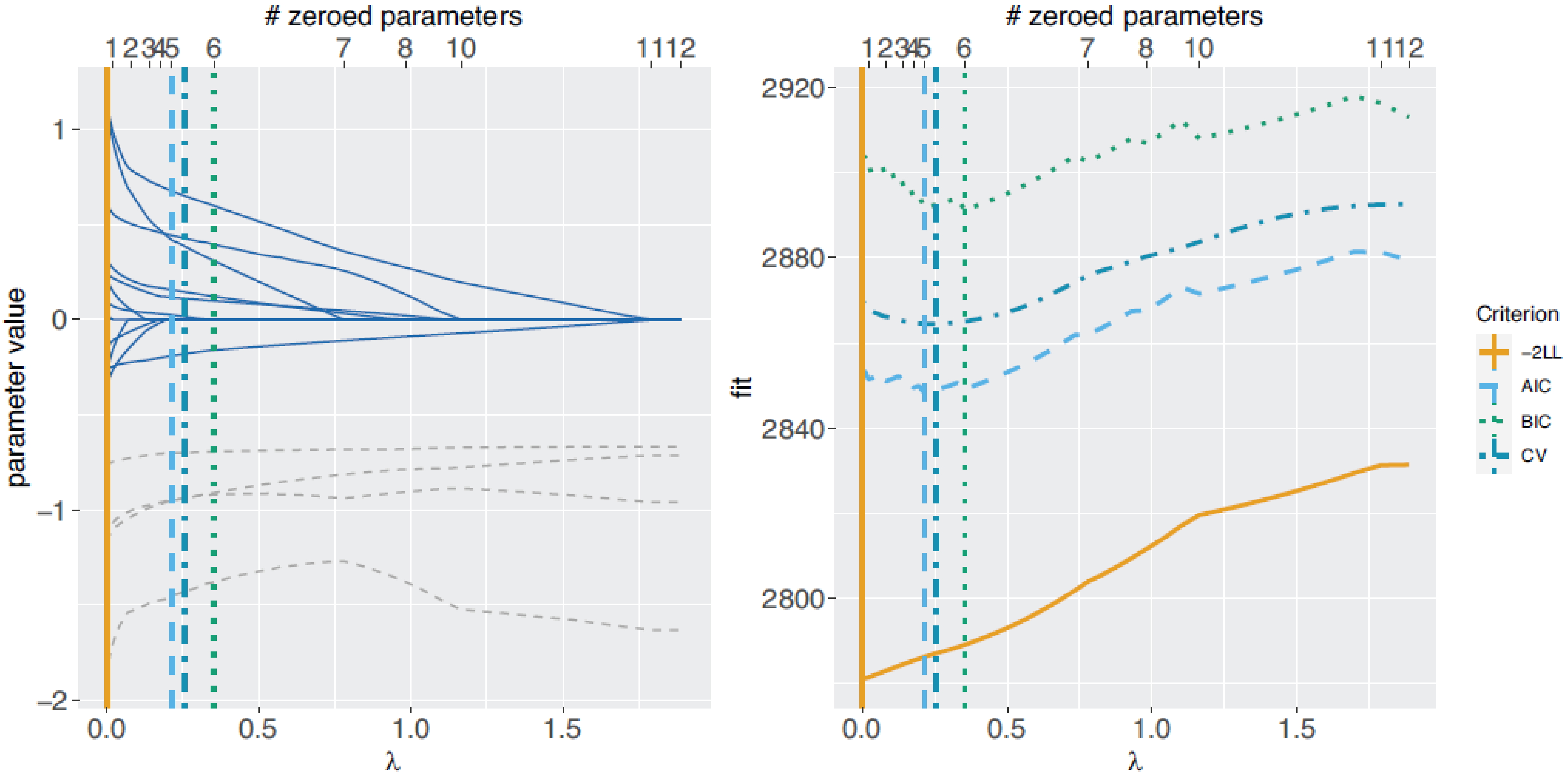
An introduction to regularization

2. Model selection:



An introduction to regularization

2. Model selection:



Regularization in continuous time models

- Although the general idea of regularization in continuous time models is straightforward, there are some problems that need to be resolved.

1. Standardization of the drift coefficients:

- So far, we assumed the same λ value for all parameters. This only makes sense when all parameters are on the same scale.

- Diagonal elements (auto-effects) are per se standardized. However, in case the (latent) variables are on different scales, parameters need to be standardized. Generally, this is done via

$$\alpha = \frac{\sigma_{\text{pred.}}}{\sigma_{\text{crit.}}} a$$

Regularization in continuous time models

- However, in continuous time models, there exists not a single $\sigma_{pred.}$, $\sigma_{crit.}$ respectively, but the covariance matrix for standardization can be chosen differently.

  a) One option is to use the initial covariance matrix $\boldsymbol{\Sigma}_{t_0}$. (e.g., Oud & Delsing, 2010).

  b) Another option is to use the asymptotic covariance matrix $\boldsymbol{P}_{asym}$ (e.g., Driver et al., 2017; Schuurman et al., 2016) computed via

$$\boldsymbol{P}_{\text{asym}} = \text{irow}\left\{-[\boldsymbol{A} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{A}]^{-1}\text{row}(\boldsymbol{G}\boldsymbol{G}^{\top})\right\}$$

- $\boldsymbol{P}_{asym}$ is a function of the drift and the diffusion matrix. This makes it difficult to impose direct constraints for standardization (e.g., by setting the diagonals to 1). Instead, we propose to use a parameter-specific tuning parameter (sLASSO)

$$f_{\text{SL}}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + N\sum_{j \in J} \lambda_j \,|\, \theta_j \,|$$

with $\lambda_j = \lambda \frac{\hat{\sigma}_{pred.}}{\hat{\sigma}_{crit.}}$ and $\frac{\hat{\sigma}_{pred.}}{\hat{\sigma}_{crit.}}$ being the ratio of the unregularized maximum likelihood estimates for the (initial or asymptotic) variances.

Regularization in continuous time models

c) A third option is to use the adaptive LASSO (aLASSO; Zou, 2006; Brandt et al. 2018). Here the tuning parameter is defined as $\lambda_j = \frac{\lambda}{|\hat{\theta}_j|^g}$ with $\hat{\theta}$ being the unregularized ML estimate and g > 0. For g = 1 follows $\frac{\lambda}{|\hat{\theta}_j|}\theta_j$ in the loss function, thus parameters are rescaled with respect to their unregularized ML estimates.

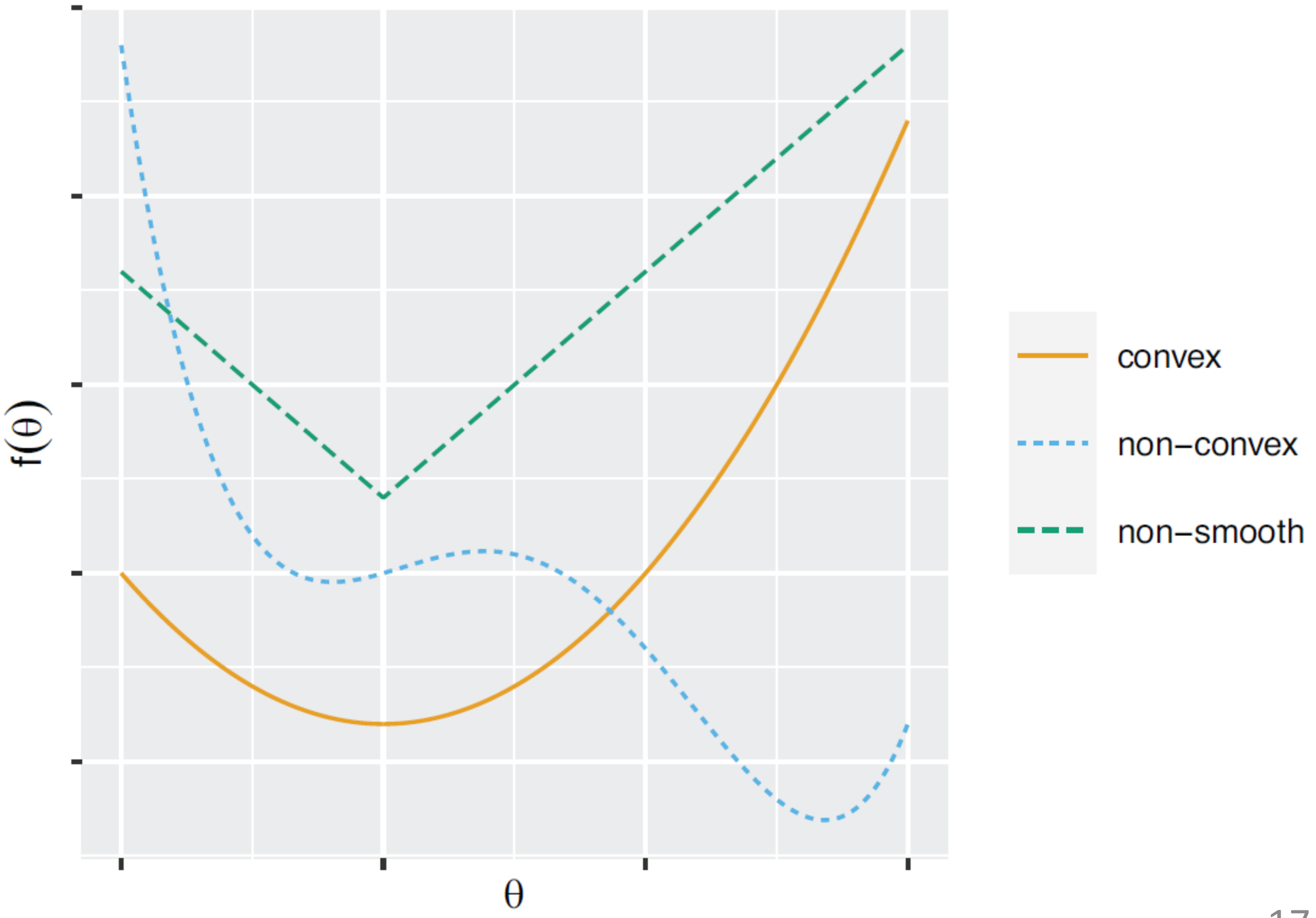
$$f_{\mathrm{SL}}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + N \sum_{j \in J} \lambda_j \mid \theta_j \mid$$

- Some work suggest that the aLASSO outperforms LASSO in first order stochastic differential equations (Gaiffas & Matulewicz, 2019) and we found that it outperformed sLASSO with regard to MSE and sensitivity but not specificity (Orzek & Voelkle, 2023).

Regularization in continuous time models

2. Optimizing the LASSO regularized fitting function

- Another (technical) challenge concerns the LASSO regularized fitting function.
- The combination of the -2 log likelihood (smooth but possibly non-convex) and the penalty (convex but not smooth) results in a function that may be neither smooth nor convex.
- This is a problem for standard optimizers (NPSOL, SOLNP, SLSQP).



Regularization in continuous time models

2. Optimizing the LASSO regularized fitting function

Essentially, there are two options to deal with this problem

a) One option is to *approximate* the non-differentiable penalty with a smooth function.

For unstandardized LASSO: $$f_{\mathrm{L}}^{*}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \lambda N \sum_{j \in J} \sqrt{\theta_j^2 + \epsilon_1}$$

For sLASSO & aLASSO: $$f_{\mathrm{SL}}^{*}(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + N \sum_{j \in J} \lambda_j \sqrt{\theta_j^2 + \epsilon_1}$$

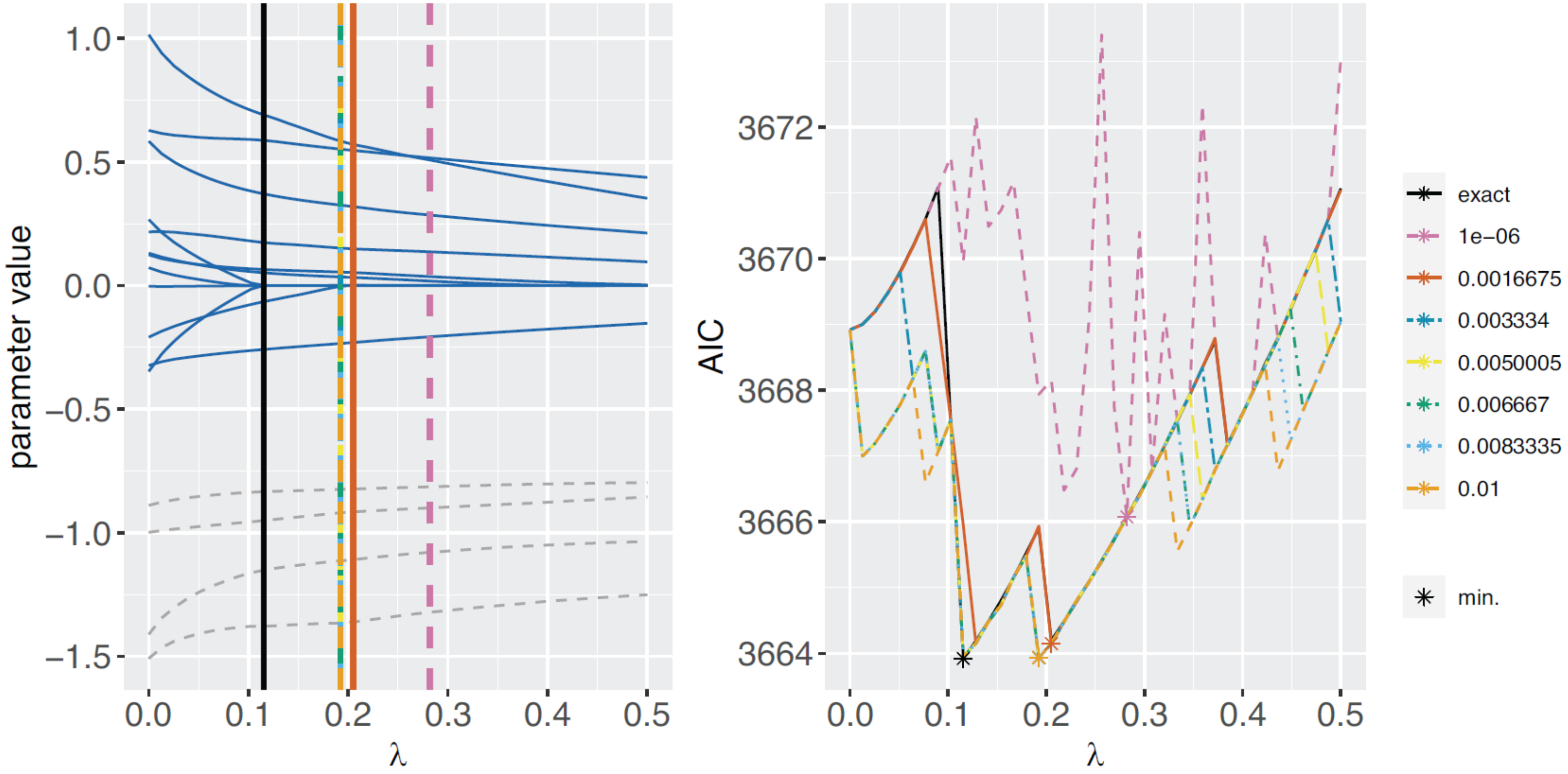
- Because $\theta_j^2 + \epsilon_1 > 0$ even if $\theta_j = 0$ both functions are differentiable and thus suited for standard optimizers.

- Unfortunately, the approximation will not result in a sparse solution. To this end, another threshold parameter ϵ_2 must be implemented that defines the cut-off according to which a parameter is evaluated a zero (if $\left|\hat{\theta}_j\right| < \epsilon_2$ the parameter is treated as zero).

Regularization in continuous time models

2. Optimizing the LASSO regularized fitting function

Choosing ϵ_1 and ϵ_2 is not trivial and can have a substantial effect on results (see Orzek, Arnold, Voelkle, 2023). Pay attention to the defaults and test different values.



Regularization in continuous time models

2. Optimizing the LASSO regularized fitting function

b) Another option is to develop a specialized optimizer that takes the nondifferentiability into account.

- Based on Friedman et al. (2010), Huang (2020) developed such an optimizer for lslx, which is referred to as GLMNET in regCtsem.

- The general iterative shrinkage and thresholding algorithm (GIST, Gong et al., 2013) is an alternative to GLMNET that is implemented/adapted in regCtsem (default).

An introduction to regCtsem

From theory to practice… an example using regCtsem:

```
#if(!require(devtools))install.packages("devtools")
#devtools::install_github("jhorzek/regCtsem") #install regCtsem
library(regCtsem)
set.seed(123)

#### PART 1: Data Simulation ####
# Population parameter values:
DRIFT <- matrix(c(-0.973, 0, 0.434,
                  0.1, -0.795, 0,
                  0.264, 0, -2.065),3,3, TRUE)

DIFFUSIONchol <- matrix(c(1.275, 0,0,
                          0.367, 1.177, 0,
                         .806, -.153, 1.414),3,3,TRUE)

generatingModel <- ctModel(LAMBDA = diag(3), n.manifest = 3, n.latent = 3,
                           T0VAR = diag(3), T0MEANS = 0, MANIFESTMEANS = 0,
                           MANIFESTVAR = 0, DRIFT = DRIFT,
                           DIFFUSION = DIFFUSIONchol, TRAITVAR = NULL, Tpoints = 10)

simulatedData <- ctGenerate(ctmodelobj = generatingModel, n.subjects = 100, burnin = 100, wide = T)
```

Script: regCtsem_demonstration.R

An introduction to regCtsem

From theory to practice… an example using regCtsem:

```
#### PART 2: Specify & estimate an unregularized CTSEM ####

DiffusionEstim <- matrix(paste0("Diff",rep(1:3,each = 3), paste0("_Diff", 1:3)),
                                              nrow = 3, ncol = 3, byrow = T)
DiffusionEstim[upper.tri(DiffusionEstim)] <- "0"

T0VAREstim <- matrix(paste0("T0VAR", rep(1:3, each = 3), rep(1:3)),3,3, T)
T0VAREstim[upper.tri(T0VAREstim)] <- "0"

analysisModel <- ctModel(LAMBDA = diag(3), n.manifest = 3,  n.latent = 3,
               T0VAR = T0VAREstim,
               T0MEANS = paste0("T0MEANS", 1:3),
               MANIFESTMEANS = 0,
               MANIFESTVAR = 0,
               DRIFT = "auto",
               DIFFUSION = DiffusionEstim,
               TRAITVAR = NULL,
               Tpoints = 10)

fit.Model <- ctFit(dat = simulatedData, ctmodelobj = analysisModel)
```

Script: regCtsem_demonstration.R

An introduction to regCtsem

From theory to practice… an example using regCtsem:

```
#### PART 3: Specify & estimate a regularized CTSEM ####

# Which parameters do we want to regularize?
regIndicators <- fit.Model$mxobj$DRIFT$labels[!diag(T, 3)] # all cross-effects
print(regIndicators)

# regularization
regModel <- try(regCtsem::regCtsem(ctsemObject = fit.Model,
            dataset = simulatedData,
            regIndicators = regIndicators,
            lambdas = "auto", # the maximally required lambda will be computed automatically
            lambdasAutoLength  = 20, # note: we should  use as many lambdas as possible; restricted here to reduce the runtime.
            penalty = "adaptiveLasso"))

# Plot results and extract best estimates:
plot(regModel)
plot(regModel, what = "fit")
getFinalParameters(regCtsemObject = regModel, criterion = "BIC")
```

Script: regCtsem_demonstration.R

Selected References

Brandt, H., Cambria, J., & Kelava, A. (2018, 2018/11/02). An Adaptive Bayesian Lasso Approach with Spike-and-Slab Priors to Identify Multiple Linear and Nonlinear Effects in Structural Equation Models. Structural Equation Modeling: A Multidisciplinary Journal, 25(6), 946-960. https://doi.org/10.1080/10705511.2018.1474114

Bulteel, K., Mestdagh, M., Tuerlinckx, F., & Ceulemans, E. (2018). VAR(1) based models do not always outpredict AR(1) models in typical psychological applications. Psychological Methods, 23, 740-756. https://doi.org/10.1037/met0000178

Driver, C. C., Oud, J. H. L., & Voelkle, M. C. (2017). Continuous time structural equation modeling with R package ctsem. Journal of Statistical Software, 77(5), 1-35. https://doi.org/10.18637/jss.v077.i05

Friedman, J. H., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1 - 22. https://doi.org/10.18637/jss.v033.i01

Gaïffas, S., & Matulewicz, G. (2019). Sparse inference of the drift of a high-dimensional Ornstein–Uhlenbeck process. Journal of Multivariate Analysis, 169, 1-20. https://doi.org/https://doi.org/10.1016/j.jmva.2018.08.005

Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems Proceedings of the 30th International Conference on Machine Learning, Proceedings of Machine Learning Research. https://proceedings.mlr.press/v28/gong13a.html

Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. Psychometrika, 82(2), 329-354. https://doi.org/10.1007/s11336-017-9566-9

Huang, P.-H. (2018). A penalized likelihood method for multi-group structural equation modelling. British Journal of Mathematical and Statistical Psychology, 71(3), 499-522. https://doi.org/https://doi.org/10.1111/bmsp.12130

Huang, P.-H. (2020). lslx: Semi-Confirmatory Structural Equation Modeling via Penalized Likelihood. Journal of Statistical Software, 93(7), 1 - 37. https://doi.org/10.18637/jss.v093.i07

Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016, 04/12). Regularized Structural Equation Modeling. Structural equation modeling : a multidisciplinary journal, 23(4), 555-566. https://doi.org/10.1080/10705511.2016.1154793

Loossens, T. (2021). Toward parsimonious modeling of affect dynamics in daily life (Doctoral dissertation). Katholieke Universiteit Leuven.

Orzek, J. H., & Voelkle, M. C. (2023). Regularized continuous time structural equation models: A network perspective. Psychological Methods, 28(6), 1286–1320

Orzek, J. H., Arnold, M., & Voelkle, M. C. (2023). Striving for Sparsity: On Exact and Approximate Solutions in Regularized Structural Equation Models. Structural Equation Modeling: A Multidisciplinary Journal, 30(6), 956–973. https://doi.org/10.1080/10705511.2023.2189070

Oud, J. H. L., & Delsing, M. J. M. H. (2010). Continuous time modeling of panel data by means of SEM. In K. van Montfort, J. H. L. Oud, & A. Satorra (Eds.), Longitudinal research with latent variables (pp. 201-244). Springer. https://doi.org/10.1007/978-3-642-11760-2_7

Schuurman, N. K., Ferrer, E., de Boer-Sonnenschein, M., & Hamaker, E. L. (2016). How to Compare Cross-Lagged Associations in a Multilevel Autoregressive Model. Psychological Methods, 21, 206-221. https://doi.org/10.1037/met0000062

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267-288. https://doi.org/https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Zou, H. (2006). The Adaptive Lasso and Its Oracle Properties. Journal of the American Statistical Association, 101(476), 1418-1429. https://doi.org/10.1198/016214506000000735

Study Questions

Question 1:

In your own words, what is the purpose of regularization in statistical modeling? Why might regularization be useful when modeling psychological time series data with continuous-time SEM?

Question 2:

Explain the two main steps of LASSO regularization and how they are applied to structural equation models.

Question 3:

Why is the optimization of LASSO-regularized ct models non-trivial, and how is it addressed in practice?

Question 4:

Summarize the two main steps of the regCtsem package workflow in R.