ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 1 - JUNE 2025

MODEL SELECTION IN GLM : POISSON REGRESSION

Stephen Worrall

Professor Maarten Jansen

PROBLEM

Model selection in GLM model (Poisson regression)

1. Consider the DGP (data generating process) $Y_i \sim \text{Poisson}(\mu_i)$ with $\mu_i = 5\left[\cos(5x_i^2) + 2\right]$ for $n = 100$ covariates $x_i \in [0, 1]$ (uniformly distributed on the interval).

   Select and estimate a GLM of the form $g(\boldsymbol{\mu}) = \boldsymbol{\theta} = \mathbf{X}\boldsymbol{\beta}$ where the design matrix $\mathbf{X}$ has the elements $X_{ik} = x_i^k$ for $k = 0, 1, \ldots, m$ and using the log link function $g(u) = \log(u)$. Use AIC and BIC on nested submodels $S = \{1, 2, \ldots, p\}$ (with $p \leq m$) to select the appropriate polynmial degree. Let n grow towards infinty and see what happens. What can you say about the consistency of BIC?

   Briefly discuss the obtained results (in maximum 200 words). You may use the appropriate `R`-packages, or software in `Matlab`, `Python` or others. Some Matlab code doing the job is provided in the zip-file `mfilesforGLM.zip` (the main file being `simulatePoissonregression.m`). This code has some numerical issues, but should give a satisfactory answer.

SOLUTION

Model Selection in GLM : Poisson Regression

In this task, using matlab, data generated from the Poisson regression is fit to a generalized linear model (GLM) using polynomial bases of varying degree and select among submodels using AIC and BIC penalty criteria.

Simulation Results: Upon running the provided MATLAB code, we observe that both AIC and BIC select a polynomial model, but BIC consistently chooses a smaller model (lower degree polynomial) compared to AIC, as expected due to its heavier penalty for model complexity. As the sample size (n) increases, both AIC and BIC tends to converge towards the curve of the true model (figures 1.1, 1.2, 1.3),, as outliers are drowned out by volume of generated datapoints, generally BIC tends to avoid overfitting and more reliably selects the true (or nearly true) model complexity, a property referred to as consistency. In contrast, AIC, with a lighter penalty, may still select overly complex models even for large (n).

Consistency of BIC: BIC is known to be a consistent model selection criterion for parametric models: as n goes to infinity, it selects the correct model with probability approaching 1 This was observed in simulation (figure 1.4), where the degree selected by BIC stabilized near the true underlying complexity.

```
samplesize =

   200

>> simulatePoissonregression

pAIC =

     9

pBIC =

     5
```

Executing a loop over a range of sample sizes n between 40 to 1000 and determining value least penalty value : pAIC and pBIC returns a result of sample size : n = 200 , with penalties : pAIC = 9 and pBIC = 5…

For Poisson regression with polynomial bases, BIC provides consistent order selection, especially as sample size grows, supporting its theoretical properties.

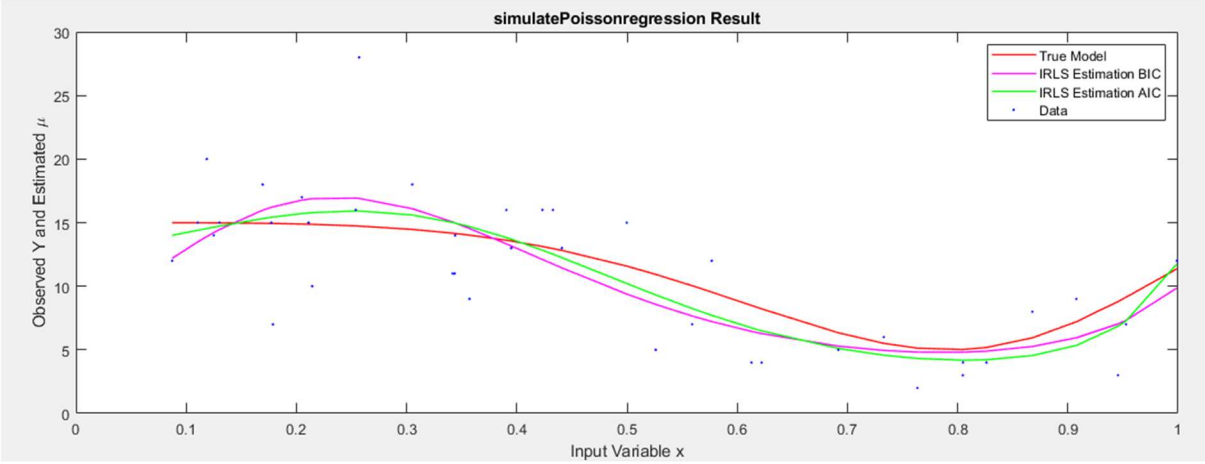

Figure 1.1 – Simulate Poisson Regression - n = 40

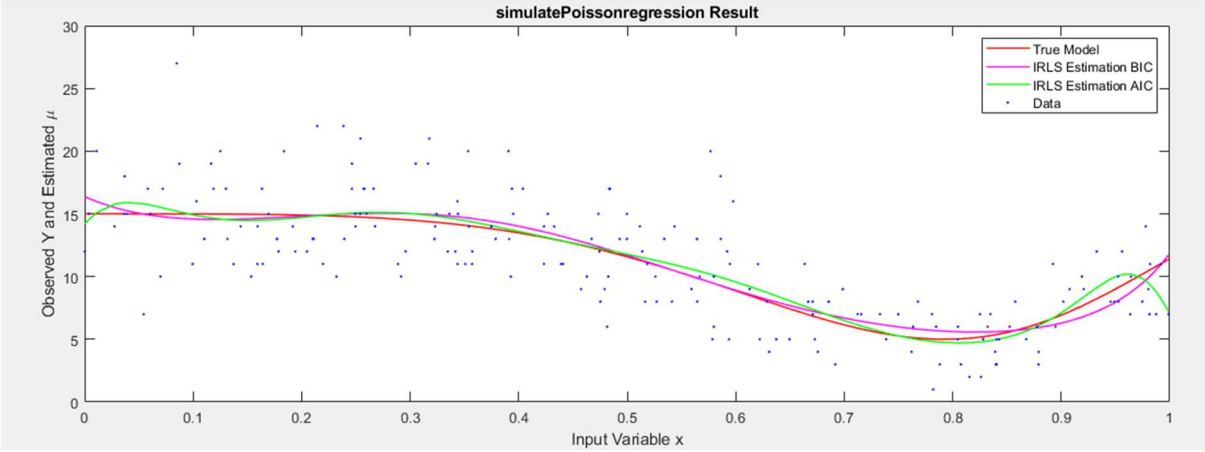

Figure 1.2 – Simulate Poisson Regression - n = 200

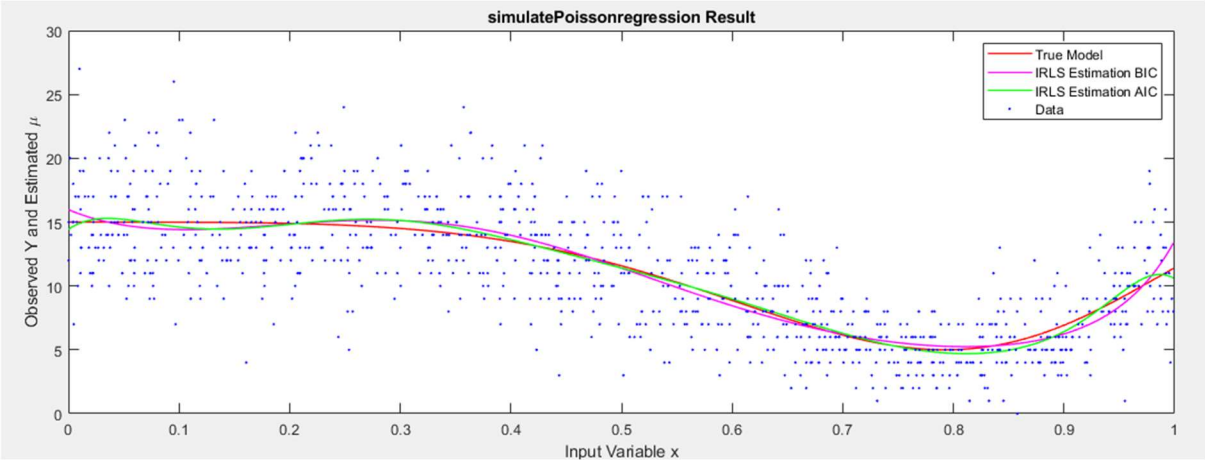

Figure 1.3 – Simulate Poisson Regression - n = 1000

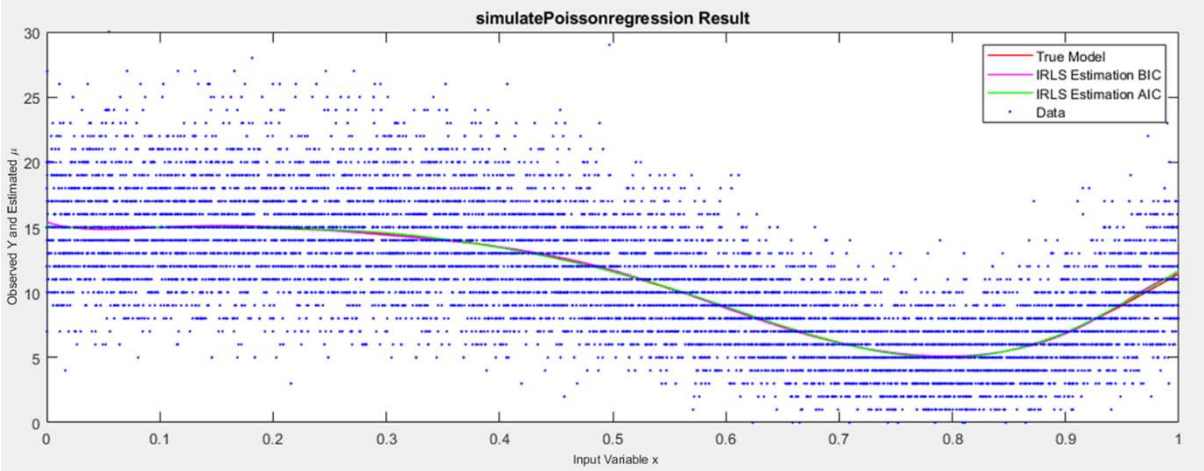

Figure 1.4 – Simulate Poisson Regression - n=10000

ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 2 - JUNE 2025

INFORMATION CRITERIA

Stephen Worrall

Professor Maarten Jansen

PROBLEM

Information criteria

2. The zip-file `mfilesforIC.zip` contains a routine `STATF408simulationsIC2025.m` running a simulation of a variable selection procedure for nested models

$$Y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \sigma Z_i,$$

with $Z_i \sim \mathrm{NID}(0,1)$, while the DGP is a polynomial of degree 7 (the same as on slide 126), defined by its zeros at

$$\begin{bmatrix} -0.1000 & 0.1555 & 0.3143 & 0.5469 & 0.6903 & 0.8730 & 1.1 \end{bmatrix}$$

The routine produces plots of AIC, BIC, Cp, PE, and sum of squared differences $\|\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}\|$. Run the algorithm as follows:

- `clear; samplesize = 30; STATF408simulationsIC2025`
- `clear; samplesize = 100; STATF408simulationsIC2025`
- `clear; samplesize = 1000; STATF408simulationsIC2025`
- `clear; samplesize = 10000; STATF408simulationsIC2025`

Then discuss the followin issues:

- The curves of $\mathrm{PE}(p)$ and $\mathrm{KL}(p)$ have an increasingly flat right half (the part beyond the optimal value of p, that is) for growing n. Discuss the origin of this phenomenon as well as its effect on the value minimizing the PE or KL.
- Briefly compare the asymptotic behavior of AIC and BIC.

SOLUTION

Information criteria

As we increase sample size, the curves for square diff (PE(p), C(p)) (see figures 2.1b, 2.2b, 2.3b, 2.4b) and KL-LogL(DGP) (see figures 2.1d, 2.2d, 2.3d, 2.4d) become increasingly flat to the right of their minimum. This happens because, with more data, the variance of parameter estimates decreases, so including extra polynomial terms (beyond the true number) does not significantly harm prediction error. The origin of this phenomenon lies in the decreasing impact of estimation noise as n grows. The minimizing value of PE or KL thus reliably pinpoints the true model for large n, with the minimum growing sharper as the sample size increases.

AIC and BIC behave differently as sample size increases. AIC tends to slightly overfit even as n grows, because its penalty for model complexity (2p) is fixed and does not keep pace with increasing data size. In contrast, BIC's penalty (log(n)p) increases with n, causing it to be consistent, BIC will almost surely select the simplest true model as n goes to infinity.

Note : For all simulations : fullmodelsize = 150

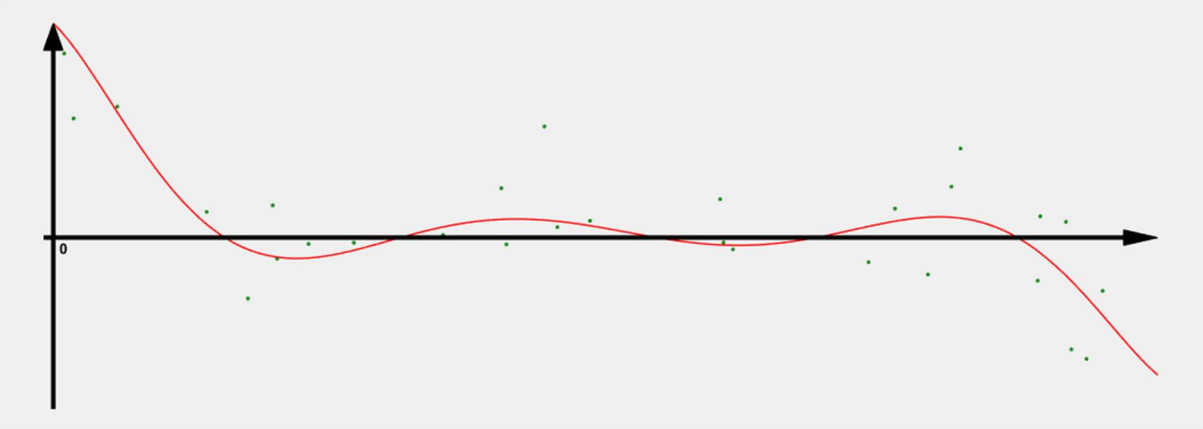

Figure 2.1a – samplesize = 30, STATF408simulationsIC2025 output : mmu (red)

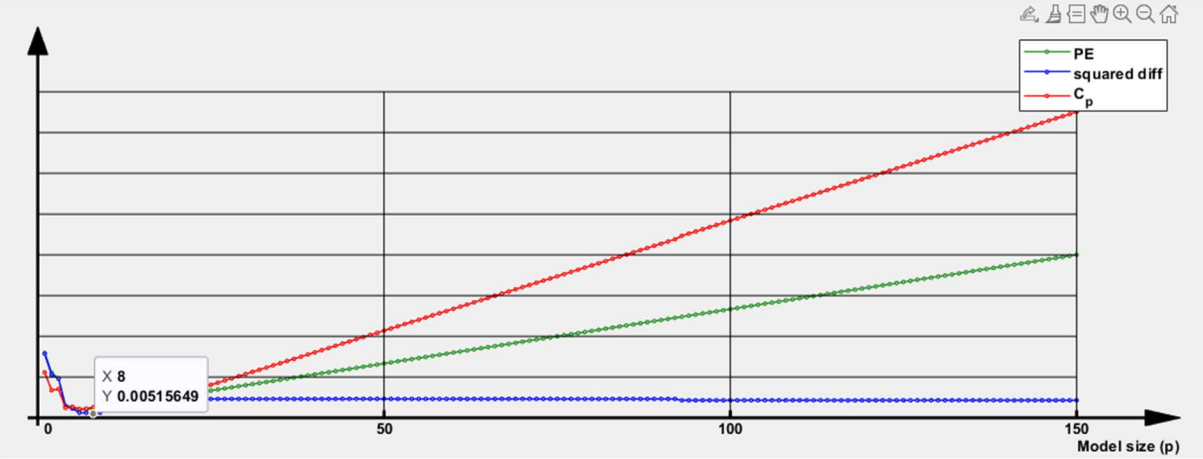

Figure 2.1b – samplesize = 30, STATF408simulationsIC2025 output :

minimum square diff = 0.00515, model size = 8

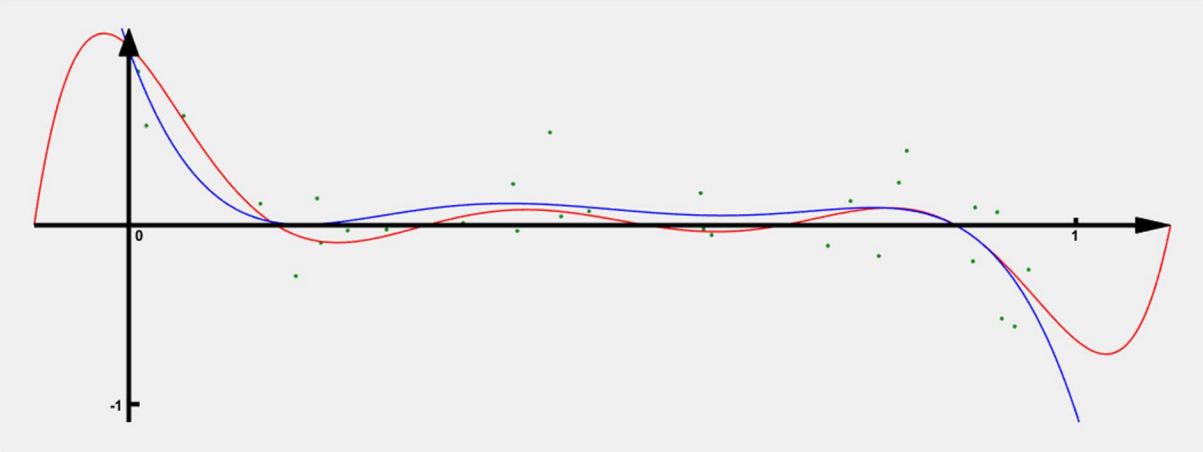

Figure 2.1c – samplesize = 30, STATF408simulationsIC2025 output : extended plot beyond 1 and 0 for mmu (red) and mmuhatopt (blue)

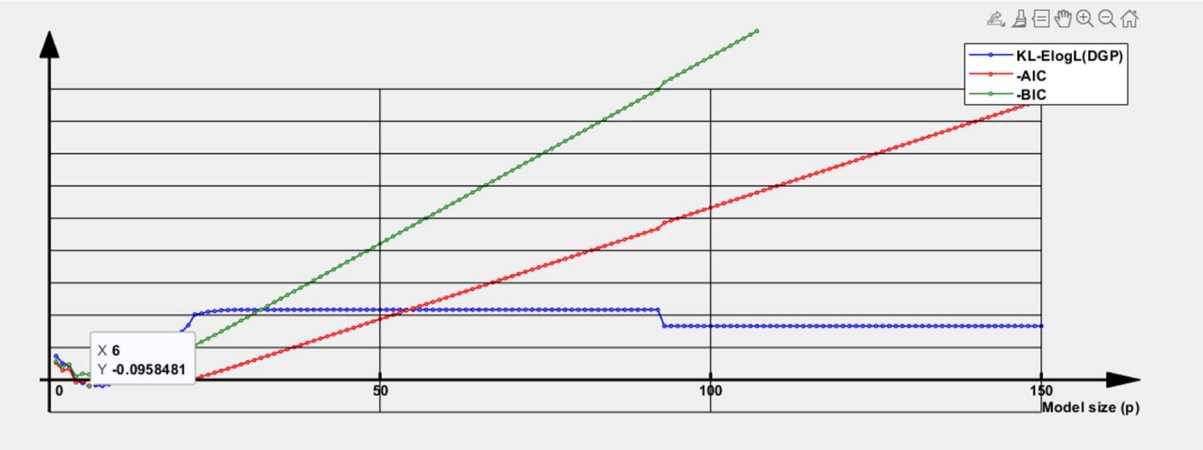

Figure 2.1d – samplesize = 30, STATF408simulationsIC2025 output :

minimum KL-LogL(DGP) = -0.0958481, model size = 6

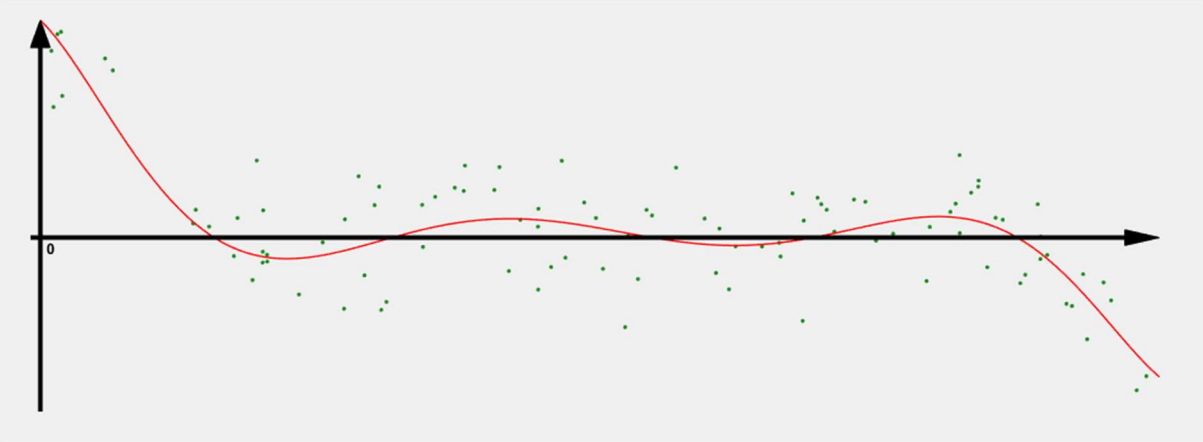

Figure 2.2a – samplesize = 100, STATF408simulationsIC2025 output : mmu (red)

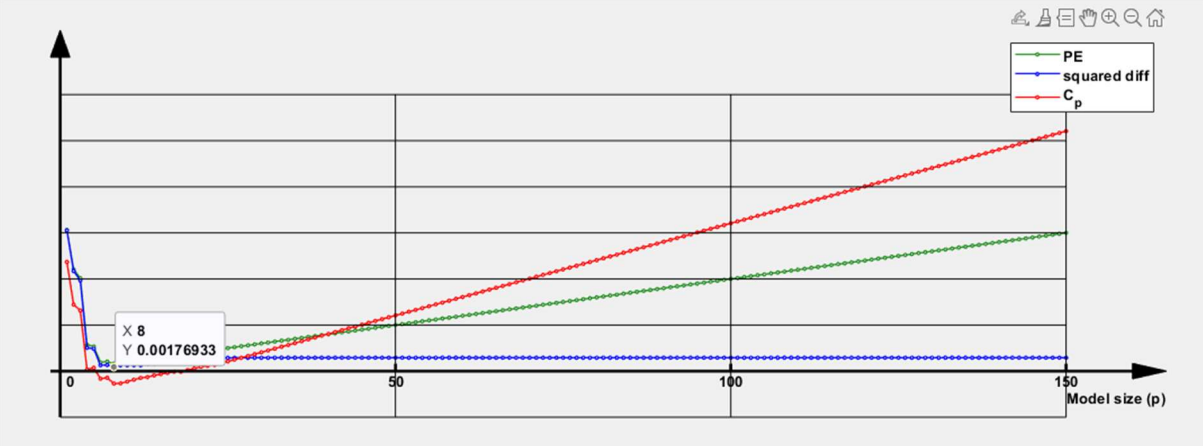

Figure 2.2b – samplesize = 100, STATF408simulationsIC2025 output :

minimum square diff = 0.00176, model size = 8

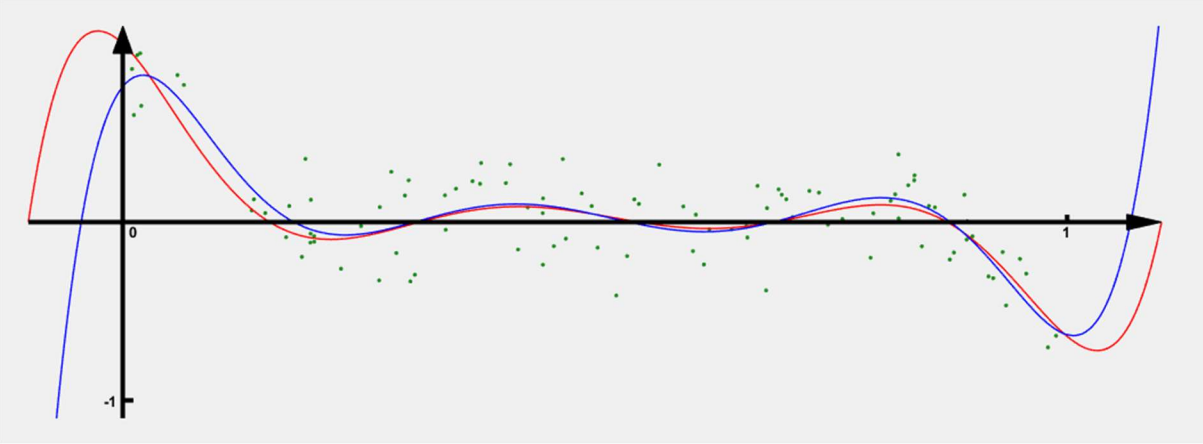

Figure 2.2c – samplesize = 100, STATF408simulationsIC2025 output : extended plot beyond 1 and 0 for mmu (red) and mmuhatopt (blue)

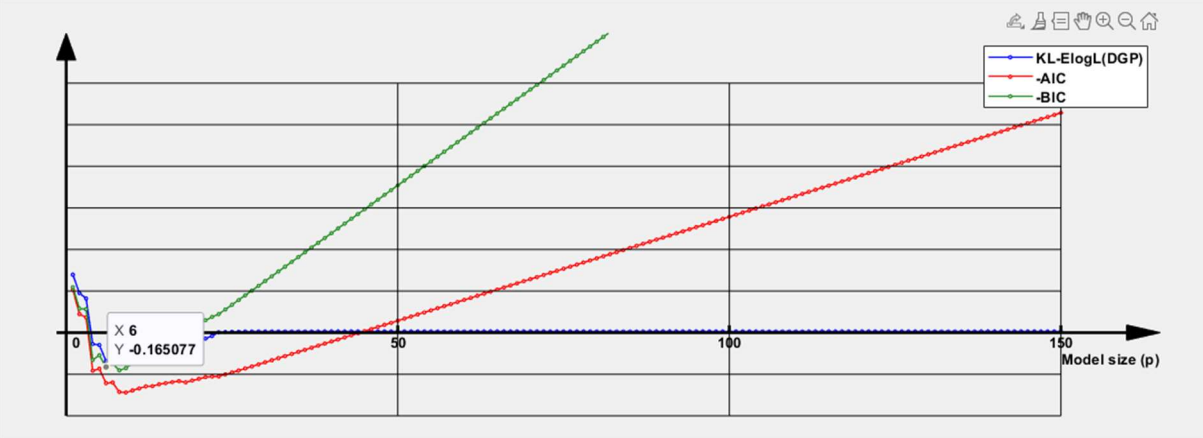

Figure 2.2d – samplesize = 100, STATF408simulationsIC2025 output :

minimum KL-LogL(DGP) = -0.165077, model size = 6

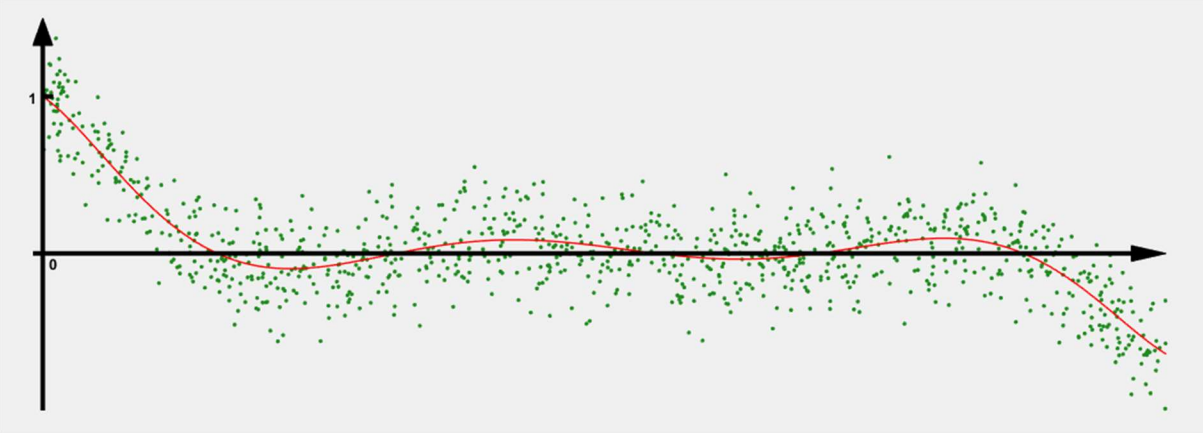

Figure 2.3a – samplesize = 1000, STATF408simulationsIC2025 output : mmu (red)

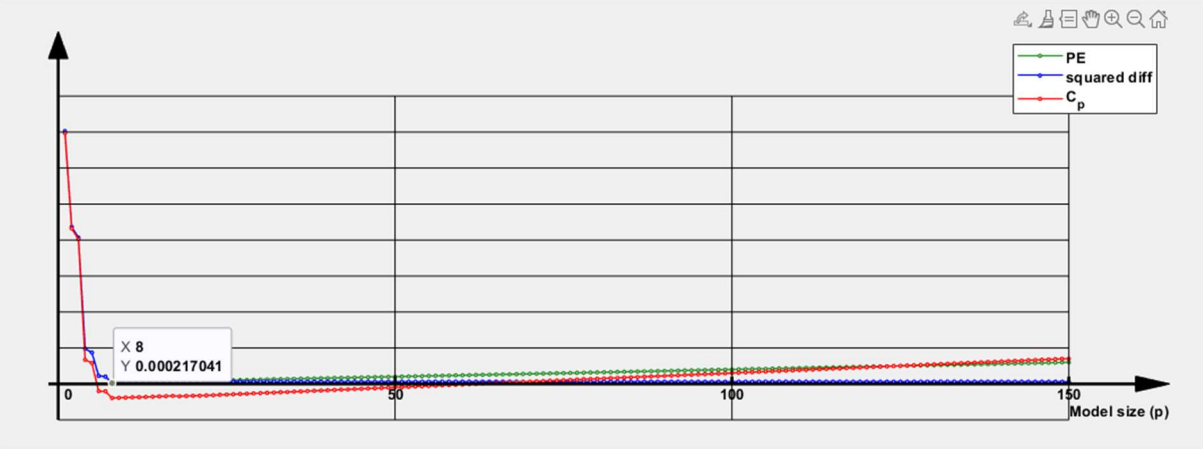

Figure 2.3b – samplesize = 1000, STATF408simulationsIC2025 output :

minimum square diff = 0.000217, model size = 8

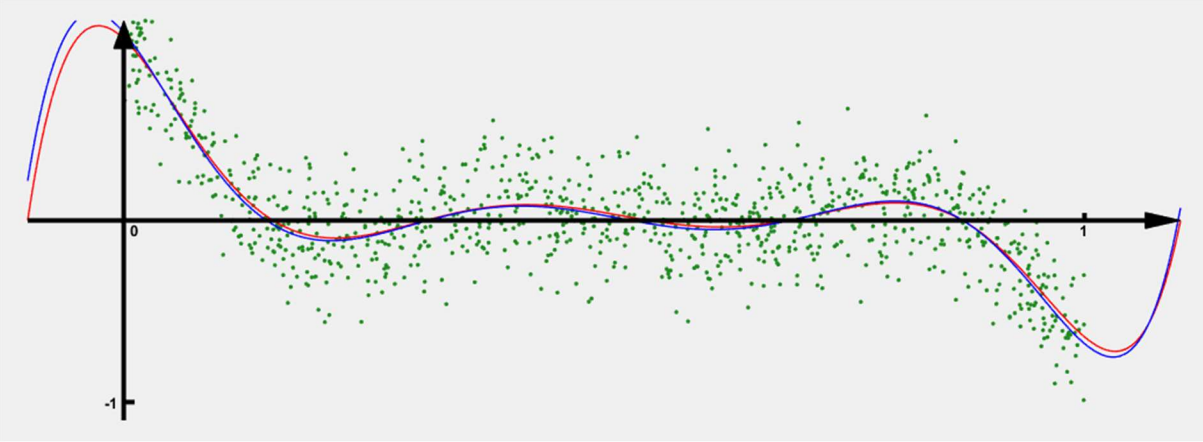

Figure 2.3c – samplesize = 1000, STATF408simulationsIC2025 output : extended plot beyond 1 and 0 for mmu (red) and mmuhatopt (blue)

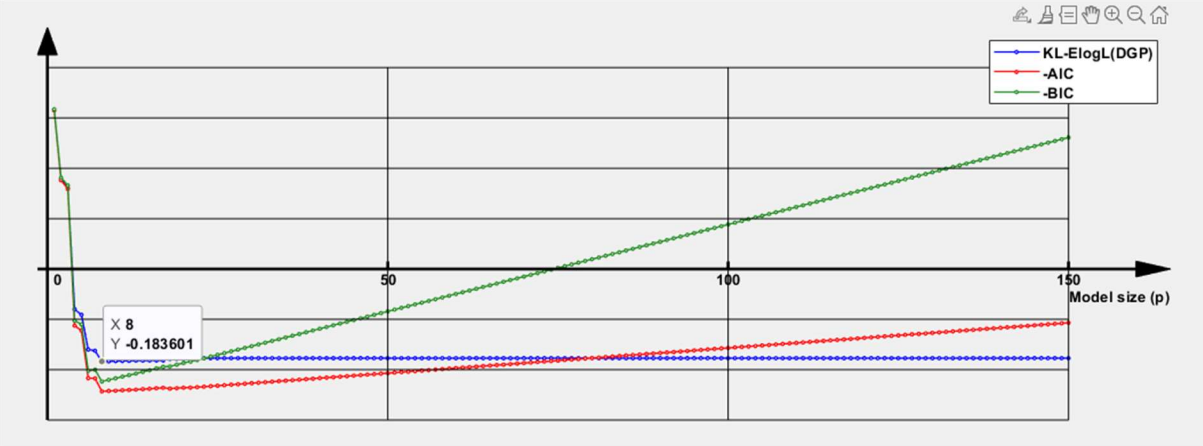

Figure 2.3d – samplesize = 1000, STATF408simulationsIC2025 output :

minimum KL-LogL(DGP) = -0.183601, model size = 8

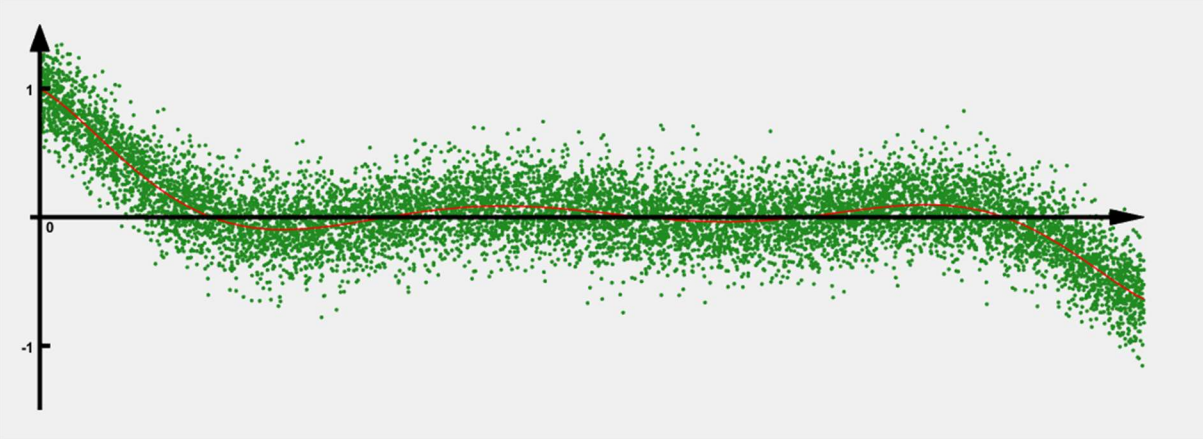

Figure 2.4a – samplesize = 10000, STATF408simulationsIC2025 output : mmu (red)

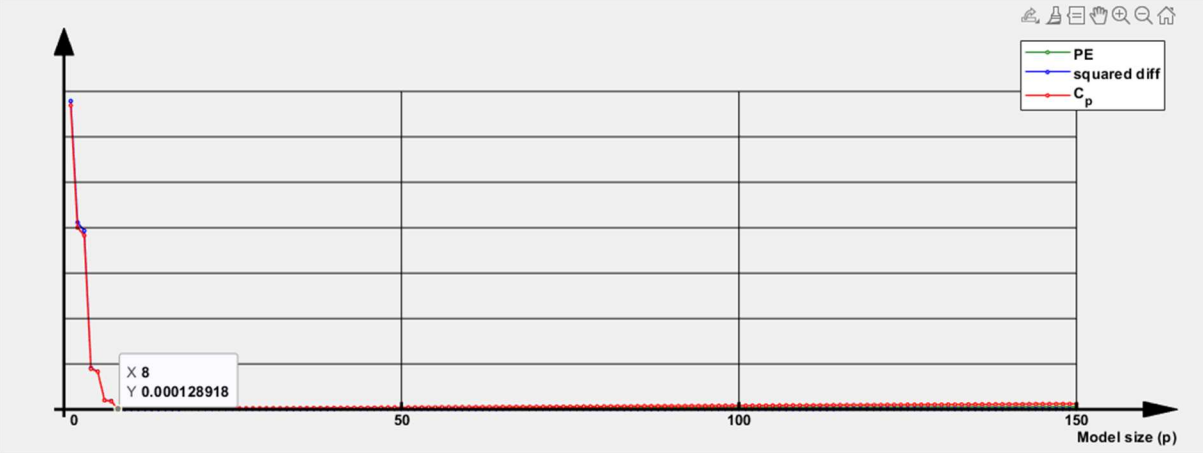

Figure 2.4b – samplesize = 10000, STATF408simulationsIC2025 output :

minimum square diff = 0.000128918, model size = 8

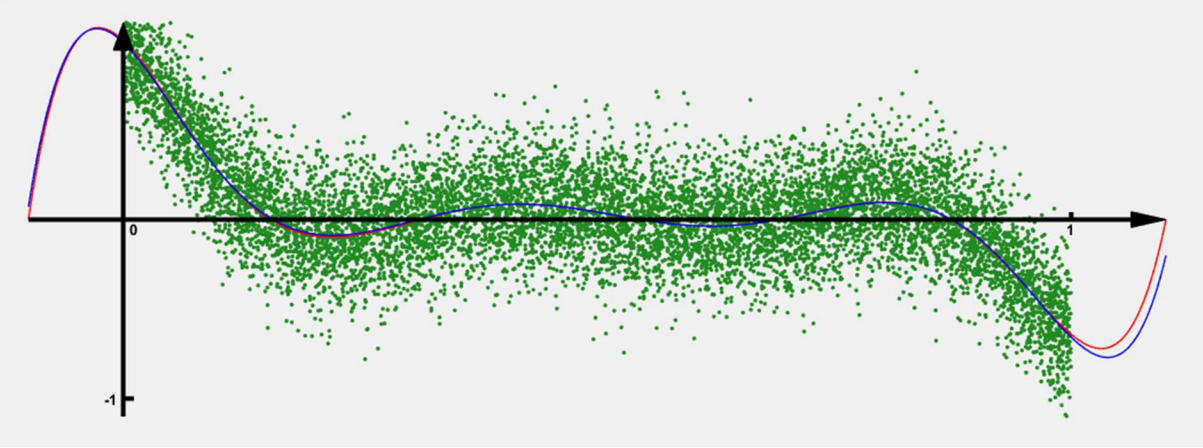

Figure 2.4c – samplesize = 10000, STATF408simulationsIC2025 output : extended plot beyond 1 and 0 for mmu (red) and mmuhatopt (blue)

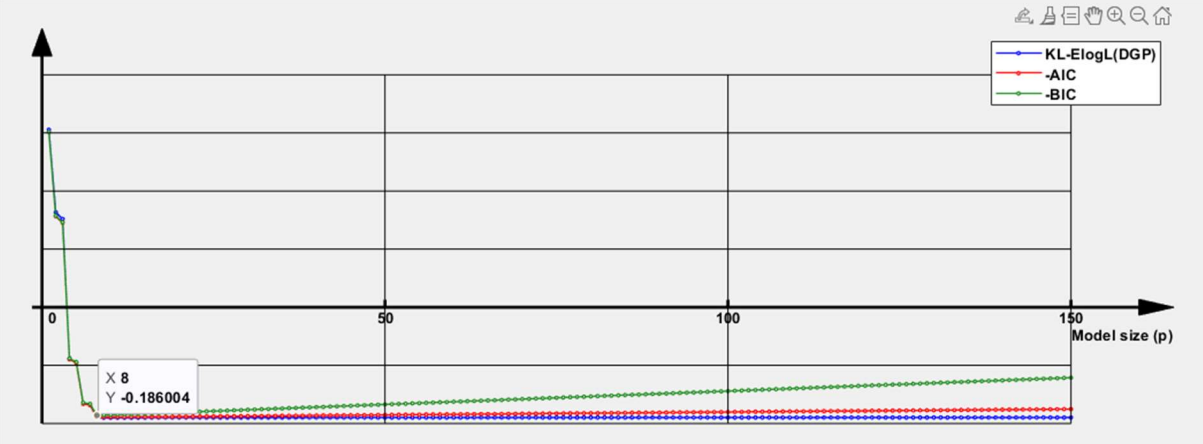

Figure 2.4d – samplesize = 10000, STATF408simulationsIC2025 output :

minimum KL-LogL(DGP) = -0.186004, model size = 8

ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 3 - JUNE 2025

HIGH DIMENSIONAL MODEL SELECTION

Stephen Worrall

Professor Maarten Jansen

PROBLEM

High dimensional model selection

3. The zip-file `mfilesforsparsemodelselection.zip` contains a routine `illustrateHigDim202503.m` running a simulation of a high-dimensional sparse problem following the model

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \sigma \boldsymbol{Z},$$

where β is sparse. This is simulated by letting β be a realization of a random variable $\boldsymbol{V}$ which has probability $1-p$ of being zero. Here, p acts as the degree of sparsity. Given a nonzero, the distribution is taken Laplace (double exponential), which generates larger outliers than the normal errors, for clear distinction between significant and non-significant values in β. The Laplace random variable is realised by taking the logarithm of a uniform random variable (see Chapter on Monte Carlo).

The default design matrix in `illustrateHigDim202503.m` is a band diagonal matrix, defined by $X_{ij} = K\left(\frac{x_i - u_j}{h}\right)$, where $K(u)$ is a kernel function with bounded support, h is the bandwidth, and where the n-vector x and the m-vector $\boldsymbol{u}$ are generated as sorted samples from a uniform random variable.

The simulation study produces graphs of the sample prediction error (PE) and Mallows's Cp for orthogonal projection and shrinkage estimators in models selected by Lasso (implemented by LARS).

- The variables `beta` and `betahat` contain the true vector of β and the shrinkage estimator within the LARS selected model. The variables `mu` and `muhat` contain the true and estimator vectors of responses $\boldsymbol{\mu}$ and $\widehat{\boldsymbol{\mu}}$ without noise. After running `illustrateHigDim202503`, perform

```
plot(beta,'r','linewidth',3)
axy = axis;
hold on
plot(betahat,'b')
axis(axy)
hold off
```

to see that $\widehat{\beta}$ is far from the true β. How repeat the same for $\boldsymbol{\mu}$:

```
plot(mu,'r','linewidth',3)
axy = axis;
hold on
plot(muhat,'b')
axis(axy)
hold off
```

What can you conclude? (Short answer)

Now check the values of λ:

```
plot(kappa,log(lambda),'b','linewidth',3)
```

What can be concluded w.r.t. the effect of the shrinkage?

Now check the Karush-Kuhn-Tucker (KKT) conditions in the optimal selection `Sopt` (Note that `LARS` has worked with `Y-mean(Y)` as response):

```
checkKKT = X'*(Y-mean(Y)-X*betahat);
plot(checkKKT(Sopt),'b','linewidth',3)
Sprime = setdiff(1:m,Sopt);
plot(checkKKT(Sprime),'r','linewidth',3)
```

Does the solution found by the `LARS` routine satisfy the KKT consitions?

SOLUTION

High Dimensional Model Selection

For this question I will walk through the MATLAB listing (code indicated by **>>**) and answer questions as the results dictate...

>> illustrateHigDim202503

'plot(beta)'

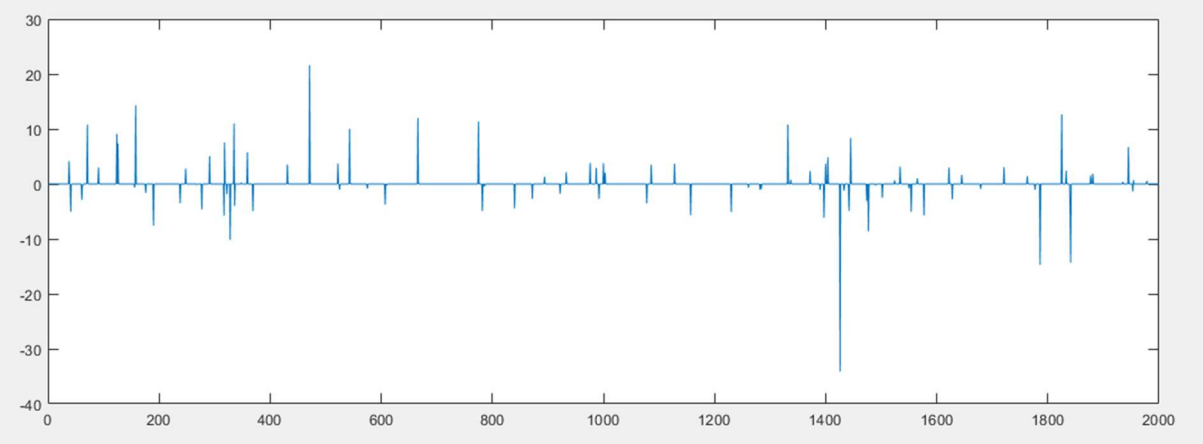

Figure 3.1 – illustrateHigDim202503 output 'plot(beta)'

plot(kappa,samplePE0,linedotoptr{:})

plot(kappa,samplePE1,linedotoptm{:})

plot(kappa,Cp1,linedotoptb{:})

plot(kappa,Cp0,linedotoptk{:})

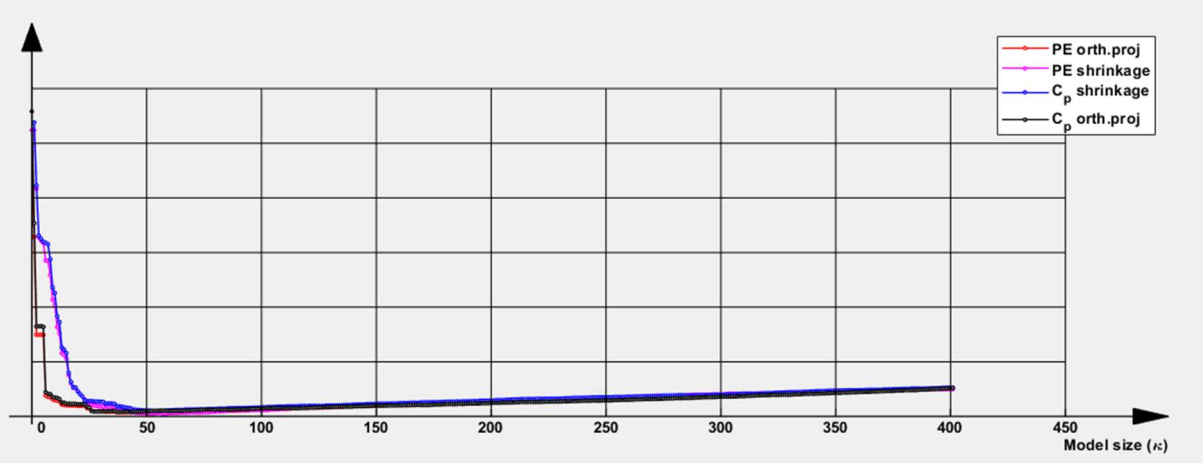

Figure 3.2 – illustrateHigDim202503 output plot of kappa : samplePE0, samplePE1, Cp0, Cp1

```
>> plot(beta,'r','linewidth',1)

>> axy = axis;

>> hold on

>> plot(betahat,'b')

>> axis(axy)

>> hold off
```

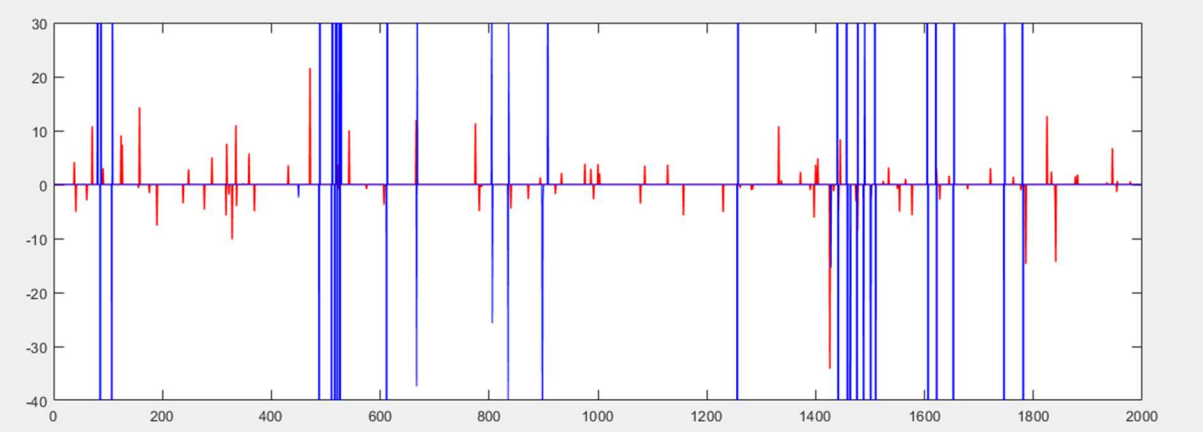

Figure 3.3 – illustrateHigDim202503 output plot of beta, betahat

Red (β): The true coefficient vector is sparse — most entries are near 0, and a few have significant values.

Blue (β̂): The Lasso estimate.

>> plot(mu,'r','linewidth',1)

>> axy = axis;

>> hold on

>> plot(muhat,'b')

>> axis(axy)

>> hold off

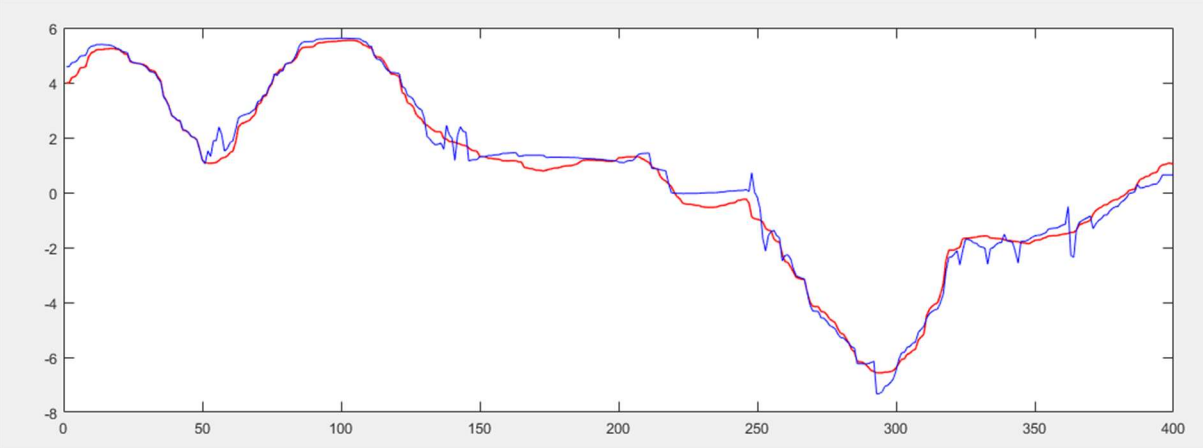

Figure 3.4 – illustrateHigDim202503 output plot of mu, muhat

Red (μ): The true mean response.

Blue (μ̂): The fitted mean from estimated coefficients.

The fitted mean (blue) tracks the true mean (red) very closely, capturing almost all of the signal's variation. Some small discrepancies are visible, reflecting the bias in β̂. Despite bias in individual coefficients, the prediction from Lasso (μ̂) generally provides a good fit to the true response (μ). Lasso reliably detects sparsity in β. The fitted mean (μ̂) tracks the true mean response (μ) closely, so prediction remains accurate, even if β̂ appears biased.

>> plot(kappa,log(lambda),'b','linewidth',1)

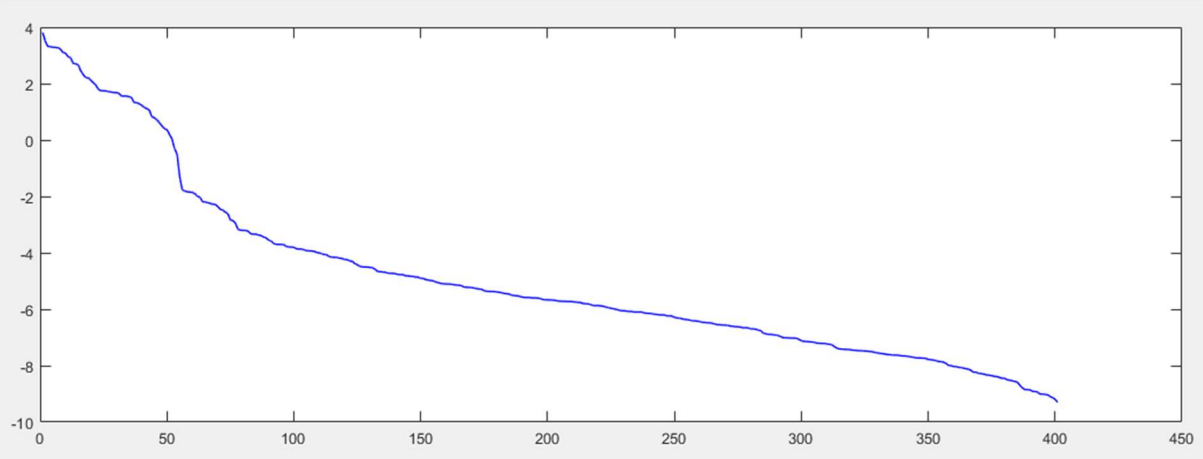

Figure 3.5 – illustrateHigDim202503 output 'plot(kappa,log(lambda))'

X-axis (kappa): Number of active predictors

Y-axis (log(lambda)): The log of the penalty parameter.

In the plot of log(λ) against the number of nonzero coefficients (κ), we see that as more variables enter the model, λ decreases, the solution becomes less sparse. With very large λ, the model includes very few predictors (high shrinkage) with very low λ, almost all predictors are included (little or no shrinkage), resulting in possible overfitting. Selecting the optimal λ is key to balancing sparsity and accuracy in high-dimensional modelling.

>> checkKKT = X'*(Y-mean(Y)-X*betahat);

>> plot(checkKKT(Sopt),'b','linewidth',1)

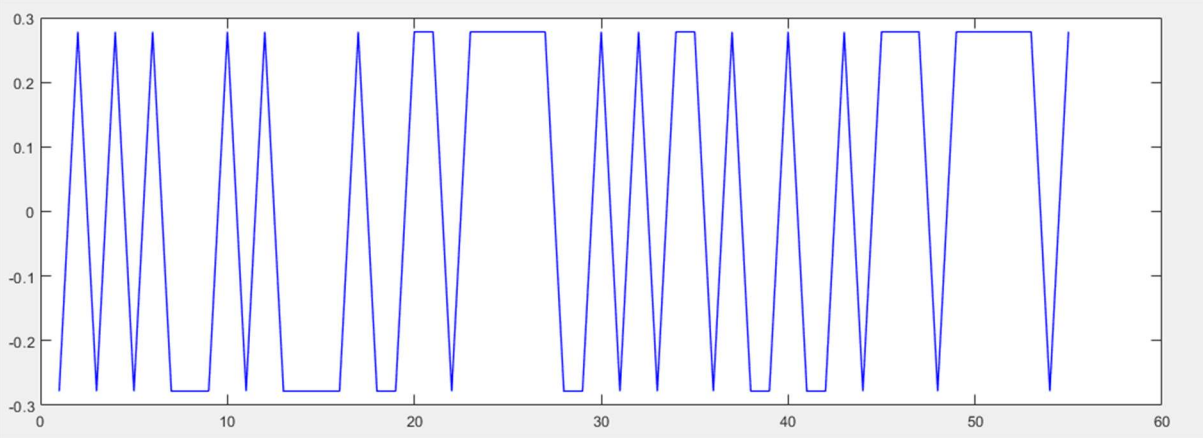

Figure 3.6 – illustrateHigDim202503 output 'plot(checkKKT(Sopt))'

>> Sprime = setdiff(1:m,Sopt);

>> plot(checkKKT(Sprime),'r','linewidth',1)

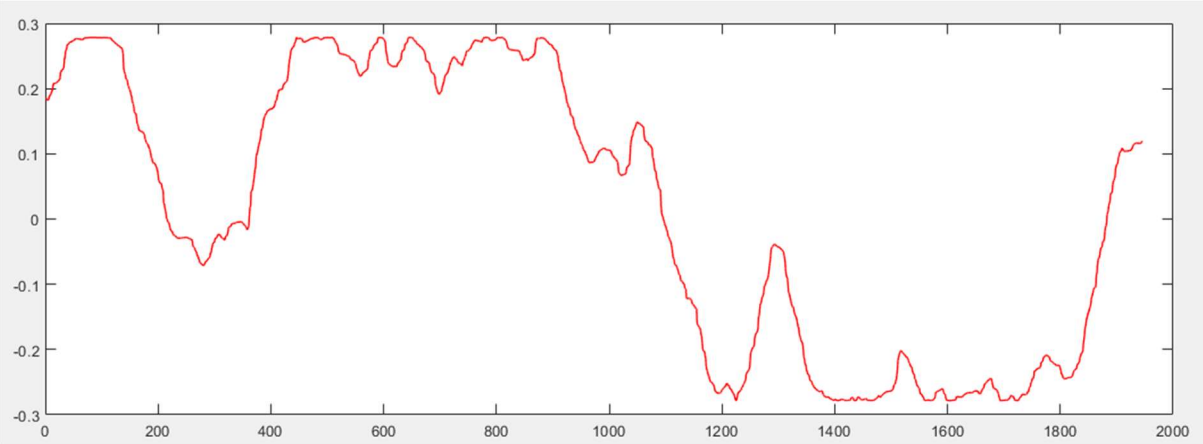

Figure 3.7 – illustrateHigDim202503 output 'plot(checkKKT(Sprime)'

The KKT (Karush-Kuhn-Tucker) conditions were checked for the Lasso solution. For indices in the active set (nonzero coefficients), the KKT values reach the penalty boundary (±0.3), as expected. For indices outside the active set (zero coefficients), the KKT values fall strictly within the bounds (−λ, λ). Thus, the solution found using LARS/Lasso satisfies the KKT conditions.

ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 4 - JUNE 2025

BOOTSTRAP

Stephen Worrall

Professor Maarten Jansen

PROBLEM

Bootstrap

4. Let $X_i \sim \text{neg.bin}(p_i, r)$, i.e., $X_i \in \mathbb{N}$ and

$$P(X_i = k) = \frac{\Gamma(r+k)}{\Gamma(r)k!} p_i^r (1-p_i)^k,$$

i.e., X_i counts the number of failures until the r-th success in an independent sequence of Bernoulli experiments with success rate p_i (Note that often a negative binomial variable is defined to count all trials, failures and successes alike). In the case of unequal p_i, the distribution of

$$Y = \sum_{i=1}^{\kappa} X_i$$

has no closed expression, but it holds that

$$\begin{aligned} \mu &= E(Y) = r \cdot \sum_{i=1}^{\kappa} \frac{1-p_i}{p_i} \\ \sigma^2 &= \text{var}(Y) = r \cdot \sum_{i=1}^{\kappa} \frac{1-p_i}{p_i^2} \end{aligned}$$

Supposing that all the p_i's are known, then so is

$$d = \frac{\sigma^2}{\mu} = \frac{\sum_{i=1}^{\kappa} \frac{1}{p_i}\left(\frac{1}{p_i} - 1\right)}{\sum_{i=1}^{\kappa}\left(\frac{1}{p_i} - 1\right)}$$

Now, complete the Matlab code in `STATF408bootstrap2025.m` for the construction of a basic bootstrap confidence interval and a bootstrap-t confidence interval for μ. Simulate this with `ns` samples of samplesize `n` and compare coverage and mean widths of both methods. What can be concluded in this example?

SOLUTION

Bootstrap

To complete the code STATF408bootstrap2025.m for : Sstar, Ustar, ustaralfa and bootstraptCI we must first define tstar...

```
        Sstar = std(Xstar, 0, 1) / sqrt(n);

        tstar = (mean(Xstar)-mustar) ./ Sstar;

        tstar_sorted = sort(tstar);

        Ustar = tstar_sorted(qq);

        ustaralfa(s,1:2) = Ustar;

end
```

bootstraptCI definition as...

```
        basicCI = 2*[T T]-tstaralfa(1:ns,[2 1]);

        bootstraptCI = [muhat' muhat'] - [ustaralfa(:,2).*S  ustaralfa(:,1).*S];
```

Running the simulations returns the following:

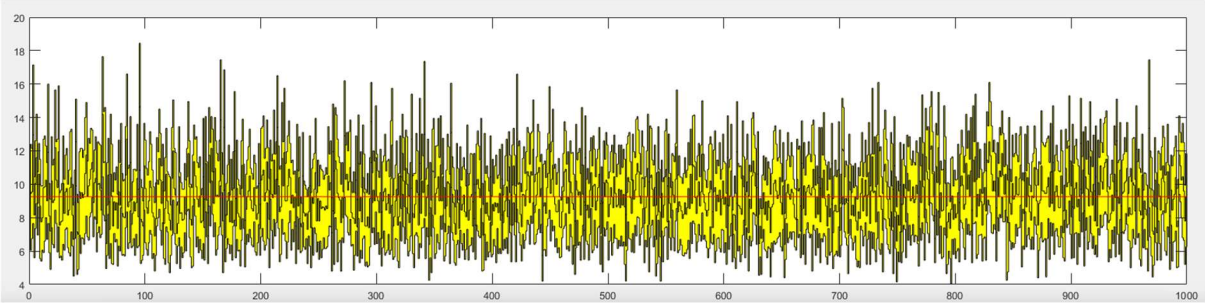

Figure 4.1 – Basic Bootstrap CI

Yellow bands indicate basic (percentile or reversed) bootstrap confidence intervals for the mean, in each simulation. Black is the mean for each simulation. The red line is the true theoretical mean. The Y-axis range matches the expected mean +/– typical sampling error. The true mean is contained within many intervals but not all, with coverage slightly under the nominal 95%.

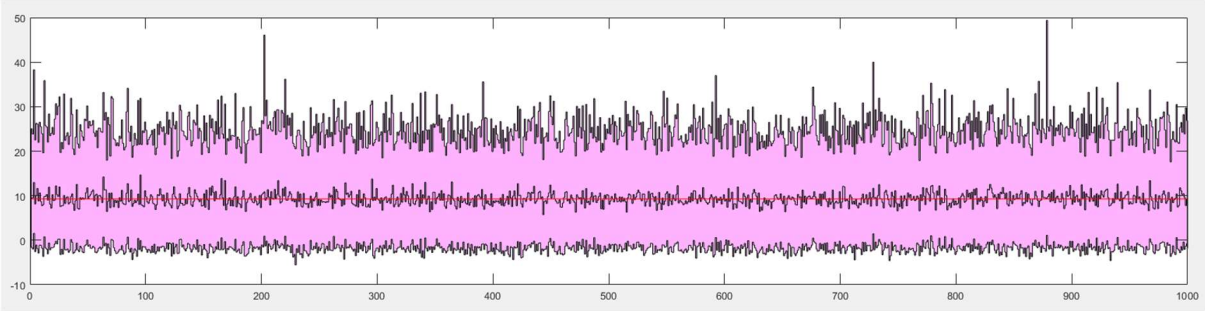

Figure 4.2 – Bootstrap-t CI

Magenta bands indicate Bootstrap-t confidence intervals, wider and more variable. lower bounds are much lower and your upper bounds much higher than in the basic bootstrap as the bootstrap-t method is extremely sensitive to sample mean or standard error, often happens in small-sample, over-dispersed settings

>> STATF408bootstrap2025

coverbasicCI = 0.9160

coverbootstraptCI = 1

meanwidthbasicCI = 4.9789

meanwidthbootstraptCI = 26.5785

Method	Coverage	Mean Width
Basic Bootstrap	91.6%	5.0
Bootstrap-t	100%	26.6

Bootstrap Confidence Intervals for a Sum of Negative Binomials

Comparing two bootstrap methods for constructing confidence intervals for the mean of a sum of negative binomial random variables with unequal probabilities. Using 1000 simulated samples (each with 20 observations), basic bootstrap and the bootstrap-t confidence intervals are evaluated in terms of empirical coverage and average interval width. The basic bootstrap confidence interval achieved an empirical coverage of 91.6% with an average interval width of 5.0, slightly under the nominal 95% level. The bootstrap-t interval, on the other hand, always contained the true mean (100% coverage) but produced intervals that were excessively wide (average width 26.6). The basic bootstrap method provides more practical and informative intervals, despite some loss in coverage. Conversely, the bootstrap-t approach is overly conservative and less useful for inference, as its intervals are unnecessarily wide. This highlights the importance of considering data properties and simulation results when choosing bootstrap methods in complex or discrete models.

ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 5 - JUNE 2025

RANDOM NUMBER GENERATOR (MONTE CARLO)

Stephen Worrall

Professor Maarten Jansen

PROBLEM

Random number generator (Monte Carlo)

5. Find a numerical value of $\mathrm{var}(X)$ where X has the density $f_X(x) = Kf(x)$ with

$$f(x) = \exp\left(-\frac{1}{\sqrt{1-x^2}}\right)$$

on $[-1, 1]$ (and zero outside this interval), using rejection sampling starting from the density $g_X(x) = Lg(x)$ where

$$g(x) = 1 - x^2.$$

(a) Make a plot showing that $f_X(x)$ can be bounded by $M \cdot g_X(x)$. Find a value of M as a function of K and L.

(b) Use the following scheme to generate data from $g_X(x)$ and explain why this works.

```
a = 2; b = 2;
S = -log(rand(a+b,n));
U = sum(S(1:a,1:n))./sum(S);
V = 2*U-1;
```

(c) Develop the rejection criterion $U \geq \dfrac{f_X(V)}{Mg_X(V)}$ for V with density function $g_X(x)$ and $U \sim \mathrm{unif}([0, 1]$ in terms of $f(X)$ and $g(X)$.

(d) Implement the rejection sampling, for instance by completing the matlab routine `randexpinvsqrt.m` (taking `randCauchyplusNormal.m` as an example; both matlab-files are avaiable in `mfilesforMC2025jun.zip`). Use the sampler to generate pseudo-observations from which the variance can be estimated.

SOLUTION

Random Number Generator (Monte Carlo)

Part (a)

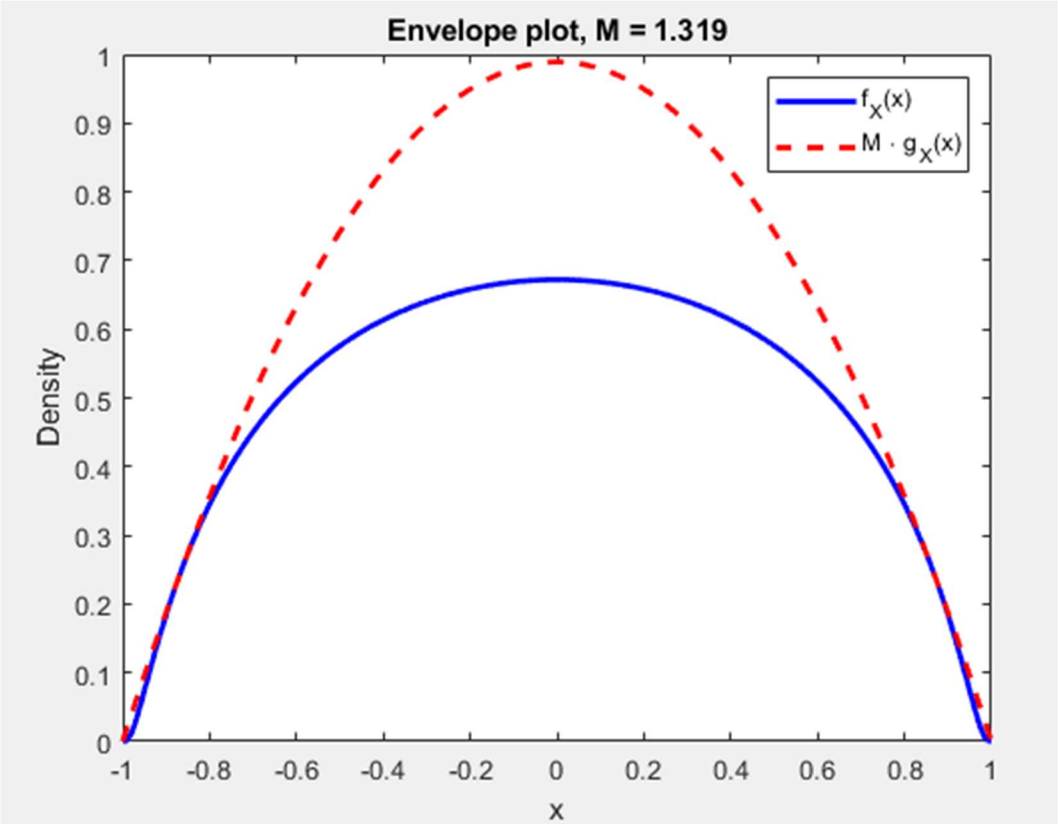

Figure 5a.2 – Envelope plot, (M = 1.319)

The blue line is your target normalized density, while the red dashed line is the proposal envelope. The envelope is always above or equal to the target, as required for valid rejection sampling. The value M = 1.319 was determined as the maximum of :

$$f_X(x) \ / \ g_X(x)$$

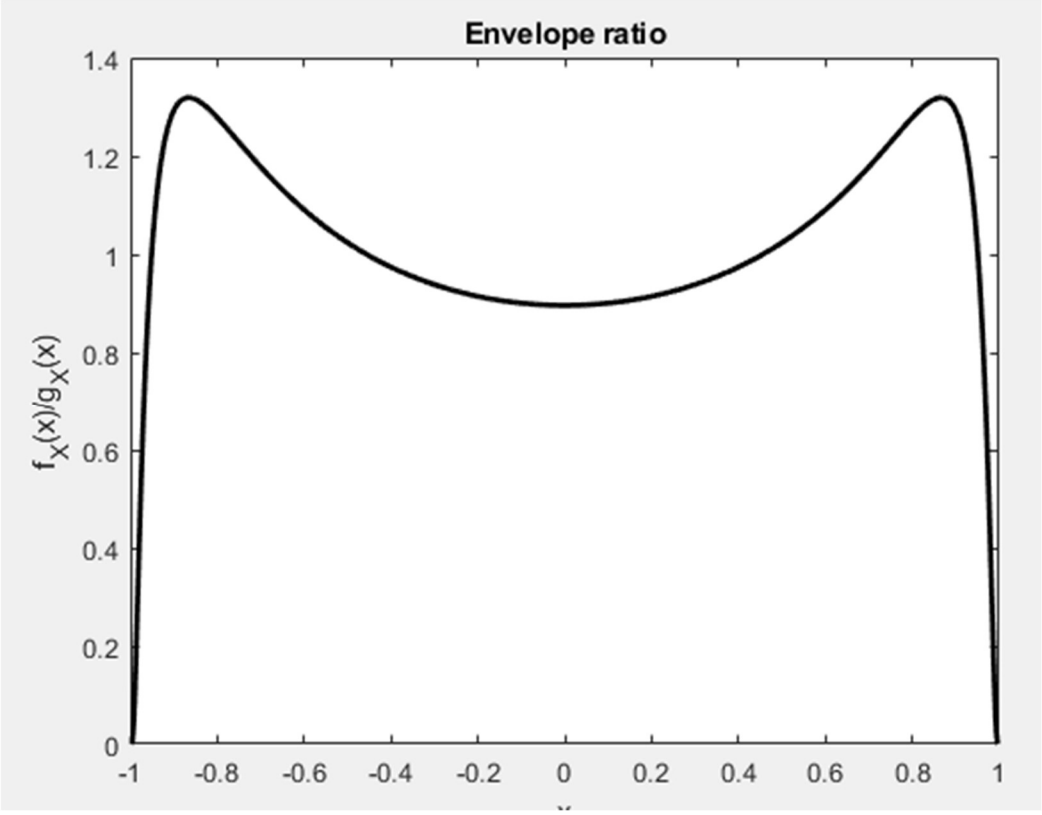

Figure 5a.2 – Envelope ratio

This plot shows :

$$f_X(x) \ / \ g_X(x)$$

...across x in the range [-1,1]. The maximum value reached is the value of M : 1.319. The curve peaks near the boundaries (but not at ±1, as both go to zero there), which is typical for this kind of ratio.

Part (b)

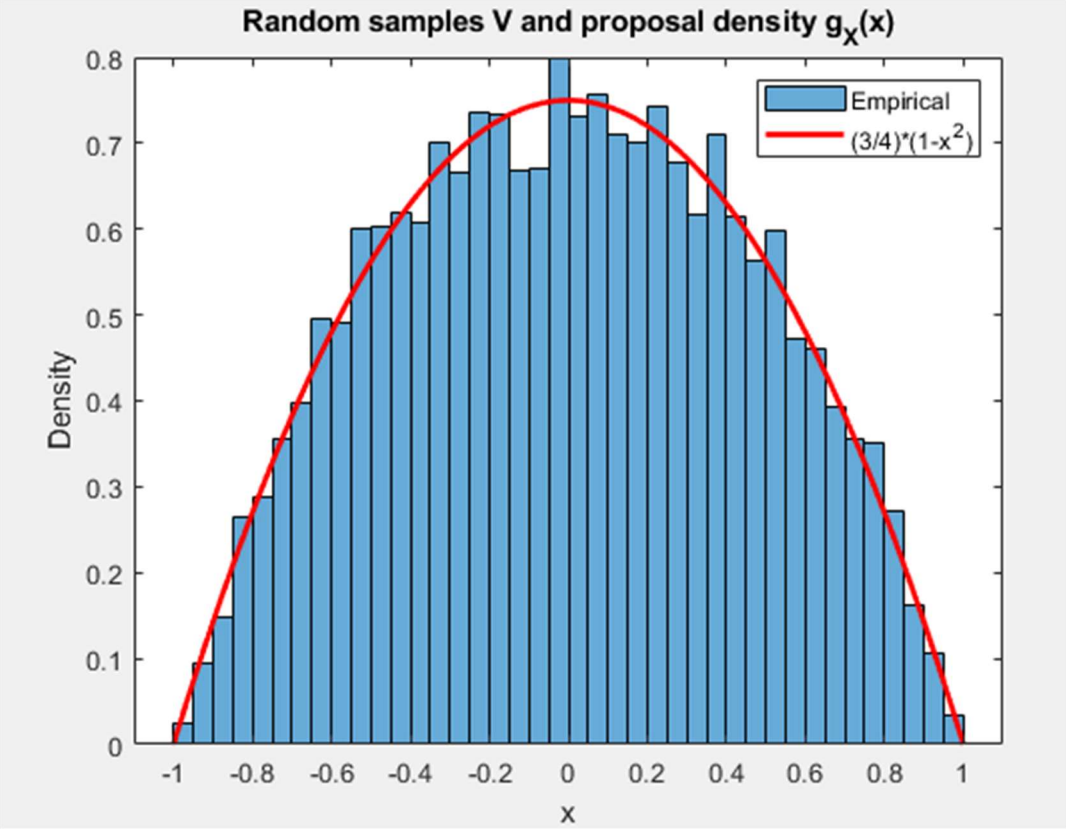

Figure 5b.1 – Random Samples and Proposal Density : n = 10000

```
a = 2; b = 2;
S = -log(rand(a+b,n));
U = sum(S(1:a,1:n))./sum(S);
V = 2*U-1;
```

This method works because the ratio of the sum of the first a to the sum of all a+b independent exponential random variables is distributed according to the Beta(a, b) distribution. When a = b = 2, this gives Beta(2,2) samples on the range [0,1], which are then linearly transformed, normalized to the range [-1,1] via V = 2U-1. The resulting variable V has the density :

$$g_X(x) = (3/4)*(1-x^2)$$

Part (c)

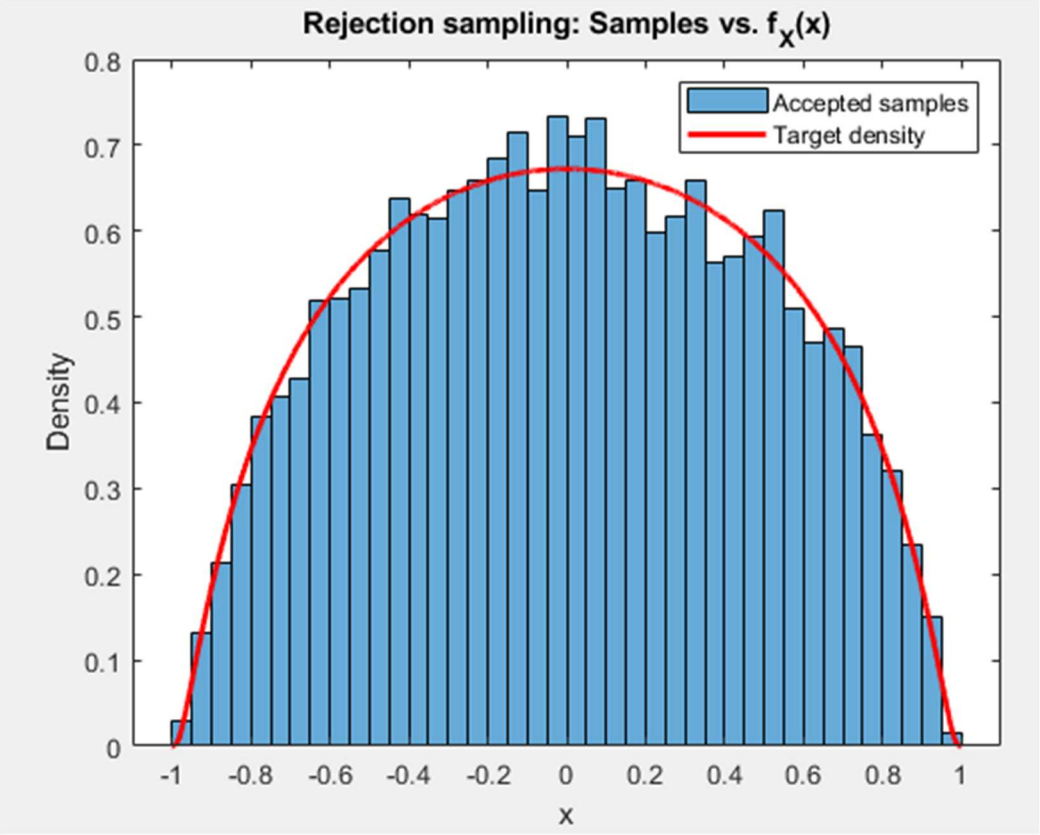

Figure 5c.1 – Rejection Sampling implemented for:

$$U \geq \frac{f_X(V)}{M g_X(V)}$$

The blue histogram shows the Monte Carlo samples. The red curve is the target density:

$$f_X(x)$$

The match between them is correlated, indicating that the samples are distributed according to the target density as desired. This confirms the rejection sampling algorithm is implemented correctly.

Part (d)

```
fXoverMgX = exp(-1 ./ sqrt(1 - V.^2)) ./ (A * (1 - V.^2));

reject = reject(U > fXoverMgX);

n = 10000
```

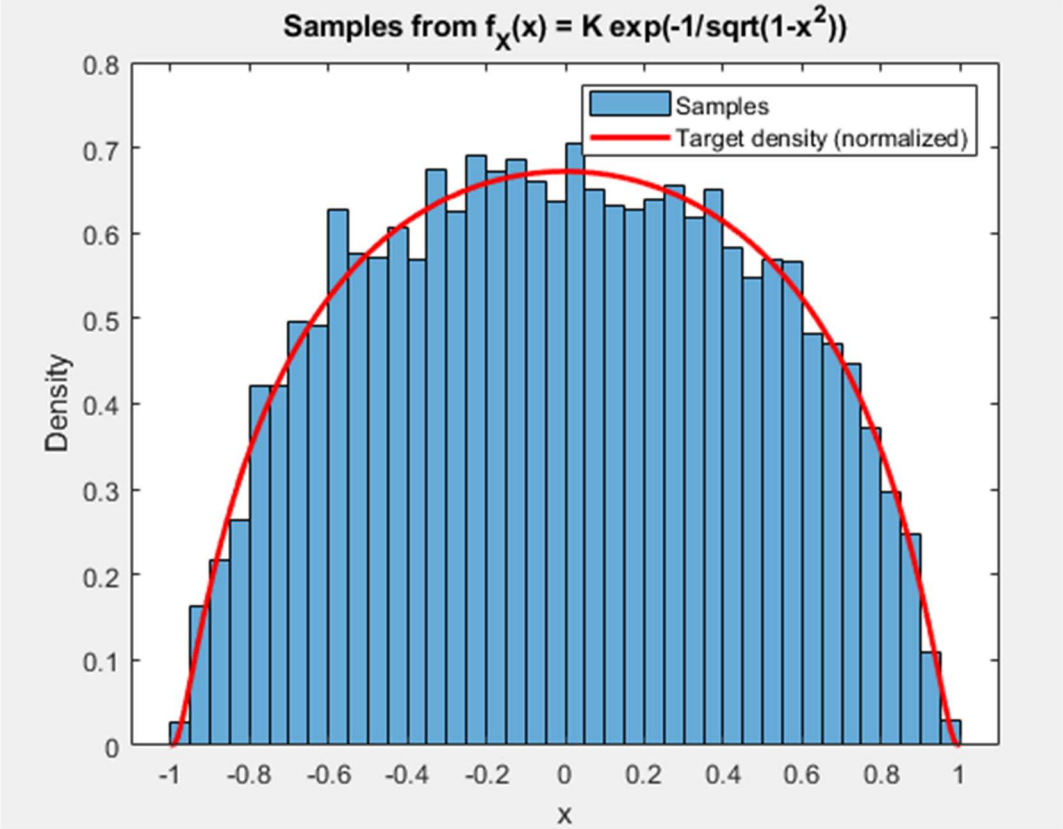

Figure 5d.1 – Samples vs Normalized Target Density

Returned Estimated variance: 0.22471

The blue histogram represents sampled data from the rejection sampler. The red curve is the target density properly normalized. Both curves are well-aligned, indicating that the code produces the correct distribution.

ECOLE
POLYTECHNIQUE
DE BRUXELLES

STAT-F408 COMPUTATIONAL STATISTICS HOMEWORK 6 - JUNE 2025

MODEL SELECTION IN GLM : POISSON REGRESSION

Stephen Worrall

Professor Maarten Jansen

PROBLEM

Random number generator (Monte Carlo)

6. The Matlab file `simulateGibbssamplerN2025.m` performs a simulation of a Gibbs sampler and a Metropolis-Hastings sampler for a multivariate normal vector with zero mean and given (known) covariance matrix. In this case, the use of Gibbs sampler is not necessary, since a multivariate normal vector $\boldsymbol{X}$ can always be generated from a vector of independent standard normal random variables $\boldsymbol{Z}$ by $\boldsymbol{X} = \mathbf{A}\boldsymbol{Z}$, provided that $\mathbf{A}\mathbf{A}^T = \boldsymbol{\Sigma}_{\boldsymbol{X}}$.

   (a) For $d = 2, 3, 4, 5, 6, 7, 8, ...$ run

   ```
   studentnumber = <your student ID>
   dimension = d;
   simulateGibbssamplerN2025
   ```

   and comment on the performance of the Gibbs and Metropolis-Hastings samplers.

   (b) Which proposal distribution is being used in the Metropolis-Hastings sampler? Explain that the use of this proposal distribution in the context of 1D rejection sampling for the generation of normal random variables would not be possible. Explain also why this problem is irrelevant in the Metropolis-Hastings sampler.

SOLUTION

Random number generator (monte carlo)

Part (a)

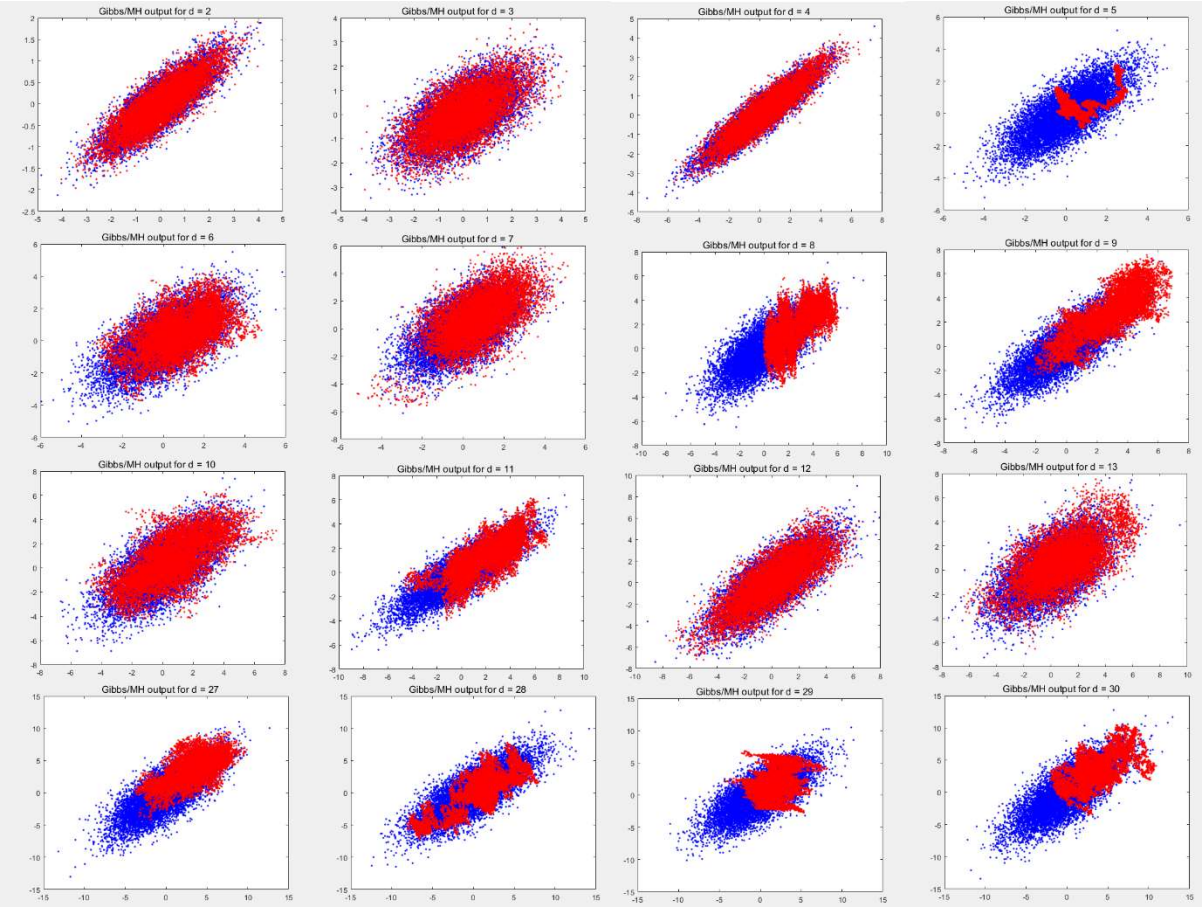

Figure 6a.1 – monte carlo random distribution (blue) vs Gibbs Sampler (red)

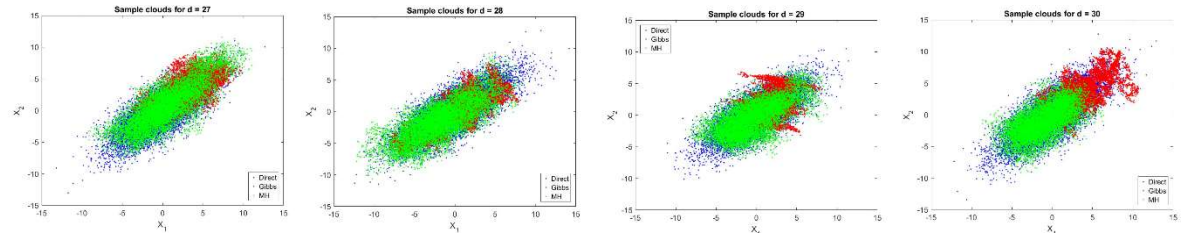

Figure 6a.2 – For high values of d = [27:30] monte carlo vs Gibbs Sampler and Metropolis-Hastings sampler MATLAB

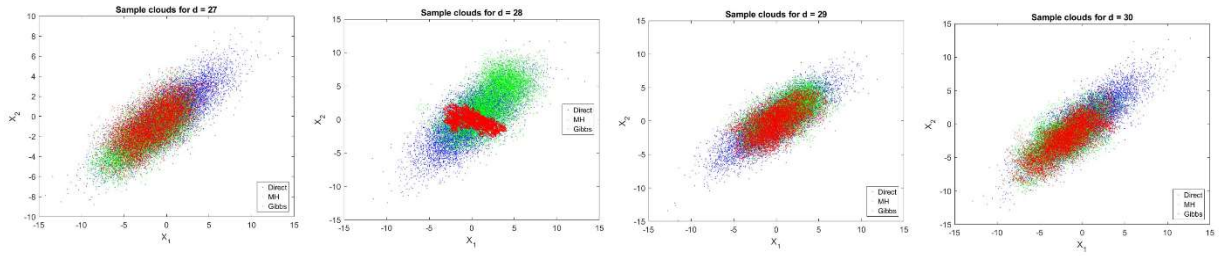

Figure 6a.3 – For the same high values of d = [27:30] monte carlo vs Gibbs Sampler and Metropolis-Hastings sampler from Octave

In figure 6a.1 Gibbs Sampler is encapsulated and run in a loop for d up to 30, I ran this on both MATLAB (shown) and Octave figure 6a.2 and 6a.3, the results looked different, likely different seeding methods used between the programs, but the tight clustering in Gibbs Sampler apparent in d = 5, 8, 29, 30 was seen also in the octave plot but for different values of d (28).

Ignoring these clustering plots, as the dimension d rises, in some select there is a stark contrast between the mixing behaviour of the Gibbs sampler (red) and the Metropolis-Hastings sampler (green). For small d, both samplers produce points that fill the Monte Carlo generated data distribution well, but as d increases, the Metropolis-Hastings samples show strong autocorrelation and clusters, indicating poor mixing and poor exploration of the target distribution. In contrast, the Gibbs sampler remains efficient even as d grows, with new samples quickly forgetting their starting points and covering the target distribution robustly, while Gibbs sampling, when conditionals are tractable, remains effective.

Part (b)

The Metropolis-Hastings sampler uses a uniform proposal distribution, where a candidate is generated by adding an independent range[-1,1] to each coordinate of the current state. This type of uniform proposal could not be used for direct rejection sampling of a normal distribution, because the normal density is nonzero everywhere but the uniform is zero outside its finite interval, and so no global bounding constant exists. However, in the Metropolis-Hastings framework, the algorithm constructs a Markov chain that does not require the proposal to envelope the target everywhere; only local moves and acceptance probabilities are needed.