We demonstrate I-prior modelling on a toy data set to illustrate the Nyström method, as well as three other real-data examples. All of the analyses were conducted in R, and I-prior model estimation was done using the **iprior** package (Jamil, 2017). The **iprior** package comes documented with usage examples in the vignette. The complete source code for replication is found at `http://myphdcode.haziqj.ml`. Note that in all of these examples, **??**–**??** were assumed.

0.0.1 Random effects models

In this section, a comparison between a standard random effects model and the I-prior approach for estimating varying intercept and slopes model is illustrated. The example concerns control data[1] from several runs of radioimmunoassays (RIA) for the protein insulin-like growth factor (IGF-I) (explained in further detail in Davidian and Giltinan, 1995, §3.2.1). RIA is an in vitro assay technique which is used to measure concentration of antigens—in our case, the IGF-I proteins. When an RIA is run, control samples at known concentrations obtained from a particular lot are included for the purpose of assay quality control. It is expected that the concentration of the control material remains stable as the machine is used, up to a maximum of about 50 days, at which point control samples from a new batch is used to avoid degradation in assay performance.

```
R> data(IGF, package = "nlme")
R> head(IGF)
```

```
## Grouped Data: conc ~ age | Lot
##   Lot age conc
## 1   1   7 4.90
## 2   1   7 5.68
## 3   1   8 5.32
## 4   1   8 5.50
## 5   1  13 4.94
## 6   1  13 5.19
```

The data consists of IGF-I concentrations (`conc`) from control samples from 10 different lots measured at differing `age`s of the lot. The data were collected with the aim of identifying possible trends in control values `conc` with `age`, ultimately investigating whether or not the usage protocol of maximum sample age of 50 days is justified. J. C. Pinheiro and Bates (2000) remarks that this is not considered a longitudinal problem because different samples were used at each measurement.

[1]This data is available in the R package **nlme** (J. Pinheiro et al., 2017).

We shall model the IGF data set using the I-prior methodology using the ANOVA-decomposed regression function

$$f(\texttt{age}, \texttt{Lot}) = f_1(\texttt{age}) + f_2(\texttt{Lot}) + f_{12}(\texttt{age}, \texttt{Lot})$$

where f_1 lies in the linear RKHS $\mathcal{F}_1$, f_2 in the Pearson RKHS $\mathcal{F}_2$ and f_{12} in the tensor product space $\mathcal{F}_{12} = \mathcal{F}_1 \otimes \mathcal{F}_2$. The regression function f then lies in the RKHS $\mathcal{F} = \mathcal{F}_1 \oplus \mathcal{F}_2 \oplus \mathcal{F}_{12}$ with kernel equal to the sum of the kernels from each of the RKHSs. The explanation here is that the `conc` levels are assumed to be related to both `age` and `Lot`, and in particular, the contribution of `age` on `conc` varies with each individual `Lot`. This gives the intended effect of a linear mixed-effects model, which is thought to be suitable in this case, in order to account for within-lot and between-lot variability. We first fit the model using the **iprior** package, and then compare the results with the standard random effects model using the R command `lme4::lmer()`. The command to fit the I-prior model using the EM algorithm is

```
R> mod.iprior <- iprior(conc ~ age * Lot, IGF, method = "em")

## ========================================
## Converged after 57 iterations.

R> summary(mod.iprior)

## Call:
## iprior(formula = conc ~ age * Lot, data = IGF, method = "em")
##
## RKHS used:
## Linear (age)
## Pearson (Lot)
##
## Residuals:
##     Min.  1st Qu.   Median 3rd Qu.    Max.
## -4.4889 -0.3798 -0.0090   0.2563   4.3973
##
## Hyperparameters:
##             Estimate   S.E.      z P[|Z>z|]
## lambda[1]    0.0000 0.0002 -0.004    0.997
## lambda[2]    0.0007 0.0030  0.238    0.812
## psi          1.4576 0.1366 10.672   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Closed-form EM algorithm. Iterations: 57/100
```

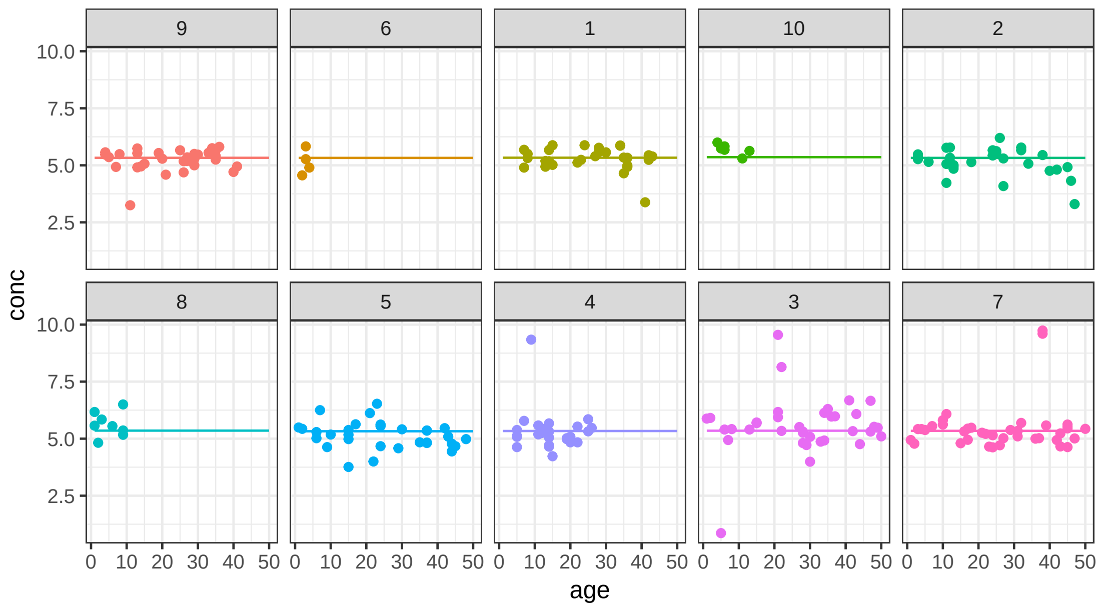

Figure 1: Plot of fitted regression line for the I-prior model on the IGF data set, separated into each of the 10 lots.

fig:IGF.mod.iprior.plot

```
## Converged to within 1e-08 tolerance. Time taken: 2.882966 secs
## Log-likelihood value: -291.9033
## RMSE of prediction: 0.8273639 (Training)
```

To make inference on the covariates, we look at the scale parameters `lambda`. We see that both scale parameters for `age` and `Lot` are close to zero, and a test of significance is not able to reject the hypothesis that these parameters are indeed null. We conclude that neither `age` nor `Lot` has a linear effect on the `conc` levels. The plot of the fitted regression line in Figure 1 does show an almost horizontal line for each `Lot`.

The standard random effects model, as explored by Davidian and Giltinan (1995) and J. C. Pinheiro and Bates (2000), is

$$\texttt{conc}_{ij} = \beta_{0j} + \beta_{1j}\texttt{age}_{ij} + \epsilon_{ij}$$
$$\begin{pmatrix}\beta_{0j}\\ \beta_{1j}\end{pmatrix} \sim \mathrm{N}\left(\begin{pmatrix}\beta_0\\ \beta_1\end{pmatrix}, \begin{pmatrix}\sigma_0^2 & \sigma_{01}\\ \sigma_{01} & \sigma_1^2\end{pmatrix}\right)$$
$$\epsilon_{ij} \sim \mathrm{N}(0, \sigma^2)$$

for $i = 1, \dots, n_j$ and the index j representing the 10 `Lots`. Fitting this model using `lmer`, we can test for the significance of the fixed effect β_0, for which we find that it is not (p-value $= 0.616$), and arrive at the same conclusion as in the I-prior model.

```
R> (mod.lmer <- lmer(conc ~ age + (age | Lot), IGF))

## Linear mixed model fit by REML ['lmerModLmerTest']
## Formula: conc ~ age + (age | Lot)
##    Data: IGF
## REML criterion at convergence: 594.3662
## Random effects:
##  Groups   Name        Std.Dev. Corr
##  Lot      (Intercept) 0.082507
##           age         0.008092 -1.00
##  Residual             0.820628
## Number of obs: 237, groups:  Lot, 10
## Fixed Effects:
## (Intercept)          age
##    5.374974    -0.002535

R> coef(summary(mod.lmer))

##                 Estimate  Std. Error        df    t value
## (Intercept)  5.374974121 0.107488043 41.575698 50.0053212
## age         -0.002534995 0.005044317  9.513561 -0.5025448
##                 Pr(>|t|)
## (Intercept) 9.096157e-39
## age         6.267116e-01
```

However, we notice that the package reports a perfect negative correlation between the random effects, σ_{01}. This indicates a potential numerical issue when fitting the model—a value of exactly -1, 0 or 1 is typically imposed by the package to force through estimation in the event of non-positive definite covariance matrices arising. We can inspect the eigenvalues of the covariance matrix for the random effects to check that they are indeed non-positive definite. One of the eigenvalues was found to be negative, so the covariance matrix is non-positive definite.

```
R> eigen(VarCorr(mod.lmer)$Lot)

## eigen() decomposition
## $values
## [1]  6.872939e-03 -1.355253e-20
##
## $vectors
##              [,1]        [,2]
## [1,] -0.99522490 -0.09760839
## [2,]  0.09760839 -0.99522490
```

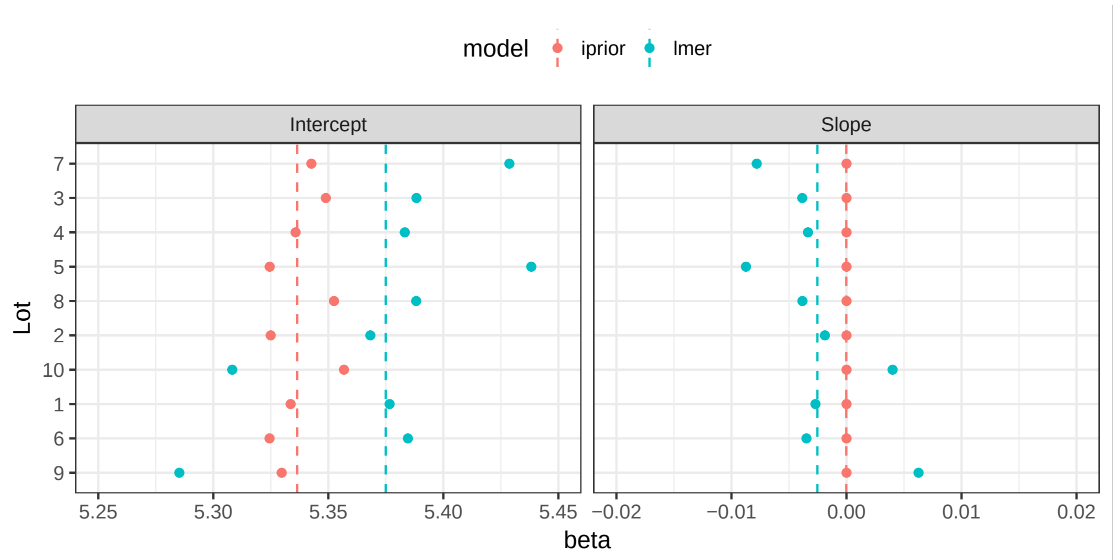

Figure 2: A comparison of the estimates for random intercepts and slopes (denoted as points) using the I-prior model and the standard random effects model. The dashed vertical lines indicate the fixed effect values.

fig:IGF.plot.beta

Degenerate covariance matrices often occur in models with a large number of random coefficients, and in cases where values of the variance components are estimated at the boundary. These are typically solved by setting restrictions which then avoids overparameterising the model. One advantage of the I-prior method for varying intercept/slopes model is that the positive-definiteness is automatically taken care of. Furthermore, I-prior models typically require fewer parameters to fit a similar varying intercept/slopes model—in the above example, the I-prior model estimated only three parameters, while the standard random effects model estimated a total of six parameters.

It is also possible to "recover" the estimates of the standard random effects model from the I-prior model, albeit in a slighly manual fashion (refer to **??**). Denote by f^j the individual linear regression lines for each of the $j = 1, \dots, 10$ `Lots`. Then, each of these f^j has a slope and intercept for which we can estimate from the fitted values $\hat{f}^j(x_{ij})$, $i = 1, \dots, n_j$. This would give us the estimate of the posterior mean of the random intercepts and slopes; these would typically be obtained using empirical-Bayes methods in the case of the standard random effects model.

Furthermore, σ_0^2 and σ_1^2 gives a measure of variability of the intercepts and slopes of the different groups, and this can be calculated from the estimates of the random intercepts and slopes. In the same spirit, $\rho_{01} = \sigma_{01}/(\sigma_0\sigma_1)$, which is the correlation between the random intercept and slope, can be similarly calculated. Finally, the fixed effects can be estimated from the intercept and slope of the best fit line running through the I-prior estimated `conc` values. The intuition for this is that the fixed effects are essentially the ordinary least squares (OLS) of a linear model if the groupings are disregarded. Figure 2 illustrates the differences in the estimates for the random coefficients, while Table 1 illustrates the differences in the estimates for the covariance matrix. Minor differences do exist, with the most noticeable one being that the slopes in the I-prior model are categorically estimated as zero, and the sign of the correlation ρ_{01} being opposite in both models. Even so, the conclusions from both models are similar.

Table 1: A comparison of the estimates for the covariance matrix of the random effects using the I-prior model and the standard random effects model.

tab:igf

Parameter	iprior	lmer
σ_0	0.012	0.083
σ_1	0.000	0.008
ρ_{01}	0.690	-1.000

0.0.2 Longitudinal data analysis

sec:cows

We consider a balanced longitudinal data set consisting of weights in kilograms of 60 cows, 30 of which were randomly assigned to treatment group A, and the remaining 30 to treatment group B. The animals were weighed 11 times over a 133-day period; the first 10 measurements for each animal were made at two-week intervals and the last measurement was made one week later. This experiment was reported by Kenward (1987), and the data set is included as part of the package **jmcm** (J. Pan and Y. Pan, 2017) in R. The variable names have been renamed for convenience.

```
R> data(cattle, package = "jmcm")
R> names(cattle) <- c("id", "time", "group", "weight")
R> cattle$id <- as.factor(cattle$id)  # convert to factors
R> levels(cattle$group) <- c("Treatment A", "Treatment B")
R> str(cattle)
```

```
## 'data.frame': 660 obs. of  4 variables:
##  $ id    : Factor w/ 60 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1..
##  $ time  : num  0 14 28 42 56 70 84 98 112 126 ...
##  $ group : Factor w/ 2 levels "Treatment A",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ weight: int  233 224 245 258 271 287 287 287 290 293 ...
```

The response variable of interest are the `weight` growth curves, and the aim is to investigate whether a treatment effect is present. The usual approach to analyse a longitudinal data set such as this one is to assume that the observed growth curves are realizations of a Gaussian process. For example, Kenward (1987) assumed a so-called ante-dependence structure of order k, which assumes an observation depends on the previous k observations, but given these, is independent of any preceeding observations.

tab:cowmodel

Table 2: A brief description of the five models fitted using I-priors.

Model	Explanation	Formula (`weight ~ ...`)
1	Growth does not vary with treatment nor among cows	`time`
2	Growth varies among cows only	`id * time`
3	Growth varies with treatment only	`group * time`
4	Growth varies with treatment and among cows	`id * time + group * time`
5	Growth varies with treatment and among cows, with an interaction effect between treatment and cows	`id * group * time`

Using the I-prior, it is not necessary to assume the growth curves were drawn randomly. Instead, it suffices to assume that they lie in an appropriate function class. For this example, we assume that the function class is the fBm RKHS, i.e., we assume a smooth effect of time on weight. The growth curves form a multidimensional (or functional) response equivalent to a "wide" format of representing repeated measures data. In our analysis using the **iprior** package, we used the "long" format and thus our (unidimensional) sample size n is equal to 60 cows $\times$ 11 repeated measurements. We also have two covariates potentially influencing growth, namely the cow subject `id` and also treatment `group`. The regression model can then be thought of as

$$\begin{gathered}\texttt{weight} = \alpha + f(\texttt{id}, \texttt{group}, \texttt{time}) + \epsilon \\ \epsilon \sim \mathrm{N}(0, \psi^{-1}).\end{gathered}$$

We assume iid errors, and in addition to a smooth effect of `time`, we further assume a nominal effect of both cow `id` and treatment `group` using the Pearson RKHS. In the **iprior** package, factor type objects are treated with the Pearson kernel automatically, and the only `model` option we need to specify is the `kernel = "fbm"` option for the `time` variable. We have opted not to estimate the Hurst coefficient in the interest of computational time, and instead left it at the default value of 0.5. Table 2 explains the five models we have fitted.

The simplest model fitted was one in which the growth curves do not depend on the treatment effect or individual cows. We then added treatment effect and the cow `id` as covariates, separately first and then together at once. We also assumed that both of these covariates are time-varying, and hence added also the interaction between these covariates and the `time` variable. The final model was one in which an interaction between treatment effect and individual cows was assumed, which varied over time.

All models were fitted using the `mixed` estimation method. Compared to the EM algorithm alone, we found that the combination of direct optimisation with the EM algorithm fits the model about six times faster for this data set due to slow convergence of EM algorithm. Here is the code and output for fitting the first model:

```
R> # Model 1: weight ~ f(time)
R> set.seed(456)
R> (mod1 <- iprior(weight ~ time, cattle, kernel = "fbm", method = "mixed"))

## Running 5 initial EM iterations
## ======================================================================
## Now switching to direct optimisation
## final  value 1394.615062
## converged
## Log-likelihood value: -2789.231
##
##  lambda     psi
## 0.83592 0.00375
```

Table 3: Summary of the five I-prior models fitted to the cow data set. Error S.D. refers to the inverse square root of the error precision, $\psi^{-1/2}$.

tab:cowresults

Model	Formula (`weight ~ ...`)	Log-likelihood	Error S.D.	Number of parameters
1	`time`	-2789.23	16.33	1
2	`id * time`	-2791.42	16.39	2
3	`group * time`	-2295.16	3.68	2
4	`id * time + group * time`	-2270.85	3.39	3
5	`id * group * time`	-2249.26	3.90	3

The results of the model fit are summarised in Table 3. We can test for a treatment effect by testing Model 4 against the alternative that Model 2 is true. The log-likelihood ratio test statistic is $D = -2(-2791.42 - (-2270.85)) = 1041.14$, which has an asymptotic chi-squared distribution with $3 - 2 = 1$ degree of freedom. The p-value for this likelihood ratio test is less than 10^{-6}, so we conclude that Model 4 is significantly better.

We can next investigate whether the treatment effect differs among cows by comparing Model 5 against Model 4. As these models have the same number of parameters, we can simply choose the one with the higher likelihood, which is Model 5. We conclude that treatment does indeed have an effect on growth, and that the treatment effect differs among cows. A plot of the fitted regression curves onto the cow data set is shown in Figure 3.

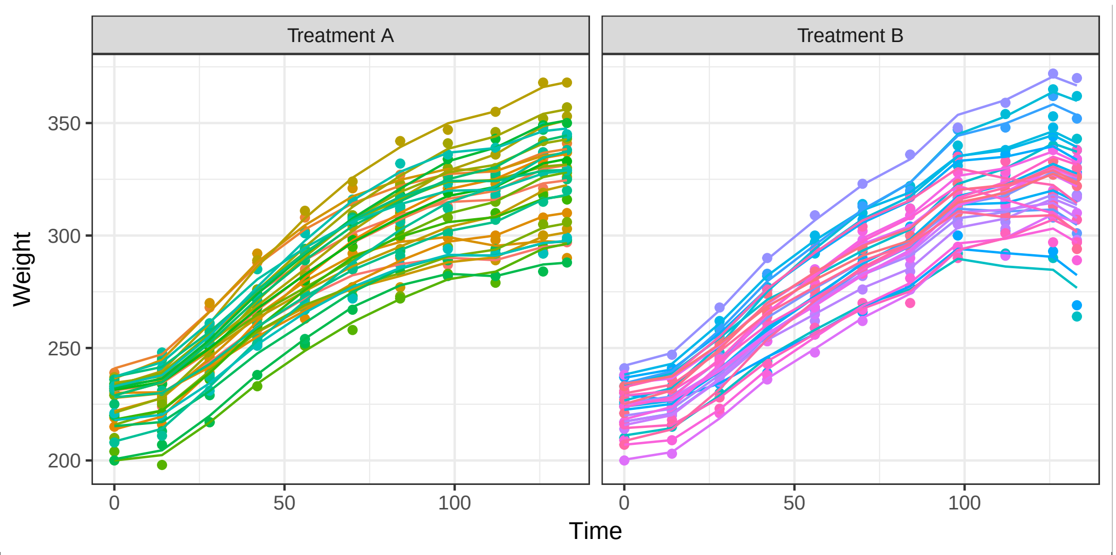

Figure 3: A plot of the I-prior fitted regression curves from Model 5. In this model, growth curves differ among cows and by treatment effect (with an interaction between cows and treatment effect), thus producing these 60 individual lines, one for each cow, split between their respective treatment groups (A or B).

fig:cows.plot

0.0.3 Regression with a functional covariate

We illustrate the prediction of a real valued response with a functional covariate using a widely analysed data set for quality control in the food industry. The data[2] contain samples of spectrometric curve of absorbances of 215 pieces of finely chopped meat, along with their water, fat and protein content. These data are recorded on a Tecator Infratec Food and Feed Analyzer working in the wavelength range 850–1050 nm by the Near Infrared Transmission (NIT) principle. Absorption data has not been measured continuously, but instead 100 distinct wavelengths were obtained. Figure 4 shows a sample of 10 such spectrometric curves.

For our analyses and many others' in the literature, the first 172 observations in the data set are used as a training sample for model fitting, and the remaining 43 observations as a test sample to evaluate the predictive performance of the fitted model. The focus here is to use the **iprior** package to fit several I-prior models to the Tecator data set, and calculate out-of-sample predictive error rates. We compare the predictive performance of I-prior models against Gaussian process regression and the many other different methods applied on this data set. These methods include neural networks (Thodberg, 1996), kernel smoothing (Ferraty and Vieu, 2006), single and multiple index functional regression models (Chen et al., 2011), sliced inverse regression (SIR) and sliced average variance estimation (SAVE), multivariate adaptive regression splines (MARS),

[2]Obtained from Tecator (see `http://lib.stat.cmu.edu/datasets/tecator` for details). We used the version made available in the dataframe `tecator` from the R package **caret** (Kuhn et al., 2017).

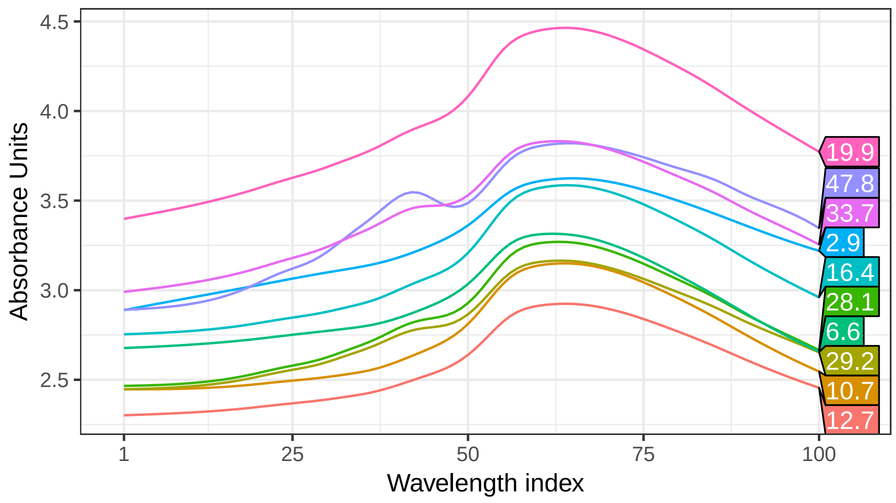

Figure 4: Sample of spectrometric curves used to predict fat content of meat. For each meat sample the data consists of a 100 channel spectrum of absorbances and the contents of moisture, fat (numbers shown in boxes) and protein measured in percent. The absorbance is $-\log 10$ of the transmittance measured by the spectrometer. The three contents, measured in percent, are determined by analytic chemistry.

fig:tecator.data

partial least squares (PLS), and functional additive model with and without component selection (FAM & CSEFAM). An analysis of this data set using the SIR and SAVE methods were conducted by Lian and Li (2014), while the MARS, PLS and (CSE)FAM methods were studied by Zhu et al. (2014). Table 4 tabulates the results of all of these methods from these various references.

Assuming a regression model as in **??**, we would like to model the `fat` content y_i using the spectral curves x_i. Let $x_i(t)$ denote the absorbance for wavelength $t = 1, \dots, 100$. From Figure 4, it appears that the curves are smooth enough to be differentiable, and therefore it is reasonable to assume that they lie in the Sobolev-Hilbert space as discussed in **??**. We take first differences of the 100-dimensional matrix, which leaves us with the 99-dimensional covariate saved in the object named `absorp`. The `fat` and `absorp` data have been split into `*.train` and `*.test` samples, as mentioned earlier. Our first modelling attempt is to fit a linear effect by regressing the responses `fat.train` against a single high-dimensional covariate `absorp.train` using the linear RKHS and the direct optimisation method.

```
R> # Model 1: Canonical RKHS (linear)
R> (mod1 <- iprior(y = fat.train, absorp.train))

## iter   10 value 222.653144
## final  value 222.642108
## converged
## Log-likelihood value: -445.2844
```

```
##
##     lambda        psi
## 4576.86595    0.11576
```

Our second and third model uses polynomial RKHSs of degrees two and three, which allows us to model quadratic and cubic terms of the spectral curves respectively. We also opted to estimate a suitable offset parameter, and this is called to `iprior()` with the option `est.offset = TRUE`. Each of the two models has a single scale parameter, an offset parameter, and an error precision to be estimated. The direct optimisation method has been used, and while both models converged regularly, it was noticed that there were multiple local optima that hindered the estimation (output omitted).

```
R> # Model 2: Polynomial RKHS (quadratic)
R> mod2 <- iprior(y = fat.train, absorp.train, kernel = "poly2",
+                 est.offset = TRUE)
R> # Model 3: Polynomial RKHS (cubic)
R> mod3 <- iprior(y = fat.train, absorp.train, kernel = "poly3",
+                 est.offset = TRUE)
```

Next, we attempt to fit a smooth dependence of fat content on the spectrometric curves using the fBm RKHS. By default, the Hurst coefficient for the fBm RKHS is set to be 0.5. However, with the option `est.hurst = TRUE`, the Hurst coefficient is included in the estimation procedure. We fit models with both a fixed value for Hurst (at 0.5) and an estimated value for Hurst. For both of these models, we encountered numerical issues when using the direct optimisation method. The L-BFGS algorithm kept on pulling the hyperparameter towards extremely high values, which in turn made the log-likelihood value greater than the machine's largest normalised floating-point number (`.Machine$double.xmax = 1.797693e+308`). To circumvent this issue, we used the EM algorithm to estimate the fixed Hurst model, and the `mixed` method for the estimated Hurst model. For both models, the `stop.crit` was relaxed and set to `1e-3` for quicker convergence, though this did not affect the predictive abilities compared to a more stringent `stop.crit`.

```
R> # Model 4: fBm RKHS (default Hurst = 0.5)
R> (mod4 <- iprior(y = fat.train, absorp.train, kernel = "fbm",
+                  method = "em", control = list(stop.crit = 1e-3)))

## ==============================================
## Converged after 65 iterations.
## Log-likelihood value: -204.4592
##
##     lambda        psi
##    3.24112 1869.32897
```

```
R> # Model 5: fBm RKHS (estimate Hurst)
R> (mod5 <- iprior(fat.train, absorp.train, kernel = "fbm", method = "mixed",
+                    est.hurst = TRUE, control = list(stop.crit = 1e-3)))
```

```
## Running 5 initial EM iterations
## ======================================================================
## Now switching to direct optimisation
## iter   10 value 115.648462
## final  value 115.645800
## converged
## Log-likelihood value: -231.2923
##
##    lambda     hurst       psi
## 204.97184   0.70382   9.96498
```

Finally, we fit an I-prior model using the SE RKHS with lengthscale estimated. Here we illustrate the use of the `restarts` option, in which the model is fitted repeatedly from different starting points. In this case, eight random initial parameter values were used and these jobs were parallelised across the eight available cores of the machine. The additional `par.maxit` option in the `control` list is an option for the maximum number of iterations that each parallel job should do. We have set it to 100, which is the same number for `maxit`, but if `par.maxit` is less than `maxit`, the estimation procedure continues from the model with the best likelihood value. We see that starting from eight different initial values, direct optimisation leads to (at least) two log-likelihood optima sites, -231.5 and -680.5.

```
R> # Model 6: SE kernel
R> (mod6 <- iprior(fat.train, absorp.train, est.lengthscale = TRUE,
+                    kernel = "se", control = list(restarts = TRUE,
+                                                  par.maxit = 100)))
```

```
## Performing 8 random restarts on 8 cores
## ======================================================================
## Log-likelihood from random starts:
##     Run 1     Run 2     Run 3     Run 4     Run 5     Run 6     Run 7
## -231.5440 -680.4636 -680.4636 -680.4637 -680.4637 -231.5440 -231.5440
## Continuing on Run 6
## final  value 115.771932
## converged
## Log-likelihood value: -231.544
```

```
##
##      lambda lengthscale         psi
##    96.11515     0.09269     6.15426
```

Predicted values of the test data is obtained using `predict()`. An example for obtaining the first model's predicted values is shown below. The `predict()` method for `ipriorMod` objects also return the test MSE if the vector of test data is supplied.

```
R> predict(mod1, newdata = list(absorp.test), y.test = fat.test)
```

```
## Test RMSE: 2.890353
##
## Predicted values:
##  [1] 43.607 20.444  7.821  4.491  9.044  8.564  7.935 11.615 13.807
## [10] 17.359
## # ... with 33 more values
```

These results are summarised in Table 4. For the I-prior models, a linear effect of the functional covariate gives a training RMSE of 2.89, which is improved by both the qudratic and cubic model. The training RMSE is improved further by assuming a smooth RKHS of functions for f, i.e. the fBm and SE RKHSs. When it comes to out-of-sample test error rates, the cubic model gives the best RMSE out of the I-prior models for this particular data set, with an RMSE of 0.58. This is followed closely by the fBm RKHS with estimated Hurst coefficient (fBm-0.70) and also the fBm RKHS with default Hurst coefficient (fBm-0.50). The best performing I-prior model is only outclassed by the neural networks of Thodberg (1996), who also performed model selection using automatic relevance determination (ARD). The I-prior models also give much better test RMSE than Gaussian process regression.

Table 4: A summary of the root mean squared error (RMSE) of prediction for the I-prior models and various other methods in literature conducted on the Tecator data set. Values for the methods under *Others* were obtained from the corresponding references cited earlier.

tab:tecator

	RMSE	
Model	Train	Test
I-prior		
Linear	2.89	2.89
Quadratic	0.72	0.97
Cubic	0.37	0.58
Smooth (fBm-0.50)	0.00	0.68
Smooth (fBm-0.70)	0.19	0.63
Smooth (SE-0.09)	0.35	1.85
Gaussian process regression[a]		
Linear	0.18	2.36
Smooth (SE-7.04)	0.17	2.10
Others		
Neural network[b]		0.36
Kernel smoothing[c]		1.49
Single/multiple indices model[d]		1.55
Sliced inverse regression		0.90
Sliced average variance estimation		1.70
MARS[e]		0.88
Partial least squares[e]		1.01
FAM[e]		0.92
CSEFAM[e]		0.85

[a] GPR models were fit using `gausspr()` in **kernlab**.
[b] Neural network best results with automatic relevance determination (ARD) quoted.
[c] Data set used was a 160/55 training/test split.
[d] These are results of a leave-one-out cross-validation scheme.
[e] Data set used was an extended version with $n = 240$, and a random 185/55 training/test split.

0.0.4 Using the Nyström method

We investigate the use of the Nyström method of approximating the kernel matrix in estimating I-prior models. Let us revisit the data set generated by **??** described in **??**. The features of this regression function are two large bumps at the centres of the mixed Gaussian PDFs, and also a small bump right after $x > 4.5$ caused by the additional exponential function. The true regression function tends to positive infinity as x increases, and to zero as x decreases. Samples of (x_i, y_i), $i = 1, \dots, 2000$ have been generated by the built-in `gen_smooth()` function, of which the first few lines of the data are shown below.

```
R> dat <- gen_smooth(n = 2000, xlim = c(-1, 5.5), seed = 1)
R> head(dat)
```

```
##            y         X
## 1  0.6803514 -2.608953
## 2  3.6747031 -2.554039
## 3 -1.1563508 -2.381275
## 4  2.2657657 -2.280259
## 5  2.5398243 -2.214122
## 6  1.2929592 -2.170532
```

One could fit the regression model using all available data points, with an I-prior from the fBm-0.5 RKHS of functions as follows (note that the `silent` option is used to suppress the output from the `iprior()` function):

```
R> (mod.full <- iprior(y ~ X, dat, kernel = "fbm",
+                      control = list(silent = TRUE)))
```

```
## Log-likelihood value: -4355.075
##
##  lambda     psi
## 2.30244 0.23306
```

To implement the Nyström method, the option `nystrom = 50` was added to the function call, which uses 50 randomly selected data points for the Nyström approximation.

```
R> (mod.nys <- iprior(y ~ X, dat, kernel = "fbm", nystrom = 50,
+                     control = list(silent = TRUE)))
```

```
## Log-likelihood value: -1945.33
##
##  lambda     psi
## 1.64833 0.13538
```

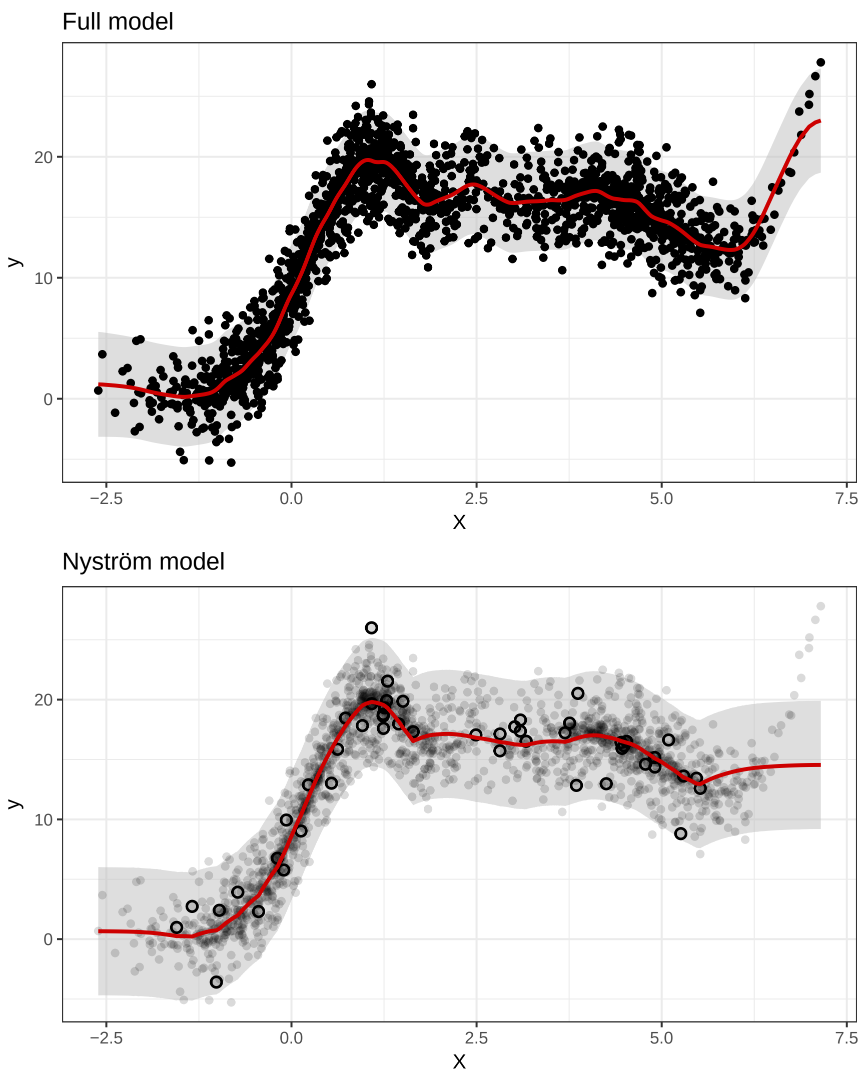

Figure 5: Plot of predicted regression function for the full model (top) and the Nyström approximated method (bottom). For the Nyström plot, the data points that were active are shown by circles with bold outlines.

fig:nystrom.plot

```
R> get_time(mod.full); get_size(mod.full, "MB"); get_prederror(mod.full)

## 12.10819 mins
## 128.2 MB
## Training RMSE
##      2.054232

R> get_time(mod.nys); get_size(mod.nys); get_prederror(mod.nys)

## 1.287808 secs
## 982.2 kB
## Training RMSE
##      2.171928
```

The hyperparameters estimated for both models are slightly different. The log-likelihood is also different, but this is attributed to information loss due to the approximation procedure. Nevertheless, we see from Figure 5 that the estimated regression functions are quite similar in both the full model and the approximated model. The main difference is that the the Nyström method was not able to extrapolate the right hand side of the plot well, because it turns out that there were no data points used from this region. This can certainly be improved by using a more intelligent sampling scheme. The full model took a little over 12 minutes to converge, while the Nyström method took seconds without compromising too much on root mean squared error of predictions. Storage savings is significantly higher with the Nyström method as well.