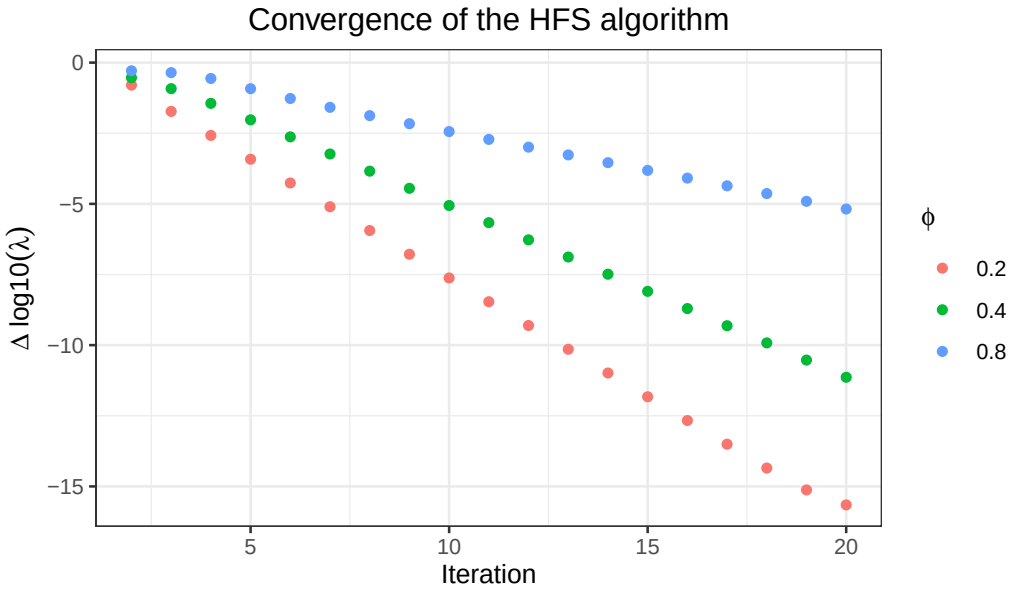
Slow convergence of HFS algorithm



Speed of convergence of the HFS (Harville-Fellner-Schall) algorithm for different values of the standard deviation of the noise (ϕ) in the simulation. R code in `f-HFS-convergence.R`

```
# Slow convergence of Harville-Fellner-Schall algorithm
# A graph in the book 'Practical Smoothing. The Joys of P-splines'
# Paul Eilers and Brian Marx, 2019

library(JOPS)
library(ggplot2)

# Simulate data
m = 100
set.seed(2013)
x = seq(0, 1, length = m)
r = rnorm(m)

# P-spline preparations
xmin = 0
xmax = 1
nseg = 20
B = bbase(x, xmin, xmax, nseg)
n = ncol(B)
d = 2
D = diff(diag(n), diff = d)
P = t(D) %*% D
BtB = t(B) %*% B

L = Las = NULL
nit = 20
Yhat = Y = NULL
phis = c(0.2, 0.4, 0.8)
for (phi in phis) {
  y = sin(10 * x) + r * phi
  Bty = t(B) %*% y

  # Find  lambda with Schall algorithem
  lambda = 1
  las = NULL
```

```
  for (it in 1:nit) {
    a = solve(BtB + lambda * P, Bty)
    G = solve(BtB + lambda * P, BtB)
    ed = sum(diag(G))
    yhat = B %*% a
    s2 = sum((y - yhat) ^ 2) / (m - d - ed)
    v2 = sum((D %*% a) ^2) / ed
#    cat(it, lambda, '\n')
    las = c(las, lambda)
    lambda = s2 / v2
    cat(phi, it, lambda, '\n')
  }

  # Compute convergence rate
  dla =  log10(abs(diff(log10(las))))
  Lsub = data.frame(it = 2:nit, phi = as.factor(phi), dla = dla)
  L = rbind(L, Lsub)

  # Save data and fit for plotting
  Yhat = cbind(Yhat, yhat)
  Y = cbind(Y, y)
  Las = cbind(Las, las)
}

ylab = bquote(Delta  ~ log10(lambda))
plt2 = ggplot(L, aes(x = it, y = dla, group = phi)) +
  geom_point(aes(colour = phi)) +
  xlab("Iteration") +
  ylab(ylab) +  labs(color = bquote(phi)) +
  ggtitle('Convergence of the HFS algorithm') +
  JOPS_theme()

# Plot and save pdf
print(plt2)
```