Supplemental material

A Coefficients, gain and phase shift functions

Figure 1: Coefficients, gain and phase shift functions for the Linear-Constant (LC) filter with $I/C = 3.5$.

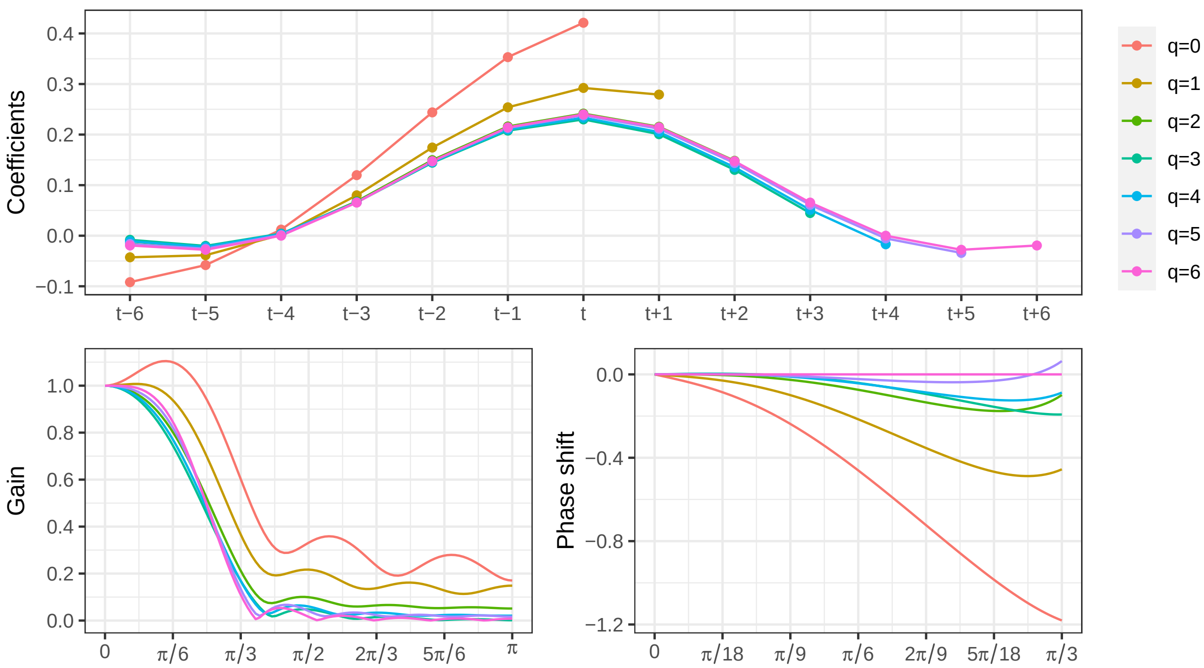


Figure 2: Coefficients, gain and phase shift functions for the Quadratic-Linear (QL) filter with $I/C = 3.5$.

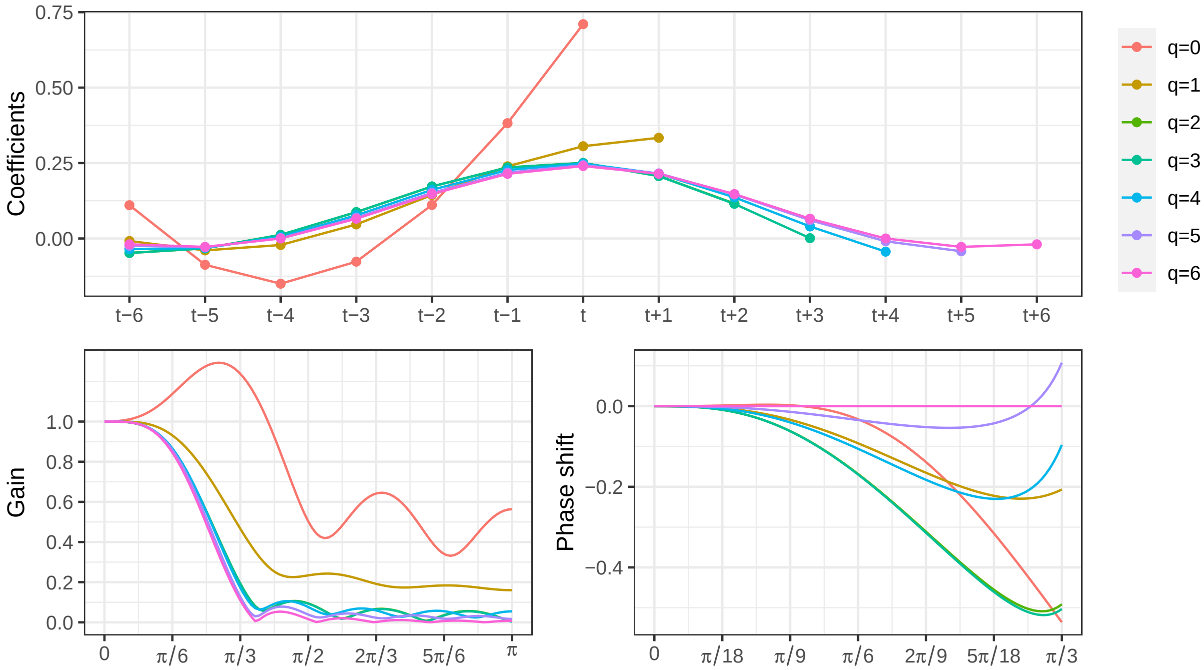


Figure 3: Coefficients, gain and phase shift functions for the Cubic-Quadratic (CQ) filter with $I/C = 3.5$.

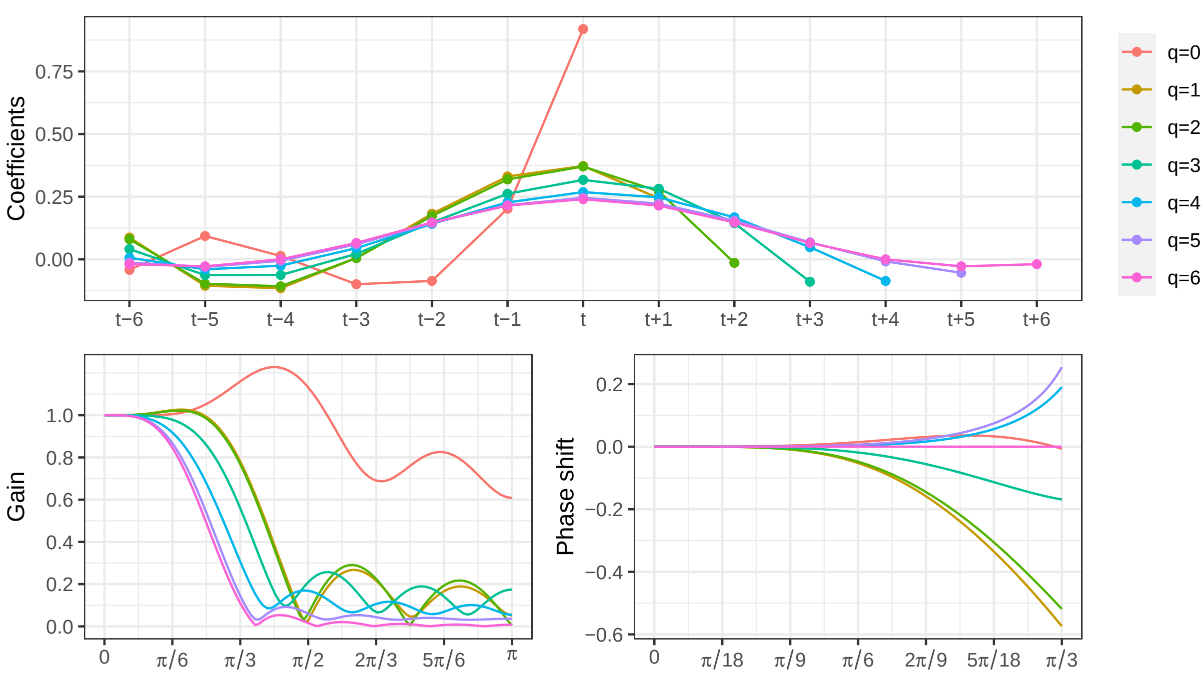
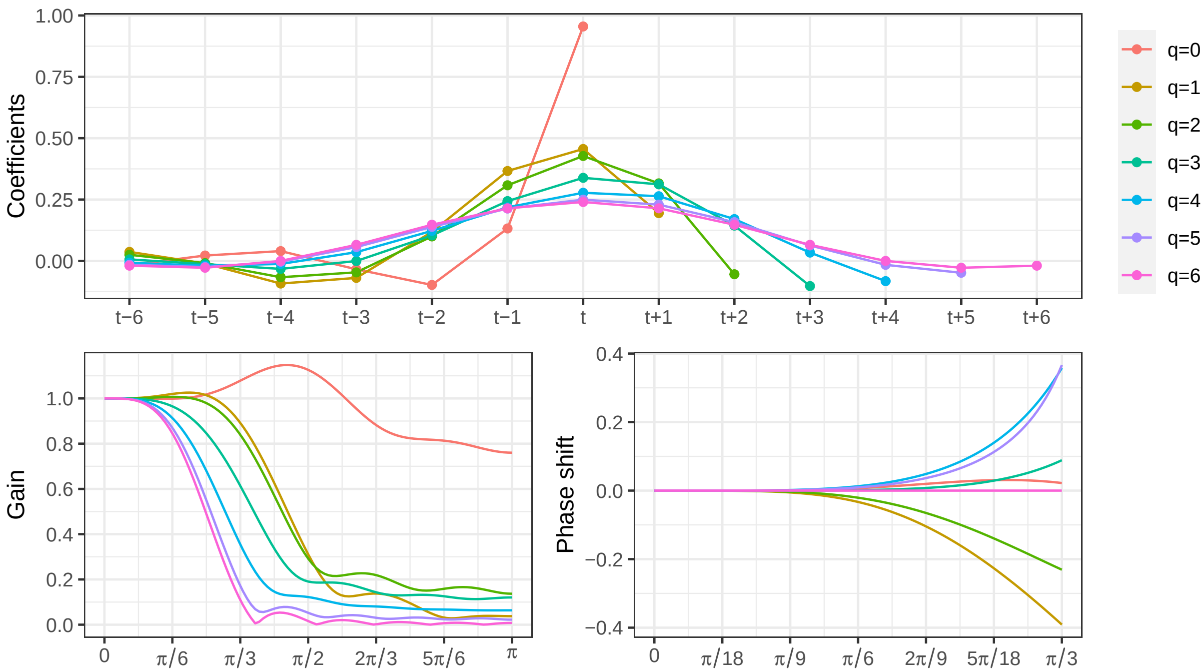


Figure 4: Coefficients, gain and phase shift functions for the direct asymmetric filter (DAF).



B R code example

In this appendix we present the R code that can be used to estimate trend-cycle components for the US Employment with local polynomial filters.

```
# # To install rjd3filters
# remotes::install_github("rjdemetra/rjd3toolkit")
# remotes::install_github("rjdemetra/rjd3x11plus")
# remotes::install_github("rjdemetra/rjd3filters")
library(rjd3filters)
library(ggplot2)
library(patchwork)
library(zoo)
library(forecast)
file <-"https://files.stlouisfed.org/files/htdocs/fred-md/monthly/2022-11.csv"
data <- read.csv(file)
data <- data[-1,] # First line removed: it contains the transformation codes
```

```r
y <- ts(log(data[, "CE16OV"]), # We study US employment
         start = 1959, frequency = 12)
y <- window(y, end = c(2001, 9))
last_dates <- c(tail(time(y), 8))
names(last_dates) <- as.character(zoo::as.yearmon(last_dates))
last_est <- lapply(last_dates, function(x) window(y, end = x))

# MA of length 13 is appropriated:
select_trend_filter(y)
# Final estimates:
tc_f <- henderson(y, length = 13, musgrave = FALSE)
plot(y)
lines(tc_f, col = "red")
forecast::autoplot(ts.union(y, tc_f))

# IC-ratio computation
icr <- sapply(last_est, function(x) {
  ic_ratio(x, henderson(x, length = 13, musgrave = FALSE))
})
lp_est <- lapply(c("LC", "QL", "CQ", "DAF"), function(method) {
  res <- lapply(seq_along(icr), function(i) {
    lp_coef <- lp_filter(horizon = 6,
                         kernel = "Henderson",
                         endpoints = method,
                         ic = icr[i])
    rjd3filters::filter(last_est[[i]], lp_coef)
  })
  names(res) <- names(last_est)
```

```
  res
})
lp_if <- lapply(c("LC", "QL", "CQ", "DAF"), function(method) {
  res <- lapply(seq_along(icr), function(i) {
    lp_coef <- lp_filter(horizon = 6,
                         kernel = "Henderson",
                         endpoints = method,
                         ic = icr[i])
    implicit_forecast(last_est[[i]], lp_coef)
  })
  names(res) <- names(last_est)
  res
})
names(lp_est) <- names(lp_if) <-  c("LC", "QL", "CQ", "DAF")


# Local estimates of IC-ratios
# We replicate the direct estimates to have
# estimators of the slope and the concavity
gen_MM <- function(p=6, q=p, d=2){
  X_gen <- function(d = 1, p = 6, q = p){
    sapply(0:d, function(exp) seq(-p, q)^exp)
  }
  k = rjd3filters::get_kernel("Henderson", h = p)
  k = c(rev(k$coef[-1]), k$coef[seq(0,q)+1])
  K = diag(k)
  X = X_gen(d=d, p = p, q = q)
  e1 = e2 = e3 = matrix(0, ncol = 1, nrow = d+1)
  e1[1] = 1
```

```
  e2[2] = 1
  e3[3] = 1
  # Estimator of the constant
  M1 = K %*% X %*% solve(t(X) %*% K %*% X, e1)
  # Estimator of the slope
  M2 = K %*% X %*% solve(t(X) %*% K %*% X, e2)
  # Estimor of the concavity
  M3 = K %*% X %*% solve(t(X) %*% K %*% X, e3)
  mm <- list(const = M1, slope = M2, concav = M3)
  lapply(mm, moving_average, lags = -p)
}
all_mm <- lapply(6:0, gen_MM, p = 6, d = 2)
est_slope <- finite_filters(all_mm[[1]]$slope,
                            lapply(all_mm[-1], `[[`, "slope"))
est_concav <- finite_filters(all_mm[[1]]$concav,
                             lapply(all_mm[-1], `[[`, "concav"))

henderson_f <- lp_filter(h=6)@sfilter
lp_filter2 <- function(icr, method = "LC", h = 6, kernel = "Henderson"){
  all_coef = lapply(icr, function(ic){
    lp_filter(horizon = h,
              kernel = kernel,
              endpoints = method,
              ic = ic)
  })
  rfilters = lapply(1:h, function(i){
    q=h -i
    all_coef[[i]][,sprintf("q=%i", q)]
```

```
  })
  finite_filters(henderson_f, rfilters = rfilters)
}
loc_lc_est <-
  lapply(last_est, function(x) {
    est_loc_slope <- c(tail(est_slope * x, 6))
    sigma2 <- var_estimator(x, henderson_f)
    icr = 2/(sqrt(pi) * (est_loc_slope / sqrt(sigma2)))
    lp_coef = lp_filter2(ic = icr,
                         method = "LC", h = 6, kernel = "Henderson")
    rjd3filters::filter(x, lp_coef)
  })
loc_lc_if <-
  lapply(last_est, function(x) {
    est_loc_slope <- c(tail(est_slope * x, 6))
    sigma2 <- var_estimator(x, henderson_f)
    icr = 2/(sqrt(pi) * (est_loc_slope / sqrt(sigma2)))
    lp_coef = lp_filter2(ic = icr,
                         method = "LC", h = 6, kernel = "Henderson")
    implicit_forecast(x, lp_coef)
  })
loc_ql_est <-
  lapply(last_est, function(x) {
    est_loc_concav <- c(tail(est_concav * x, 6))
    sigma2 <- var_estimator(x, henderson_f)
    icr = 2/(sqrt(pi) * (est_loc_concav / sqrt(sigma2)))
    lp_coef = lp_filter2(ic = icr,
                         method = "QL", h = 6, kernel = "Henderson")
```

```
    rjd3filters::filter(x, lp_coef)
  })
loc_ql_if <-
  lapply(last_est, function(x) {
    est_loc_concav <- c(tail(est_concav * x, 6))
    sigma2 <- var_estimator(x, henderson_f)
    icr = 2/(sqrt(pi) * (est_loc_concav / sqrt(sigma2)))
    lp_coef = lp_filter2(ic = icr,
                         method = "QL", h = 6, kernel = "Henderson")
    implicit_forecast(x, lp_coef)
  })

## Plots
# Plots with all the estimates
plot_est <- function(data, nperiod = 6) {
  joint_data <- do.call(ts.union, data)
  joint_data <-
    window(joint_data,
           start = last_dates[1] - nperiod * deltat(joint_data))

  data_legend <-
    data.frame(x = last_dates,
               y = sapply(data, tail, 1),
               label = colnames(joint_data))

  forecast::autoplot(joint_data) + theme_bw() +
    scale_x_continuous(labels = zoo::as.yearmon) +
    geom_text(aes(x = x, y = y, label = label, colour = label),
```

```
              data = data_legend,
              check_overlap = TRUE, hjust = 0, nudge_x = 0.01,
              size = 2, inherit.aes = FALSE) +
    theme(legend.position = "none")  +
    labs(x = NULL, y = NULL)
}

plot_prevs <- function (data, nperiod = 6) {
  joint_data <- do.call(ts.union, lapply(data, function(x) {
    first_date <- time(x)[1] - deltat(x)
    # The last observed data is added for readability
    ts(c(window(y, start = first_date, end = first_date), x),
       start = first_date, frequency = frequency(x))
  }))

  data_legend <-
    data.frame(x = sapply(data, function(x) tail(time(x), 1)),
               y = sapply(data, tail, 1),
               label = colnames(joint_data))
  forecast::autoplot(joint_data, linetype = "dashed") +
    forecast::autolayer(
      window(y, start = last_dates[1] - nperiod * deltat(y)),
      colour = FALSE
    ) +
    theme_bw() +
    scale_x_continuous(labels = zoo::as.yearmon) +
    geom_text(aes(x = x, y = y, label = label, colour = label),
              data = data_legend,
```

```
                check_overlap = TRUE, hjust = 0, nudge_x = 0.01,
                size = 2, inherit.aes = FALSE) +
    theme(legend.position = "none") +
    labs(x = NULL, y = NULL)
}


all_est <- c(lp_est, list("LC local param." = loc_lc_est),
             list("QL local param." = loc_ql_est))
all_if <- c(lp_if, list("LC local param." = loc_lc_if),
            list("QL local param." = loc_ql_if))
y_lim <- NULL
all_plots_est <- lapply(
  names(all_est),
  function(x) plot_est(all_est[[x]]) +
    ggtitle(latex2exp::TeX(sprintf("Trend-cycle with %s", x))) +
    coord_cartesian(ylim = y_lim)
)
all_plots_prev <- lapply(
  names(all_if),
  function(x) plot_prevs(all_if[[x]]) +
    ggtitle(latex2exp::TeX(sprintf("Implicit forecasts with %s", x)))
)
# Combine all plots:
wrap_plots(all_plots_est, ncol = 2)
wrap_plots(all_plots_prev, ncol = 2)
```