S6) Baseline comparison

This notebook contains the code of the paper "Bayesian Calibration of Imperfect Computer Models using Physics-Informed Priors". The models are fitted in rstan and the code is available in the folder "STAN/Baseline_comparison".

Load packages

```
# uncomment to install
# install.packages("rstan")
# install.packages("ggplot2")
library(rstan)
library(ggplot2)

rstan_options(auto_write = TRUE)
options(mc.cores = 3) # allocate 3 cores (for each model we run 3 chains in parallel)

# Numerical simulator of the WK3 model
source("functions/WK2and3_sim_fn.R")
# Load flow data
d = readRDS("Data/Inflow_time.rds")
```

This notebook congtains the baseline comparison in Section 6. More specifically, three methods are compared. The Bayesian calibration method proposed by Kennedy and O'Hagan, the physics-informed prior (Raissi et al.) and the proposed approach. More details can be found in Section 6.

Reality and modelling choice

$$\mathcal{R}: \quad \frac{dP(t)}{dt} + \frac{P(t)}{R_2 C} = \frac{Q(t)}{C}\left(1 + \frac{R_1}{R_2}\right) + R_1 \frac{dQ(t)}{dt} \qquad \text{(the misspesified model we use to fit the data)} \quad \text{[WK3]} \tag{1}$$

$$\eta: \quad Q(t) = \frac{1}{R}P(t) + C\frac{dP(t)}{dt} \qquad \text{(the model we use to simulate data)} \quad \text{[WK2]} \tag{2}$$

Data simulation

```
# choose some reasonable physical parameter values
Rtrue = 1; Ctrue = 1.1; Ztrue = 0.05
flow = d$inflow*0.95
time = d$time

nP = 12 # number of pressure data
nI = 17 # number of inflow data
nc = 3  # number of cardiac cycles
nflow = length(flow)
# 1. simulate WK3 data (R=R_2, Z=R_1)
Psim = WK3_simulate(flow = flow, time = time, R = Rtrue, C = Ctrue, Z=Ztrue) # simulate WK3 data for a
```

```r
P_true = Psim
# 2. choose pressure and inflow indices
indP = round(seq(1, nflow, length.out = nP)); indI = round(seq(1, nflow, length.out = nI))
yP_real = Psim[indP]; yI_real = flow[indI] # noise free fimulated pressure and flow
# 3. Add noise
# set.seed(0)
set.seed(123)
Pnoise = rnorm(nP*nc, 0, 4) # sample pressure noise from N(0, 4^2)
Inoise = rnorm(nI*nc, 0, 10) # sample flow noise from N(0,10^2)
yP_real = rep(yP_real,nc) # create 2 replicates (2 cardiac cycles/heart beats)
yI_real = rep(yI_real,nc) # create 2 replicates (2 cardiac cycles/heart beats)
# 4. store individual data in the population matrices
yP = yP_real + Pnoise # add noise
yI = yI_real + Inoise # add noise
tP = time[indP] # corresponding time (synchronized for the two cycles)
tI = time[indI] # corresponding time (synchronized for the two cycles)
```

Model 1 (PI optimization)

```r
#--------------------------------------------------
### Model 1 (no-without delta in paper, Figure 6)
# WK2 PI prior / no delta (magenta model)
nP_pred = nI_pred = 30
ind_pred = round(seq(1,101,length.out = nP_pred))
tP_pred = tI_pred=time[ind_pred]
data_PI = list(nP=nc*nP, nI=nc*nI, tP=rep(tP,3), tI=rep(tI,3), yP=yP, yI=yI,
                 nP_pred=nP_pred, nI_pred=nI_pred, tP_pred=tP_pred, tI_pred=tI_pred)
WK2_PI_opt = stan_model("STAN/Baseline_comparison/PI_opt/WK2_PI.stan")
lfit = list()
set.seed(123)
lval = topt = rep(NA,10)
for(i in 1:10){
  tic = Sys.time()
  lfit[[i]] = optimizing(WK2_PI_opt, data=data_PI, hessian=FALSE)
  toc = Sys.time()
  topt[i] = toc-tic
  lval[i] = lfit[[i]]$value
}
sum(topt)
```

```
[1] 2.781348
```

PI Predictions

```r
opt_par = lfit[[which.max(lval)]]$par
pred_data = list(nP=nc*nP, nI=nc*nI, tP=rep(tP,3), tI=rep(tI,3), yP=yP, yI=yI,
                 rho=opt_par["rho"],alpha=opt_par["alpha"], sigmaP=opt_par["sigmaP"],
                 sigmaI=opt_par["sigmaI"],R=opt_par["R"],C=opt_par["C"],
                 nP_pred=nP_pred, nI_pred=nI_pred, tP_pred=tP_pred, tI_pred=tI_pred)

pred_fit = stan(file="STAN/Baseline_comparison/PI_opt/WK2_PI_pred.stan", data=pred_data, iter=1000, warm
                     chains=1, seed=123, refresh=1000, algorithm="Fixed_param")
```

```
SAMPLING FOR MODEL 'WK2_PI_pred' NOW (CHAIN 1).
Chain 1: Iteration:    1 / 1000 [  0%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                3.08152 seconds (Sampling)
Chain 1:                3.08152 seconds (Total)
Chain 1:
```

```r
smr=summary(pred_fit)$summary

pred=data.frame(smr[grep("y_P", rownames(smr)), c("mean", "2.5%", "97.5%")])
rmse=function(actual, pred) sqrt(mean((actual - pred)^2))
(rmse_opt=rmse(actual=P_true[ind_pred],pred=pred[,"mean"]))
```

```
[1] 6.846298
```

Model 2 (BCPI)

```r
# Function for extracting the posterior summary
post_smr.fn = function(post,quant=c(0.05, 0.95), t_pred){
  df = data.frame(
    mean = colMeans(post),
    lower = apply(post, 2, quantile, probs = quant[1]),
    upper = apply(post, 2, quantile, probs = quant[2]),
    time = t_pred
  )
}
```

```r
fit_BCPI = stan(file="STAN/Baseline_comparison/BCPI/WK2_delta_SE.stan",
                data=data_PI,
                chains=3,
                iter=1000,
                seed=0
)
fit_BCPI
```

```
Inference for Stan model: WK2_delta_SE.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

           mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff Rhat
rho        0.15    0.00 0.01    0.13    0.14    0.15    0.15    0.16  1396 1.00
alpha     20.46    0.21 6.17    9.33   16.15   19.87   24.42   34.02   834 1.00
rho_d      0.18    0.00 0.04    0.10    0.15    0.18    0.21    0.26   781 1.00
alpha_d   13.92    0.21 5.45    6.69    9.80   12.93   16.85   26.98   693 1.00
mu_wk2    94.57    0.28 9.95   75.97   87.96   94.33  100.75  114.66  1246 1.00
sigmaP     4.10    0.02 0.61    3.10    3.67    4.03    4.46    5.51  1412 1.00
sigmaI     8.68    0.03 1.10    6.87    7.92    8.55    9.30   11.06  1189 1.00
R          1.06    0.02 0.26    0.77    0.92    1.01    1.13    1.67   255 1.01
C          1.08    0.02 0.35    0.68    0.86    0.99    1.18    2.09   346 1.01
lp__    -263.84    0.14 2.49 -269.80 -265.12 -263.39 -262.04 -260.34   323 1.01
```

```
Samples were drawn using NUTS(diag_e) at Mon Nov 28 12:54:33 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

BCPI predictions

```r
post_BCPI =  rstan::extract(fit_BCPI)
N_samples = length(post_BCPI$rho)
data_pred = list(nP=nc*nP, nI=nc*nI, tP=rep(tP,3), tI=rep(tI,3), yP=yP, yI=yI
                   , tP_pred = tP_pred, tI_pred=tI_pred, nP_pred=nP_pred, nI_pred=nI_pred
                   , alpha=post_BCPI$alpha, rho=post_BCPI$rho, alpha_d=post_BCPI$alpha_d
                   , rho_d=post_BCPI$rho_d, sigmaP=post_BCPI$sigmaP, sigmaI=post_BCPI$sigmaI
                   , R=post_BCPI$R, C=post_BCPI$C, N_samples=N_samples
  )

pred_BCPI = stan(file = "STAN/Baseline_comparison/BCPI/WK2_delta_pred.stan",
              data = data_pred,
              chains = 1, iter = 1, seed=123,
              algorithm = "Fixed_param")
```

```
SAMPLING FOR MODEL 'WK2_delta_pred' NOW (CHAIN 1).
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                5.20676 seconds (Sampling)
Chain 1:                5.20676 seconds (Total)
Chain 1:
```

```r
ex_pred=rstan::extract(pred_BCPI)
```

```r
Psam = ex_pred$y_P[1,,]
Isam = ex_pred$y_I[1,,]
P_post = post_smr.fn(Psam,quant=c(0.05, 0.95), tP_pred)
(rmse_BCPI=rmse(actual=P_true[ind_pred],pred=P_post[,"mean"]))
```

```
[1] 1.612807
```

```r
library(lhs)
nd=12
set.seed(123)
RC_des=data.frame(maximinLHS(nd,2)*2.5 +0.5)
colnames(RC_des)=c("R", "C")
# plot(RC_des)
```

```r
sim_list=list()
for (i in 1:nd) {
  # remember that flow in practice is noisy
  R = RC_des$R[i]
  C = RC_des$C[i]
  Psim = WK2_simulate(flow = flow, time = time, R = R, C = C)
  indP = round(seq(1, nflow, length.out = nP)); indI = round(seq(1, nflow, length.out = nP))
  t = time[indP]
  sim_list[[i]] = data.frame(P =Psim[indP], t=t, Q=flow[indP]/diff(range(flow)), R=rep(R,nP), C=rep(C,n
}
```

```r
model_data = do.call(rbind, sim_list)
```

Prepare data for the KOH model fit

```r
eta = model_data$P
x_M = model_data[, c("t", "Q")]
t_M = model_data[, c("R", "C")]
y = yP
# remember again that flow in practice is noisy
x_F = data.frame(t=rep(tP,3), Q=rep(flow[indP],3)/diff(range(flow)))
m = nrow(x_M)
n = nrow(x_F)
p=2
q=2
data_KOH = list(m=m, n=n, p=p, q=q, eta=eta, y=y, x_M=x_M, t_M=t_M, x_F=x_F, delta=1e-8)
```

```r
fit_KOH = stan(file="STAN/Baseline_comparison/KOH/KOH.stan",
               data=data_KOH,
               chains=3,
               iter=1000,
               seed=123
)
```

```r
fit_KOH
```

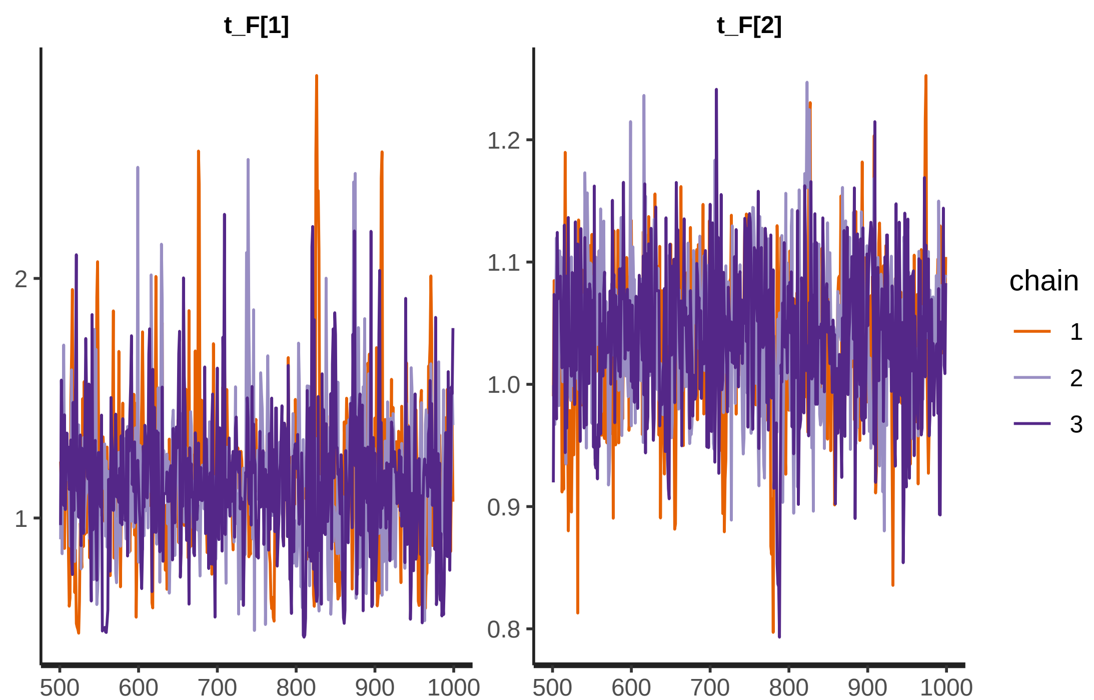
```
Inference for Stan model: KOH.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

            mean se_mean    sd    2.5%     25%     50%     75%   97.5% n_eff
alpha_M    78.04    0.09  3.53   71.11   75.69   78.11   80.50   84.48  1393
alpha_B    34.31    0.67 19.50    9.22   19.96   29.89   44.45   84.04   853
rho_M[1]    0.28    0.00  0.01    0.26    0.27    0.28    0.29    0.30   820
rho_M[2]    0.79    0.00  0.06    0.68    0.75    0.79    0.83    0.91  1050
rho_M[3]    1.69    0.00  0.09    1.51    1.63    1.69    1.75    1.86   933
rho_M[4]    2.07    0.00  0.13    1.82    1.98    2.06    2.15    2.32   829
rho_B[1]    2.54    0.04  1.37    0.72    1.52    2.23    3.25    5.84  1284
rho_B[2]    1.35    0.03  0.77    0.35    0.77    1.20    1.75    3.23   866
sigma       3.94    0.01  0.51    3.06    3.58    3.88    4.27    5.03  1538
mu_M      167.33    0.25  9.47  148.87  160.96  167.36  173.74  185.93  1439
t_F[1]      1.15    0.01  0.30    0.62    0.98    1.13    1.28    1.83   816
t_F[2]      1.04    0.00  0.06    0.91    1.00    1.04    1.08    1.15   820
lp__     -216.39    0.11  2.53 -221.85 -217.95 -216.15 -214.46 -212.32   539
         Rhat
alpha_M     1
alpha_B     1
rho_M[1]    1
rho_M[2]    1
rho_M[3]    1
rho_M[4]    1
rho_B[1]    1
rho_B[2]    1
sigma       1
mu_M        1
```

```
t_F[1]      1
t_F[2]      1
lp__        1

Samples were drawn using NUTS(diag_e) at Mon Nov 28 13:07:15 2022.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

```
stan_trace(fit_KOH, pars = "t_F")
```




```
KOH_post=rstan::extract(fit_KOH)
tf=data.frame(t=time, Q=flow/diff(range(flow)))
x_Fpred=tf[ind_pred,]
data_pred_KOH = list(
  m=m, n=length(y), p=p, q=q, eta=eta, y=y, x_M=x_M, t_M=t_M, x_F=x_F, delta=1e-8,
  n_pred=nP_pred, x_Fpred=x_Fpred, N_samples=nrow(KOH_post$rho_M),
  alpha_M=KOH_post$alpha_M, alpha_B=KOH_post$alpha_B,
  rho_M=KOH_post$rho_M, rho_B=KOH_post$rho_B,
  sigma=KOH_post$sigma, t_F=KOH_post$t_F
  )
```

KOH predictions

```
pred_KOH = stan(file = "STAN/Baseline_comparison/KOH/KOH_pred.stan",
                 data = data_pred_KOH,
                 chains = 1, iter = 1, seed=123,
                 algorithm = "Fixed_param")
```

```
SAMPLING FOR MODEL 'KOH_pred' NOW (CHAIN 1).
Chain 1: Iteration: 1 / 1 [100%]  (Sampling)
Chain 1:
```

```
Chain 1:  Elapsed Time: 0 seconds (Warm-up)
Chain 1:                24.4279 seconds (Sampling)
Chain 1:                24.4279 seconds (Total)
Chain 1:
```

```r
ex_pred_KOH=rstan::extract(pred_KOH)
Psam = ex_pred_KOH$y_P[1,,]
P_post_KOH = post_smr.fn(Psam,quant=c(0.05, 0.95), tP_pred)
(rmse_KOH=rmse(actual=P_true[ind_pred],pred=P_post_KOH[,"mean"]))
```

```
[1] 1.915173
```

```r
# Psam = ex_pred_KOH$y_P[1,,]
# P_post_KOH = post_smr.fn(Psam,quant=c(0.05, 0.95), tP_pred)
# (rmse_KOH=rmse(actual=P_true[ind_pred],pred=P_post_KOH[,"mean"]))
```

```r
pr_BCPI=rstan::extract(pred_BCPI)$y_P
BCPI_rmse_vec=apply(pr_BCPI[1,,],1,rmse, actual=P_true[ind_pred])
mean(BCPI_rmse_vec); sd(BCPI_rmse_vec)
```

```
[1] 4.909706
```

```
[1] 1.011549
```

Table in Section 6

```r
pr = c("R", "C", "sigmaP", "sigmaI")
opt_par[pr]
```

```
        R         C    sigmaP    sigmaI
0.9311425 0.9617584 7.8931358 8.3501029
```

```r
opt_mu_CI = data.frame(mu = opt_par[pr], lower=rep(NA,4), upper=rep(NA,4))
opt_mu_CI$mu=round(opt_mu_CI$mu,2)
rownames(opt_mu_CI) = pr
smr_BCPI = summary(fit_BCPI, pars=pr, probs = c(0.05, 0.95))$summary
smr_BCPI[,c("mean", "5%", "95%")]
```

```
           mean        5%       95%
R      1.058963 0.8014303  1.464309
C      1.077127 0.7256030  1.741661
sigmaP 4.101236 3.2144190  5.190075
sigmaI 8.675639 7.0895515 10.685798
```

```r
BCPI_mu_CI = data.frame(mu = c(smr_BCPI[,"mean"]),
                        lower = c(smr_BCPI[,"5%"]),
                        upper =  c(smr_BCPI[,"95%"]))

smr_KOH = summary(fit_KOH, pars=c("t_F", "sigma"), probs = c(0.05, 0.95))$summary
rownames(smr_KOH) = pr[1:3]
smr_KOH[,c("mean", "5%", "95%")]
```

```
           mean        5%      95%
R      1.152612 0.6854677 1.646368
C      1.038509 0.9409588 1.133727
sigmaP 3.941126 3.1704890 4.877751
```

```r
KOH_mu_CI = data.frame(mu = c(smr_KOH[,"mean"],NA),
                       lower = c(smr_KOH[,"5%"],NA),
                       upper =  c(smr_KOH[,"95%"],NA))
rownames(KOH_mu_CI) = pr
res_opt = with(opt_mu_CI, paste0(mu, "(", lower, ",", upper, ")"))
res_BCPI = with(round(BCPI_mu_CI,2), paste0(mu, "(", lower, ",", upper, ")"))
res_KOH = with(round(KOH_mu_CI,2), paste0(mu, "(", lower, ",", upper, ")"))
res_all = rbind(res_opt, res_KOH, res_BCPI)
rownames(res_all) = c("PI opt", "KOH", "BCPI")
colnames(res_all) = pr
res_all=data.frame(res_all)
res_all$RMSE = as.character(round(c(rmse_opt, rmse_KOH, rmse_BCPI),2))
rt_BCPI = get_elapsed_time(fit_BCPI)
rt_KOH = get_elapsed_time(fit_KOH)
res_all$run_time = round(c(sum(topt), max(apply(rt_KOH,1,sum)), max(apply(rt_BCPI,1,sum))))
res_all
```

```
                      R               C          sigmaP           sigmaI RMSE
PI opt     0.93(NA,NA)     0.96(NA,NA)     7.89(NA,NA)      8.35(NA,NA) 6.85
KOH    1.15(0.69,1.65) 1.04(0.94,1.13) 3.94(3.17,4.88)         NA(NA,NA) 1.92
BCPI    1.06(0.8,1.46) 1.08(0.73,1.74)  4.1(3.21,5.19) 8.68(7.09,10.69) 1.61
       run_time
PI opt        3
KOH         703
BCPI         27
```

```r
sessionInfo()
```

```
R version 4.0.3 (2020-10-10)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] lhs_1.1.3            rstan_2.21.3         ggplot2_3.3.5
[4] StanHeaders_2.21.0-7

loaded via a namespace (and not attached):
 [1] tidyselect_1.1.1   xfun_0.29          purrr_0.3.4        colorspace_2.0-2
 [5] vctrs_0.3.8        generics_0.1.2     htmltools_0.5.2    stats4_4.0.3
 [9] loo_2.4.1          yaml_2.2.2         utf8_1.2.2         rlang_1.0.0
[13] pkgbuild_1.3.1     pillar_1.7.0       glue_1.6.1         withr_2.4.3
[17] DBI_1.1.2          matrixStats_0.61.0 lifecycle_1.0.1    stringr_1.4.0
[21] munsell_0.5.0      gtable_0.3.0       codetools_0.2-18   evaluate_0.14
[25] labeling_0.4.2     inline_0.3.19      knitr_1.37         callr_3.7.0
[29] fastmap_1.1.0      ps_1.6.0           parallel_4.0.3     fansi_1.0.2
```

```
[33] Rcpp_1.0.8         scales_1.1.1       RcppParallel_5.1.5 farver_2.1.0
[37] gridExtra_2.3      digest_0.6.29      stringi_1.7.6      processx_3.5.2
[41] dplyr_1.0.7        grid_4.0.3         cli_3.1.1          tools_4.0.3
[45] magrittr_2.0.2     tibble_3.1.6       crayon_1.4.2       pkgconfig_2.0.3
[49] ellipsis_0.3.2     prettyunits_1.1.1  assertthat_0.2.1   rmarkdown_2.11
[53] rstudioapi_0.13    R6_2.5.1           compiler_4.0.3
```