```
## INCREASE THE CELL WIDTH FOR BIGGER FIGURES
html"""
<style>
    main {
        margin: 0 auto;
        max-width: 2000px;
        padding-left: max(160px, 10%);
        padding-right: max(160px, 10%);
    }
  svg {
    width: 100%;
  }
</style>
"""
```

Load required packages (dependencies specified in the `Project.toml` file)

```
begin
    using Pkg
    Pkg.activate("../Project.toml")

    using PolyChaos
    using LinearAlgebra
    using DelimitedFiles
    using StatsPlots

    using Plots.PlotMeasures
    using Printf

    using CSV
    using DataFrames

    using NetCDF
    using Dates
    using LaTeXStrings
    using PlutoUI

    using Random
    using JLD
end
```

```
begin
    using Revise
    include("./gsa_utils.jl")
end
```

Include script with different helper functions. If script is changed, this cell will have to be rerun for changes to reflect.

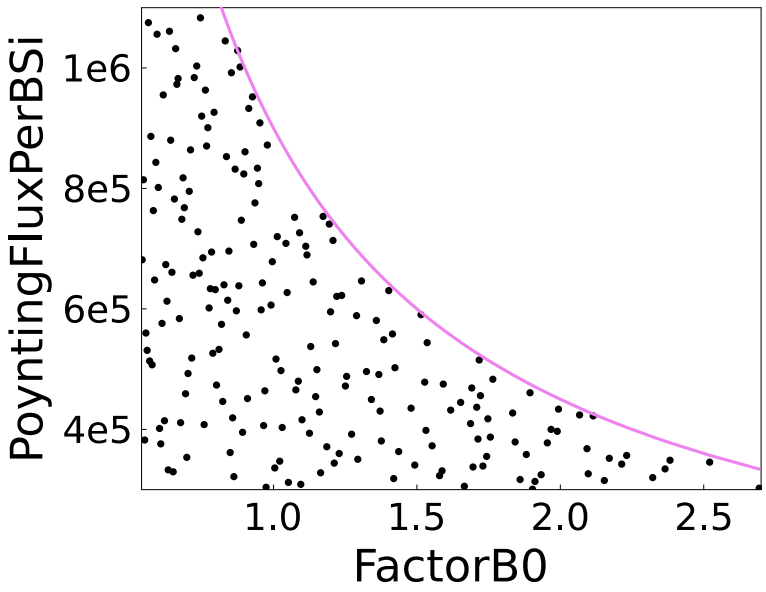
Note: Figures are plotted here in the order that they appear in the manuscript.

Preprocessing

	FactorBo	nChromoSi_AWSoM	PoyntingFluxPerBS
1	0.71496	3.4784e18	518400.0
2	1.07784	8.672e17	465600.0
3	1.19448	2.9888e18	740800.0
4	0.63288	1.7504e18	332800.0
5	1.00872	2.912e17	516800.0
6	0.76248	2.1056e18	963200.0
7	0.5724	3.7952e18	886400.0
8	1.90296	3.4688e18	300800.0
9	0.54216	2.8064e18	681600.0
10	0.6804	4.832e18	748800.0
⋮ more			
200	1.89432	2.5376e18	460800.0

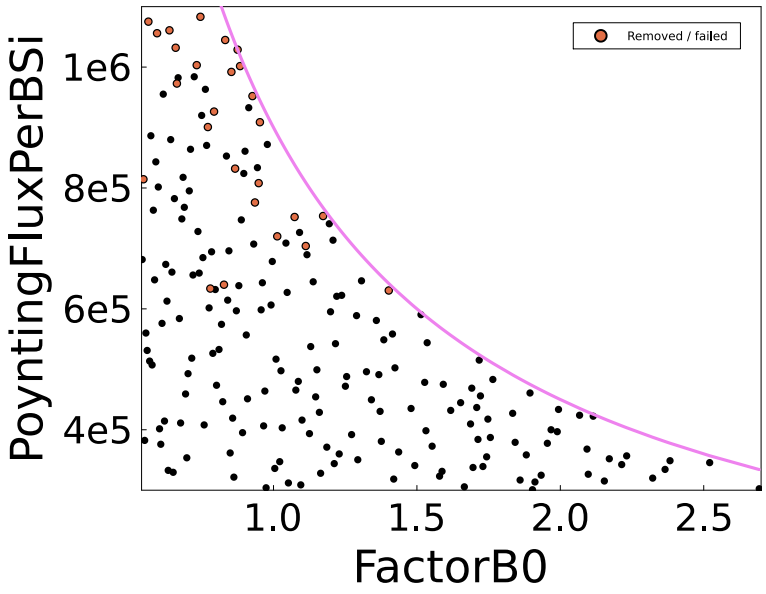
```
begin
    X_design =
CSV.read("./data/design/X_background_CR2152_update
d.csv", DataFrame)
    pfss = replace(X_design.PFSS, 1=>"HARMONICS",
2=>"FDIPS")
    surfaceWaveRefl =
replace(X_design.UseSurfaceWaveRefl, 1=>"true",
2=>"false")
    select!(X_design, Not([:PFSS,
:UseSurfaceWaveRefl]))
    insertcols!(X_design, :PFSS=>pfss,
:UseSurfaceWaveRefl=>surfaceWaveRefl)
end
```

Make selected scatterplots - figure 2 (others can be made in the same fashion by selecting appropriate column names in the plotting arguments)

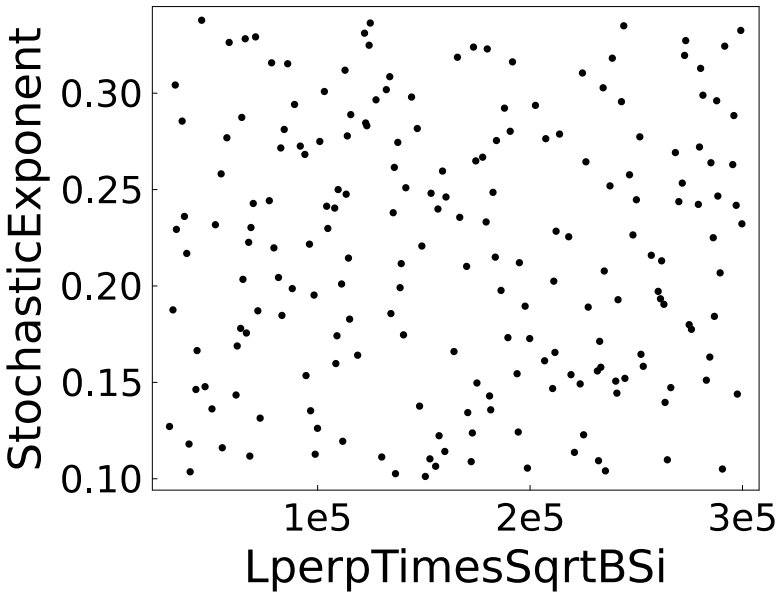



```
begin
    FactorB0PF = scatter(X_design[:, "FactorB0"],
X_design[:, "PoyntingFluxPerBSi"],
                                #
zcolor=shiftWLRMSE.PTRMSE,
                                marker=(:black,
:circle, 4),

xlabel="FactorB0",

ylabel="PoyntingFluxPerBSi",

markerstrokewidth=0,
                                label="",
                                dpi = 300,
                                grid=false
                                )
    plot!(sort(X_design.FactorB0), 9e5 ./
(sort(X_design.FactorB0)), line=(:violet, 3.1),
label="")
    plot!(xlims=(0.54, 2.7))
    plot!(ylims=(0.3e6, 1.1e6))
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(yticks=([4e5, 6e5, 8e5, 1e6], ["4e5",
"6e5", "8e5", "1e6"]))
    plot!(size=(800, 600))
end
```

Figure for Supporting Information Document

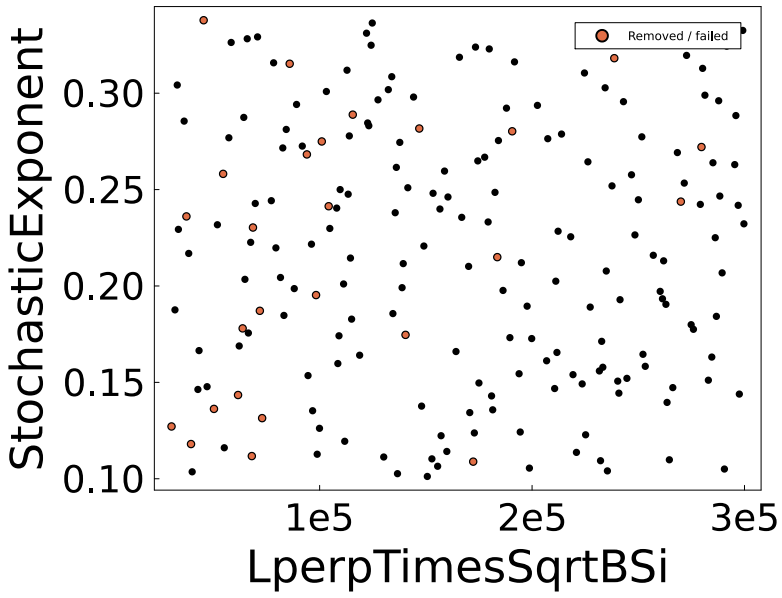



```
begin
    FactorB0PFAll = scatter(X_design[:,
"FactorB0"], X_design[:, "PoyntingFluxPerBSi"],
                                #
zcolor=shiftWLRMSE.PTRMSE,
                                marker=(:black,
:circle, 4),

xlabel="FactorB0",

ylabel="PoyntingFluxPerBSi",

markerstrokewidth=0,
                                label="",
                                dpi = 300,
                                grid=false
                                )
    scatter!(X_design[excludeRunIdx, "FactorB0"],
X_design[excludeRunIdx, "PoyntingFluxPerBSi"],
label="Removed / failed")
    plot!(sort(X_design.FactorB0), 9e5 ./
(sort(X_design.FactorB0)), line=(:violet, 3.1),
label="")
    plot!(xlims=(0.54, 2.7))
    plot!(ylims=(0.3e6, 1.1e6))
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(yticks=([4e5, 6e5, 8e5, 1e6], ["4e5",
"6e5", "8e5", "1e6"]))
    plot!(size=(800, 600))
end
```

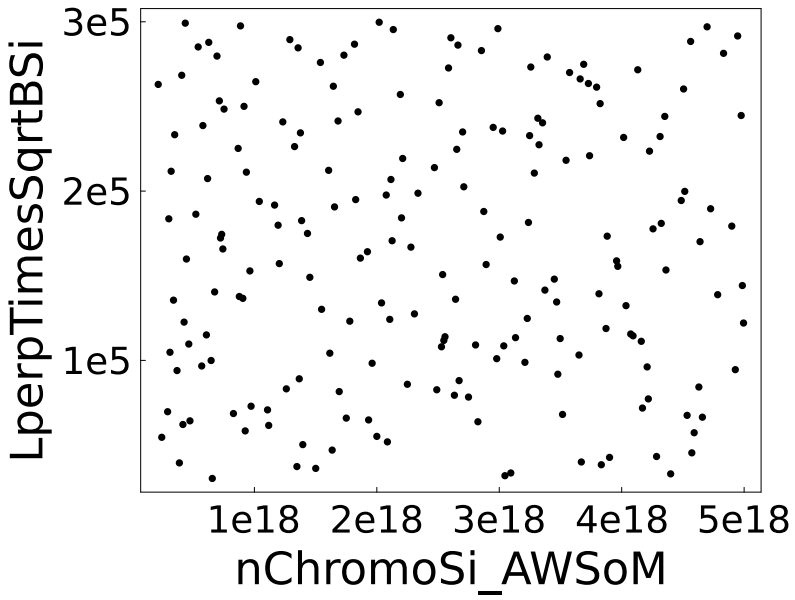



```
begin
    LperpStoch = scatter(X_design[:,
"LperpTimesSqrtBSi"], X_design[:,
"StochasticExponent"],
                                    #
zcolor=shiftWLRMSE.PTRMSE,
                                    marker=
(:black, :circle, 4),

xlabel="LperpTimesSqrtBSi",

ylabel="StochasticExponent",

markerstrokewidth=0,
                                    label="",
                                    dpi = 300,
                                    grid=false
                                    )
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(xticks=([1e5, 2e5, 3e5], ["1e5", "2e5",
"3e5"]))
    plot!(size=(800, 600))
end
```

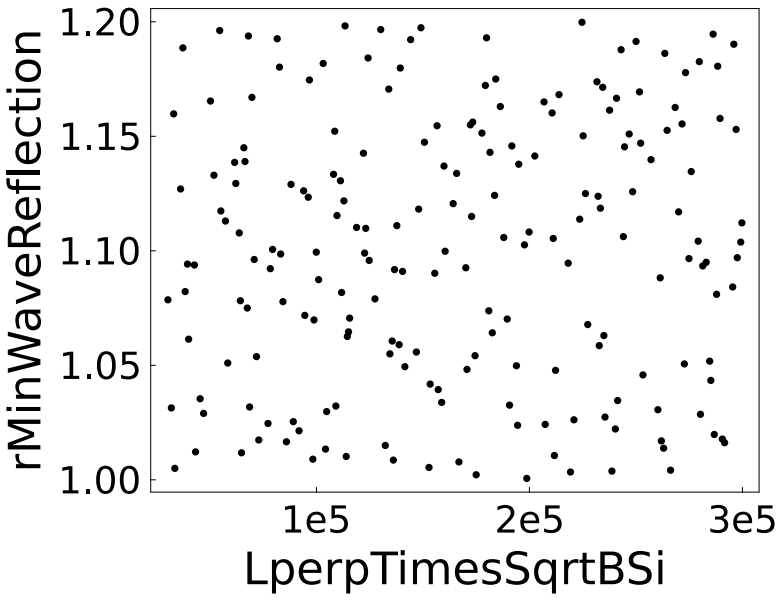
Figure for supporting information document




```
begin
    LperpStoch2 = scatter(X_design[:,
"LperpTimesSqrtBSi"], X_design[:,
"StochasticExponent"],
                                    #
zcolor=shiftWLRMSE.PTRMSE,
                                    marker=
(:black, :circle, 4),

xlabel="LperpTimesSqrtBSi",

ylabel="StochasticExponent",

markerstrokewidth=0,
                                    label="",
                                    dpi = 300,
                                    grid=false
                                    )
    scatter!(X_design[excludeRunIdx,
"LperpTimesSqrtBSi"], X_design[excludeRunIdx,
"StochasticExponent"], label="Removed / failed")
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(xticks=([1e5, 2e5, 3e5], ["1e5", "2e5",
"3e5"]))
    plot!(size=(800, 600))
end
```




```
begin
    chromoLperp = scatter(X_design[:,
"nChromoSi_AWSoM"], X_design[:,
"LperpTimesSqrtBSi"],
                                    #
zcolor=shiftWLRMSE.PTRMSE,
                                    marker=
(:black, :circle, 4),

xlabel="nChromoSi_AWSoM",

ylabel="LperpTimesSqrtBSi",

markerstrokewidth=0,
                                    label="",
                                    dpi = 300,
                                    grid=false
                                    )
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(xticks=([1e18, 2e18, 3e18, 4e18, 5e18],
["1e18", "2e18", "3e18", "4e18", "5e18"]))
    plot!(yticks=([1e5, 2e5, 3e5], ["1e5", "2e5",
"3e5"]))
    plot!(size=(800, 600))
end
```




```
begin
    LperpRMin = scatter(X_design[:,
"LperpTimesSqrtBSi"], X_design[:,
"rMinWaveReflection"],
                                    #
zcolor=shiftWLRMSE.PTRMSE,
                                    marker=
(:black, :circle, 4),

xlabel="LperpTimesSqrtBSi",

ylabel="rMinWaveReflection",

markerstrokewidth=0,
                                    label="",
                                    dpi = 300,
                                    grid=false
                                    )
    plot!(guidefontsize=30)
    plot!(tickfontsize=25)
    plot!(framestyle=:box)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(right_margin=8mm)
    plot!(xticks=([1e5, 2e5, 3e5], ["1e5", "2e5",
"3e5"]))
    plot!(size=(800, 600))
end
```

Scale input parameters to [0-1] range

	FactorBo	nChromoSi_AWSoM	PoyntingFluxPerBS
1	0.081	0.683	0.273
2	0.249	0.139	0.207
3	0.303	0.581	0.551
4	0.043	0.323	0.041
5	0.217	0.019	0.271
6	0.103	0.397	0.829
7	0.015	0.749	0.733
8	0.631	0.681	0.001
9	0.001	0.543	0.477
10	0.065	0.965	0.561
⋮ more			
200	0.627	0.487	0.201

```
begin
        lbBg = [0.54, 2e17, 0.3e6, 0.3e5, 0.1, 1]
        ubBg = [2.7, 5e18, 1.1e6, 3e5, 0.34, 1.2]

        paramsBgScaled = (X_design[!, 1:6] .-
lbBg') ./ (ubBg' - lbBg')
end
```

```
inputNames =
▸ ["FactorB0", "nChromoSi_AWSoM", "PoyntingFluxPerBSi",
```

```
inputNames = names(paramsBgScaled)
```

Filter out runs that are not physically meaningful based on the exclusion criteria described in section 4.1.

```
md"""
Filter out runs that are not physically
meaningful based on the exclusion criteria
described in section 4.1.
"""
```

```
excludeRunIdx =
▸ [11, 14, 16, 20, 34, 48, 50, 54, 68, 72, 74, 88, 94, 1(
```

```
excludeRunIdx = [11, 14, 16, 20, 34, 48, 50, 54,
68, 72, 74, 88, 94, 101, 117, 124, 126, 137, 142,
146, 149, 164, 169, 175, 194, 198]
```

```
#
CSV.write("/Users/ajivani/Desktop/Research/SWQUPaper/XDesignRemovedCR2152.csv",
X_design[excludeRunIdx, :])
```

```
Enter cell code...
```

```
finalRuns =
▸ [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 17, 18, 19,
```

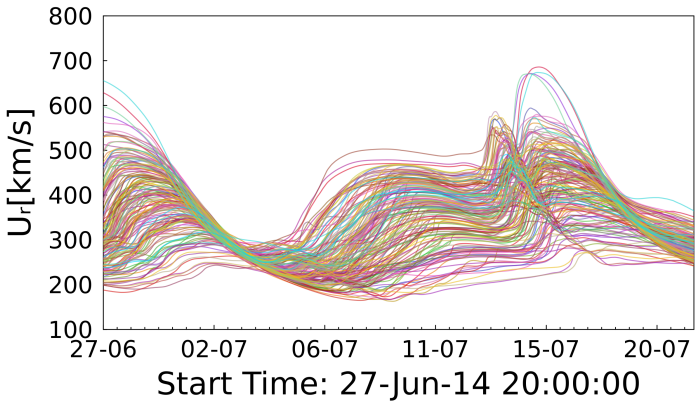
```
finalRuns = setdiff(1:200, excludeRunIdx)
```

Extract QoIs and observations

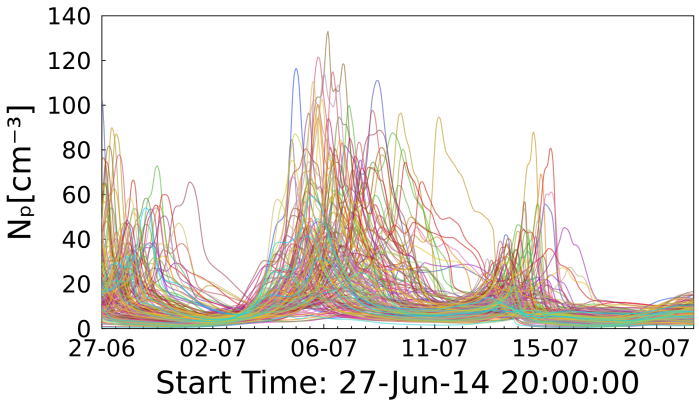
```
▸ [2014-06-27T20:00:00, 2014-06-27T21:00:00, 2014-06-27
```

```
begin
    fn = "./data/background_sims/bg_CR2152.nc"
    UrSim = ncread(fn, "UrSim")
    NpSim = ncread(fn, "NpSim")
    TSim  = ncread(fn, "TSim")
    BSim  = ncread(fn, "BSim")

    UrObs = ncread(fn, "UrObs")
    NpObs = ncread(fn, "NpObs")
    TObs  = ncread(fn, "TObs")
    BObs  = ncread(fn, "BObs")

    startTime = ncgetatt(fn, "time", "startTime")
    timeElapsed = Dates.Hour.(ncread(fn, "time"))
    times = timeElapsed .+
Dates.DateTime(startTime, "yyyy_mm_ddTHH:MM:SS")
end
```

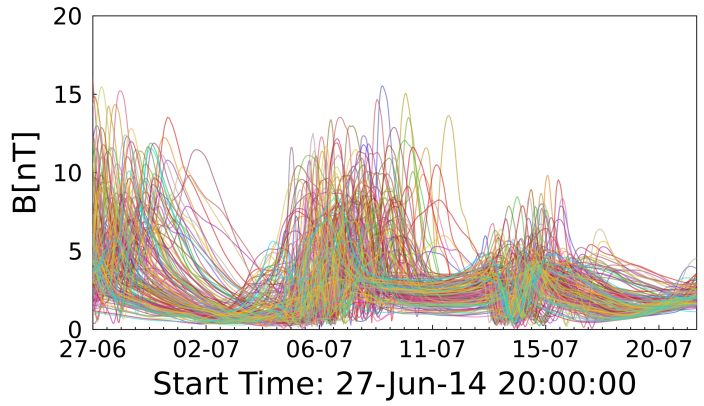
Plot the QoIs (figure 3). If we want to plot observations, supply `plotObs=true` as a keyword.



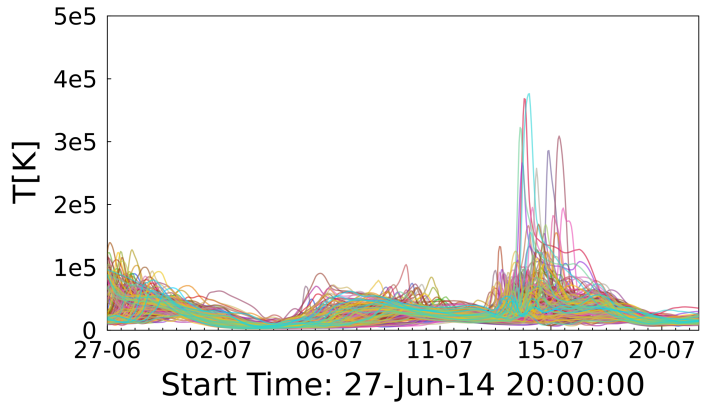

```
begin
    # make plots of the runs that are successful
and satisfy the constraints
    pUrSimObsF = plotSimObs(UrSim, UrObs, times,
collect(1:200); simIdx=finalRuns, plotArgsUr...,
ylims=(100, 800), dateFormat="dd-mm")
    plot!(size=(920, 470))
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)
end
```




```
begin
    pNpSimObsF = plotSimObs(NpSim, NpObs, times,
collect(1:200); simIdx=finalRuns, plotArgsNp...,
ylims=(0, 140), dateFormat="dd-mm")
    plot!(size=(920, 470))
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)

end
```




```
begin
    pBSimObsF  = plotSimObs(BSim, BObs, times,
collect(1:200); simIdx=finalRuns, plotArgsB...,
dateFormat="dd-mm")
    plot!(size=(920, 470))
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)
end
```




```
begin
    pTSimObsF = plotSimObs(TSim, TObs, times,
collect(1:200); simIdx=finalRuns, plotArgsT...,
ylims=(0, 5e5), dateFormat="dd-mm")
    plot!(yticks=(0:1e5:5e5, [0;
[@sprintf("%de5", i) for i in 1:5]][:]))
    plot!(size=(920, 470))
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)
end
```

UQ and GSA

Build surrogate

```
174×577 Matrix{Float64}:
  17.1584    16.2863    15.1031    …   4.82005   4.814
  15.9413    16.616     17.5474        5.91221   5.878
  20.4089    19.5313    18.6234       13.6465   13.618
   5.31195    4.80451    4.41201       2.24787   2.229
  10.4009    10.5956    10.7959        6.5112    6.598
 103.544     98.9527    90.7347    …  14.6699   14.952
   5.37366    5.35807    5.34244      10.7472   10.813
   ⋮                                  ⋱
  14.6357    15.3083    16.0386        7.46509   7.481
   8.93437    8.48892    8.04946       5.12626   5.157
   0.670335   0.663485   0.657108  …   1.39292   1.422
  18.3569    18.2715    18.1705        7.55713   7.552
  16.7034    16.7236    16.7499        5.75784   5.771
  18.4304    19.2834    20.2115       10.5687   10.580
```

```
begin
    XTrainFinal =
Matrix(paramsBgScaled[finalRuns, :])
    YTrainUr = Array{Float64, 2}(UrSim[:,
finalRuns]')
    YTrainNp = Array{Float64, 2}(NpSim[:,
finalRuns]')
end
```

```
ATrainFinal =
174×28 Matrix{Float64}:
 1.0  -1.45146    0.633931   -0.786351  …  -0.47457
 1.0  -0.86949   -1.25054    -1.01498        1.08259
 1.0  -0.682428   0.280592    0.176669      -0.0282863
 1.0  -1.58309   -0.613146   -1.59002        1.61088
 1.0  -0.980341  -1.66623    -0.793279      -0.996951
 1.0  -1.37525   -0.356802    1.13969   …    1.44408
 1.0  -1.68009    0.862561    0.807136      -0.95273
 ⋮                                    ⋱    ⋮
 1.0   0.980341   0.453797   -1.4861         2.03784
 1.0  -1.59695    0.523079   -0.114315      -0.894772
 1.0  -1.55538   -0.564649   -1.60388   …  -0.820176
 1.0   0.204382   0.467654   -1.22283       -0.565157
 1.0   0.197454   0.294449   -1.49303        1.03934
 1.0   0.439941  -0.0450333  -1.03577        2.16932
```

```
# build coefficient matrix!
ATrainFinal = buildCoefficientMatrix(XTrainFinal;
pceDegree=2)
```

```
28×577 Matrix{Float64}:
 17.3647    17.4585     17.572      …   7.32804      7
  2.05547    2.16281     2.36894          0.876979     0
 -0.971757  -1.02722    -0.915538        -0.175053    -0
 12.3385    12.508      12.7408           3.08811      3
  2.75197    2.1775      1.6117           0.351816     0
  0.919863   0.612864    0.293812   …    -0.00874819  -0
  3.71062    3.66168     3.52078          0.556068     0
  ⋮                                   ⋱                ⋮
 -2.69933   -2.83859    -2.97176         -0.155432    -0
  1.68617    1.63318     1.59091          0.100926     0
 -1.06656   -1.08772    -1.12144         -0.290205    -0
  1.37797    1.33982     1.25772    …     0.103589     0
 -0.123657  -0.0297994   0.0292054        0.00373966  -0
  2.03138    2.10591     2.13198          0.103107     0
```

```
begin
    # Perform ridge regression
    betaUrFinal = solveRegPCE(ATrainFinal,
YTrainUr; λ=lambdaUrFinal)
    betaNpFinal = solveRegPCE(ATrainFinal,
YTrainNp; λ=lambdaNpFinal)
end
```

```
5
```

```
begin
    # set regularization coefficients
    lambdaUrFinal = 0.4
    lambdaNpFinal = 5
end
```

```
Enter cell code...
```

Load matrix of test points

```
XTestFinal =
400×6 Matrix{Float64}:
 0.0965  0.0455  0.1125  0.1585  0.3965  0.2025
 0.1325  0.0895  0.6275  0.5335  0.5325  0.5565
 0.0095  0.5905  0.4725  0.5645  0.7315  0.1965
 0.2955  0.7375  0.1945  0.4465  0.7665  0.8515
 0.6785  0.0105  0.0025  0.6555  0.2515  0.5445
 0.0665  0.9795  0.2575  0.3155  0.4915  0.8585
 0.1535  0.0155  0.7265  0.5925  0.8775  0.0795
 ⋮                                       ⋮
 0.2335  0.5675  0.0595  0.2395  0.8755  0.1855
 0.5365  0.3805  0.0505  0.2995  0.2535  0.6355
 0.1815  0.6035  0.2885  0.9575  0.0735  0.6495
 0.0335  0.9125  0.4005  0.0515  0.7125  0.2135
 0.1625  0.1905  0.2655  0.0305  0.4015  0.7505
 0.0945  0.4005  0.4575  0.2685  0.4335  0.1365
```

```
XTestFinal =
load("./data/design/CR2152TestFinal.jld",
"XTestFinal")[:, 1:6]
```

```
ATest = buildCoefficientMatrix(XTestFinal[:,
1:6]; pceDegree=2);
```

```
577×400 Matrix{Float64}:
 329.497  294.512  470.253  302.48   247.377  …  497.1
 333.548  297.848  474.869  304.366  247.34      498.4
 337.018  301.246  479.031  306.298  247.244     499.6
 339.9    304.859  482.97   308.294  247.21      500.6
 342.518  308.654  486.648  310.239  247.196     501.5
 344.89   312.64   490.06   312.14   247.045  …  502.1
 347.188  316.717  493.052  314.002  246.811     502.5
   ⋮                                   ⋱
 294.882  257.654  296.437  291.417  301.057     322.8
 294.713  256.982  296.45   290.925  300.521     322.3
 294.403  256.32   296.375  290.305  299.81      321.6
 294.204  255.704  296.419  289.805  299.23      321.1
 294.006  255.11   296.466  289.331  298.647  …  320.5
 293.755  254.562  296.509  288.829  297.992     319.8
```

```
begin
    ATestFinalCoeffs = ATest[:, 2:end]
    ATestFinalFiltered = ATestFinalCoeffs .-
mean(ATestFinalCoeffs, dims=1)
    yPredNpFinal = betaNpFinal[1, :] .+
Matrix((ATestFinalFiltered * betaNpFinal[2:end,
:])');
    yPredUrFinal = betaUrFinal[1, :] .+
Matrix((ATestFinalFiltered * betaUrFinal[2:end,
:])');
end
```

Show cross validation procedure

```
lambdaVals =
▸ [0.01, 0.0126638, 0.0160372, 0.0203092, 0.0257191, 0.
```

```
lambdaVals = 10 .^(range(-2, 2, length=40))
```

```
kFoldErrorsUr =
577×40 Matrix{Float64}:
 0.293932   0.2936     0.293181   0.292655   …  0.1623
 0.288032   0.287712   0.287309   0.286803      0.1616
 0.282782   0.282473   0.282084   0.281595      0.1608
 0.278584   0.278283   0.277904   0.277428      0.1601
 0.27459    0.274297   0.273928   0.273463      0.1593
 0.270405   0.270119   0.269759   0.269306   …  0.1583
 0.265298   0.265021   0.264673   0.264235      0.1571
 ⋮                                          ⋱
 0.026653   0.0266488  0.026644   0.0266385     0.0468
 0.0273355  0.0273394  0.0273446  0.0273518     0.0473
 0.0284433  0.0284537  0.0284672  0.0284846     0.0479
 0.0298777  0.029893   0.0299126  0.0299375     0.0484
 0.0316076  0.0316258  0.031649   0.0316784  …  0.0489
 0.033753   0.0337723  0.0337969  0.033828      0.0495
```

```
kFoldErrorsUr = kFoldCV(XTrainFinal, YTrainUr,
lambdaVals; pceDegree=2, nFolds=10)
```

kFoldErrorsNp =

```
577×40 Matrix{Float64}:
 1.59921  1.59653  1.59315  1.5889   …  0.544352  0.54
 1.61889  1.61616  1.61272  1.60839      0.546465  0.54
 1.65496  1.65215  1.6486   1.64413      0.550214  0.54
 1.69834  1.69542  1.69174  1.6871       0.553475  0.55
 1.73933  1.73631  1.73251  1.72772      0.556768  0.55
 1.78012  1.777    1.77308  1.76814  …   0.560265  0.55
 1.81968  1.81648  1.81244  1.80735      0.56372   0.56
 ⋮                                  ⋱
 1.06198  1.06017  1.0579   1.05505      0.26313   0.25
 1.07754  1.07571  1.0734   1.0705       0.265044  0.26
 1.09145  1.08959  1.08725  1.0843       0.266704  0.26
 1.10479  1.10291  1.10053  1.09754      0.268449  0.26
 1.11746  1.11555  1.11314  1.11012  …   0.270228  0.26
 1.12858  1.12665  1.12421  1.12115      0.271776  0.26
```

```
kFoldErrorsNp = kFoldCV(XTrainFinal, YTrainNp, lambdaVals; pceDegree=2, nFolds=10)
```

lambdaOptIdxUr =

```
[36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 35, 35, 35, 35
```

```
lambdaOptIdxUr = [argmin(kFoldErrorsUr[i, :]) for i in 1:length(times)]
```

```
0.08727220209634957
```

```
kFoldErrorsUr[286, 30]
```

lambdaOptIdxNp =

```
[39, 38, 38, 39, 39, 39, 39, 39, 40, 40, 40, 40, 40, 40
```

```
lambdaOptIdxNp = [argmin(kFoldErrorsNp[i, :]) for i in 1:length(times)]
```

lambdaOptUr =

```
[38.8816, 38.8816, 38.8816, 38.8816, 38.8816, 38.8816
```

```
lambdaOptUr = lambdaVals[lambdaOptIdxUr]
```

lambdaOptNp =

```
[78.9652, 62.3551, 62.3551, 78.9652, 78.9652, 78.9652
```

```
lambdaOptNp = lambdaVals[lambdaOptIdxNp]
```

```
577×40 Matrix{Float64}:
 0.427381  0.401262   0.375716   0.351479   …  0.15327
 0.423899  0.398091   0.372843   0.348882      0.15300
 0.423992  0.398164   0.372882   0.348879      0.15292
 0.42428   0.398461   0.373177   0.349164      0.15290
 0.423743  0.398036   0.372855   0.348936      0.15282
 0.422527  0.397001   0.371995   0.34824    …  0.15261
 0.421188  0.395794   0.370913   0.347275      0.15204
 ⋮                                          ⋱
 0.10405   0.0923711  0.0808023  0.0697474     0.03752
 0.105868  0.0939324  0.08211    0.0708152     0.03793
 0.108563  0.0962762  0.0841034  0.0724714     0.03835
 0.110266  0.0977332  0.0853193  0.0734605     0.03878
 0.111284  0.0985913  0.0860239  0.0740262  …  0.03920
 0.112597  0.0997093  0.0869527  0.0747808     0.03963
```

```
begin
kFoldErrorsUrDeg1 = kFoldCV(XTrainFinal,
YTrainUr, lambdaVals; pceDegree=1, nFolds=10)
kFoldErrorsUrDeg3 = kFoldCV(XTrainFinal,
YTrainUr, lambdaVals; pceDegree=3, nFolds=10)
end
```

```
kFoldErrorsUrDeg4 =
577×40 Matrix{Float64}:
 0.52519    0.52314    0.520603   0.517479   …  0.1758
 0.534046   0.53193    0.529311   0.526085      0.1759
 0.54222    0.540034   0.537327   0.533995      0.1760
 0.549273   0.547019   0.544228   0.540792      0.1762
 0.555118   0.552798   0.549927   0.546392      0.1764
 0.559368   0.556989   0.554046   0.550423   …  0.1764
 0.561264   0.558835   0.555829   0.55213       0.1760
 ⋮                                           ⋱
 0.0716209  0.0714096  0.0711475  0.0708241     0.0462
 0.0718479  0.0716391  0.0713801  0.0710603     0.0465
 0.072088   0.0718826  0.0716277  0.071313      0.0469
 0.0726337  0.0724301  0.0721774  0.0718653     0.0472
 0.0734636  0.0732613  0.07301    0.0726996  …  0.0476
 0.0743609  0.0741601  0.0739107  0.0736024     0.0480
```

```
kFoldErrorsUrDeg4 = kFoldCV(XTrainFinal,
YTrainUr, lambdaVals; pceDegree=4, nFolds=10)
```

```
577×40 Matrix{Float64}:
 1.36135   1.35701   1.35162   1.34497   …  0.519304
 1.56574   1.55995   1.55277   1.54393      0.520236
 1.7517    1.7447    1.73605   1.72541      0.526709
 1.90754   1.89961   1.88982   1.87779      0.537489
 2.01339   2.00472   1.99402   1.98088      0.548976
 2.05943   2.05024   2.0389    2.02499   …  0.558502
 2.05396   2.04444   2.03271   2.01834      0.56624
 ⋮                                    ⋱
 0.452447  0.450823  0.448815  0.446345     0.239988
 0.449539  0.447957  0.446001  0.443594     0.241521
 0.444744  0.443212  0.441317  0.438987     0.24265
 0.440909  0.439421  0.437579  0.435314     0.243952
 0.437781  0.436331  0.434538  0.432331  …  0.245319
 0.433461  0.432054  0.430314  0.428173     0.246273
```

```
begin
    kFoldErrorsNpDeg1 = kFoldCV(XTrainFinal,
YTrainNp, lambdaVals; pceDegree=1, nFolds=10)
    kFoldErrorsNpDeg3 = kFoldCV(XTrainFinal,
YTrainNp, lambdaVals; pceDegree=3, nFolds=10)
    kFoldErrorsNpDeg4 = kFoldCV(XTrainFinal,
YTrainNp, lambdaVals; pceDegree=4, nFolds=10)

end
```

```
avgCVError1 = 0.504784876475921
```

```
avgCVError1 = mean([minimum(kFoldErrorsNpDeg1[i,
:]) for i in 1:length(times)])
```

```
avgCVError = 0.42686140025770264
```

```
avgCVError = mean([minimum(kFoldErrorsNp[i, :])
for i in 1:length(times)])
```

```
avgCVError4 = 0.4384702842338652
```

```
avgCVError4 = mean([minimum(kFoldErrorsNpDeg4[i,
:]) for i in 1:length(times)])
```

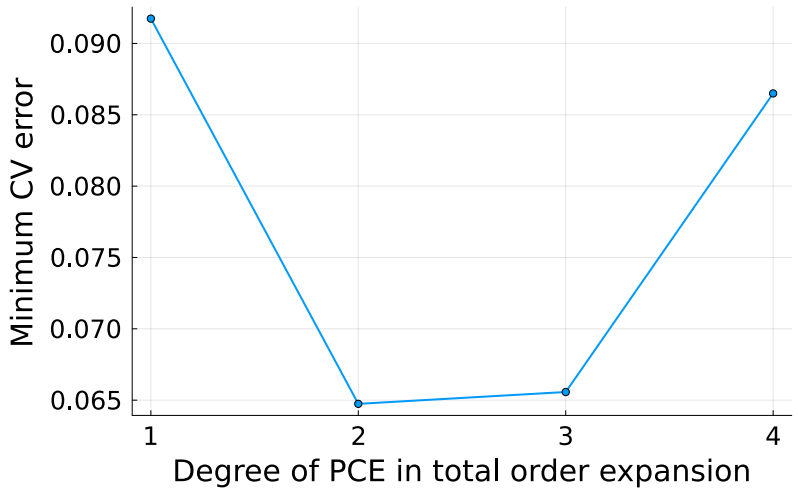
```
avgCVError3 = 0.41796835089228407
```

```
avgCVError3 = mean([minimum(kFoldErrorsNpDeg3[i,
:]) for i in 1:length(times)])
```

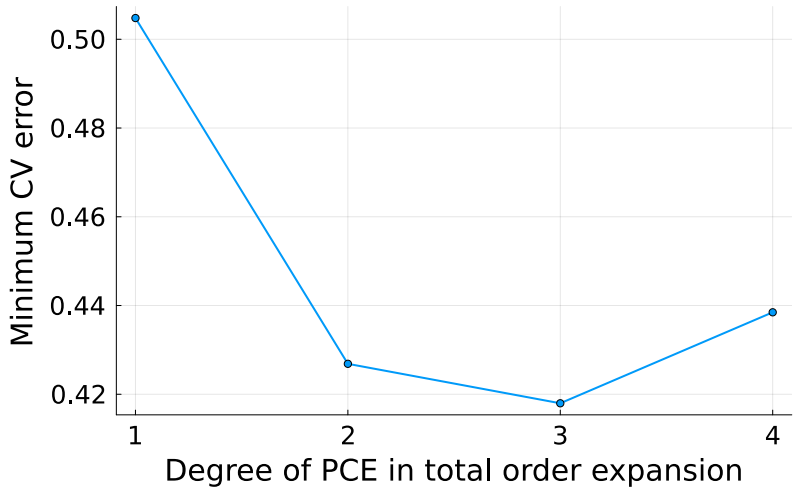
```
0.08649668229660484
```

```
begin
    avgCVError1Ur =
mean([minimum(kFoldErrorsUrDeg1[i, :]) for i in
1:length(times)])
    avgCVErrorUr = mean([minimum(kFoldErrorsUr[i,
:]) for i in 1:length(times)])
    avgCVError3Ur =
mean([minimum(kFoldErrorsUrDeg3[i, :]) for i in
1:length(times)])
    avgCVError4Ur =
mean([minimum(kFoldErrorsUrDeg4[i, :]) for i in
1:length(times)])
end
```

CV Error Plots (Figure 4)




```
begin
    pAvgUr = plot([1, 2, 3, 4], [avgCVError1Ur,
avgCVErrorUr, avgCVError3Ur, avgCVError4Ur], line=
(2), marker=(:circle), label="")
    plot!(xlabel="Degree of PCE in total order
expansion")
    plot!(ylabel="Minimum CV error")
    plot!(guidefontsize=20)
    plot!(legend=:topright)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(top_margin=2mm)
    plot!(ytickfontsize=17)
    plot!(xtickfontsize=17)
    plot!(size=(800, 500))
end
```

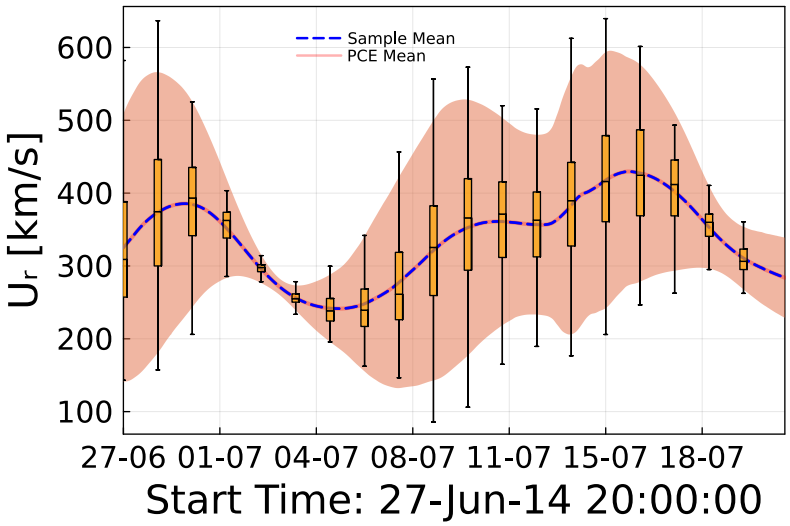



```
begin
    pAvgNp = plot([1, 2, 3, 4], [avgCVError1,
avgCVError, avgCVError3, avgCVError4], line=(2),
marker=(:circle), label="")
    plot!(xlabel="Degree of PCE in total order
expansion")
    plot!(ylabel="Minimum CV error")
    plot!(guidefontsize=20)
    plot!(legend=:topright)
    plot!(left_margin=5mm)
    plot!(bottom_margin=5mm)
    plot!(top_margin=2mm)
    plot!(ytickfontsize=17)
    plot!(xtickfontsize=17)
    plot!(size=(800, 500))
end
```

▸ [91.8676, 92.9005, 93.8162, 94.6642, 95.4176, 96.1029

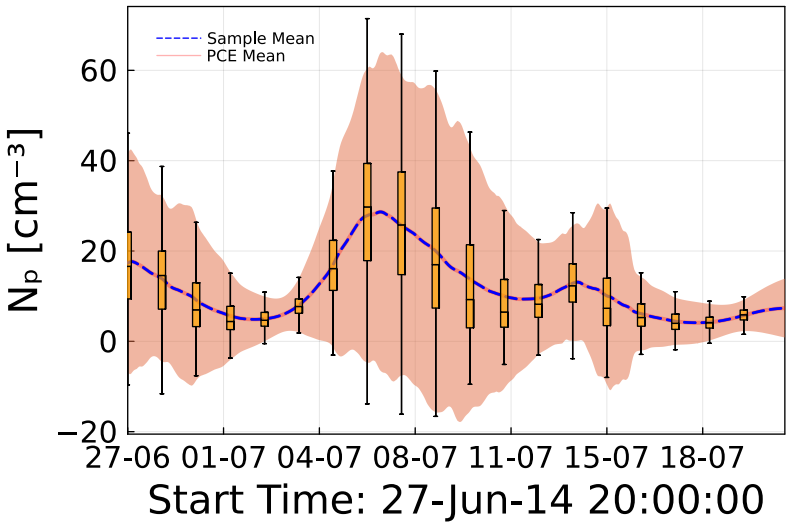
```
begin
    meanEmpiricalNpFinal = mean(yPredNpFinal;
dims=2)
    stdEmpiricalNpFinal = std(yPredNpFinal;
dims=2)[:]

    meanEmpiricalUrFinal = mean(yPredUrFinal;
dims=2)
    stdEmpiricalUrFinal = std(yPredUrFinal;
dims=2)[:]
end
```

Plot predictive uncertainty (figure 5)




```
begin
    plotUncertainty(yPredUrFinal,
meanEmpiricalUrFinal, stdEmpiricalUrFinal, UrObs,
ylabel="Uᵣ [km/s]", times, nSTD=2, trimIndices=
(1, 577))
    plot!(guidefontsize=20)
    plot!(tickfontsize=15)
    plot!(legendfontsize=15)
    plot!(labelsize=15)
    plot!(legend=:top)
    plot!(left_margin=3mm)
    plot!(bottom_margin=3mm)
end
```

```
seriestype boxplot has been moved to StatsPlots.  To
ts"); using StatsPlots`
```




```
begin
    plotUncertainty(yPredNpFinal,
meanEmpiricalNpFinal, stdEmpiricalNpFinal, NpObs,
ylabel="Nₚ [cm⁻³]", ylims=(-20, 120), times,
nSTD=2, trimIndices=(1, 577))
    plot!(guidefontsize=20)
    plot!(tickfontsize=15)
    plot!(legend=:topleft)
    plot!(labelsize=15)
    plot!(left_margin=3mm)
    plot!(bottom_margin=3mm)
end
```

```
seriestype boxplot has been moved to StatsPlots.  To
ts"); using StatsPlots`
```

Sensitivities

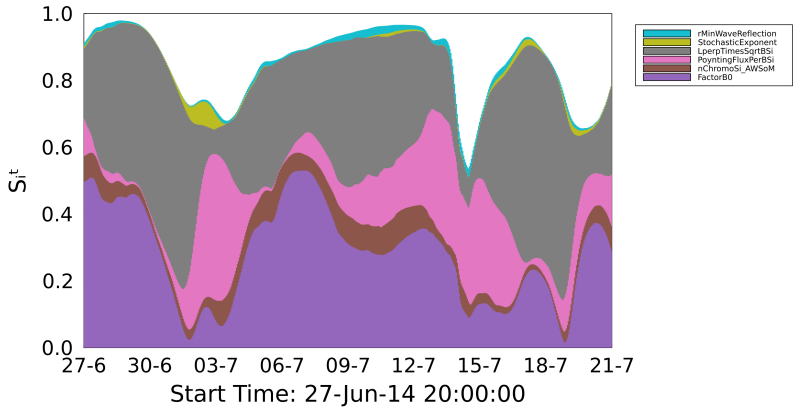
```
6×6×577 Array{Float64, 3}:
[:, :, 1] =
 0.0792493    0.00346788  0.00260307  0.0427347   0.
 0.00346788   0.00443823  0.00940849  0.0111926   0.
 0.00260307   0.00940849  0.612027    0.00435058  0.
 0.0427347    0.0111926   0.00435058  0.0596171   0.
 0.000102257  0.0012828   0.00102672  0.0114068   0.
 0.000155367  0.00148504  0.0680187   0.00456385  6.

[:, :, 2] =
 0.0845792    0.00255476  0.00266734   0.0390925   5
 0.00255476   0.0053032   0.00844868   0.0130476   0
 0.00266734   0.00844868  0.620201     0.00834743  0
 0.0390925    0.0130476   0.00834743   0.0506206   0
 5.71034e-5   0.00243127  0.000216534  0.0105491   0
 0.000187716  0.00225557  0.0656042    0.00467933  3

[:, :, 3] =
 0.0908046    0.0020431   0.0024619   0.0352059   2.
 0.0020431    0.00538104  0.00644603  0.0153008   0.
 0.0024619    0.00644603  0.632638    0.0127606   3.
 0.0352059    0.0153008   0.0127606   0.0444324   0.
 2.17803e-5   0.00406293  3.40941e-6  0.00983979  0.
 0.000182344  0.00252975  0.0586453   0.00488934  3.

;;; …

[:, :, 575] =
 0.124902     9.68884e-6   0.0516556    4.07817e-6
 9.68884e-6   0.00256601   6.54239e-5   0.00352841
 0.0516556    6.54239e-5   0.735824     0.0312985
 4.07817e-6   0.00352841   0.0312985    0.0111532
```

```
begin
    gsaUrFinal = gsa(XTrainFinal, YTrainUr;
regularize=true, pceDegree=2,
lambda=lambdaUrFinal)
    gsaNpFinal = gsa(XTrainFinal, YTrainNp;
regularize=true, pceDegree=2,
lambda=lambdaNpFinal)
end
```

```
6×577 Matrix{Float64}:
 0.0792493   0.0845792   0.0908046   …  0.124902     0
 0.00443823  0.0053032   0.00538104      0.00256601   0
 0.612027    0.620201    0.632638        0.735824     0
 0.0596171   0.0506206   0.0444324       0.0111532    0
 0.0110127   0.00858523  0.00648546      0.000814784  0
 0.0717955   0.0705686   0.0658626   …  0.024114     0
```

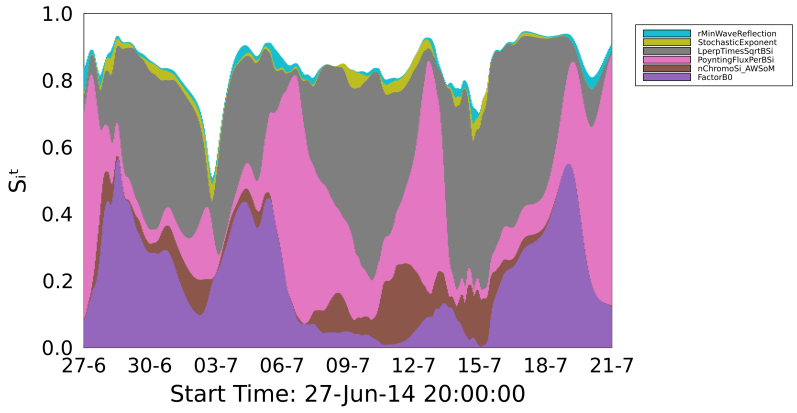
```
# Extract main effects
begin
    gsaMainUrFinal =
processMainEffects(gsaUrFinal)
    gsaMainNpFinal =
processMainEffects(gsaNpFinal)
end
```

Plot sensitivities (figures 6 and 7)

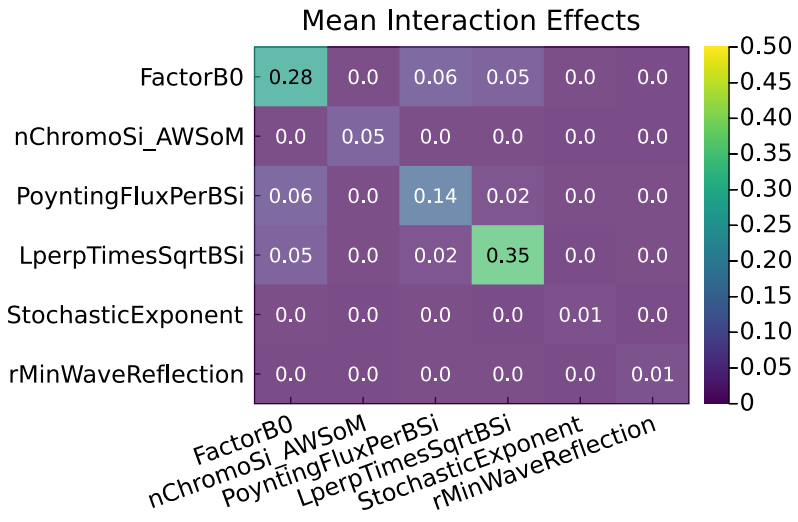



```
begin
    pMainUrFinal =
plotMainEffects(gsaMainUrFinal, times,
inputNames; title = "", dpi=300,
actualStartTime=startTime, tickFormat="dd-m")
    plot!(grid=false)
    plot!(ylabel="Sᵢᵗ")
    plot!(titlefontsize=17)
    plot!(xtickfontsize=20)
    plot!(ytickfontsize=20)
    plot!(guidefontsize=23)
    plot!(leftmargin=7mm)
    plot!(rightmargin=5mm)
    plot!(topmargin=2mm)
    plot!(bottommargin=8mm)
    plot!(size=(1200, 600))
end
```

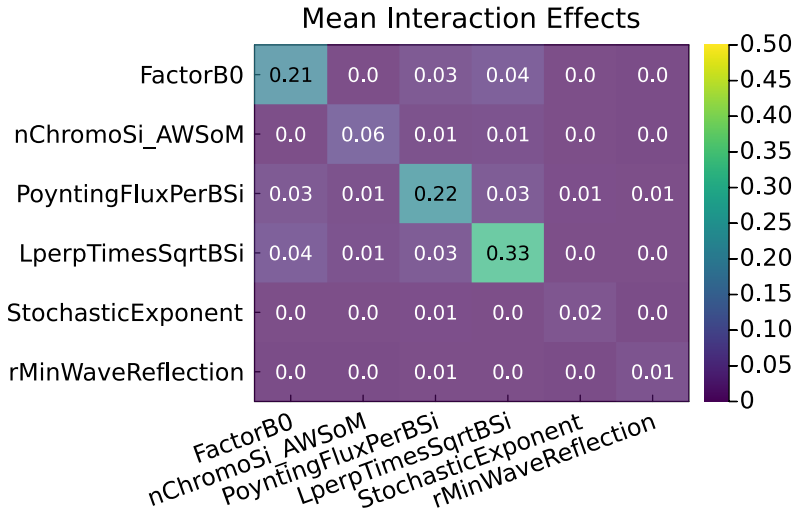
```
# begin
#   savefig(pMainUrFinal,
"/Users/ajivani/Downloads/pMainUrFinal.png")
#   savefig(pMainNpFinal,
"/Users/ajivani/Downloads/pMainNpFinal.png")

# end
```




```
begin
    pMainNpFinal =
plotMainEffects(gsaMainNpFinal, times,
inputNames; title="", dpi=300,
actualStartTime=startTime, tickFormat="dd-m")
    plot!(grid=false)
    plot!(ylabel="Sᵢᵗ")
    plot!(titlefontsize=17)
    plot!(xtickfontsize=20)
    plot!(ytickfontsize=20)
    plot!(guidefontsize=23)
    plot!(leftmargin=7mm)
    plot!(rightmargin=5mm)
    plot!(topmargin=2mm)
    plot!(bottommargin=8mm)
    plot!(size=(1200, 600))
end
```




```
begin
    pUrIntMeanFinal =
plotInteractionEffects(gsaUrFinal, times,
inputNames, dpi=300) # symmetric matrix
(interactions are read from either the upper
triangle or the lower triangle)
    plot!(tickfontsize=12)
    plot!(bottom_margin=9mm)
    plot!(right_margin=3mm)
end
```




```
begin
    pNpIntMeanFinal =
plotInteractionEffects(gsaNpFinal, times,
inputNames, dpi=300)
    plot!(tickfontsize=12)
    plot!(bottom_margin=9mm)
    plot!(right_margin=3mm)
end
```

Analysis based on log-transformed quantity

```
YTrainNpLog = Array{Float64, 2}(log.(NpSim[:, finalRuns])');
```

```
lambdaNpLogFinal = 2
```

```
lambdaNpLogFinal = 2
```

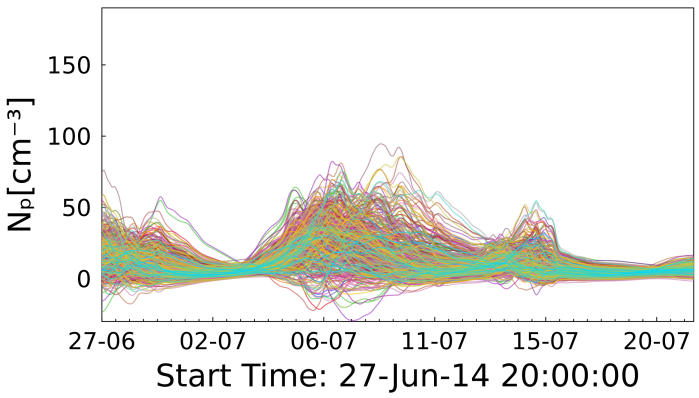
```
betaNpLogFinal =
28×577 Matrix{Float64}:
  2.57227     2.57104     2.57015    …   1.88027
  0.196904    0.20327     0.212084       0.115632
 -0.034694   -0.0443131  -0.0528377     -0.0339354
  0.506623    0.512231    0.520631       0.315389
  0.300252    0.275831    0.251365       0.0295243
 -0.0571687  -0.0603421  -0.0618873  …  -0.0254556
  0.0500137   0.0490334   0.0468386      0.0407784
  ⋮                                  ⋱
 -0.180544   -0.186789   -0.191932      -0.021068
  0.11723     0.114106    0.110489       0.0109172
 -0.0314794  -0.0379026  -0.0449436     -0.0333847
  0.0595741   0.0590162   0.0571157  …   0.0166245
 -0.0225383  -0.0214708  -0.0205065      0.000236413
  0.034486    0.0392805   0.0427157      0.00599642
```

```
betaNpLogFinal = solveRegPCE(ATrainFinal, YTrainNpLog; λ=lambdaNpLogFinal)
```

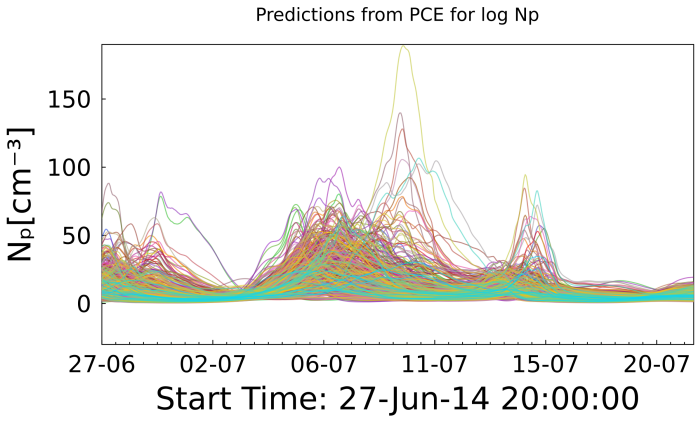
```
577×400 Matrix{Float64}:
 4.26251  32.6399  9.30266  14.4358   …  4.42614  8.83
 4.03528  32.9815  8.84372  14.5719      4.21727  8.76
 3.84493  33.3619  8.41455  14.7113      4.02234  8.71
 3.71642  33.7268  8.00291  14.8866      3.84681  8.68
 3.62803  33.9074  7.58987  15.035       3.69956  8.65
 3.55265  33.8543  7.16563  15.1338   …  3.56994  8.59
 3.48165  33.6561  6.74566  15.1839      3.45261  8.49
 ⋮                                   ⋱
 2.62007  11.2545  5.55878   6.0724      4.82749  4.79
 2.5957   11.3369  5.53718   6.06667     4.80384  4.80
 2.57499  11.4099  5.51525   6.05423     4.79362  4.80
 2.55342  11.4794  5.49279   6.04207     4.78349  4.80
 2.53166  11.539   5.46937   6.02812  …  4.77547  4.79
 2.51365  11.5873  5.44589   6.00976     4.78098  4.78
```

```
begin
    yPredNpLogFinal = betaNpLogFinal[1, :] .+
Matrix((ATestFinalFiltered *
betaNpLogFinal[2:end, :])')
    yPredNpTransformed = exp.(yPredNpLogFinal)
end
```

Plot predictions of PCE surrogate trained on Np and log(Np). Also plot time-averaged sensitivities for the latter. (figure 8)

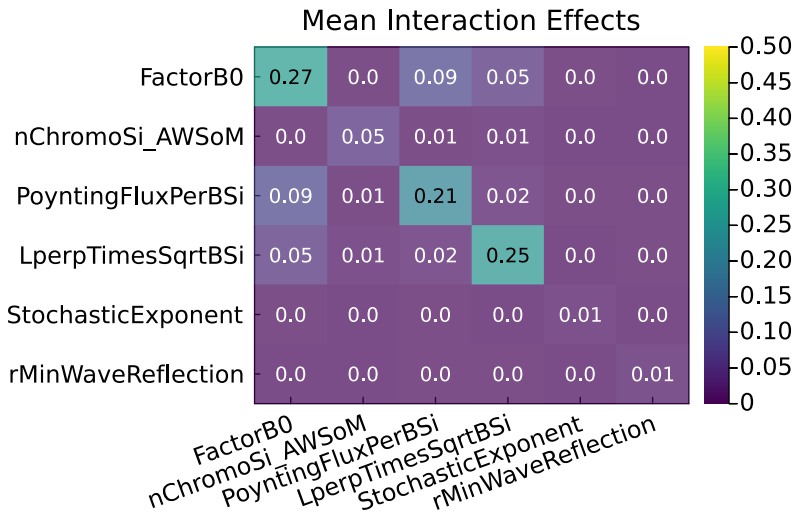



```
begin
    pNpSimObsOrigPred = plotSimObs(yPredNpFinal,
NpObs, times, collect(1:200);
simIdx=collect(1:400), plotArgsNp..., ylims=(-30,
190), dateFormat="dd-mm")
    plot!(size=(920, 470))
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)
end
```




```
begin
    pNpSimObsLogPred =
plotSimObs(yPredNpTransformed, NpObs, times,
collect(1:200); simIdx=collect(1:400),
plotArgsNp..., ylims=(-30, 190), dateFormat="dd-
mm")
    plot!(size=(920, 470))
    plot!(title="Predictions from PCE for log Np")
    plot!(guidefontsize=24)
    plot!(left_margin=7mm)
    plot!(bottom_margin=7mm)
    plot!(top_margin=5.5mm)
    plot!(ytickfontsize=18)
    plot!(xtickfontsize=18)
    plot!(dpi=300)
end
```

```
gsaNpLogFinal = gsa(XTrainFinal, YTrainNpLog;
regularize=true, pceDegree=2,
lambda=lambdaNpLogFinal);
```




```
begin
    pNpLogIntMeanFinal =
plotInteractionEffects(gsaNpLogFinal, times,
inputNames, dpi=300)
    plot!(tickfontsize=12)
    plot!(bottom_margin=9mm)
    plot!(right_margin=3mm)
end
```

Bootstrapping

Load saved bootstrap data. Performing the bootstrap can be quite slow, so the actual command for getting the data is commented out.

```
# UrBootstrap = bootstrapGSA(XTrainFinal,
YTrainUr; regularize=false, nStart=20, nEnd=140,
nStep=20)
# NpBootstrap = bootstrapGSA(XTrainFinal,
YTrainNp; regularize=false, nStart=20, nEnd=140,
nStep=20)
```

```
6×7 Matrix{Float64}:
 0.0341864  0.0568825  0.0434664  0.0360413   0.033333
 0.0295207  0.0240916  0.0153496  0.013388    0.011512
 0.0268163  0.0748956  0.0517789  0.0463773   0.040285
 0.0524918  0.0473704  0.0384021  0.0303921   0.027625
 0.0236411  0.0208347  0.0105043  0.00720721  0.005852
 0.0241669  0.0215853  0.0103924  0.00773949  0.006449
```

```
begin
    UrBootstrap =
load("./data/bootstrapping/bootstrapUr2152.jld",
"UrBootstrap")
    avgBootstrapUr = mean(UrBootstrap; dims=2)[:,
1, :, :]
    avgBootstrapRepsUr = mean(avgBootstrapUr;
dims=2)[:, 1, :]
    stdBootstrapRepsUr = std(avgBootstrapUr;
dims=2)[:, 1, :]
end
```

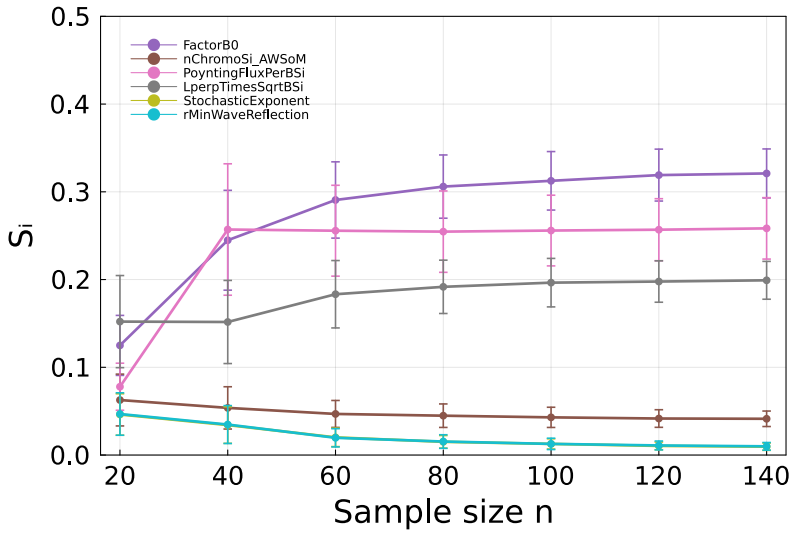
```
6×7 Matrix{Float64}:
 0.0382751  0.0420425  0.0257733  0.0218147   0.018769
 0.0290315  0.0204736  0.011135   0.00883917  0.007566
 0.0354008  0.052822   0.0275101  0.021119    0.016373
 0.0440309  0.0293069  0.020367   0.017997    0.015815
 0.02577    0.0168068  0.0083131  0.00594387  0.004521
 0.0244229  0.019108   0.0073943  0.00448848  0.003458
```

```
begin
    NpBootstrap =
load("./data/bootstrapping/bootstrapNp2152.jld",
"NpBootstrap")
    avgBootstrapNp = mean(NpBootstrap; dims=2)[:,
1, :, :]
    avgBootstrapRepsNp = mean(avgBootstrapNp;
dims=2)[:, 1, :]
    stdBootstrapRepsNp = std(avgBootstrapNp;
dims=2)[:, 1, :]
end
```

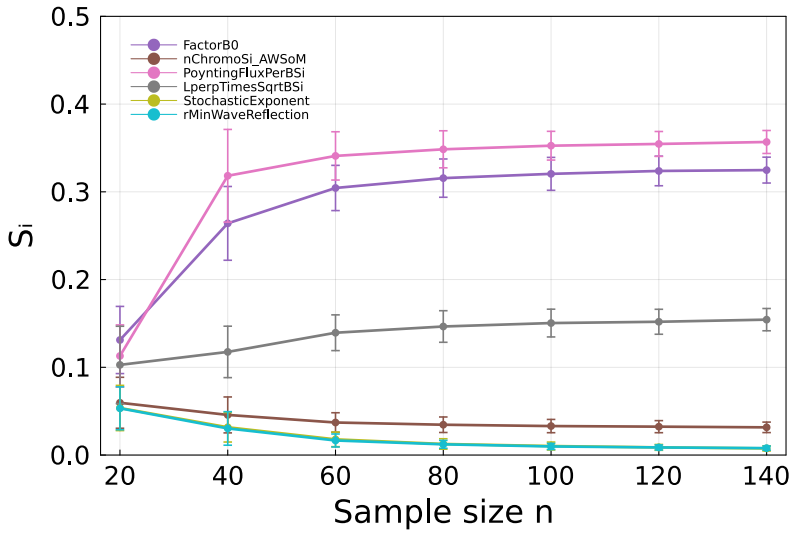
```
summaryColors =
```

```
summaryColors = palette(:tab10, rev=true)[6:-1:1]
```

Plot summary of bootstrapping results. (figure 9)




```
begin
    pLineSummaryUr = plot()
    for (idx, eachName) in enumerate(inputNames)
        meanTrend = avgBootstrapRepsUr[idx, :]
        errTrend  = stdBootstrapRepsUr[idx, :]
        plot!(20:20:140, meanTrend,
            yerr=errTrend,
            linewidth=2.5,
            linecolor=summaryColors[idx],
            marker=:circle,
            markercolor=summaryColors[idx],
            markerstrokecolor=summaryColors[idx],
            markerstrokewidth=1.5,
            label=eachName)
    end
    plot!(legend=:topleft)
    plot!(xticks=(20:20:140, string.(20:20:140)))
    plot!(xlabel="Sample size n")
    plot!(ylabel="Sᵢ")
    plot!(guidefontsize=20)
    plot!(tickfontsize=16)
    plot!(leftmargin=5mm)
    plot!(bottommargin=5mm)
    plot!(framestyle=:box)
    plot!(fg_legend=nothing)
    plot!(bg_legend=nothing)
    plot!(size=(750, 500))
    plot!(ylims=(0, 0.5))
end
```




```
begin
    pLineSummaryNp = plot()
    for (idx, eachName) in enumerate(inputNames)
        meanTrend = avgBootstrapRepsNp[idx, :]
        errTrend  = stdBootstrapRepsNp[idx, :]
        plot!(20:20:140, meanTrend,
            yerr=errTrend,
            linewidth=2.5,
            linecolor=summaryColors[idx],
            marker=:circle,
            markercolor=summaryColors[idx],
            markerstrokecolor=summaryColors[idx],
            markerstrokewidth=1.5,
            label=eachName)
    end
    plot!(legend=:topleft)
    plot!(xticks=(20:20:140, string.(20:20:140)))
    plot!(xlabel="Sample size n")
    plot!(ylabel="Sᵢ")
    plot!(guidefontsize=20)
    plot!(tickfontsize=16)
    plot!(leftmargin=5mm)
    plot!(bottommargin=5mm)
    plot!(framestyle=:box)
    plot!(fg_legend=nothing)
    plot!(bg_legend=nothing)
    plot!(size=(750, 500))
    plot!(ylims=(0, 0.5))
end
```

Miscellaneous

```
▶ Dict(:dpi => 200, :ylims => (0, 900000.0), :simWidth
```

```
begin
        plotArgsUr =
Dict(:palette=>:seaborn_bright,
                        :dateFormat=>"dd-mm HH:MM",
                        :tickInterval=>108,
                        # :simIdx => parse.(Int,
EachSimID),
                        :simAlpha=>0.6,
                        :simWidth=>1.2,
                        :ylabel=>"Ur[km/s]",
                        :title=>"",
                        :startTime=>startTime,
                        :dpi=>200,
                        :plotLabels=>:false,

                       )

        plotArgsNp =
Dict(:palette=>:seaborn_bright,
                           :dateFormat=>"dd-mm
HH:MM",
                           :tickInterval=>108,
                           # :simIdx => parse.
(Int, EachSimID),
                           :simAlpha=>0.6,
                           :simWidth=>1.2,
                           :ylabel=>"Np[cm-3]",
                           # :ylabel=>"Np [cm" *
L"^{-3}" * "]",
                           :title=>"",
                           :startTime=>startTime,
                           :dpi=>200,
                           :ylims=>(0, 80),
                           :plotLabels=>:false
                          )

        plotArgsB =
Dict(:palette=>:seaborn_bright,
                           :dateFormat=>"dd-mm
HH:MM",
                           :tickInterval=>108,
                           # :simIdx => parse.
(Int, EachSimID),
                           :simAlpha=>0.6,
                           :simWidth=>1.2,
                           :ylabel=>"B[nT]",
                           # :ylabel=>"Np [cm" *
L"^{-3}" * "]",
                           :title=>"",
                           :startTime=>startTime,
                           :dpi=>200,
                           :ylims=>(0, 20),
```

```
                            :plotLabels=>:false,
                            :subtractFactor=>0
                           )

        plotArgsT =
Dict(:palette=>:seaborn_bright,
                           :dateFormat=>"dd-mm HH:MM",
                           :tickInterval=>108,
                           # :simIdx => parse.(Int,
EachSimID),
                           :simAlpha=>0.6,
                           :simWidth=>1.5,
                           :ylabel=>"T[K]",
                           # :ylabel=>"Np [cm" *
L"^{-3}" * "]",
                           :title=>"",
                           :startTime=>startTime,
                           :dpi=>200,
                           :ylims=>(0, 9e5),
                           :plotLabels=>:false,
                           :subtractFactor=>0
                          )
end
```