Comparison with Markus Werner

Martin Ueding

August 24, 2019

In this notebook we compare the data from the new projection code with the results that Markus Werner has generated with his code.

Markus has already projected all the data for the ρ channel, we can just use this to compare. His data is not only in a different format but also uses different parametrizations for individual momenta p_i via his P' and q' values.

We want to do a fully automated complete comparison such that there are no manual steps which are additional sources of errors.

We have picked the B35.32 ensemble for no particular reason. From Markus we take the contractions for configuration 2552 in the form of HDF5 files. Also we take his projected correlators in the form of TSV files to compare to.

```
total_momentum_ref <- function (total_momentum_sq) {
  if (total_momentum_sq == 0)
    c(0, 0, 0)
  else if (total_momentum_sq == 1)
    c(0, 0, 1)
  else if (total_momentum_sq == 2)
    c(1, 1, 0)
  else if (total_momentum_sq == 3)
    c(1, 1, 1)
  else if (total_momentum_sq == 4)
    c(0, 0, 2)
}

momentum_to_string <- function (momentum) {
  paste0(sprintf("%d", momentum), collapse = '')
}

total_momentum_str_to_vec <- function (total_momentum_str) {
  parts <- stringr::str_match(total_momentum_str, '(-?\\d)(-?\\d)(-?\\d)')
  sapply(parts[1, 2:4], as.integer)
}
```

This comparison will use paramvalf within a single notebook. Users of it might miss the `pv_load` and `pv_save` calls but we are just using it as a library and not as a fully fleged framework here.

The data resides in the "work" directory where as the working directory of this notebook is the source directory. Therefore we first need to find it. This is different on the various machines, so we will just try a few candidates.

```
workdir_candidates <- c(
  '~/Lattice/NG2',
  '~/NG2'
)

for (path in workdir_candidates) {
  if (dir.exists(path)) {
    workdir <- path
    break
  }
}
```

Reading actual data

We will do all the comparisons to the total momenta and irreps that we have projected. For this we just parse all the files for a selected configuration and build a parameter table out of them.

```
split_filename <- function (path) {
  filename <- basename(path)
  parts <- stringr::str_match(filename, 'resolved_(.*)_(.*)_(.*)\\.js')
  parts[1, 2:4]
}

all_projected_paths <- Sys.glob(paste0(workdir, '/projected/resolved_*_*_2552.js'))

parts <- do.call(rbind, lapply(all_projected_paths, split_filename))
colnames(parts) <- c('total_momentum_str', 'irrep', 'config_number')
df <- as_tibble(parts)
df$path <- all_projected_paths
df$config_number <- as.integer(df$config_number)

irreps <- df_to_paramval(df, c('total_momentum_str', 'irrep', 'config_number'))
```

The parameter is the total momentum as string, we need to extract it as a three-vector and and also as a compute the $\vec{P}^2$.

```
.func <- function (param, value) {
  total_momentum_vec <- total_momentum_str_to_vec(param$total_momentum_str)
  total_momentum_sq <- sum(total_momentum_vec^2)

  list(
    total_momentum_vec = total_momentum_vec,
    total_momentum_sq = total_momentum_sq)
}

total_momentum <- pv_call(.func, irreps)
```

For each irrep we need to figure out which relative momenta q we actually have in our data. In case that there are no relative momenta available we have no coupling and therefore we can discard it.

```
.func <- function (param, value) {
  data <- jsonlite::read_json(value$path)

  q_avail_actual <- character(0)
  for (elem in data[[param$total_momentum_str]][[param$irrep]]) {
    for (inner in elem) {
      q_avail_actual <- c(q_avail_actual, names(inner))
    }
  }

  if (length(q_avail_actual) == 0) {
    return (NA)
  }

  list(actual_avail = sort(unique(q_avail_actual)))
}

actual_avail <- pv_call(.func, irreps, total_momentum)
```

Reading target data

Markus' parametrization works as

$$p_1 = \frac{P'}{2} + q' \,, \quad p_2 = \frac{P'}{2} - q' \,.$$

The parametrization used in this code here is $p_1 = P - q$ and $p_2 = q$ for the case of two particles. This means that in order to convert his parametrization into mine one has to use the relations

$$P = P' \,, \quad q = \frac{P'}{2} - q' \,.$$

The other way around works exactly the same:

$$P' = P \,, \quad q' = \frac{P}{2} - q \,.$$

His correlator matrix format contains files called `operator-indices` that give the three-vector P that is actually used for that particular moving frame, as well as the irrep row α:

```
id  p_x p_y p_z alpha
0   -1  0   0   1
1   0   -1  0   1
2   0   0   -1  1
3   0   0   1   1
4   0   1   0   1
5   1   0   0   1
```

One has to look up our $\vec{P}$ and then find out the *operator id* from this table.

The correlator matrix elements are then labeled by their q'_{source} and q'_{sink} values. These also correspond to indices that have to be read off from the lines with the correct Dirac structure, `g: \gamma_{5}, \gamma_{5}`, in the file `gevp-indices`:

```
id  element
0   p: 1, g: \gamma_{50i}
1   p: 1, g: \gamma_{i}
2   p: 1, q: (0.0, 0.0, 0.5), g: \gamma_{5}, \gamma_{5}
3   p: 1, q: (0.0, 0.0, 1.5), g: \gamma_{5}, \gamma_{5}
4   p: 1, q: (1.0, 0.0, 0.5), g: \gamma_{5}, \gamma_{5}
5   p: 1, q: (1.0, 1.0, 0.5), g: \gamma_{5}, \gamma_{5}
```

From this we can deduce which correlator matrix rows and columns correspond to my q_{source} and q_{sink}. The following function allows us to convert Markus format into my format. This way the comparison will work with my parametrization.

```r
parse_markus_gevp_indices <- function (line) {
  pattern <- 'p: (\\d+), q: \\(([\\d.-]+), ([\\d.-]+), ([\\d.-]+)\\), g: \\\\gamma_\\{5\\}, \\\\gamma_\
  match <- stringr::str_match(line, pattern)
  total_momentum_sq <- as.numeric(match[1, 2])

  if (is.na(total_momentum_sq)) {
    return (NA)
  }

  relative_momenta_markus <- as.numeric(match[1, 3:5])

  total_momentum_vec <- total_momentum_ref(total_momentum_sq)
  relative_momenta_martin <- round(total_momentum_vec / 2 - relative_momenta_markus)

  momentum_to_string(relative_momenta_martin)
}

stopifnot('-1-10' == parse_markus_gevp_indices('p: 1, q: (1.0, 1.0, 0.5), g: \\gamma_{5}, \\gamma_{5}')
stopifnot('000' == parse_markus_gevp_indices('p: 1, q: (0.0, 0.0, 0.5), g: \\gamma_{5}, \\gamma_{5}'))
stopifnot('00-1' == parse_markus_gevp_indices('p: 1, q: (0.0, 0.0, 1.5), g: \\gamma_{5}, \\gamma_{5}'))
stopifnot(is.na(parse_markus_gevp_indices('p: 1, g: \\gamma_{5}')))
```

We gather the number of correlator matrices that Markus has.

```r
glob <- sprintf('%s/reference/rho_p?_*_operator-indices.tsv', workdir)
target_filenames <- Sys.glob(glob)
```

For the correlator matrices that we have available in the actual data we also try to load the target data. They might not all exist as Markus has only created the files for the one that actually have coupling. In case the files do not exist we can therefore safely skip them.

```r
.func <- function (param, value) {
  operator_indices_path <- sprintf('%s/reference/rho_p%d_%s_operator-indices.tsv', workdir, value$total_
  if (!file.exists(operator_indices_path)) {
    return (NA)
  }
  operator_indices <- read.table(operator_indices_path, sep = '\t', header = TRUE)

  filtered <- operator_indices %>%
    filter(p_x == value$total_momentum_vec[1],
           p_y == value$total_momentum_vec[2],
           p_z == value$total_momentum_vec[3]) %>%
    select(id, alpha)
  operator_id <- filtered$id

  gevp_indices_path <- sprintf('%s/reference/rho_p%d_%s_gevp-indices.tsv', workdir, value$total_momentu

  gevp_indices <- read.table(gevp_indices_path, sep = '\t', header = TRUE, stringsAsFactors = FALSE)

  target_gevp_indices_converted <- gevp_indices %>%
    mutate(relative_momenta = sapply(element, parse_markus_gevp_indices)) %>%
    filter(!is.na(relative_momenta)) %>%
    select(id, relative_momenta)

  list(target_gevp_indices = target_gevp_indices_converted,
        target_operator_id = filtered)
}

target_avail <- pv_call(.func, irreps, total_momentum)
```

Matching up available and actual

Number of correlator matrices

There is no notational difference in the total momentum and the irreps. We must be able to load all of Markus's correlator matrices. If we cannot do that, something is off. In case that the actual data contains more than the target data we do not have a problem because these are just the ones that Markus did not write out due to non-coupling.

The following table lists all the correlator matrices that were found in both target and actual data sets.

```
semi_join(actual_avail$param, target_avail$param)
```

```
## Joining, by = c("total_momentum_str", "irrep", "config_number")

## # A tibble: 69 x 3
##    total_momentum_str irrep config_number
##    <chr>              <chr>         <int>
##  1 000                T1u            2552
##  2 00-1               A1             2552
##  3 001                A1             2552
##  4 00-1               E              2552
##  5 001                E              2552
##  6 00-2               A1             2552
##  7 002                A1             2552
##  8 00-2               E              2552
##  9 002                E              2552
## 10 0-1-1              A1             2552
## # ... with 59 more rows
```

The following table lists the correlator matrices that show coupling in the actual data but are not available in the target data. This can either be errors while loading the target data or just that Markus deemed them irrelevant for $l = 1$ and $I_3 = 1$ scattering. It seems that all these ones are understood and therefore everything that not matched up is not a problem.

```
anti_join(actual_avail$param, target_avail$param)
```

```
## Joining, by = c("total_momentum_str", "irrep", "config_number")

## # A tibble: 13 x 3
##    total_momentum_str irrep config_number
##    <chr>              <chr>         <int>
##  1 000                T2u            2552
##  2 00-1               B1             2552
##  3 001                B1             2552
##  4 00-1               B2             2552
##  5 001                B2             2552
##  6 0-10               B1             2552
##  7 010                B1             2552
##  8 0-10               B2             2552
##  9 010                 B2              2552
## 10 -100                B1              2552
## 11 100                 B1              2552
## 12 -100                B2              2552
## 13 100                 B2              2552
```

In the other direction there must be no target data sets which we do not have in the actual data set. The following table should have zero rows.

```
anti_join(target_avail$param, actual_avail$param)
```

```
## Joining, by = c("total_momentum_str", "irrep", "config_number")
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: total_momentum_str <chr>, irrep <chr>,
## #   config_number <int>
```

Momenta orbits

Markus and I have chosen different representatives for the relative momenta from each orbit. This means that at first glance there are many correlator matrix elements just missing in the other data. One must use a list of relative momenta from the orbits in order to match this up. Using Mathematica we can create a mapping from relative momentum labels to their respective orbit representative.

```
path <- sprintf('%s/momenta_orbits.js', workdir)
momentum_orbits <- jsonlite::read_json(path)
```

```
.func <- function (param, value) {
  target_relative_momenta <- as.character(momentum_orbits[[param$total_momentum_str]][value$target_gevp
  stopifnot(length(value$target_gevp_indices$relative_momenta) == length(target_relative_momenta))

  df <- tibble(
    target_gevp_id = value$target_gevp_indices$id,
    target_relative_momenta = target_relative_momenta)

  df2 <- df %>%
    filter(target_relative_momenta %in% value$actual_avail)

  if (nrow(df) != nrow(df2)) {
    str(param)
    cat(sprintf('%d -> %d\n', nrow(df), nrow(df2)))
    str(anti_join(df, df2))
    print(value$actual_avail)
  }

  if (nrow(df2) == 0) {
    str(param)
    print(sort(unique(df$target_relative_momenta)))
    print(sort(unique(value$actual_avail)))
    return (NA)
  }

  i <- 1:nrow(df2)
  k <- 1:nrow(value$target_operator_id)
  stopifnot('target_gevp_id' %in% colnames(df2))
  stopifnot('target_relative_momenta' %in% colnames(df2))

  grid <- as_tibble(expand.grid(
    i = i,
    j = i,
    k = k,
    l = k))
  attr(grid, 'out.attrs') <- NULL
  grid$target_source_id <- df2$target_gevp_id[grid$i]
  grid$target_sink_id <- df2$target_gevp_id[grid$j]
  grid$irrep_row <- value$target_operator_id$alpha[grid$k]
  grid$irrep_col <- grid$l
  grid$target_operator_id <- value$target_operator_id$id[grid$k]
  grid$source_relative_momenta <- df2$target_relative_momenta[grid$i]
  grid$sink_relative_momenta <- df2$target_relative_momenta[grid$j]
  grid$i <- NULL
  grid$j <- NULL
  grid$k <- NULL

  stopifnot(all(!is.na(grid$target_source_id)))
  stopifnot(all(!is.na(grid$target_sink_id)))

  list(matched_up_grid = grid)
}

matched_up <- pv_call(.func, actual_avail, target_avail)
```

During the process we might lose some of the data sets. The following table lists all the ones that got lost. This must not happen as it is an indication that there is no overlap in the relative momenta that are available. This likely means that either the momenta cutoffs are too strict in the new analytic projection code or that the correspondence using the momentum orbits has not worked correctly. All these cases must be understood and fixed.

```
anti_join(target_avail$param, matched_up$param)
```

```
## Joining, by = c("total_momentum_str", "irrep", "config_number")
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: total_momentum_str <chr>, irrep <chr>,
## #   config_number <int>
```

Reading target correlators

At this point we have figured out the correspondence of operators. We can just take the outer product of these operators to flesh out the whole correlator matrix and load the elements from both the target and actual data.

```
load_target_config <- function (path, config_number) {
  target <- read.table(path, header = TRUE)

  target_single <- dplyr::filter(target, cnfg == config_number)
  target_single$value
}
```

```r
.func <- function (param, value) {
  grid <- value$matched_up_grid

  grid$filename_target <- sprintf(
    '%s/reference/rho_p%d_%s_op%d_gevp%d.%d.tsv',
    workdir,
    value$total_momentum_sq,
    param$irrep,
    grid$target_operator_id,
    grid$target_source_id,
    grid$target_sink_id)

  target_loaded <- lapply(grid$filename_target, load_target_config, config_number = param$config_number
  grid$target <- target_loaded

  filename_actual <- value$path
  resolved <- jsonlite::read_json(filename_actual)
  load_actual_config <- function (q_source, q_sink, irrep_row, irrep_col) {
    irrep_col_str <- as.character(irrep_col)
    irrep_row_str <- as.character(irrep_row)
    data <- unlist(resolved[[param$total_momentum_str]][[param$irrep]][[irrep_col_str]][[irrep_row_str]
    if (is.null(data)) {
      return (NA)
    }
    return (data)
  }
  actual_loaded <- mapply(load_actual_config, grid$source_relative_momenta, grid$sink_relative_momenta,
  grid$actual <- actual_loaded

  grid$time <- lapply(1:nrow(grid), function (i) 0:(length(grid$target[[1]])-1))

  list(filename_actual = filename_actual,
       grid = grid)
}

loaded <- pv_call(.func, matched_up, total_momentum, target_avail, irreps)
```

Comparison

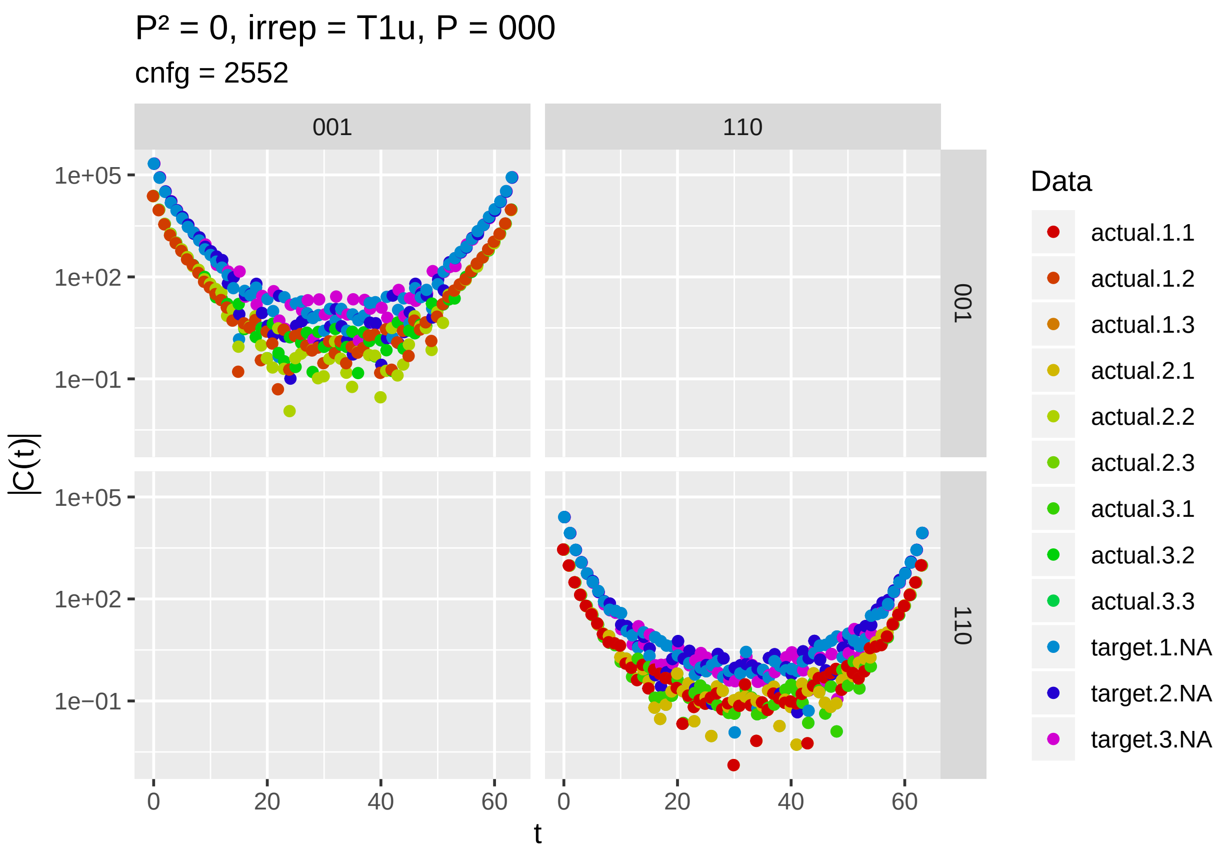
For a comparison with the ggplot library it is easiest to convert the correlator matrices into the long format and have target and actual data in the same column instead of two distinct ones.

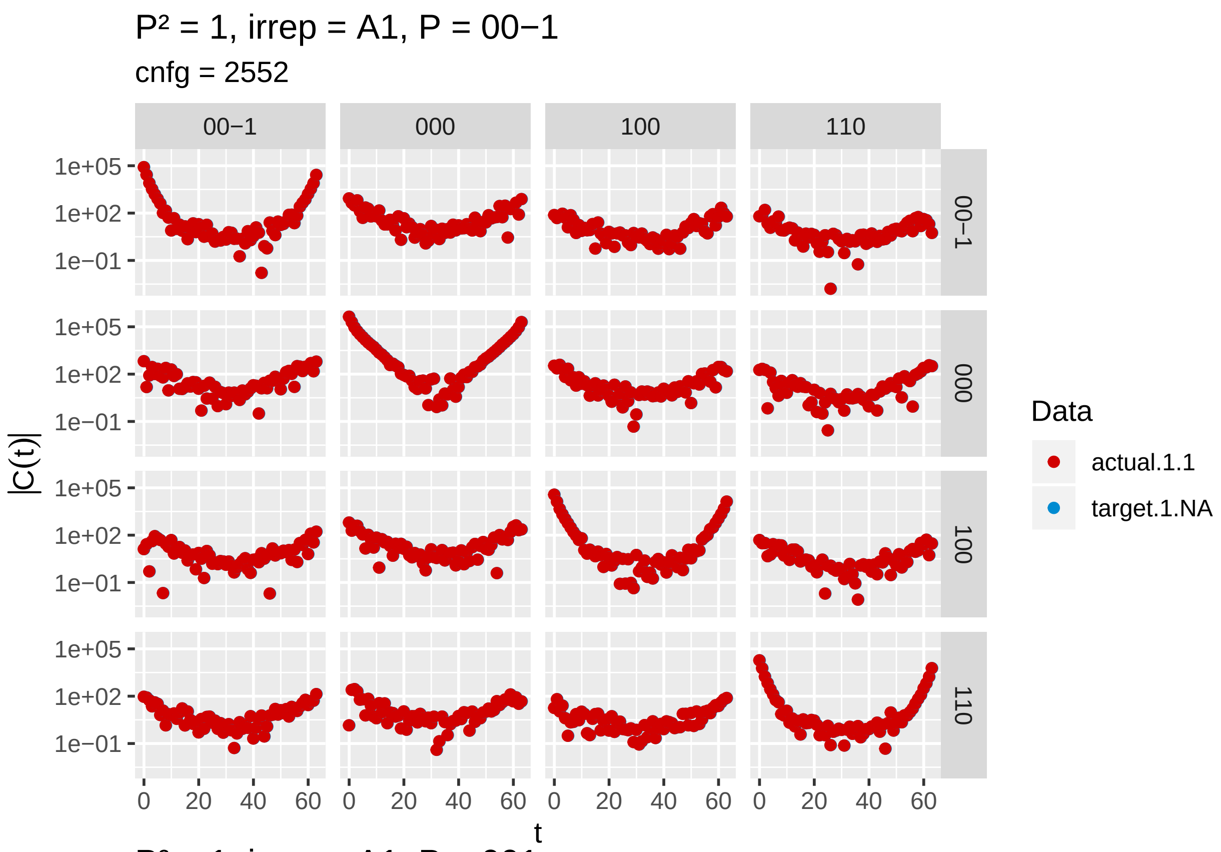
```r
.func <- function (param, value) {
  valid <- !is.na(value$grid$actual)
  grid2 <- value$grid[valid, ]
  long <- tidyr::unnest(grid2)
  long2 <- tidyr::gather(long, kind, correlator, target, actual)

  # We do not have the individual irrep columns for the target data. Therefore we just filter it to 1 a
  long3 <- long2 %>%
    filter(kind == 'actual' || irrep_col == 1)
  long3$irrep_col[long3$kind == 'target'] <- NA

  list(loaded_long = long3)
}

loaded_long <- pv_call(.func, loaded)

.func <- function (param, value) {
  list(summary = cbind(
    value$loaded_long,
    data.frame(
      total_momentum_sq = value$total_momentum_sq)))
}

loaded_summary <- pv_call(.func, loaded_long, total_momentum)
```

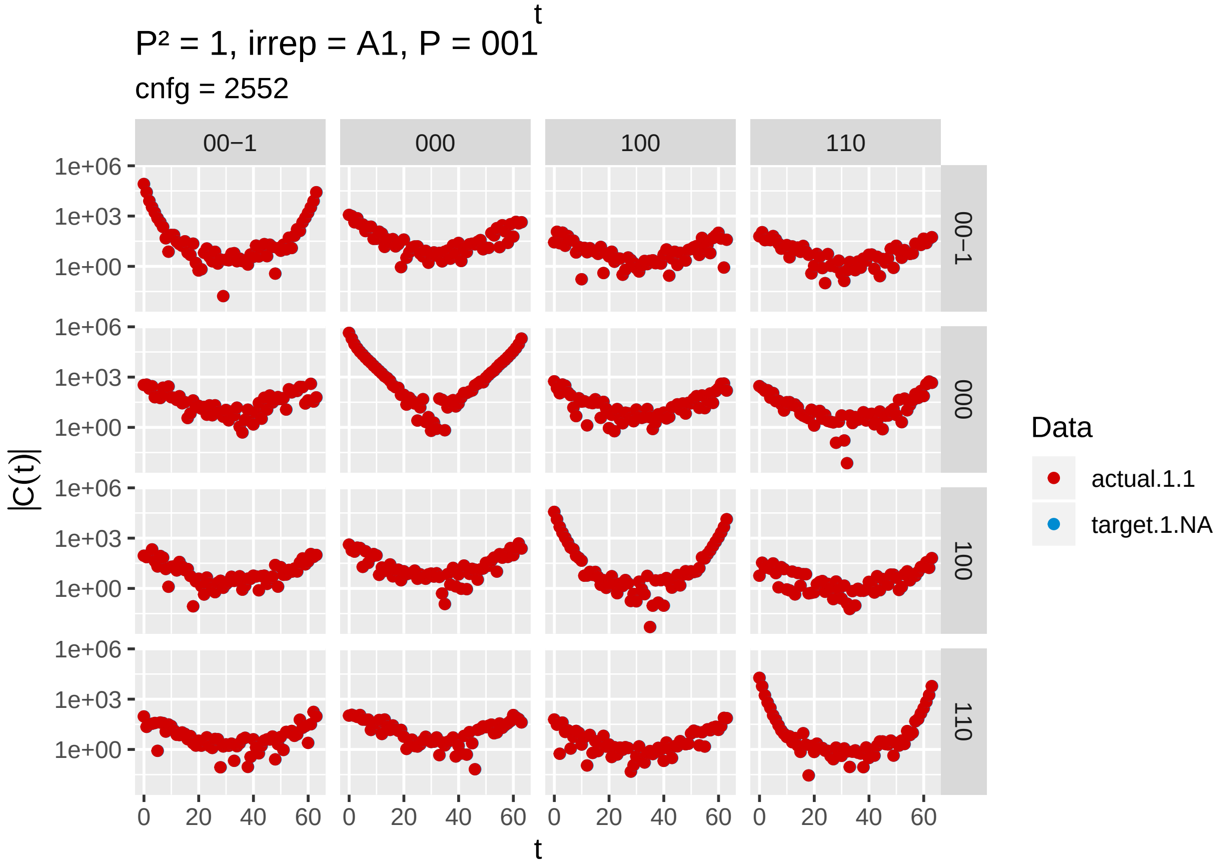
```r
ls2 <- loaded_summary %>%
  mutate(color_key = interaction(kind, irrep_row, sprintf("%d", irrep_col), lex.order = TRUE))

num_col_actual <- length(unique(unclass(filter(ls2, kind == 'actual')$color_key)))
num_col_target <- length(unique(unclass(filter(ls2, kind == 'target')$color_key)))

col_actual <- rainbow(num_col_actual, v = 0.82, start = 0.0, end = 140/360)
col_target <- rainbow(num_col_target, v = 0.82, start = 200/360, end = 300/360)

both_keys <- sort(unique(levels(ls2$color_key)[ls2$color_key]))
both_colors <- c(col_actual, col_target)

col_num <- unclass(ls2$color_key)
col_map <- integer(max(col_num))
col_map[sort(unique(col_num))] <- 1:length(both_colors)
col_value <- both_colors[col_map[col_num]]
ls2$color <- col_value

names(both_colors) <- both_keys
```

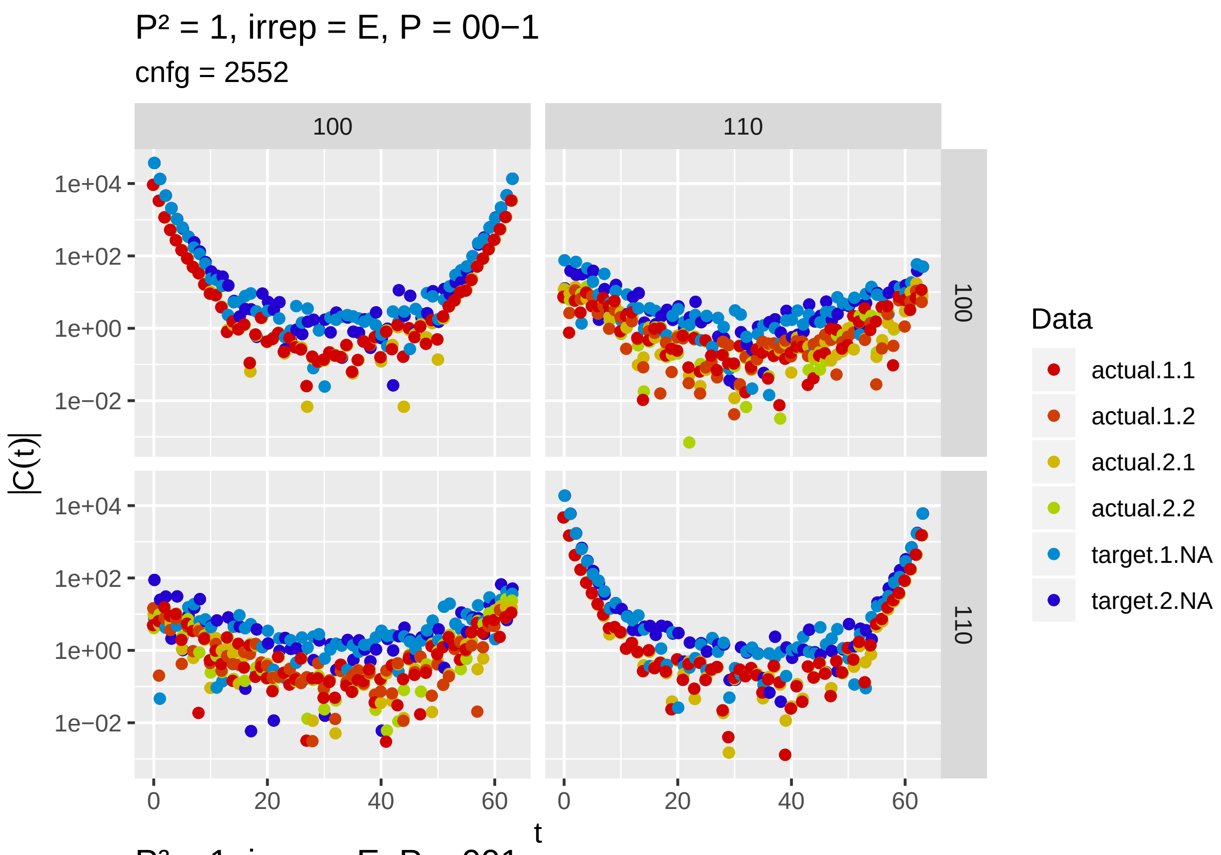
In the following we produce one comparison plot per correlator matrix, showing all the elements that could be matched up.

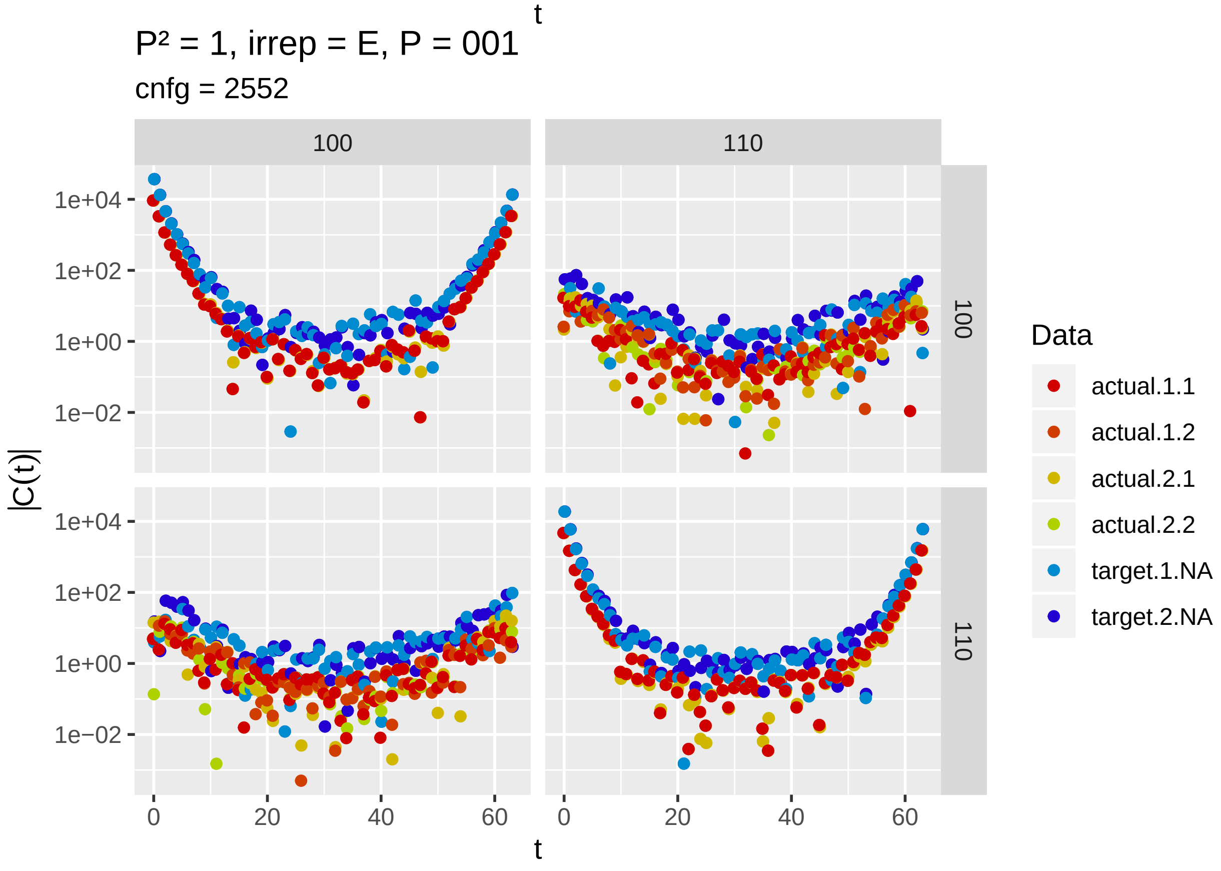
```r
make_plot <- function (.) {
  ggplot(
    .,
    aes(x = time,
        y = abs(correlator),
        color = color_key)) +
    geom_point(position = position_dodge(width = 0.3)) +
    scale_color_manual(values = both_colors) +
    scale_y_log10() +
    facet_grid(source_relative_momenta ~ sink_relative_momenta) +
    labs(title = sprintf('P² = %d, irrep = %s, P = %s',
                         .$total_momentum_sq[1],
                         .$irrep[1],
                         .$total_momentum_str[1]),
         subtitle = sprintf('cnfg = %d',
                                   .$config_number[1]),
         x = 't',
         y = expression(abs(C(t))),
         color = 'Data',
         shape = expression("irrep row" ~ alpha))
}

plots <- ls2 %>%
  arrange(total_momentum_sq, irrep, total_momentum_str) %>%
  group_by(total_momentum_sq, irrep, total_momentum_str) %>%
  do(plot = make_plot(.))

for (plot in plots$plot) {
  print(plot)
  cat('\n')
}
```




P² = 1, irrep = A1, P = 00−1
cnfg = 2552
00−1
000
100
110
|C(t)|
1e+05
1e+02
1e−01
0
20
40
60
t
Data
actual.1.1
target.1.NA

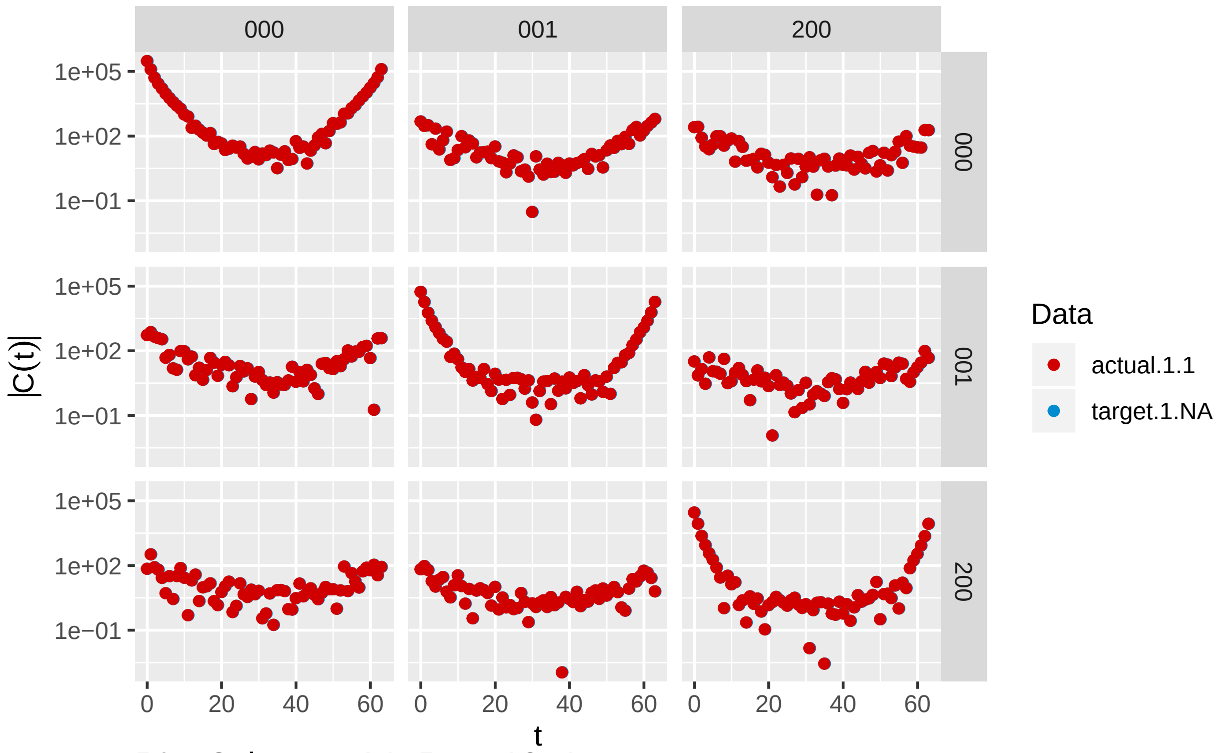

P² = 1, irrep = A1, P = 001
cnfg = 2552
00−1
000
100
110
|C(t)|
1e+06
1e+03
1e+00
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 1, irrep = E, P = 00−1
cnfg = 2552
100
110
100
110
|C(t)|
1e+04
1e+02
1e+00
1e−02
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 1, irrep = E, P = 001
cnfg = 2552
100
110
100
110
|C(t)|
1e+04
1e+02
1e+00
1e−02
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

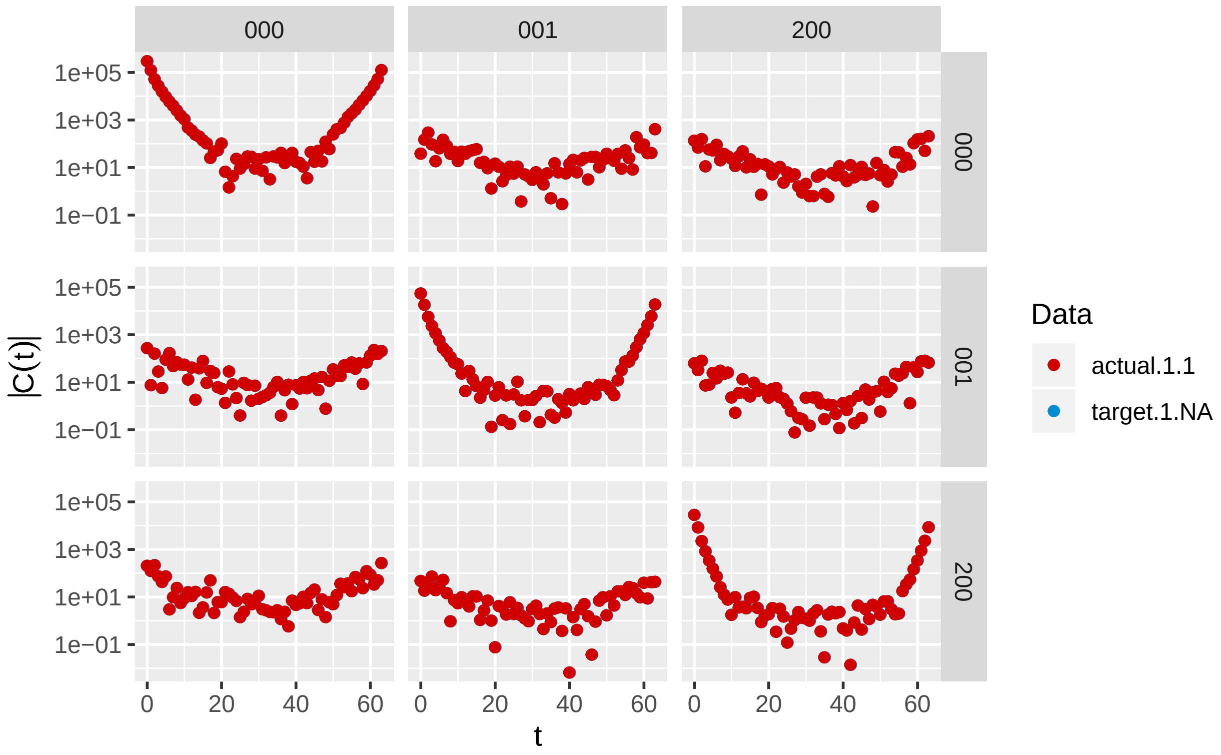
P² = 2, irrep = A1, P = −1−10

cnfg = 2552



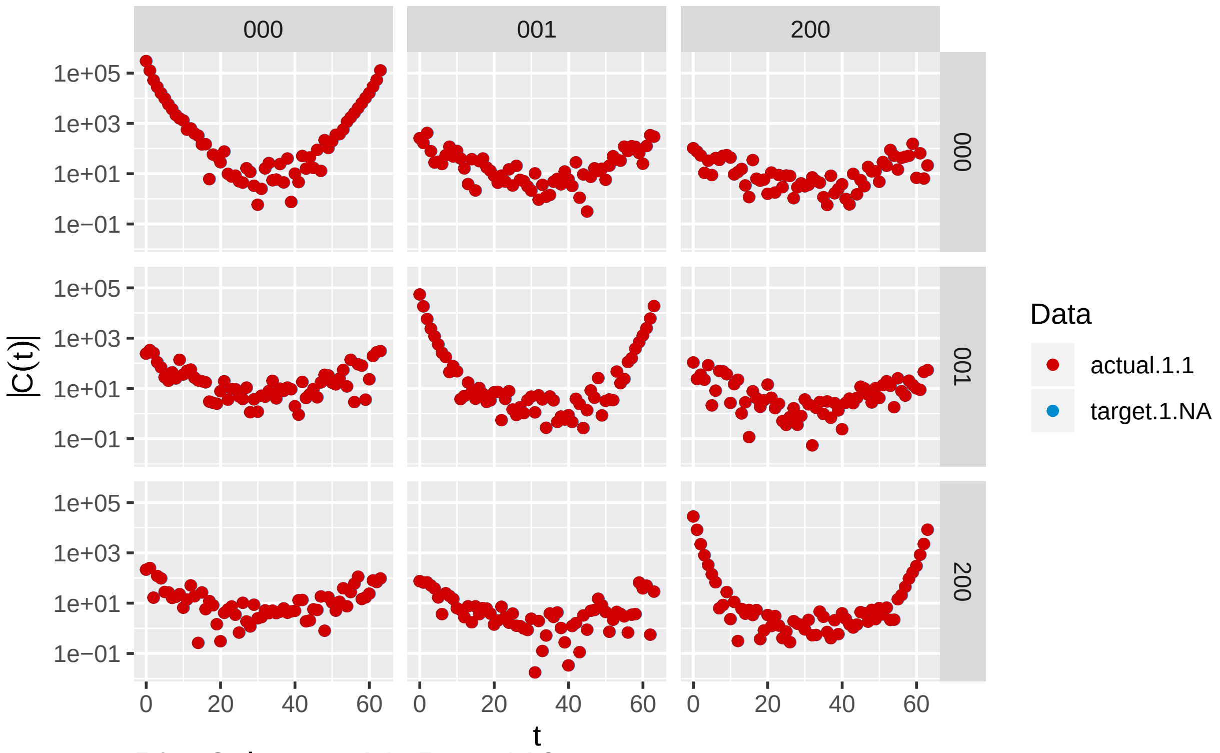
P² = 2, irrep = A1, P = −10−1

cnfg = 2552



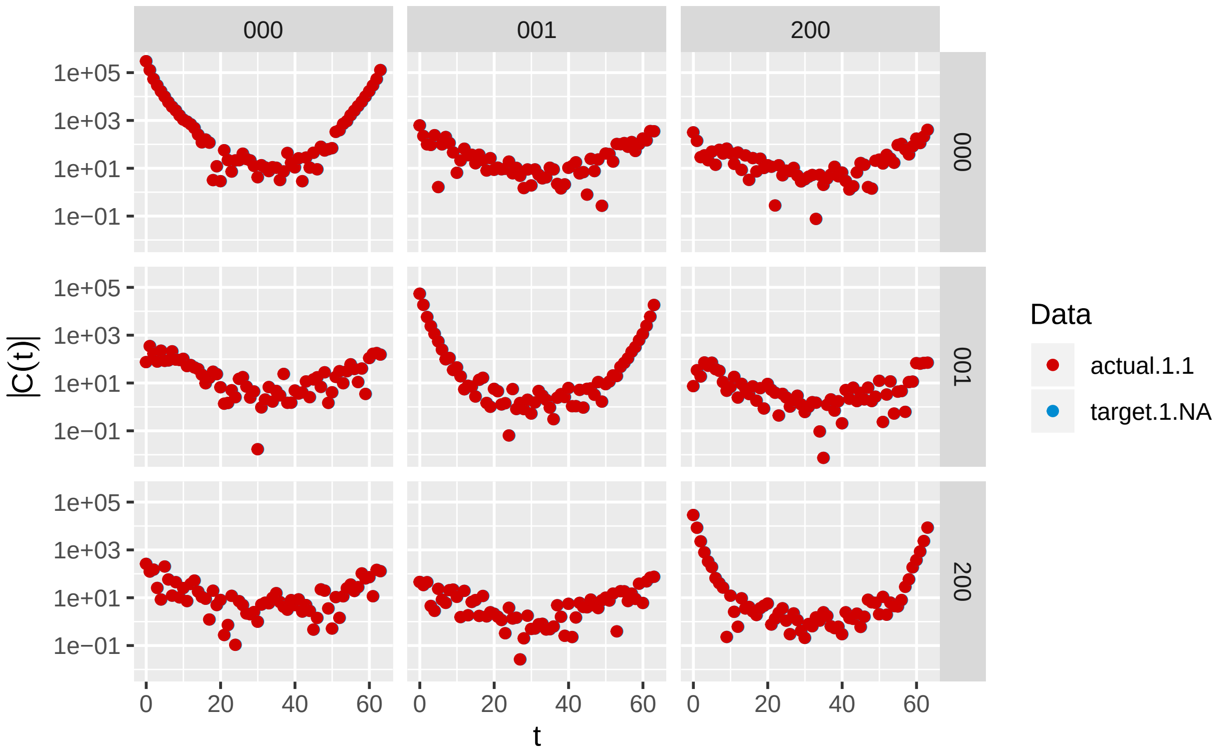
P² = 2, irrep = A1, P = −101

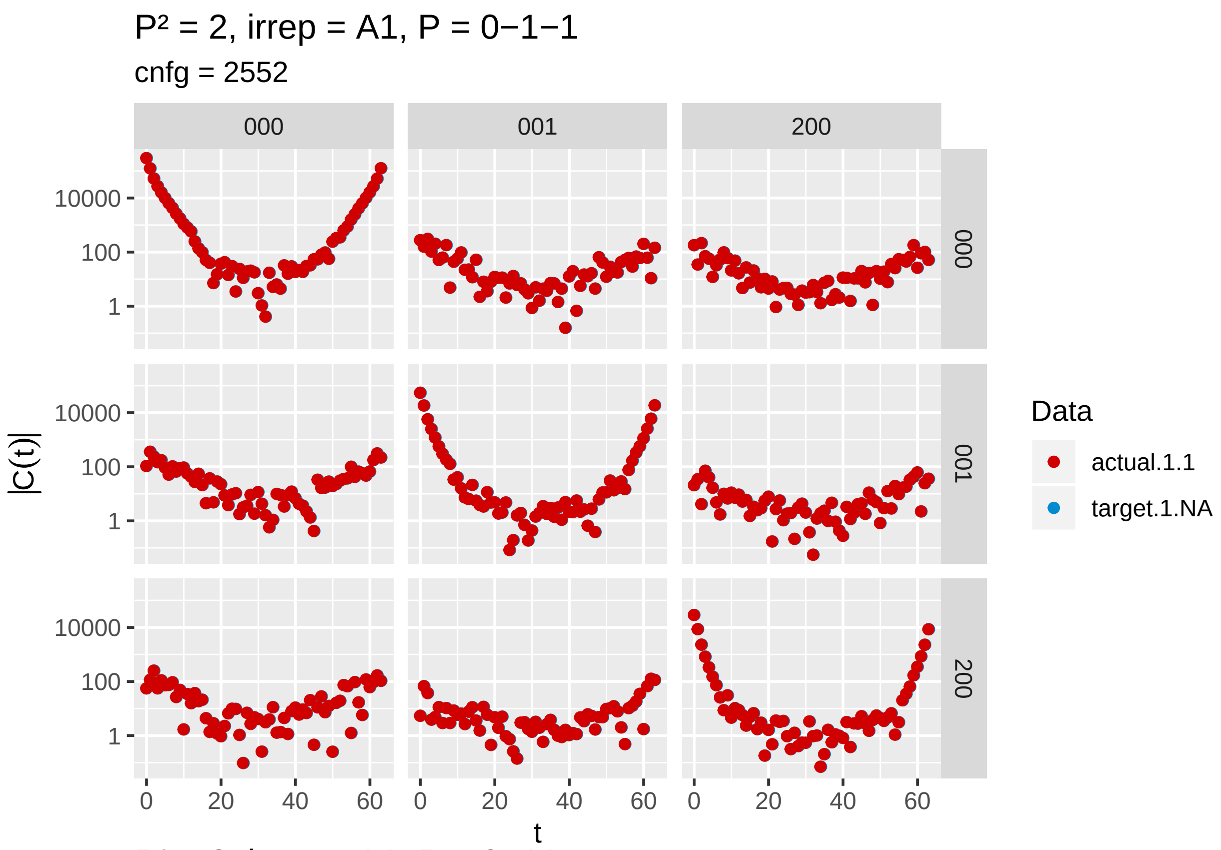
cnfg = 2552

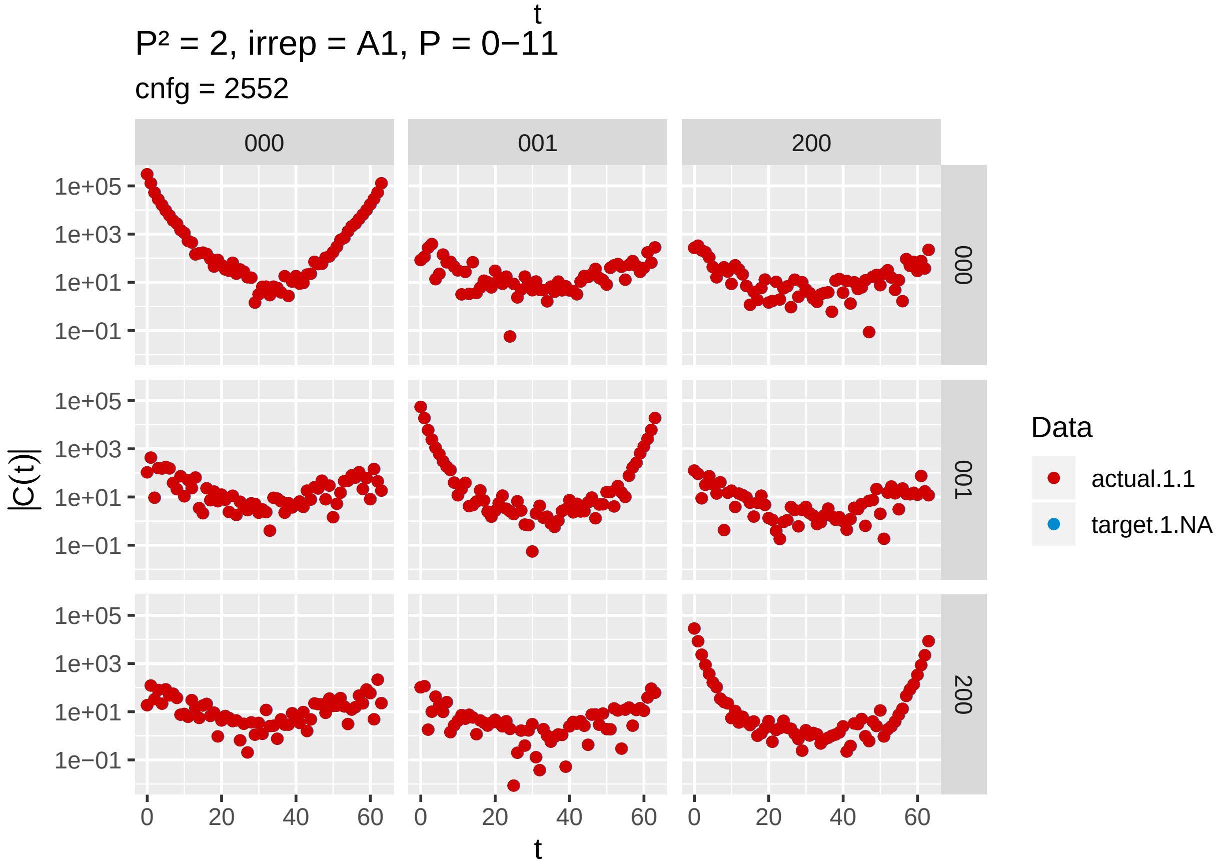


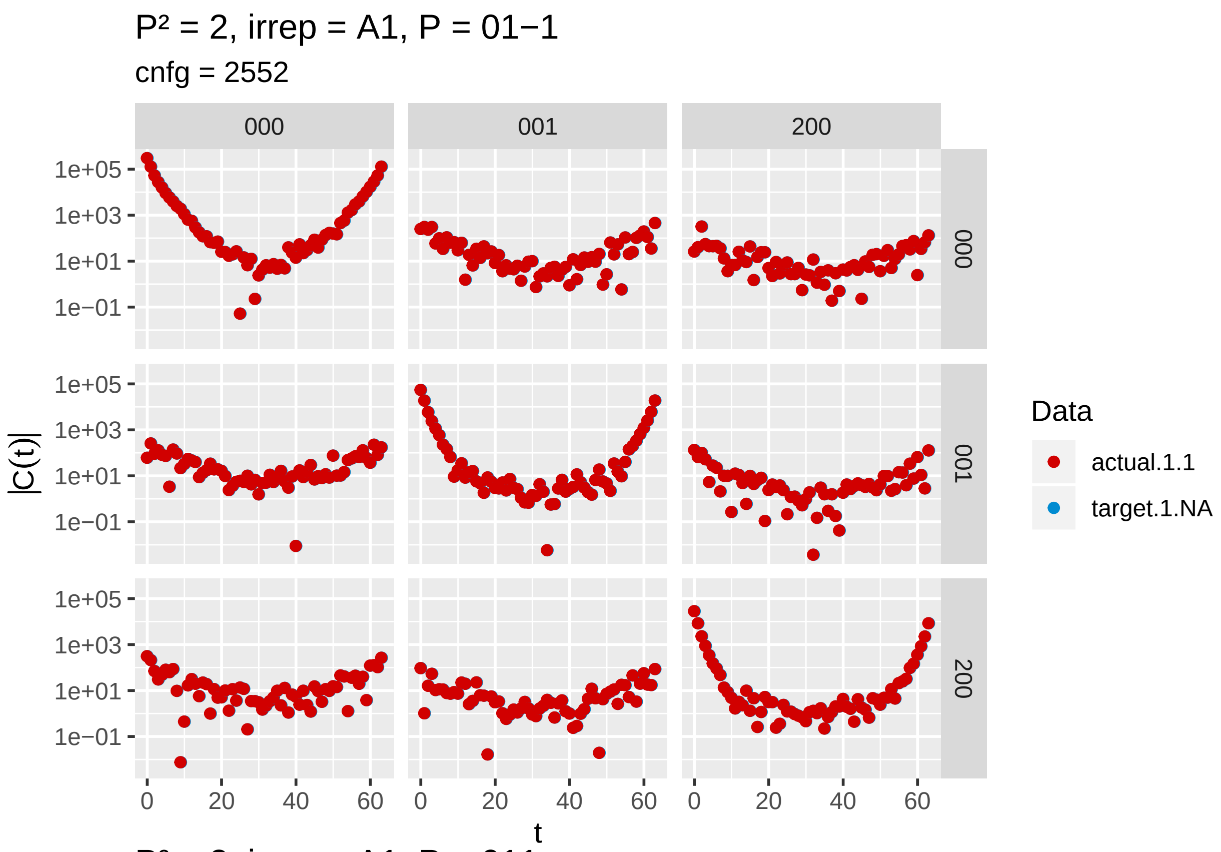
P² = 2, irrep = A1, P = −110

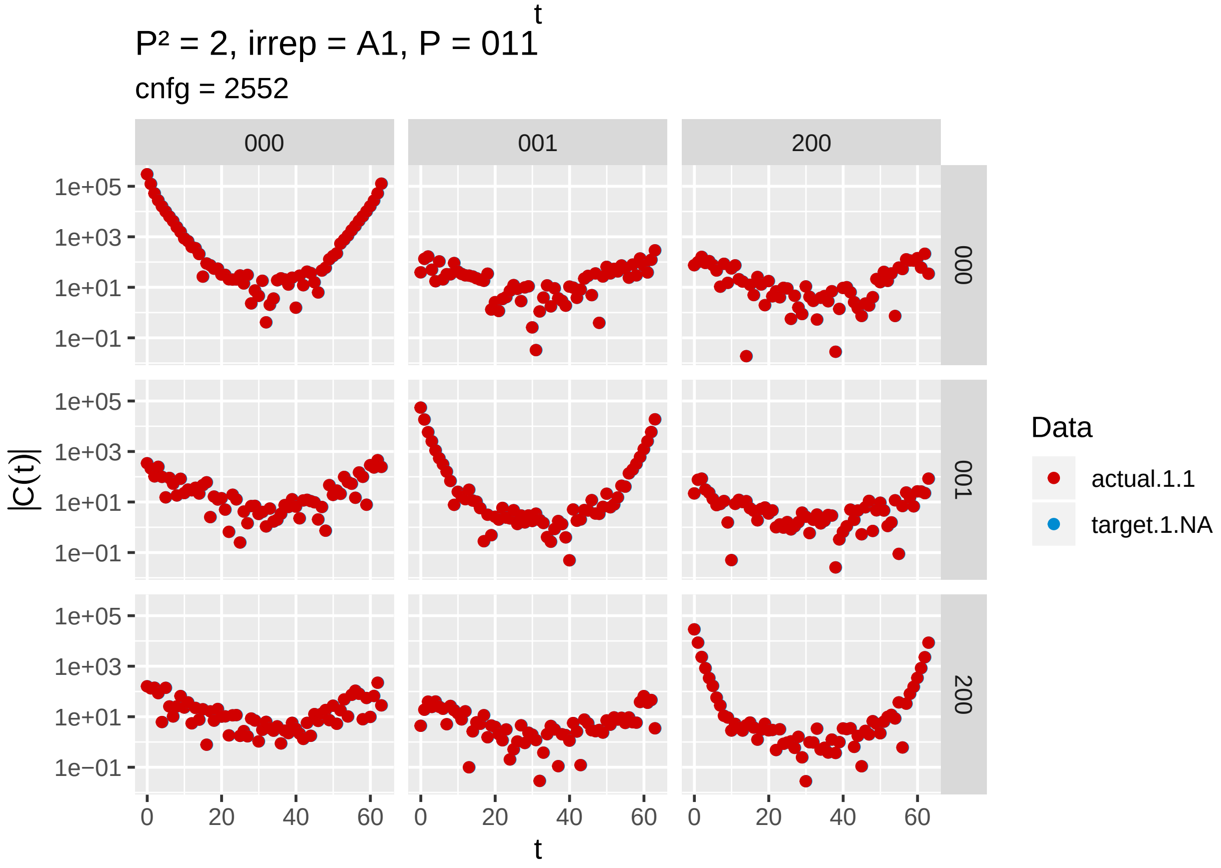
cnfg = 2552

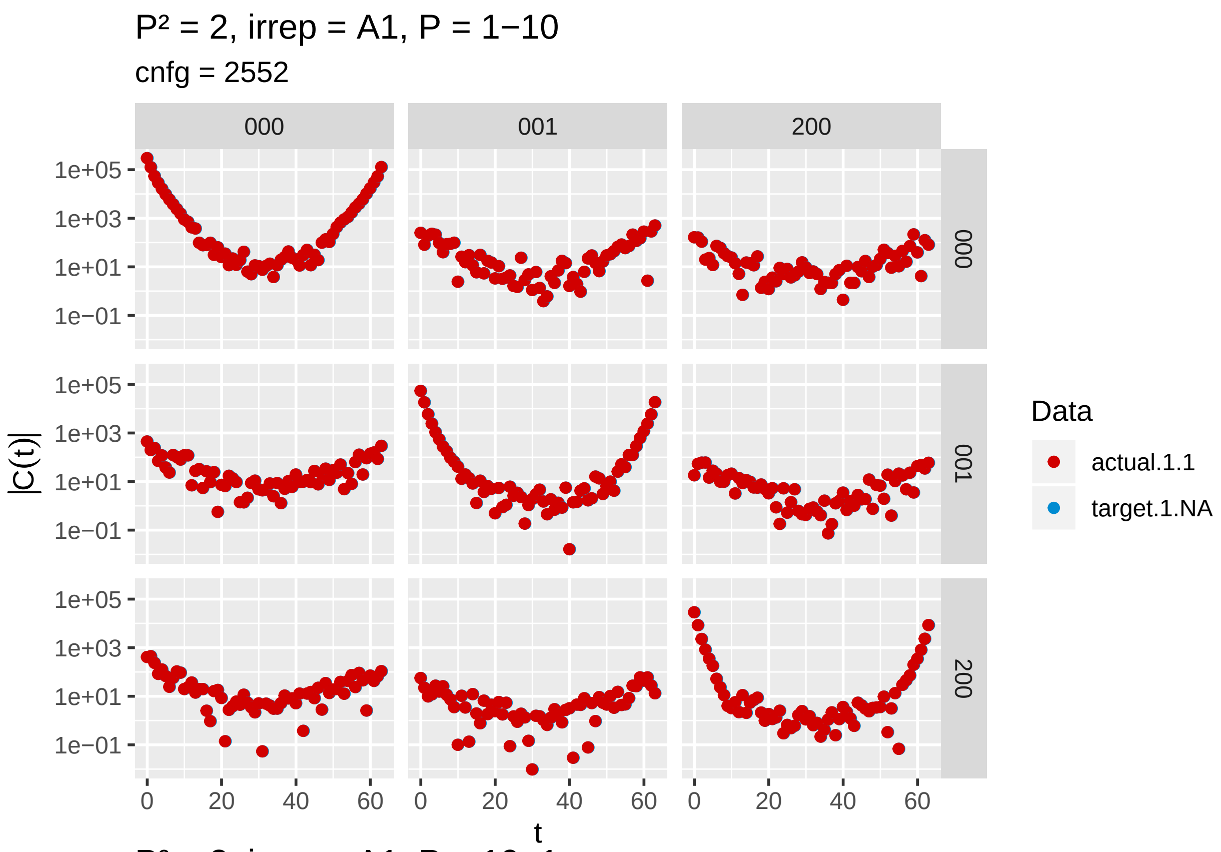



P² = 2, irrep = A1, P = 0−1−1
cnfg = 2552
000
001
200
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

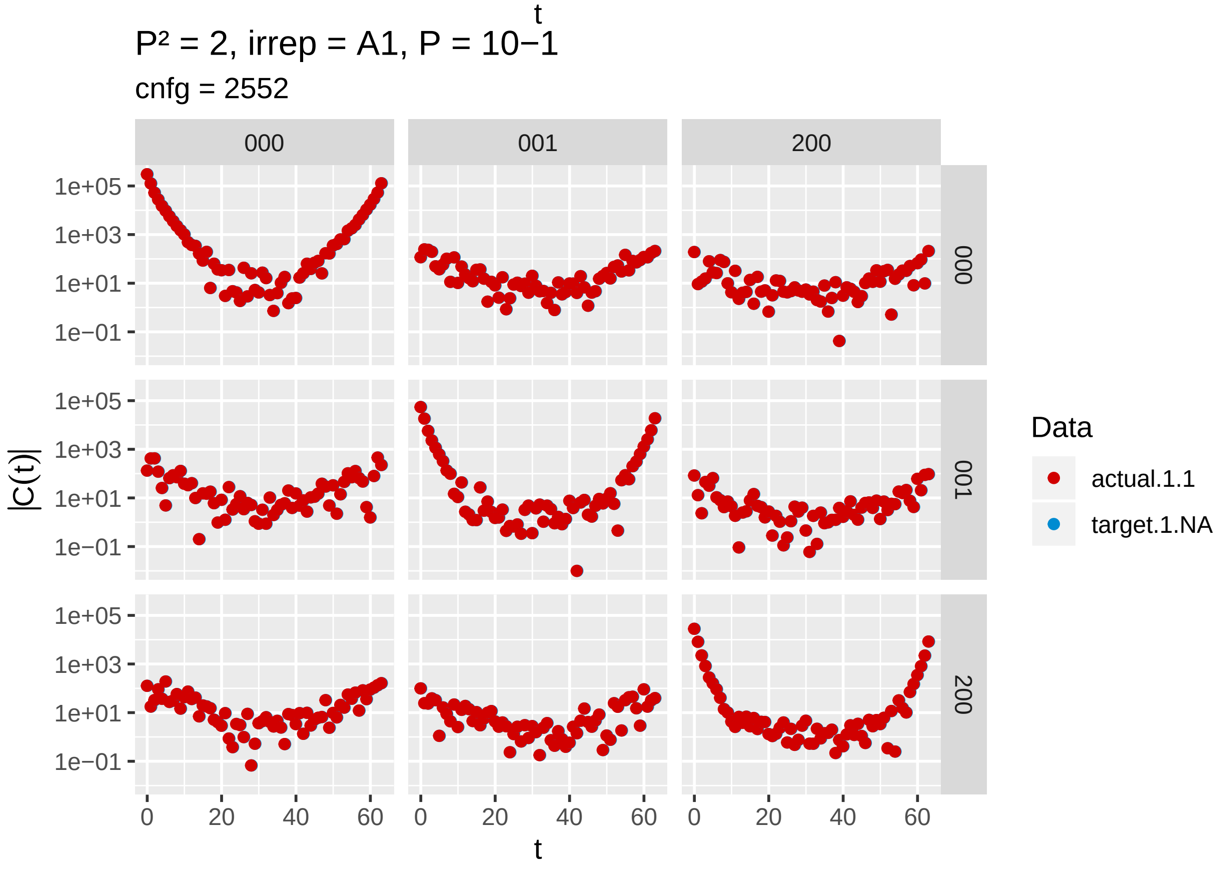

P² = 2, irrep = A1, P = 0−11
cnfg = 2552
000
001
200
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = A1, P = 01−1
cnfg = 2552
000
001
200
|C(t)|
1e+05
1e+03
1e+01
1e−01
0
20
40
60
t
Data
actual.1.1
target.1.NA

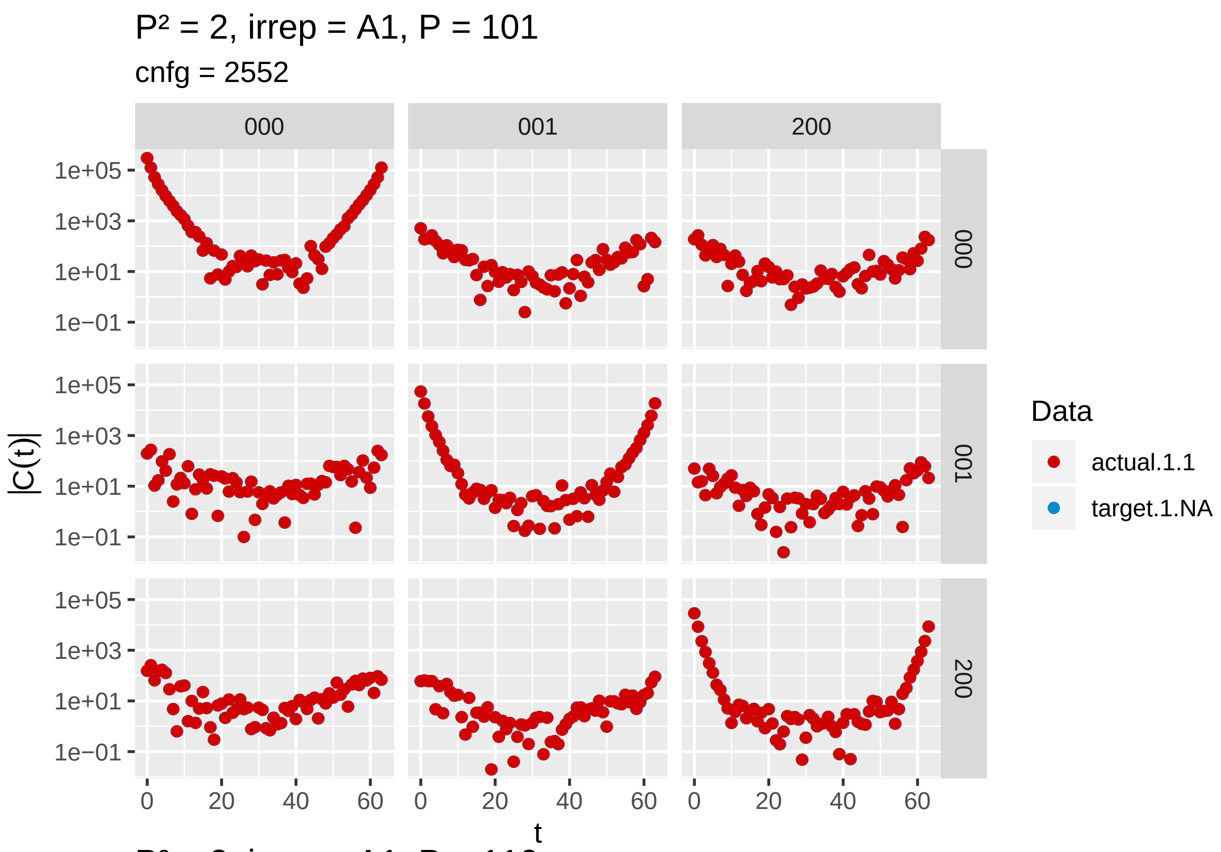

P² = 2, irrep = A1, P = 011
cnfg = 2552
000
001
200
|C(t)|
1e+05
1e+03
1e+01
1e−01
0
20
40
60
t
Data
actual.1.1
target.1.NA

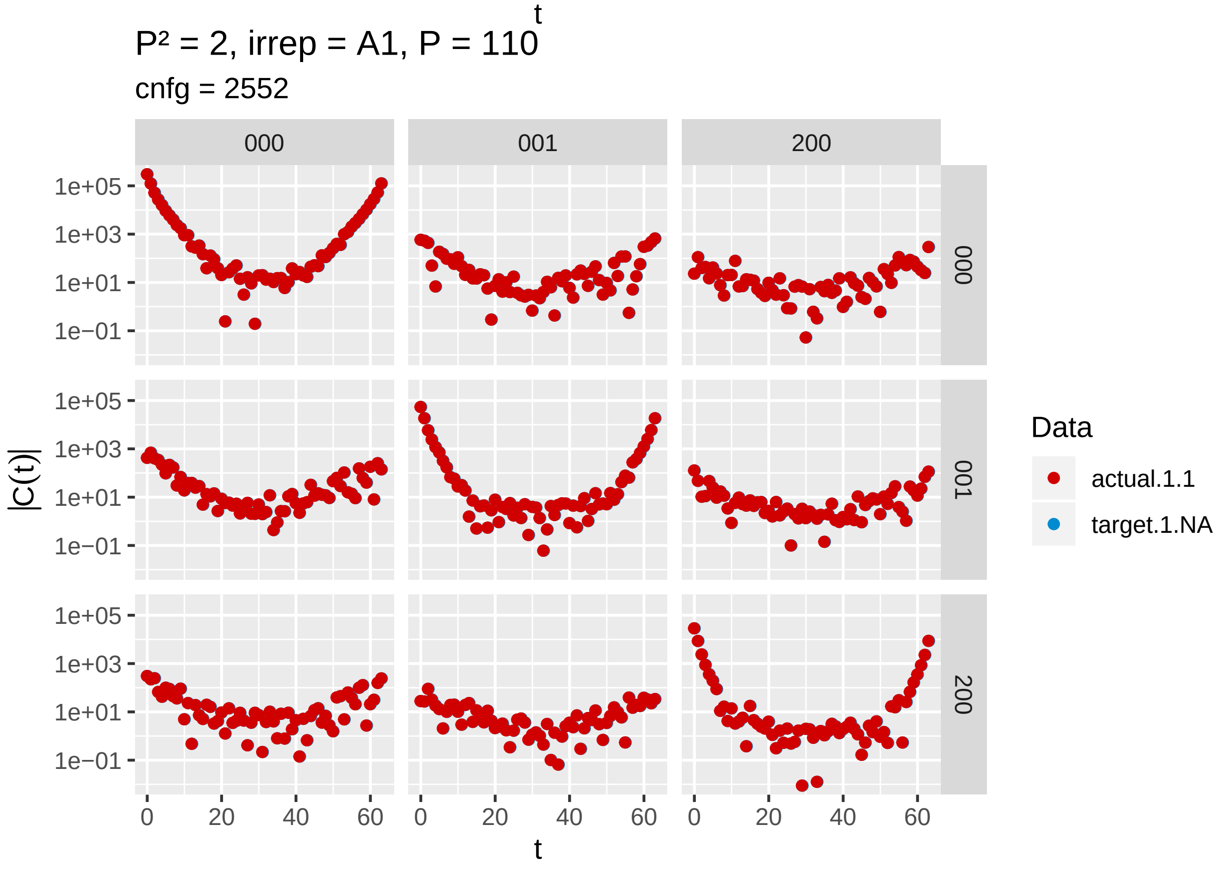


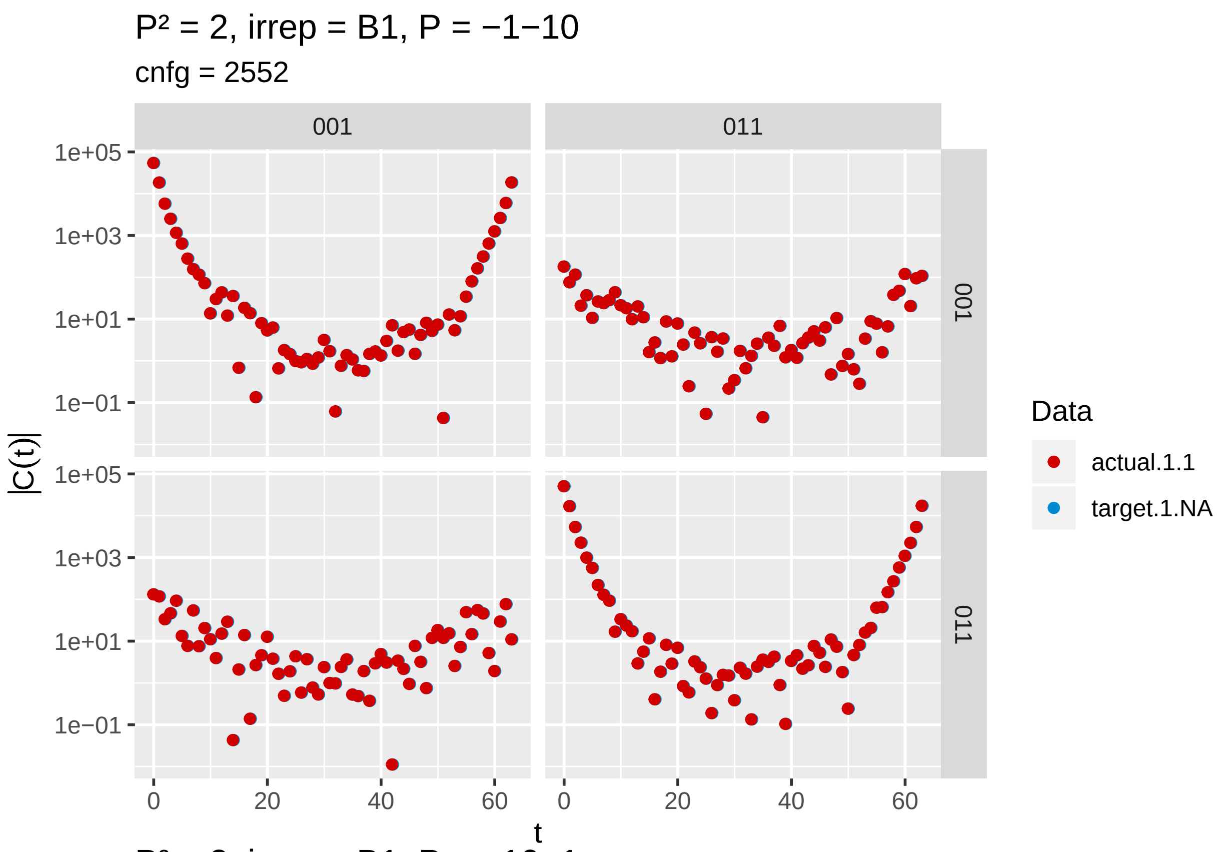
P² = 2, irrep = A1, P = 1−10
cnfg = 2552
000
001
200
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA



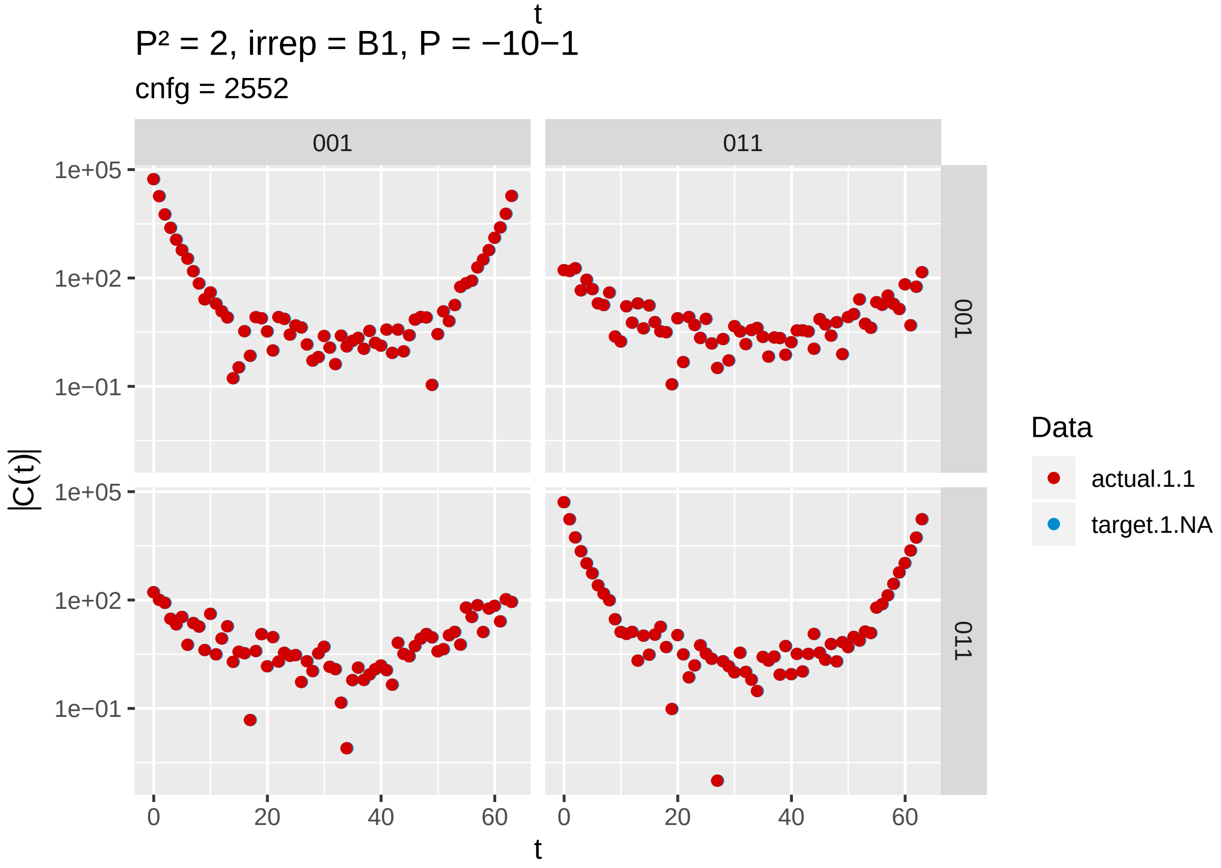
P² = 2, irrep = A1, P = 10−1
cnfg = 2552
000
001
200
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = A1, P = 101
cnfg = 2552
000
001
200
000
001
200
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA

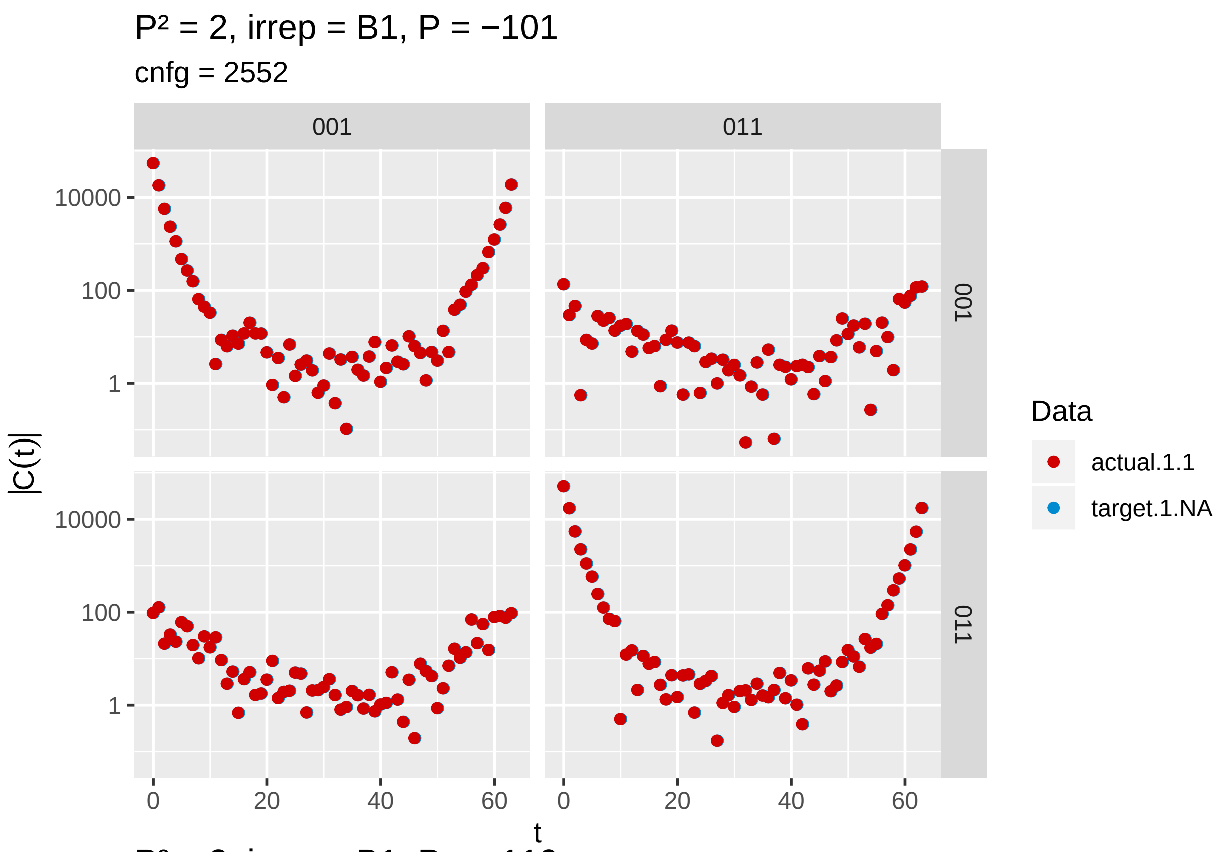

P² = 2, irrep = A1, P = 110
cnfg = 2552
000
001
200
000
001
200
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA

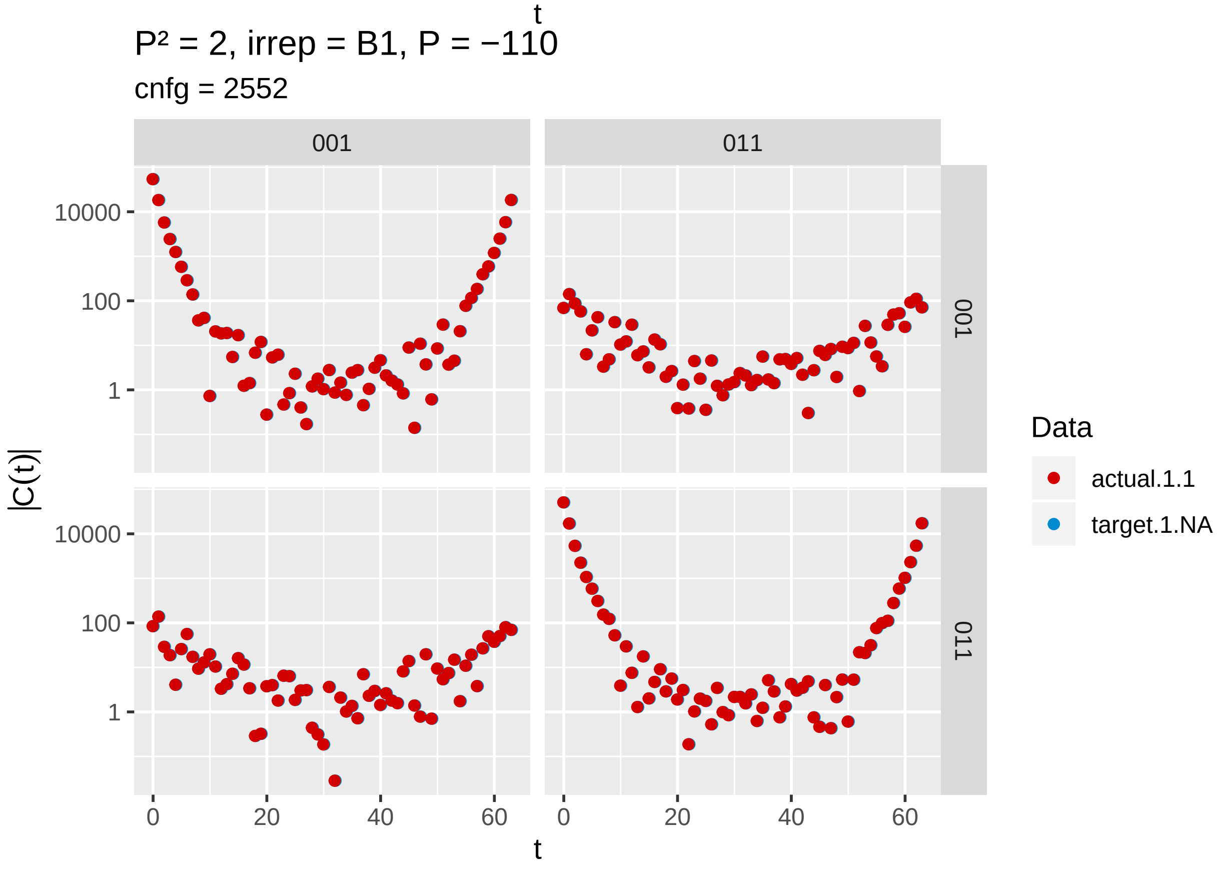


P² = 2, irrep = B1, P = −1−10
cnfg = 2552
001
011
001
011
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA



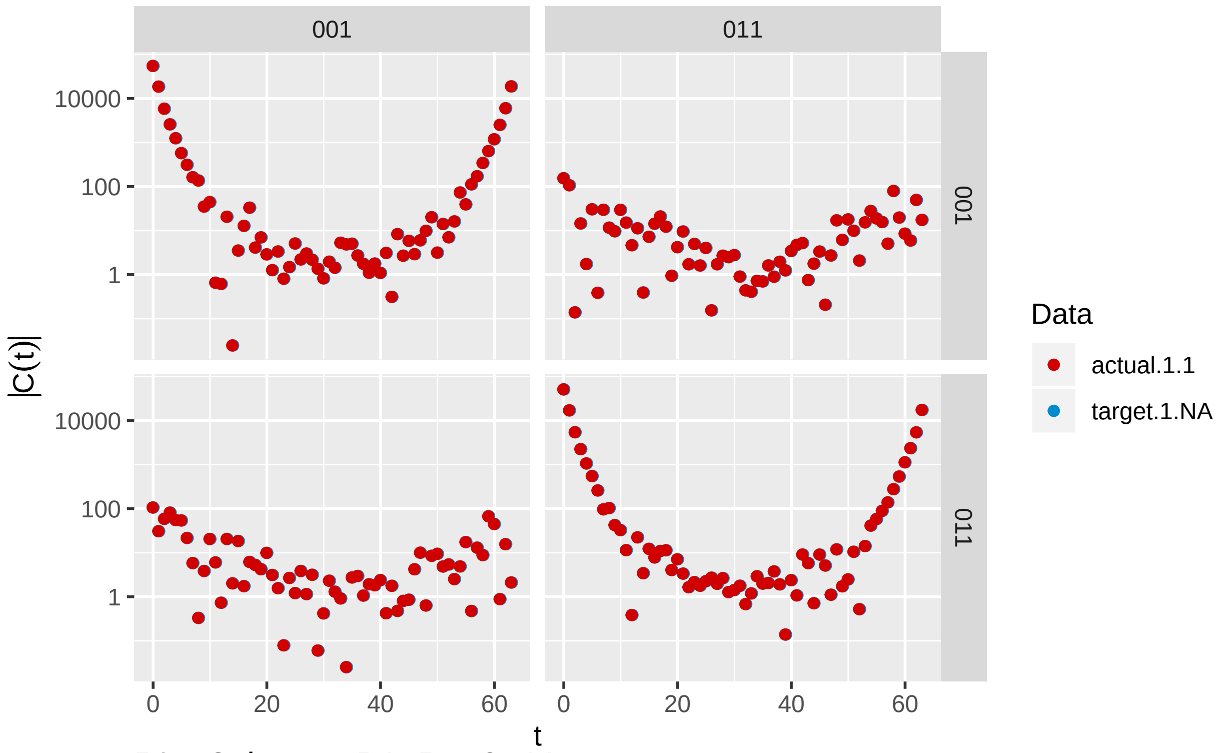
P² = 2, irrep = B1, P = −10−1
cnfg = 2552
001
011
001
110
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B1, P = −101
cnfg = 2552
001
011
001
011
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B1, P = −110
cnfg = 2552
001
011
001
011
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

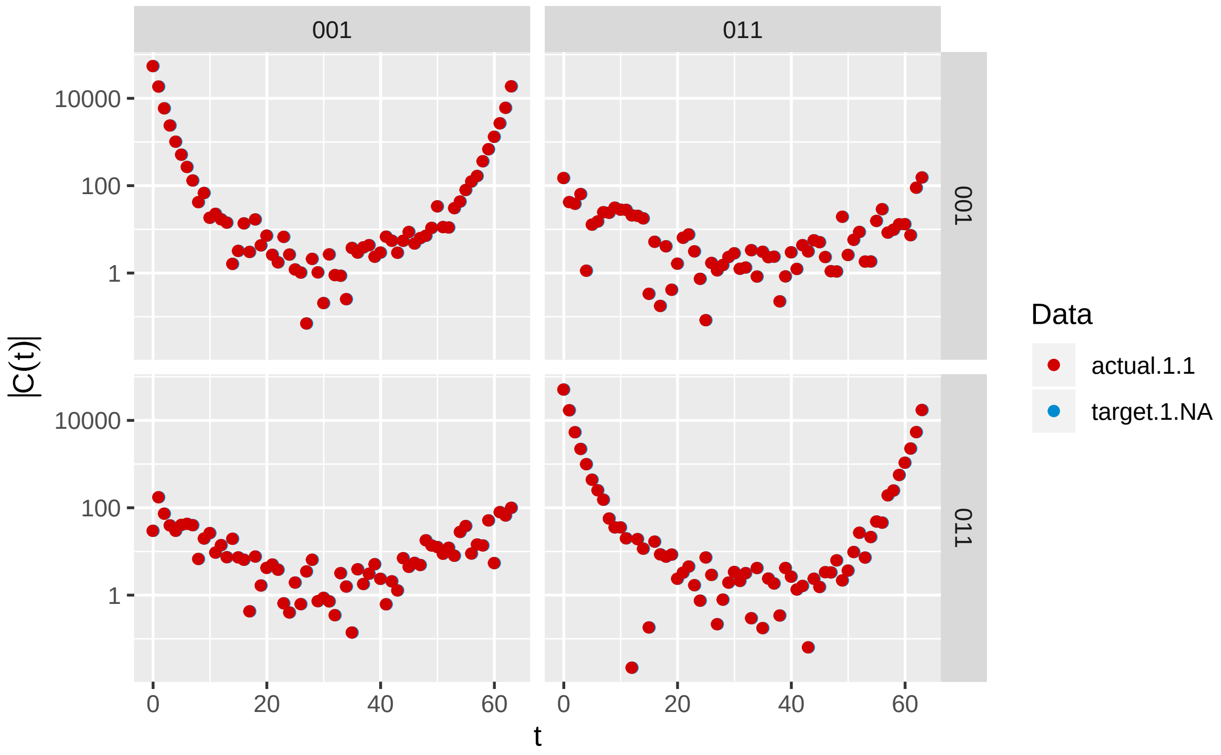
P² = 2, irrep = B1, P = 0−1−1

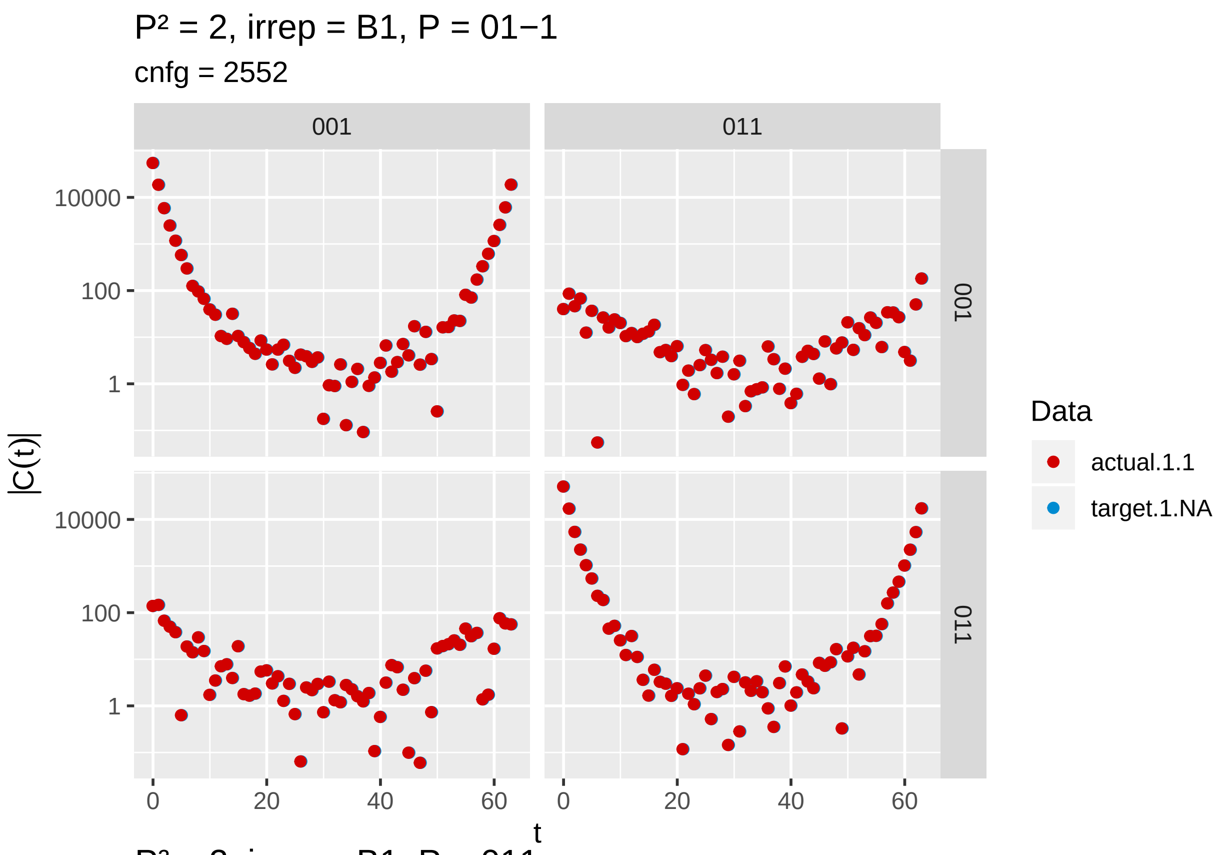
cnfg = 2552



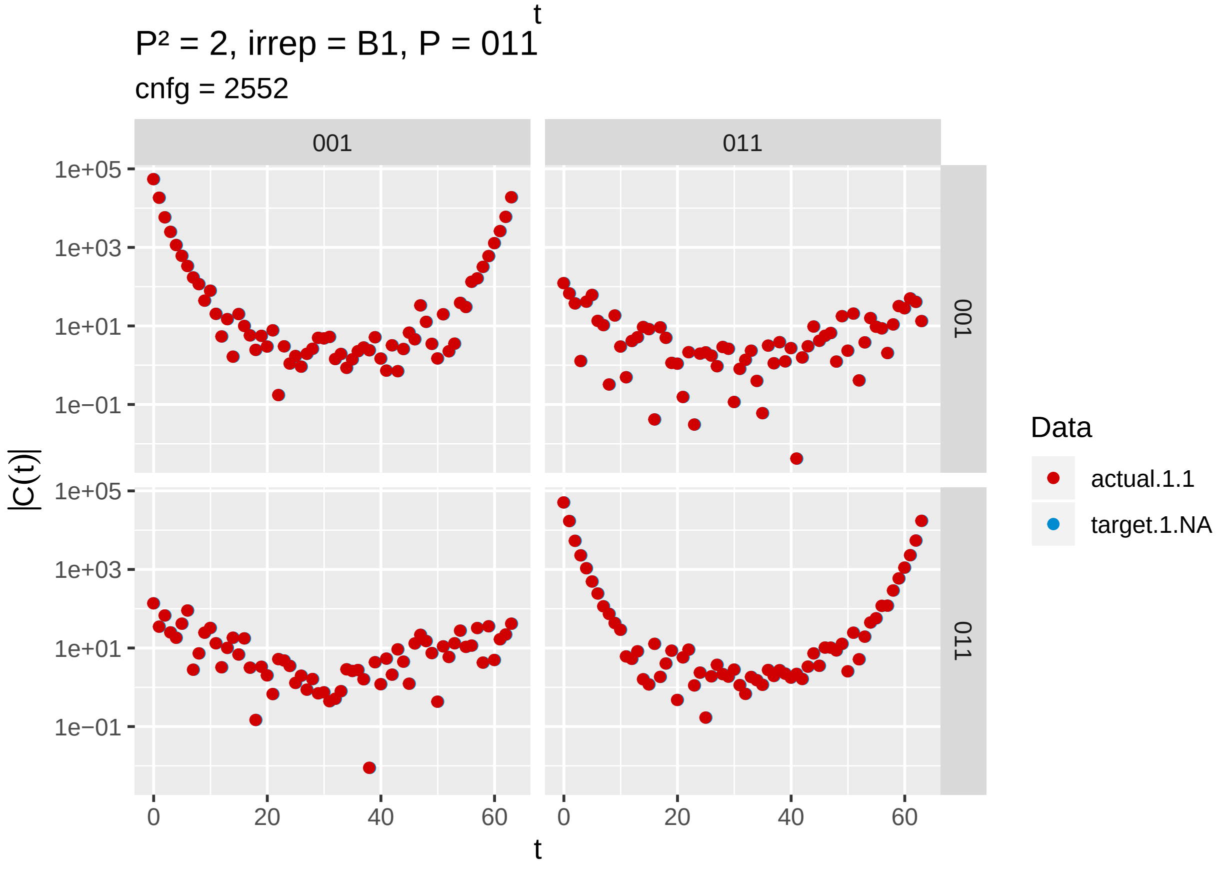
P² = 2, irrep = B1, P = 0−11

cnfg = 2552

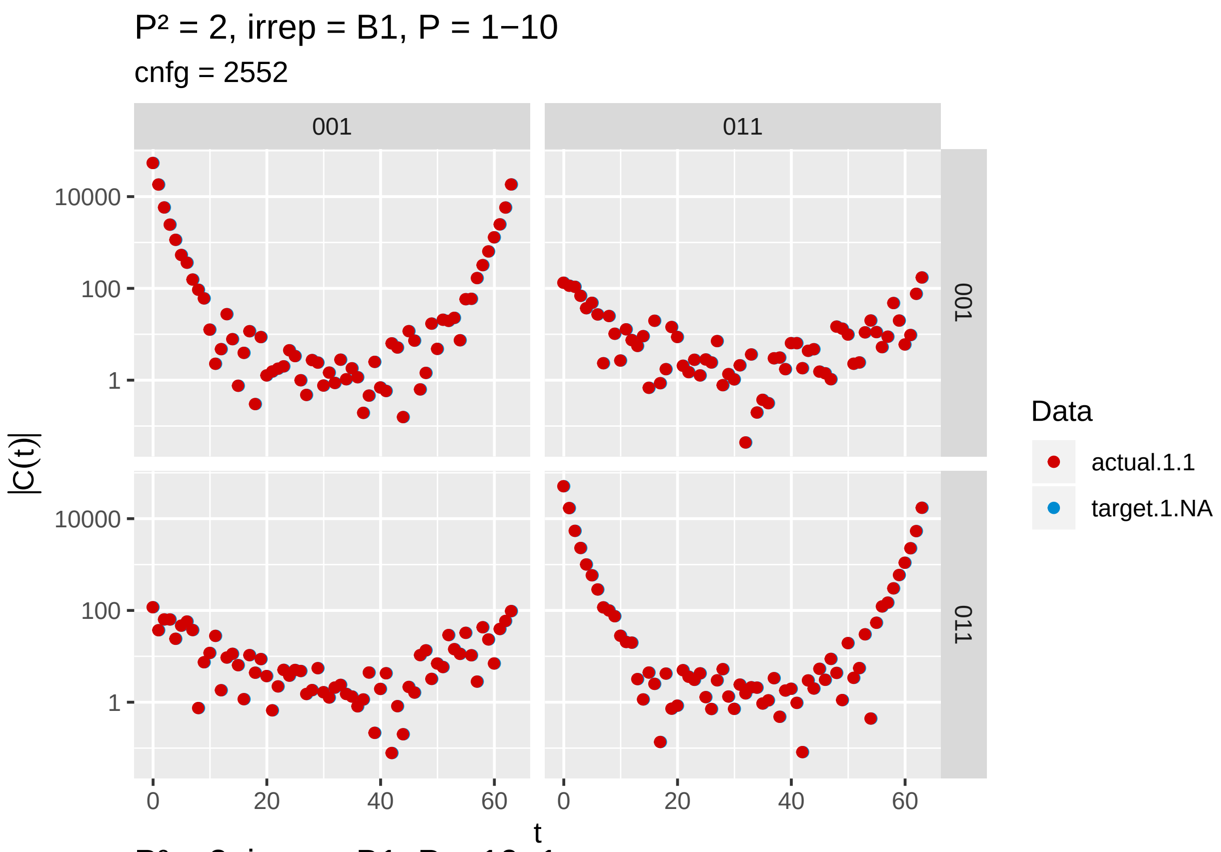


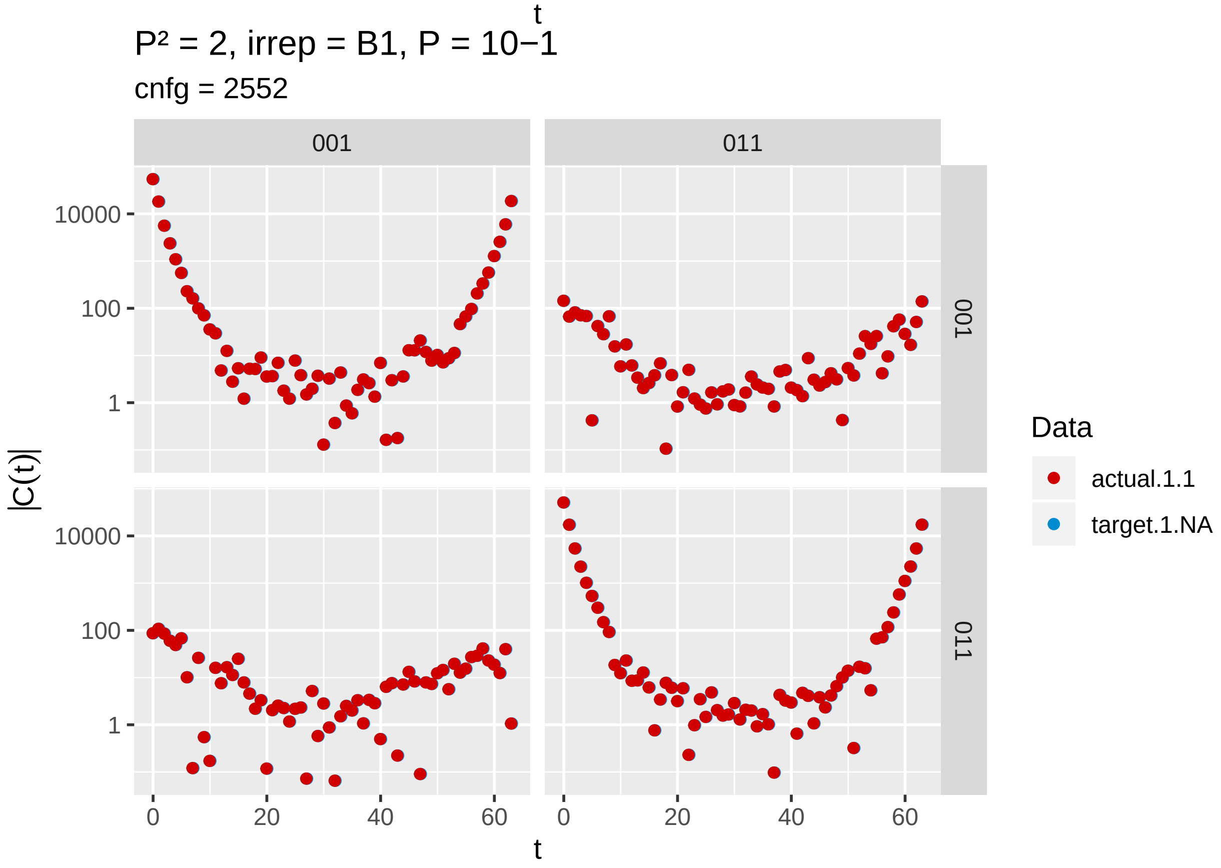


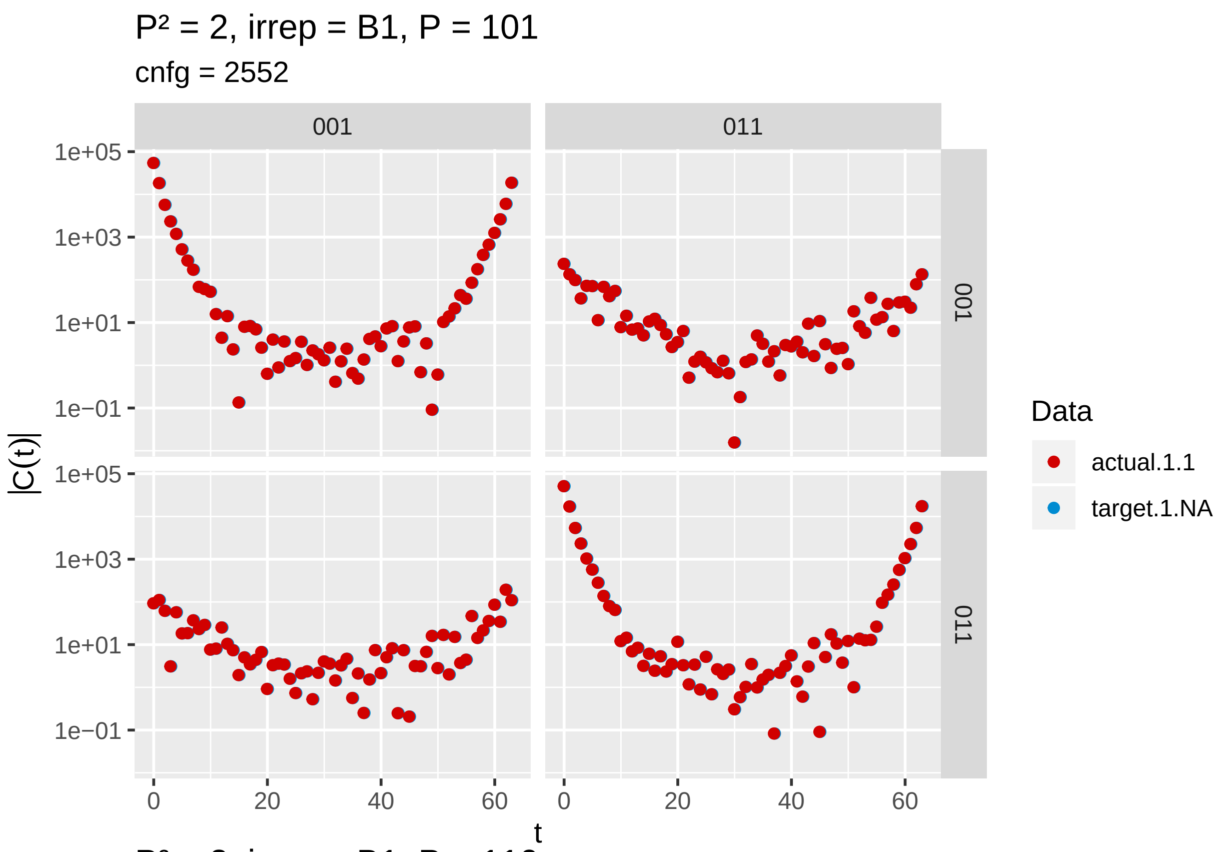
P² = 2, irrep = B1, P = 01−1
cnfg = 2552
001
011
001
011
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA



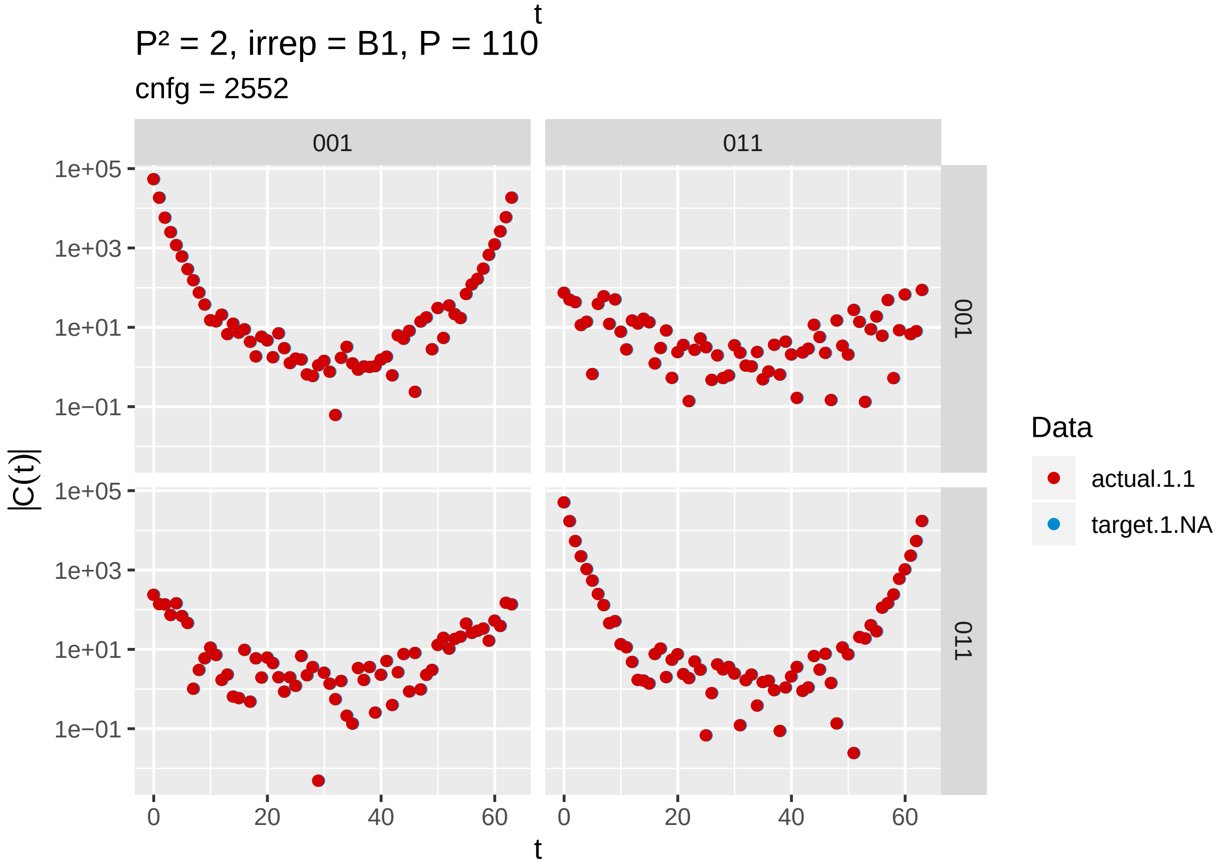
P² = 2, irrep = B1, P = 011
cnfg = 2552
001
011
001
011
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B1, P = 1−10
cnfg = 2552
001
011
001
011
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

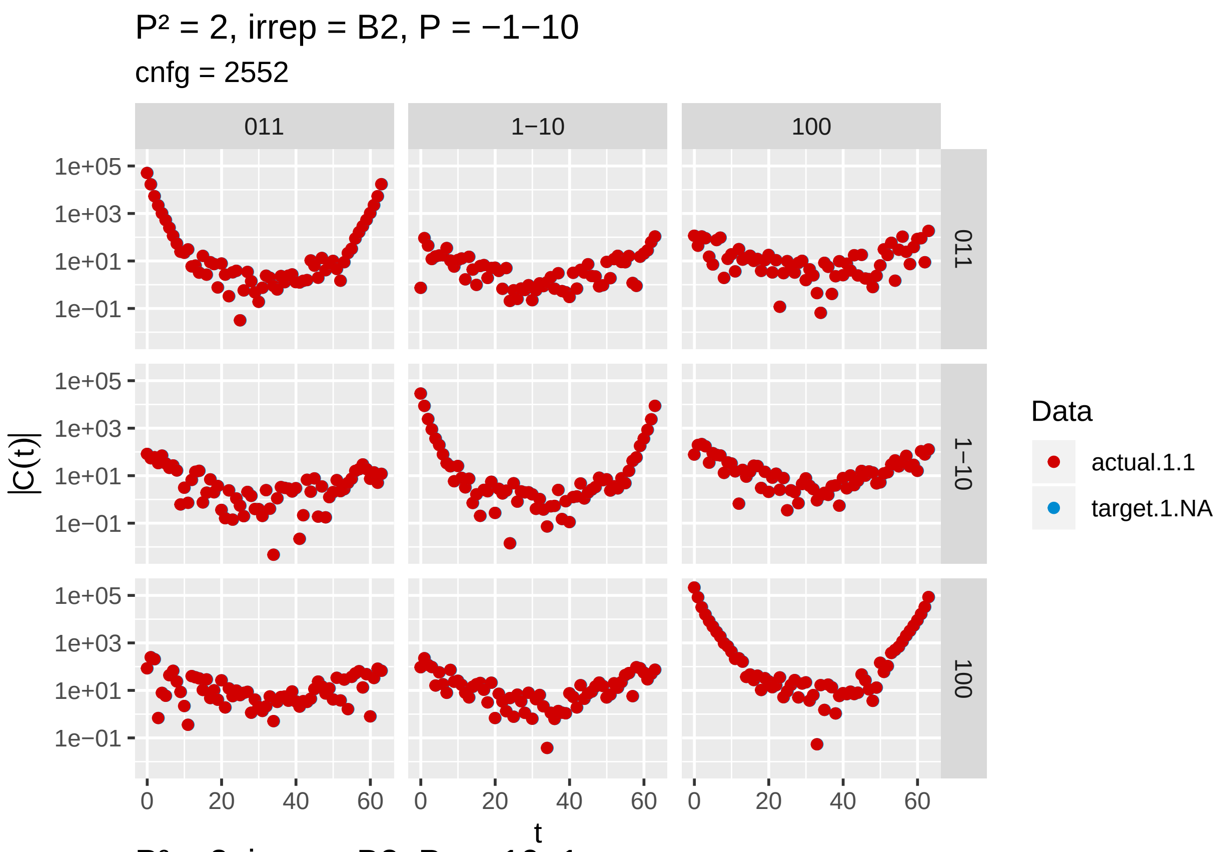

P² = 2, irrep = B1, P = 10−1
cnfg = 2552
001
011
001
011
10000
100
1
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

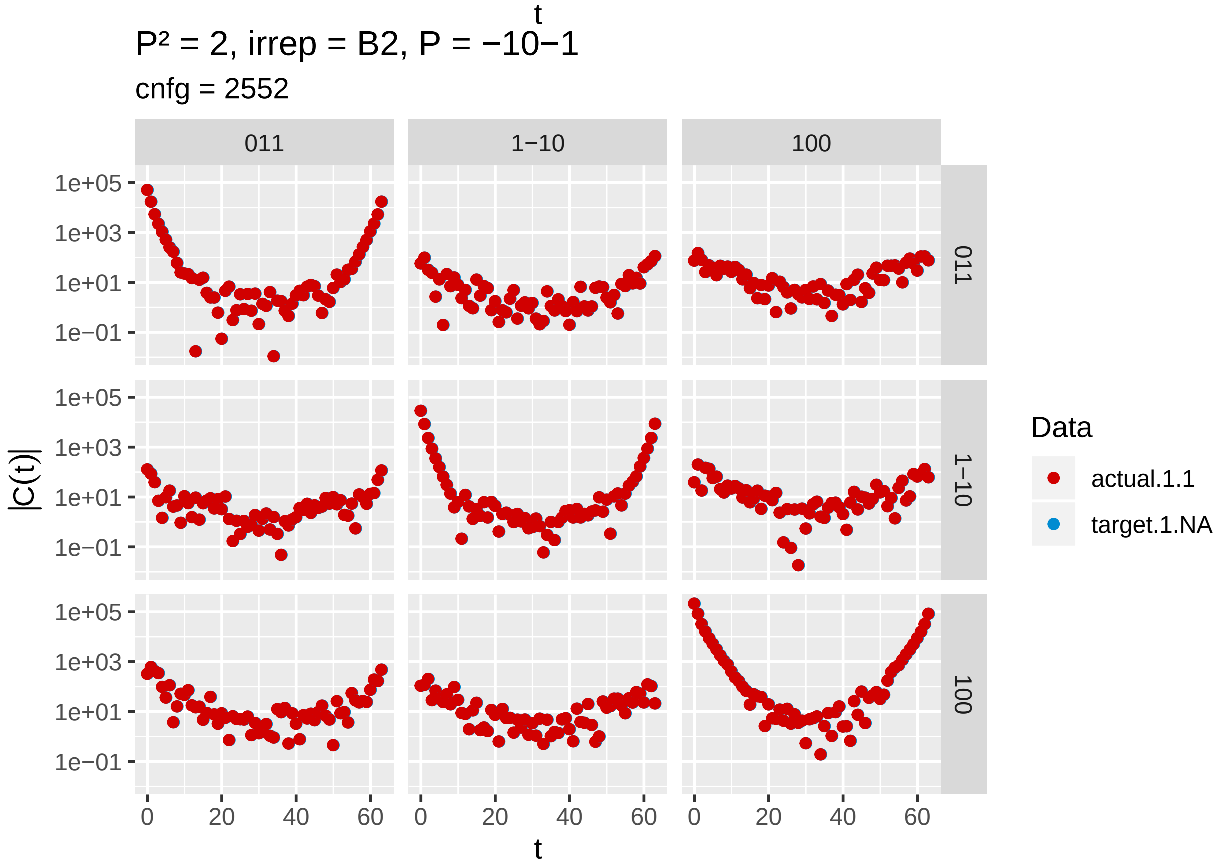


P² = 2, irrep = B1, P = 101
cnfg = 2552
001
011
001
011
|C(t)|
1e+05
1e+03
1e+01
1e−01
0
20
40
60
t
Data
actual.1.1
target.1.NA



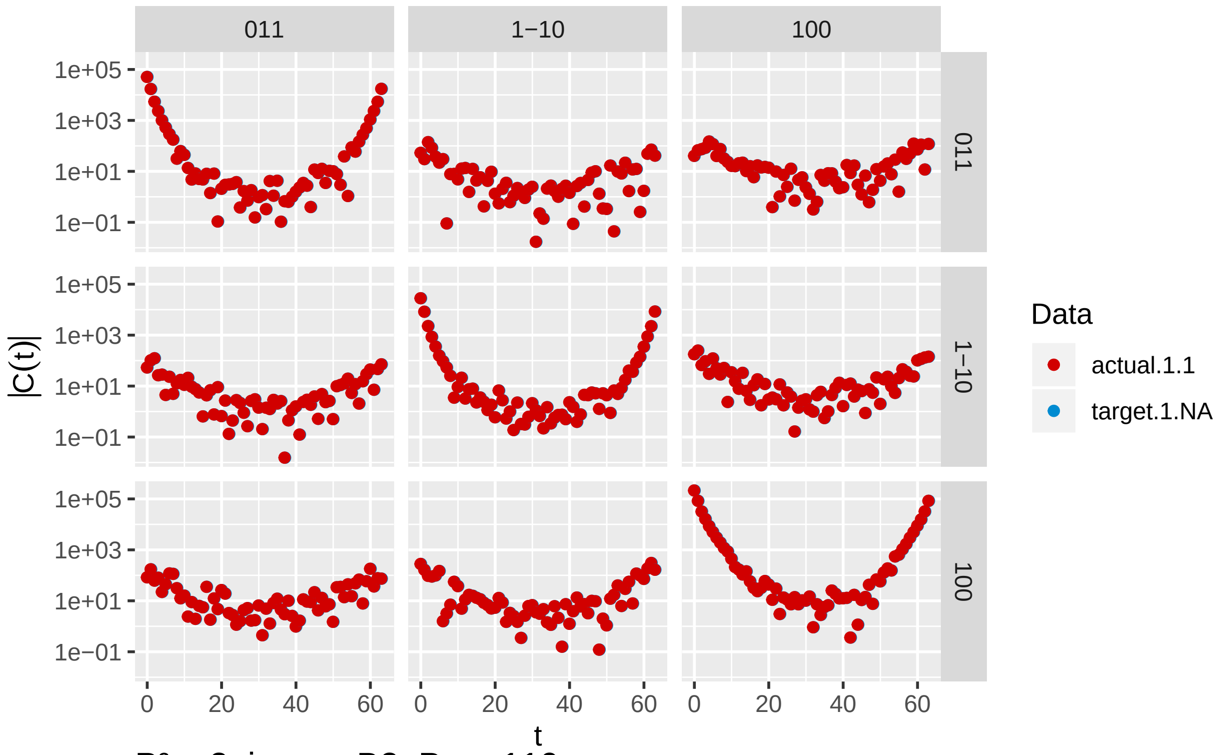
P² = 2, irrep = B1, P = 110
cnfg = 2552
001
011
001
011
|C(t)|
1e+05
1e+03
1e+01
1e−01
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B2, P = −1−10
cnfg = 2552
011
1−10
100
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA


P² = 2, irrep = B2, P = −10−1
cnfg = 2552
011
1−10
100
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA

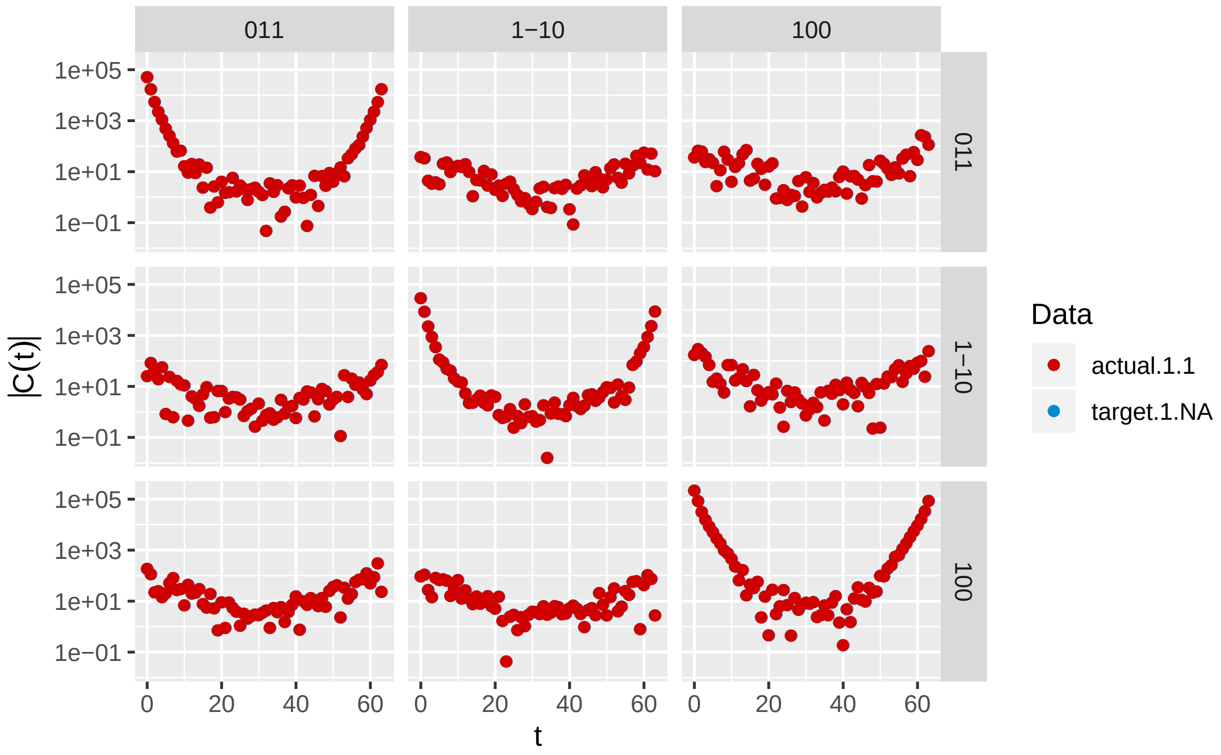
P² = 2, irrep = B2, P = −101

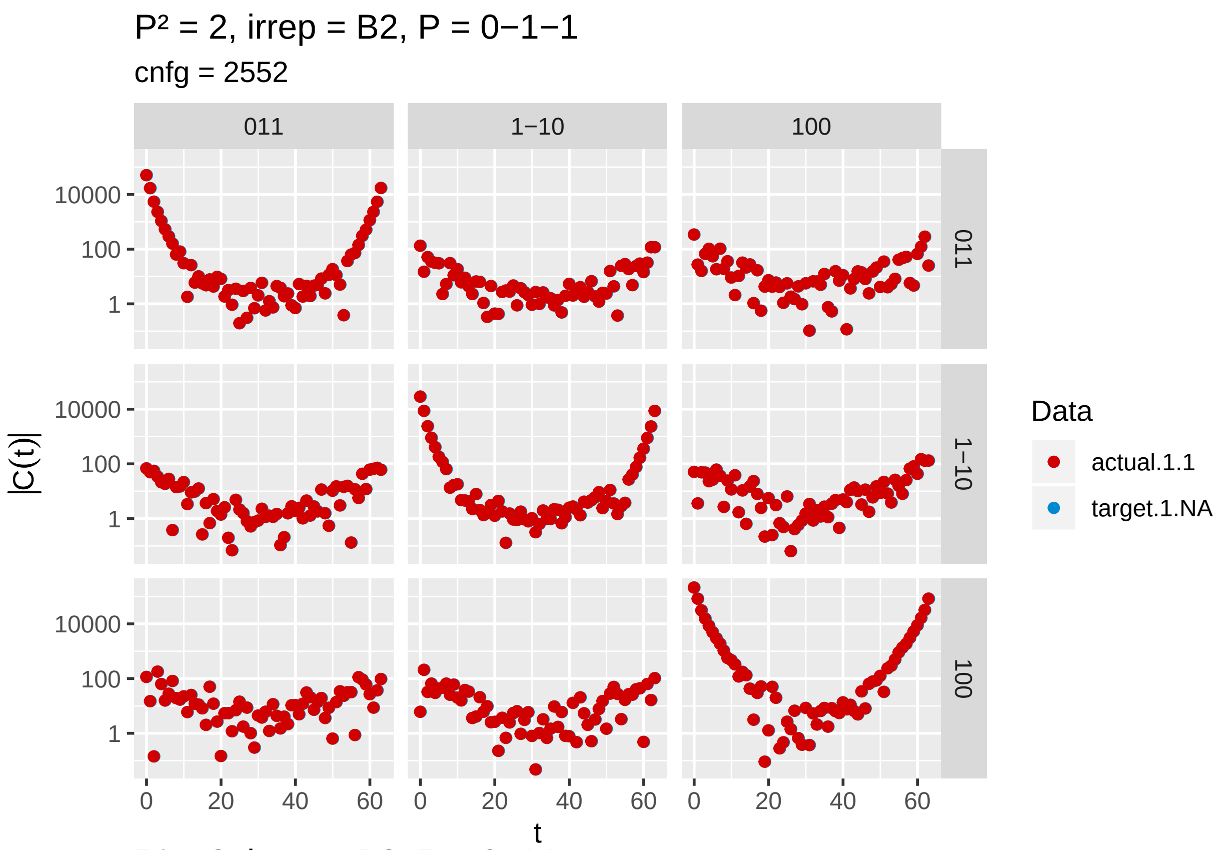
cnfg = 2552

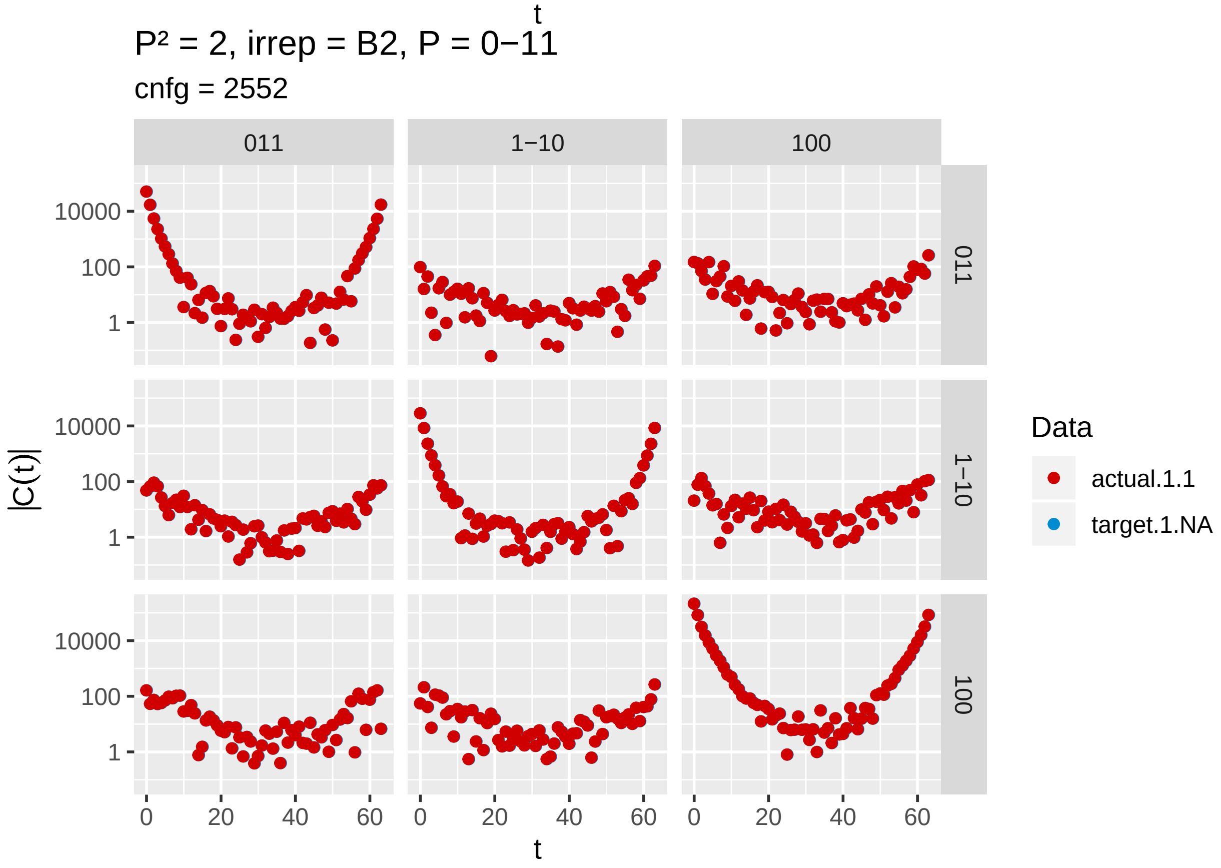


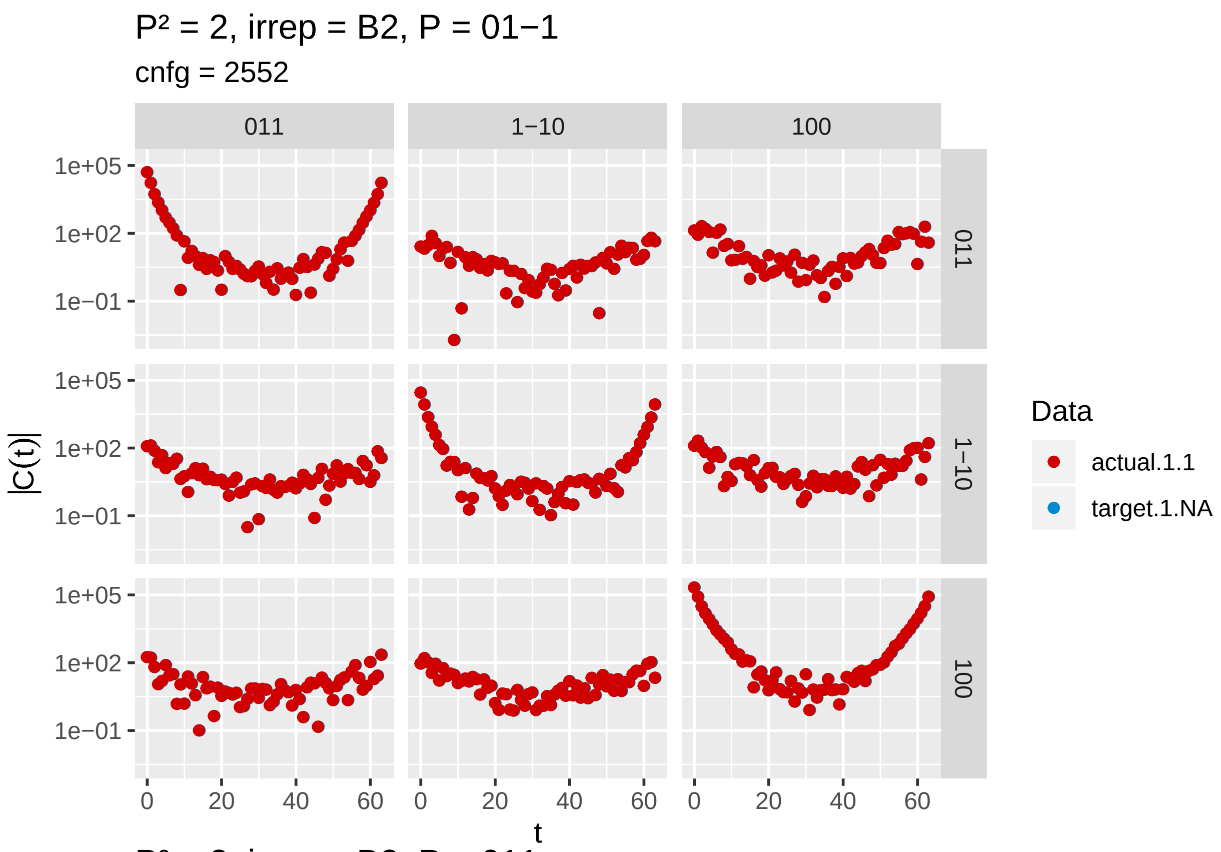
P² = 2, irrep = B2, P = −110

cnfg = 2552

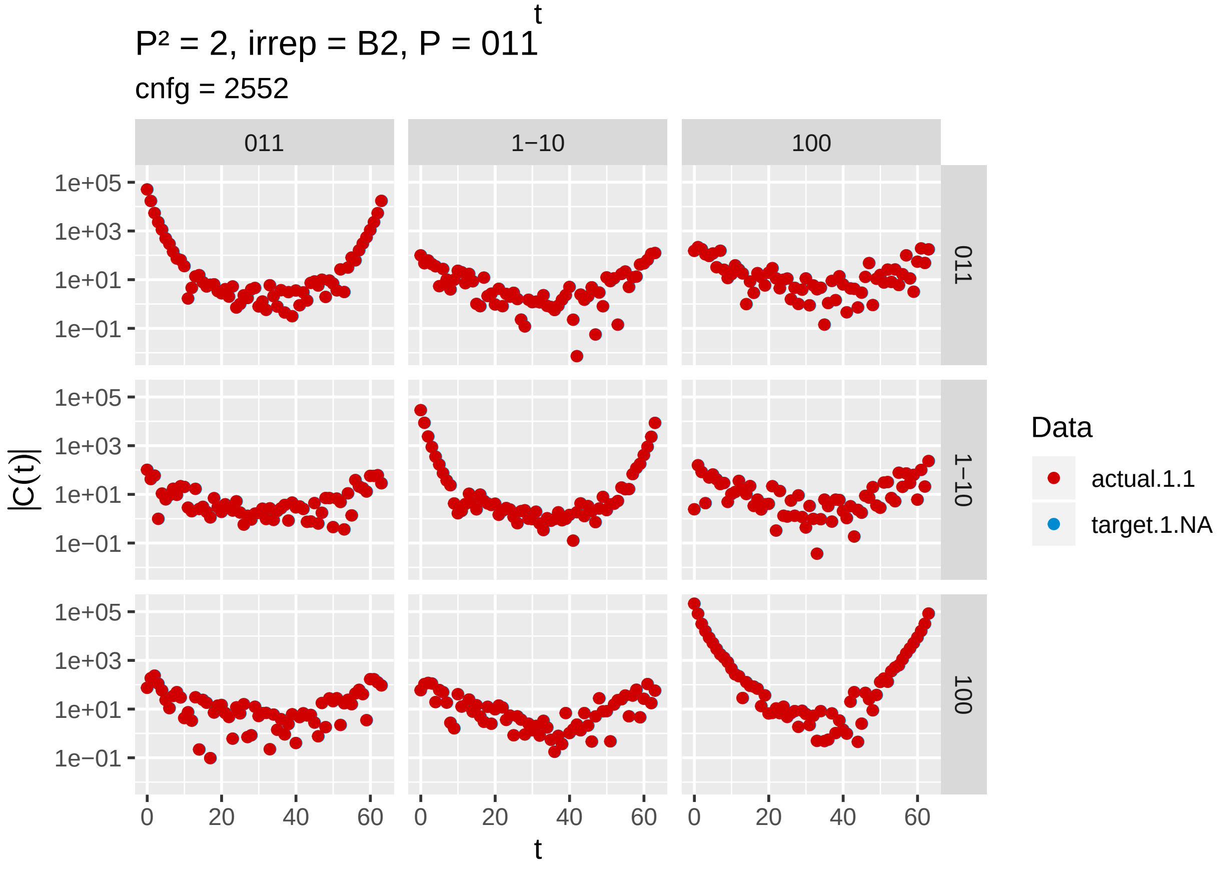



P² = 2, irrep = B2, P = 0−1−1
cnfg = 2552
011
1−10
100
10000
100
1
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA

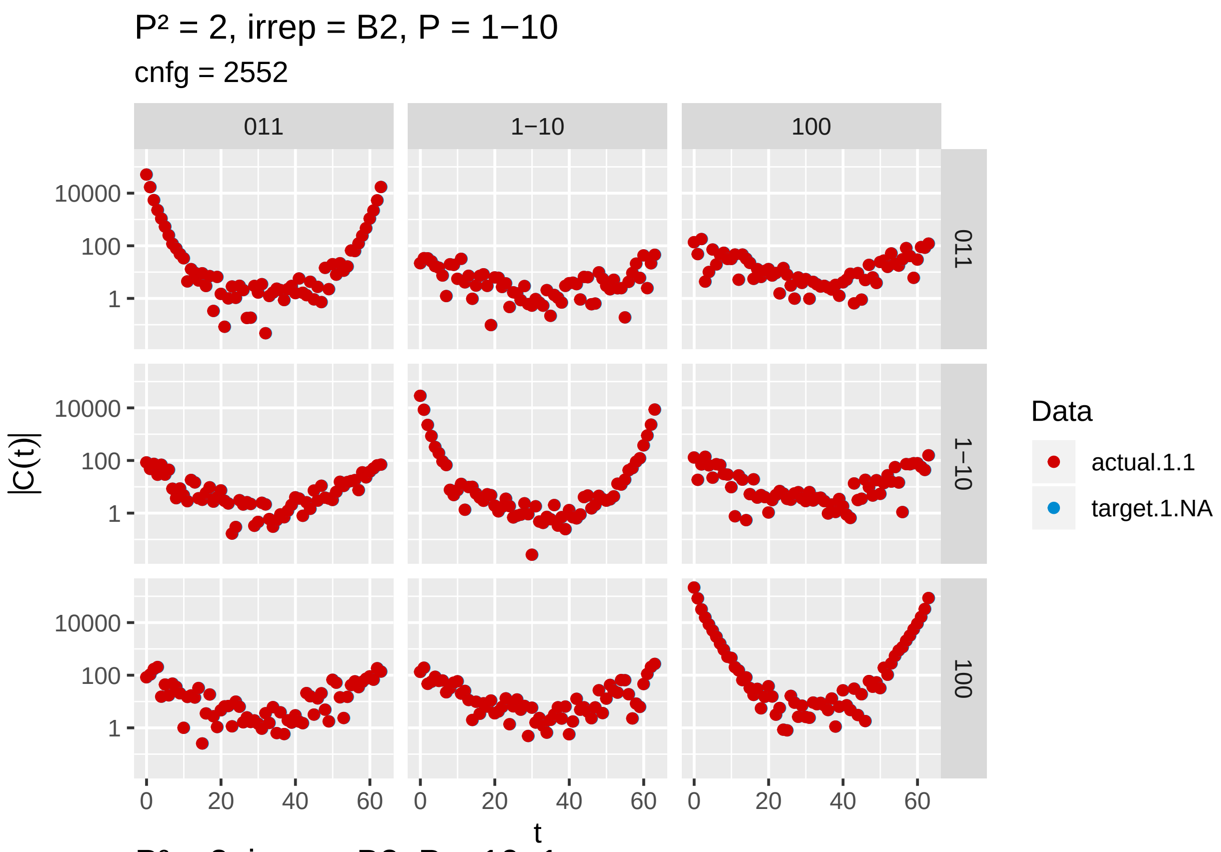

P² = 2, irrep = B2, P = 0−11
cnfg = 2552
011
1−10
100
10000
100
1
|C(t)|
0
20
40
60
t
Data
● actual.1.1
● target.1.NA

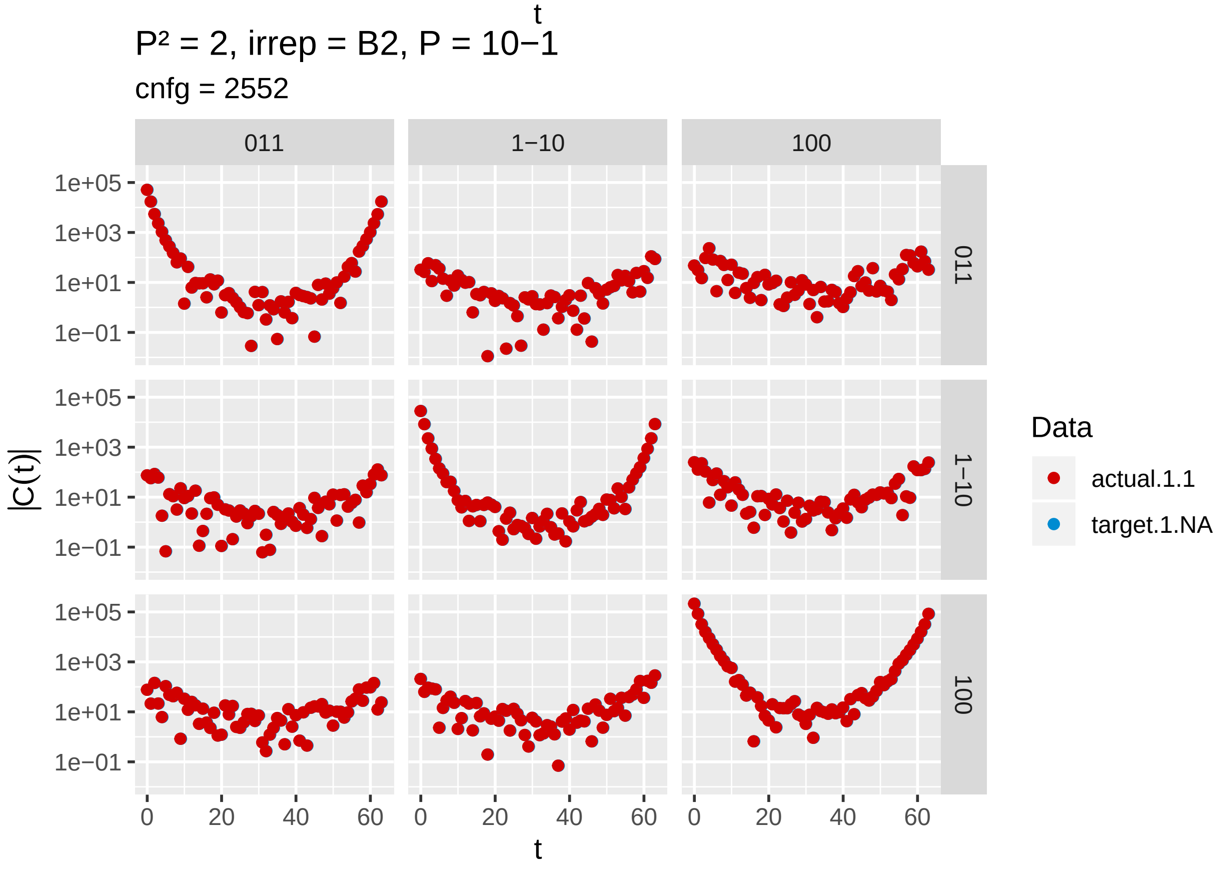


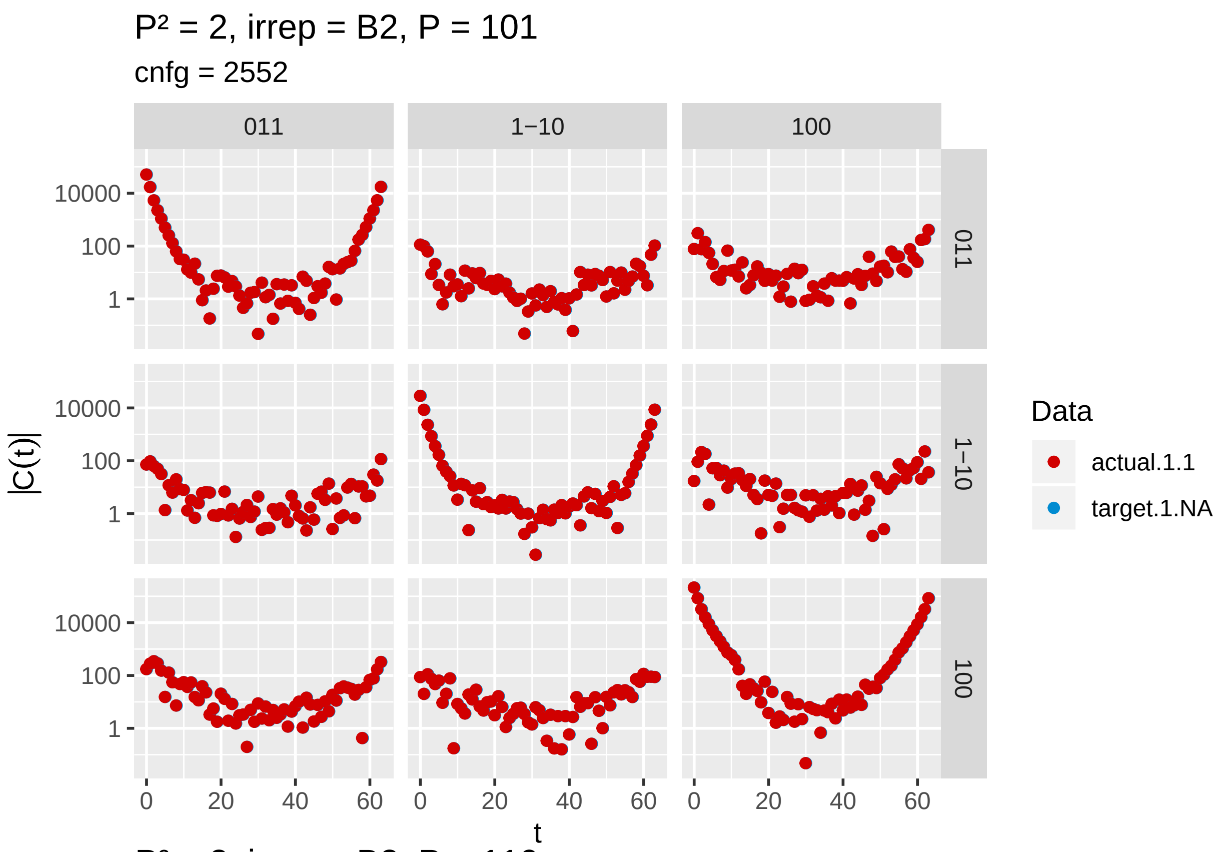
P² = 2, irrep = B2, P = 01−1
cnfg = 2552
011
1−10
100
1e+05
1e+02
1e−01
|C(t)|
t
0
20
40
60
Data
actual.1.1
target.1.NA

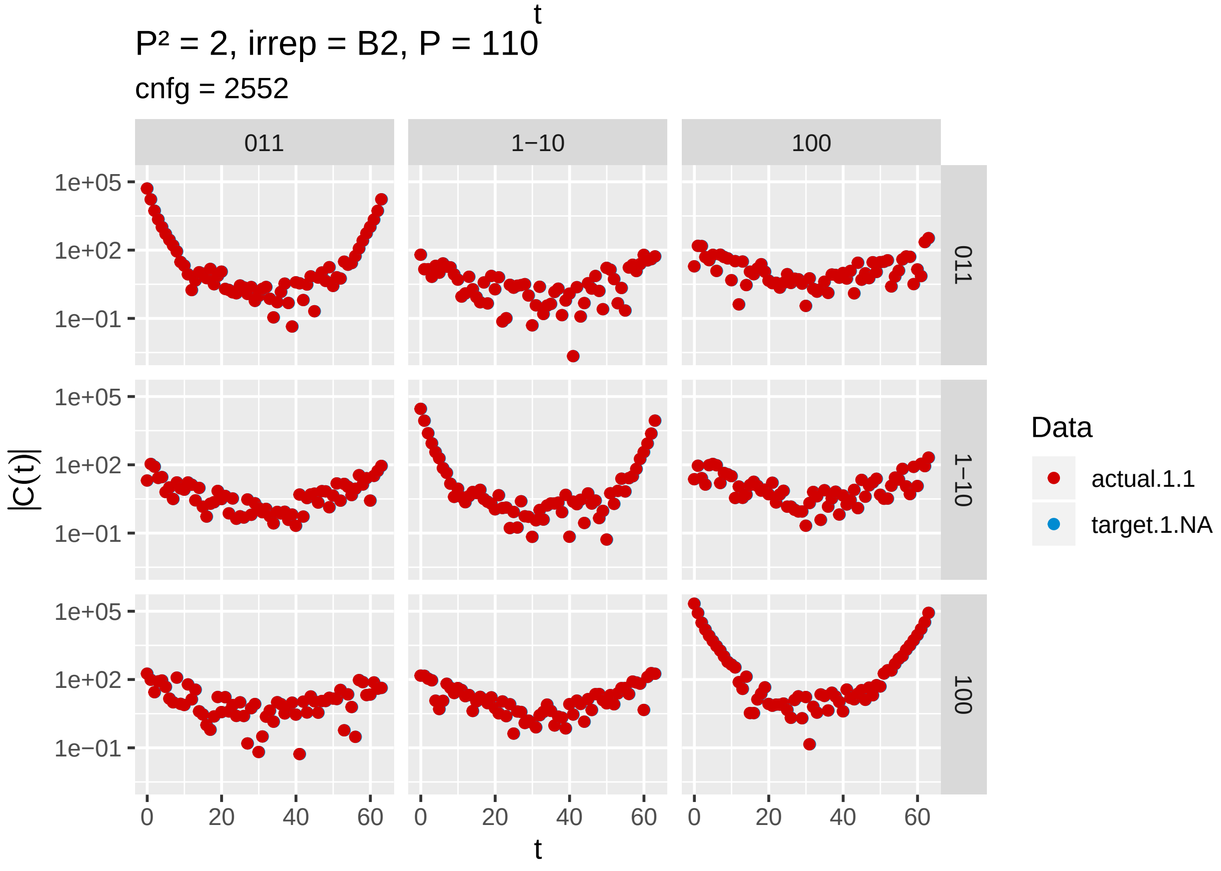


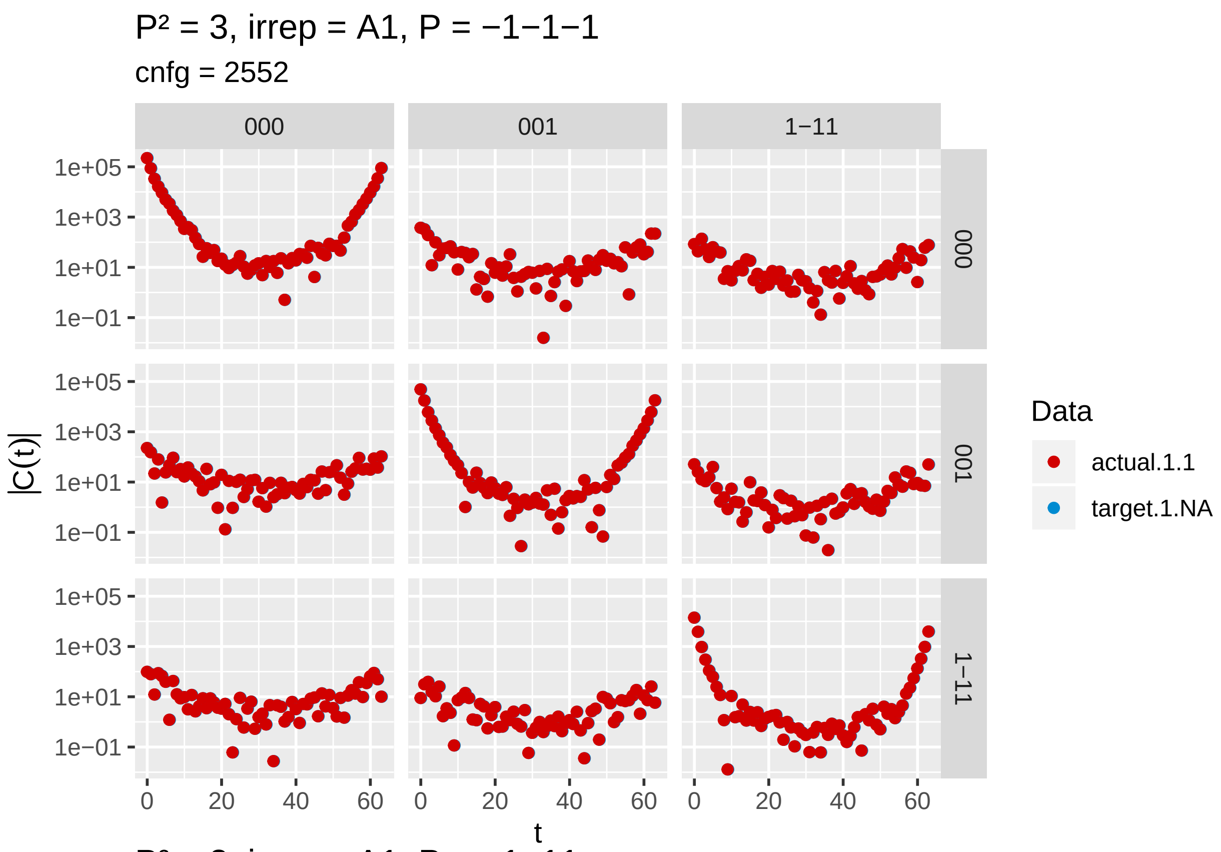
P² = 2, irrep = B2, P = 011
cnfg = 2552
011
1−10
100
1e+05
1e+03
1e+01
1e−01
|C(t)|
t
0
20
40
60
Data
actual.1.1
target.1.NA

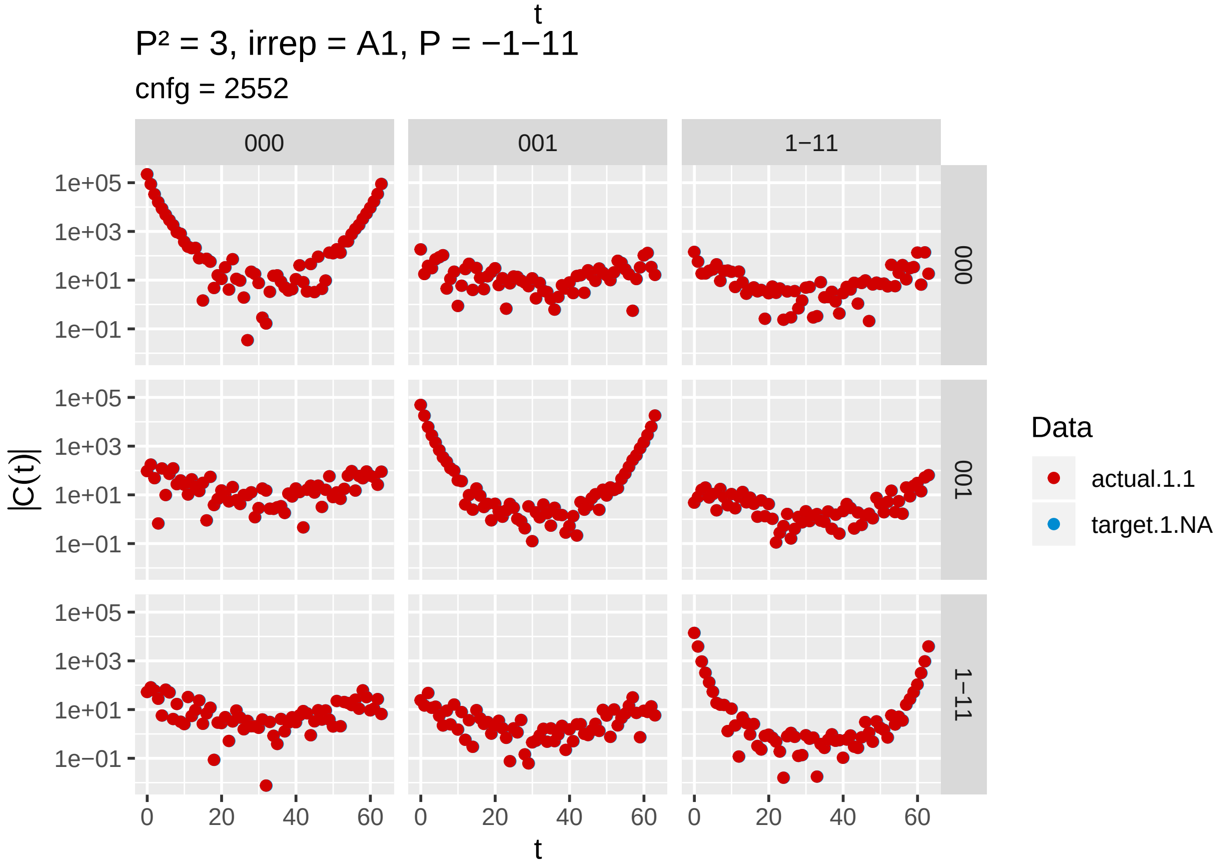

P² = 2, irrep = B2, P = 1−10
cnfg = 2552
011
1−10
100
|C(t)|
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B2, P = 10−1
cnfg = 2552
011
1−10
100
|C(t)|
t
Data
actual.1.1
target.1.NA


P² = 2, irrep = B2, P = 101
cnfg = 2552
011
1−10
100
10000
100
1
0
20
40
60
t
|C(t)|
Data
● actual.1.1
● target.1.NA

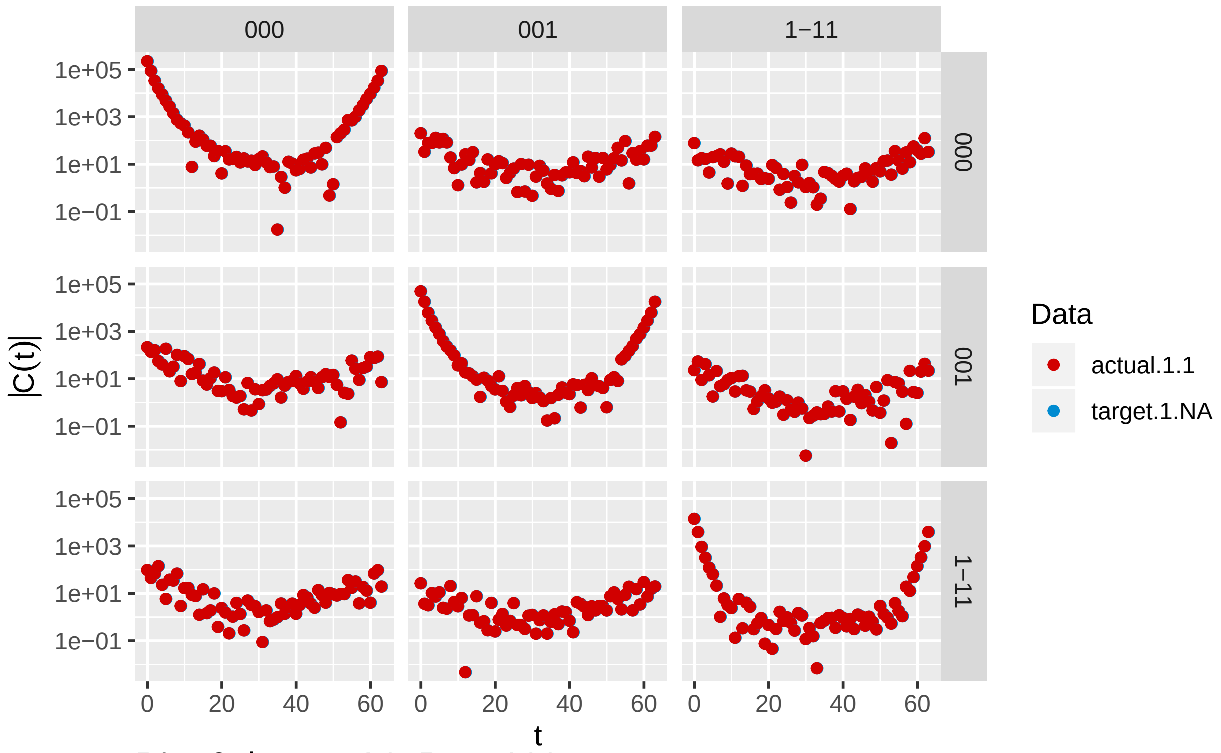

P² = 2, irrep = B2, P = 110
cnfg = 2552
011
1−10
100
1e+05
1e+02
1e−01
0
20
40
60
t
|C(t)|
Data
● actual.1.1
● target.1.NA


P² = 3, irrep = A1, P = −1−1−1
cnfg = 2552
000
001
1−11
000
001
1−11
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 3, irrep = A1, P = −1−11
cnfg = 2552
000
001
1−11
000
001
1−11
1e+05
1e+03
1e+01
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

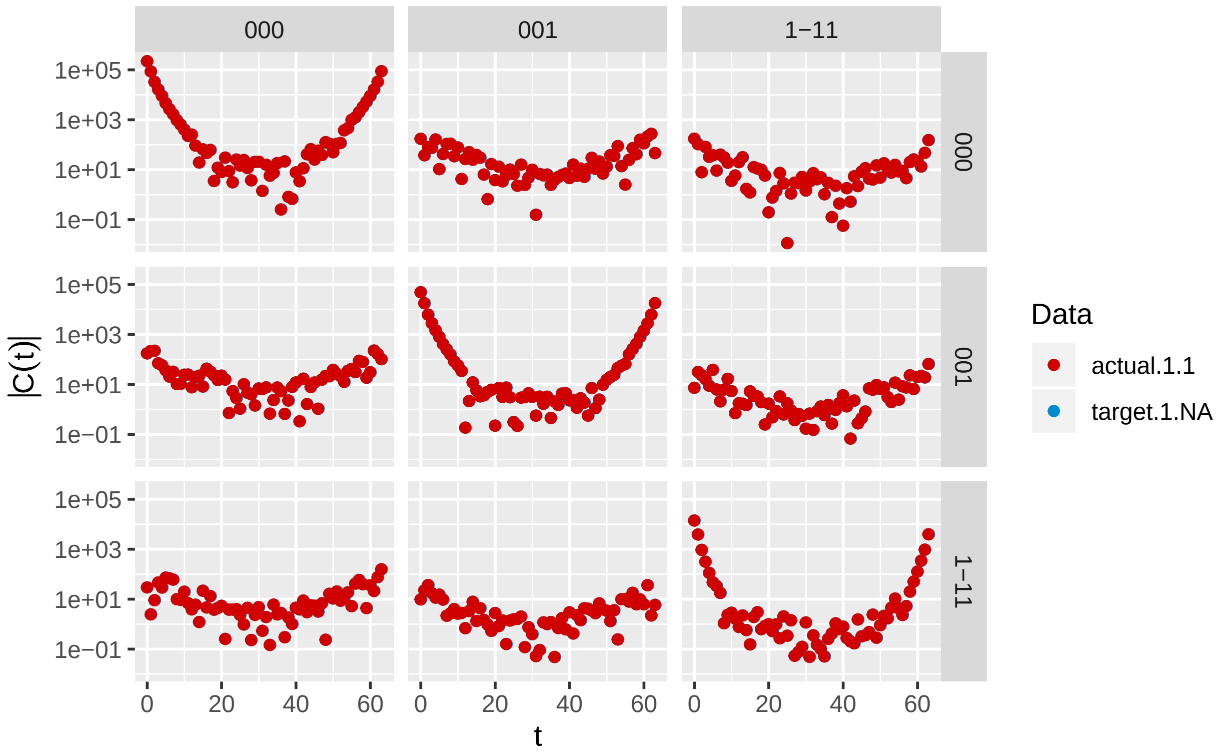
P² = 3, irrep = A1, P = −11−1

cnfg = 2552



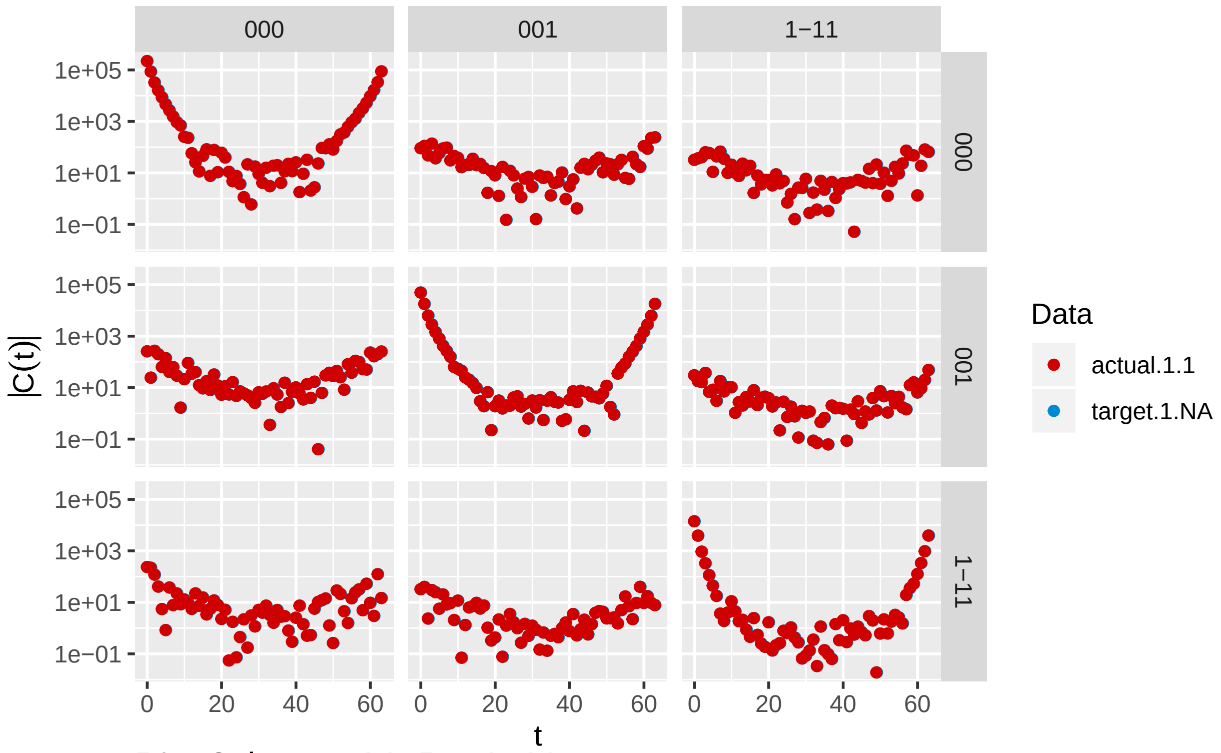
P² = 3, irrep = A1, P = −111

cnfg = 2552



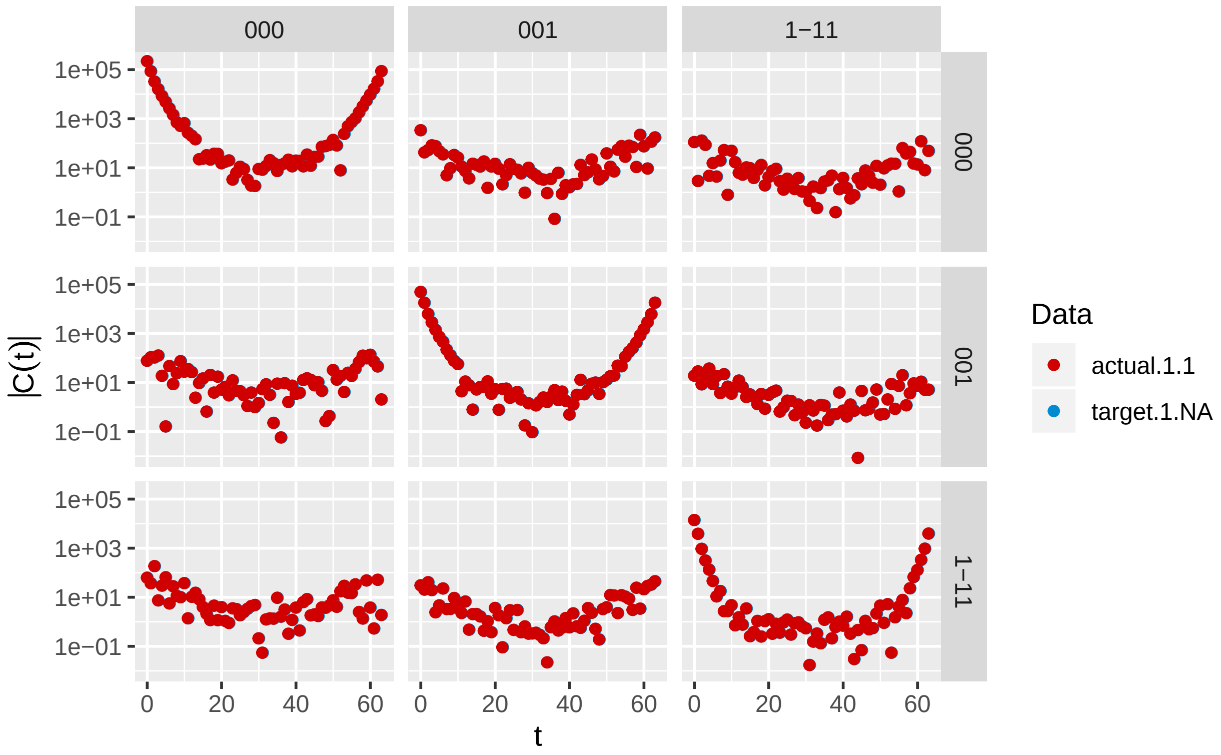
P² = 3, irrep = A1, P = 1−1−1

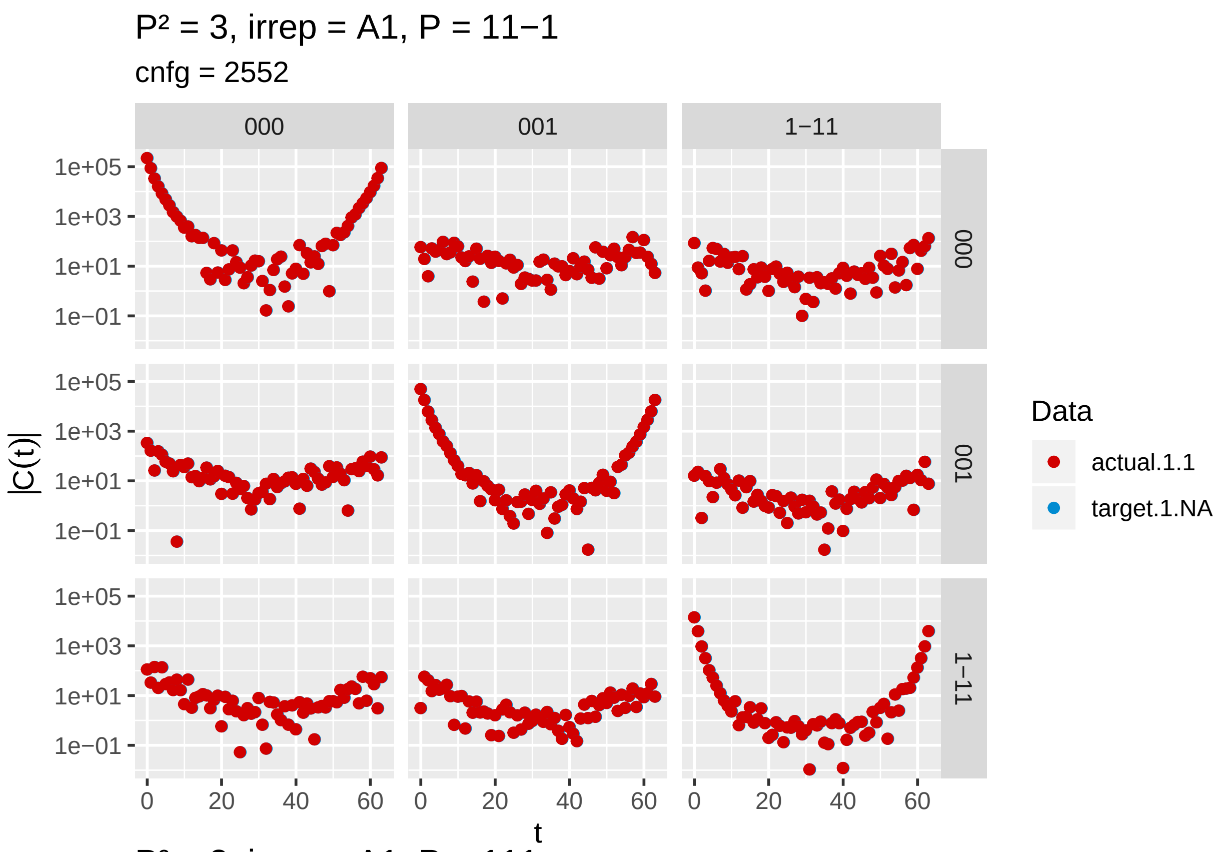
cnfg = 2552

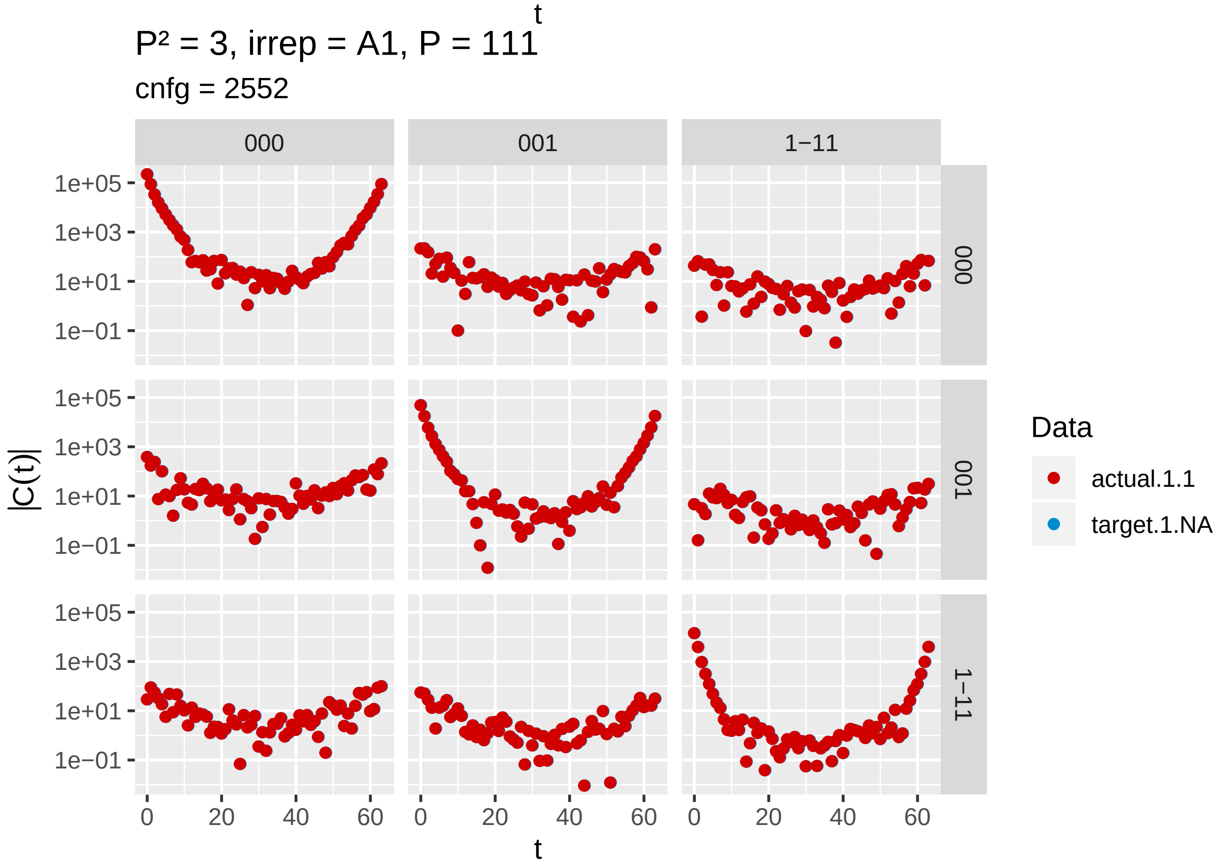


P² = 3, irrep = A1, P = 1−11

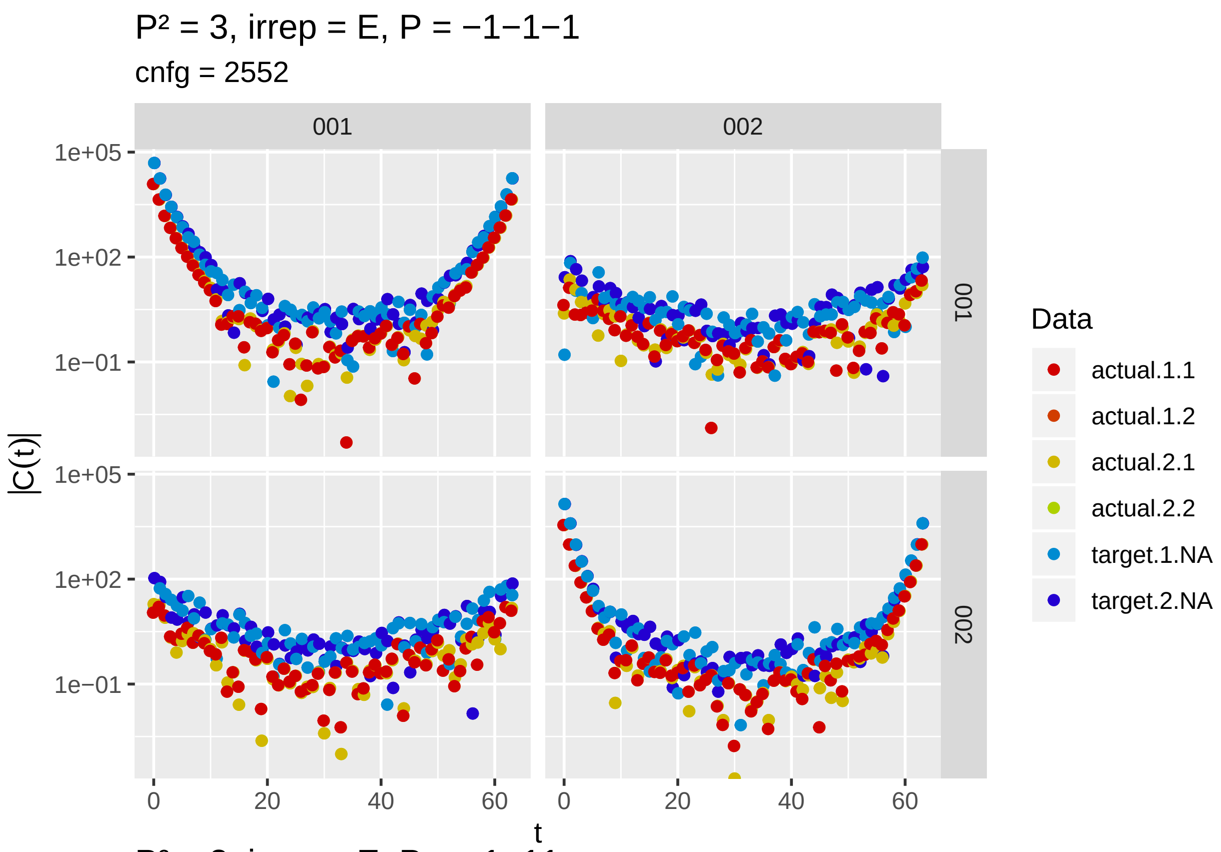
cnfg = 2552

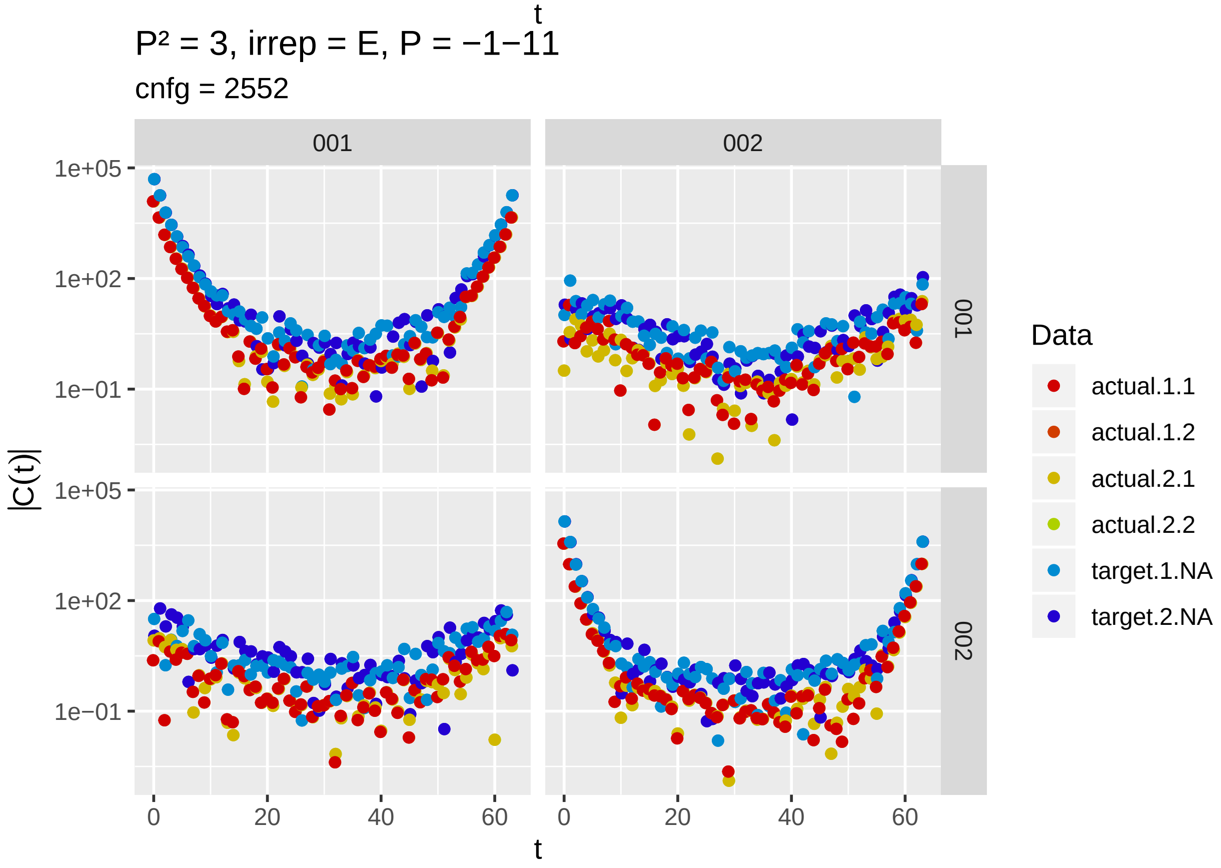


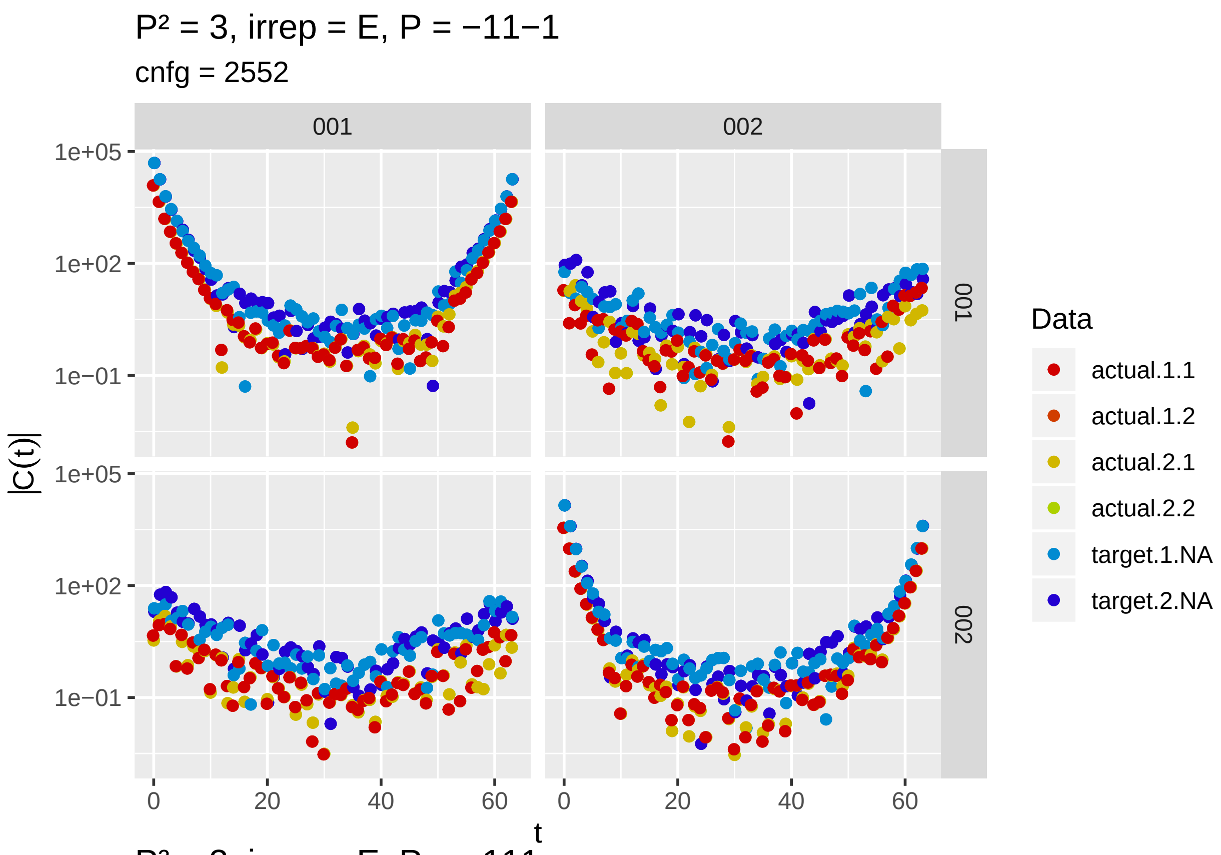


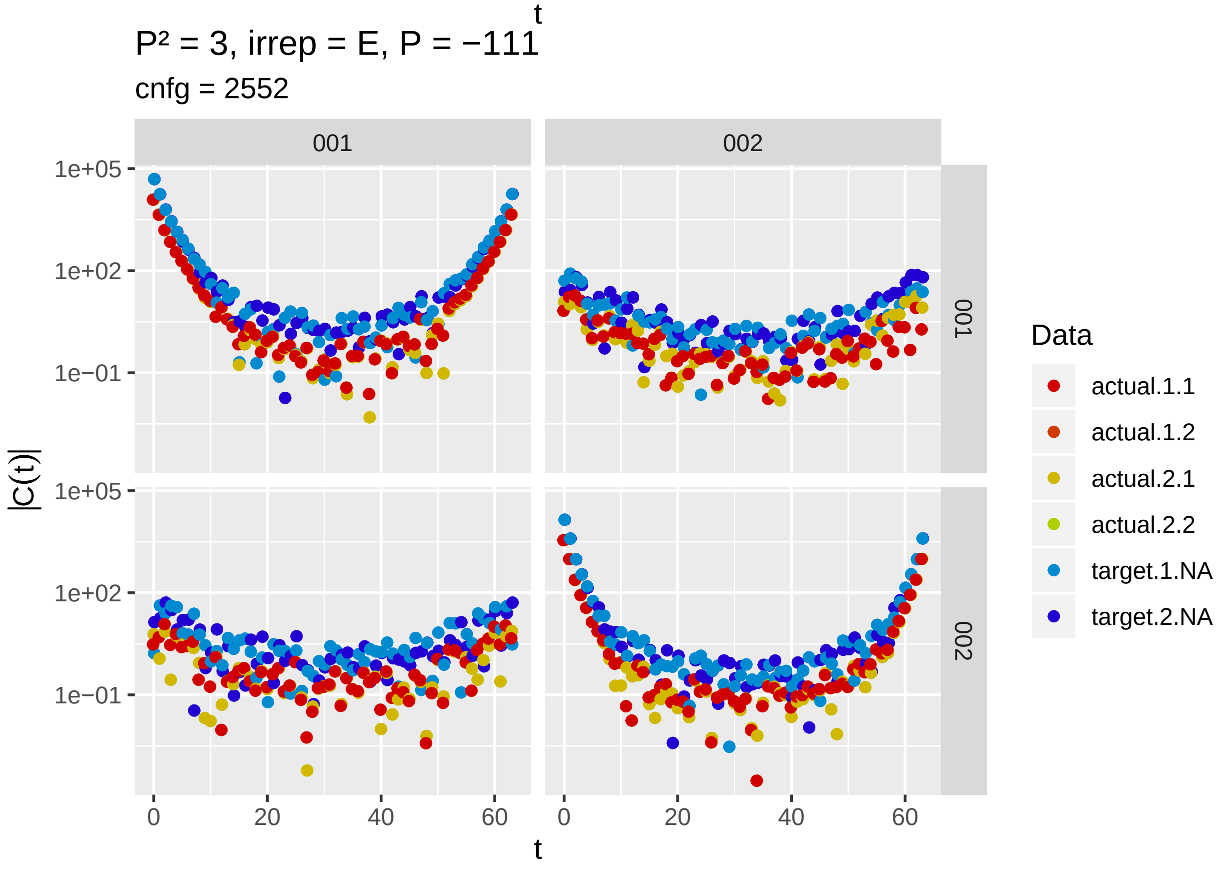


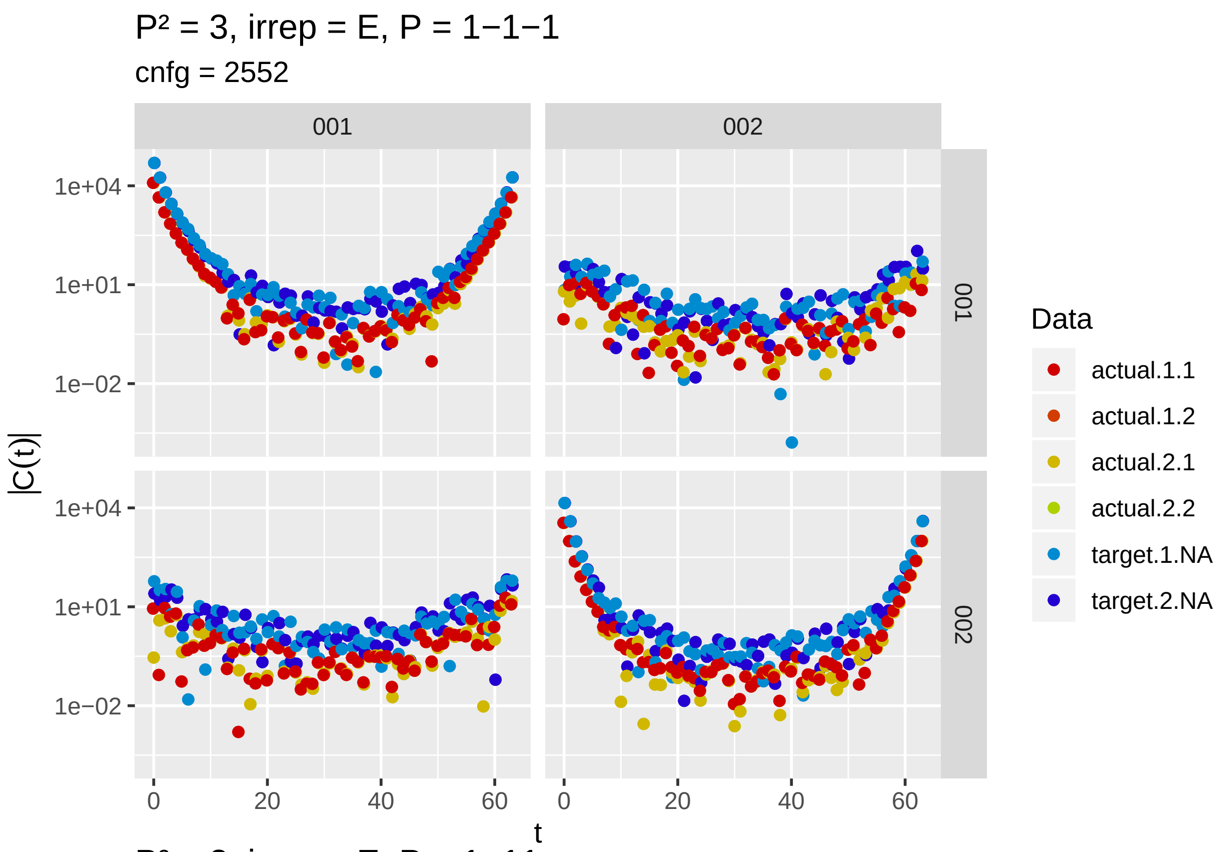
```
## Warning: Transformation introduced infinite values in continuous y-axis
```

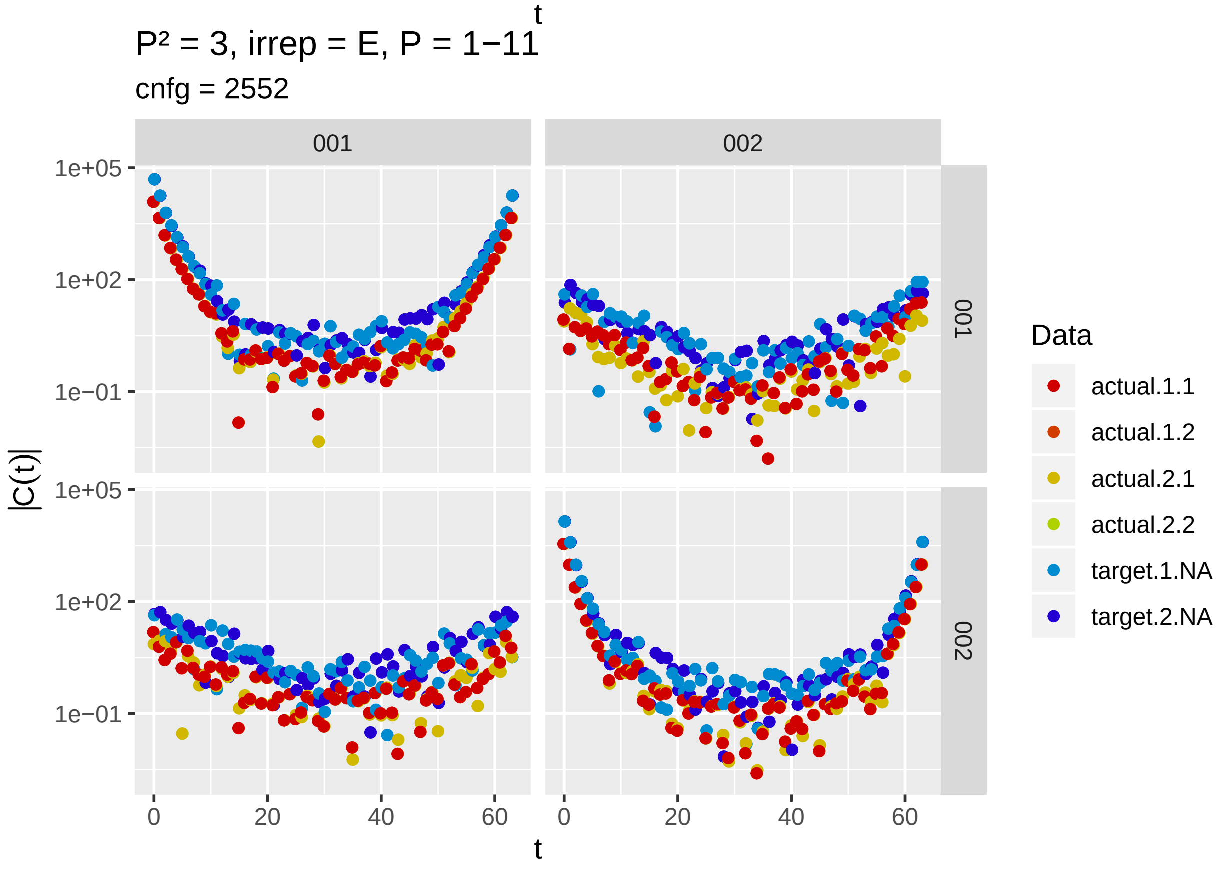

P² = 3, irrep = E, P = −1−1−1
cnfg = 2552
001
002
001
002
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

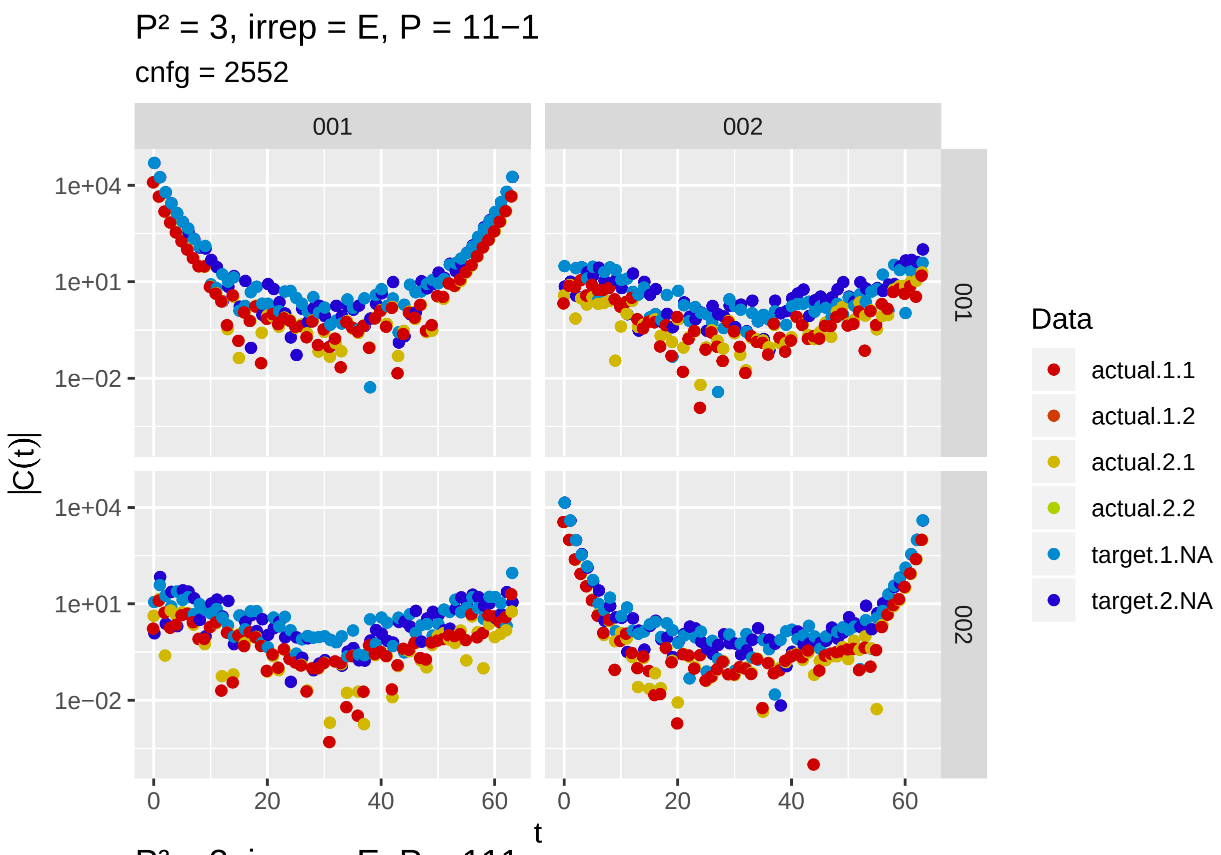

P² = 3, irrep = E, P = −1−11
cnfg = 2552
001
002
001
002
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

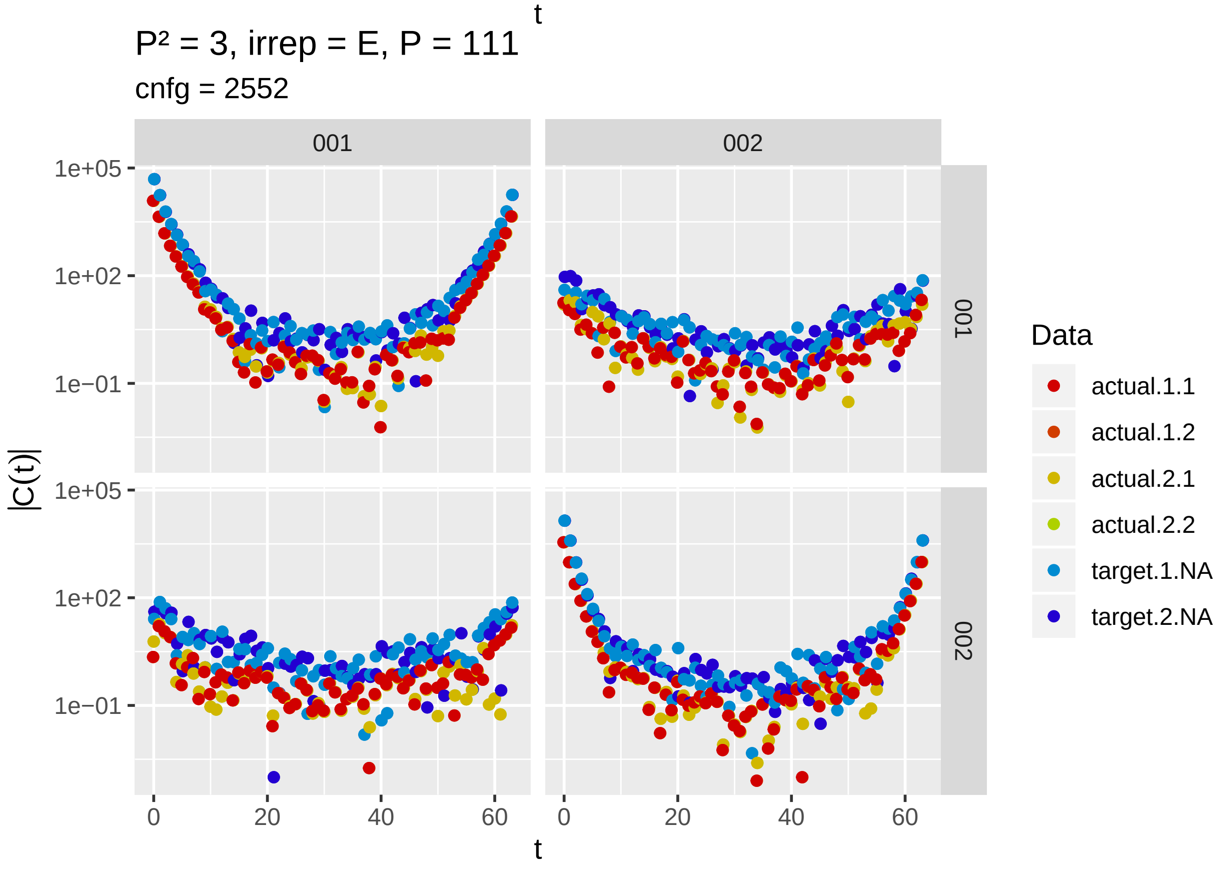

P² = 3, irrep = E, P = −11−1
cnfg = 2552
001
002
|C(t)|
1e+05
1e+02
1e−01
t
0
20
40
60
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 3, irrep = E, P = −111
cnfg = 2552
001
002
|C(t)|
1e+05
1e+02
1e−01
t
0
20
40
60
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 3, irrep = E, P = 1−1−1
cnfg = 2552
001
002
|C(t)|
1e+04
1e+01
1e−02
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

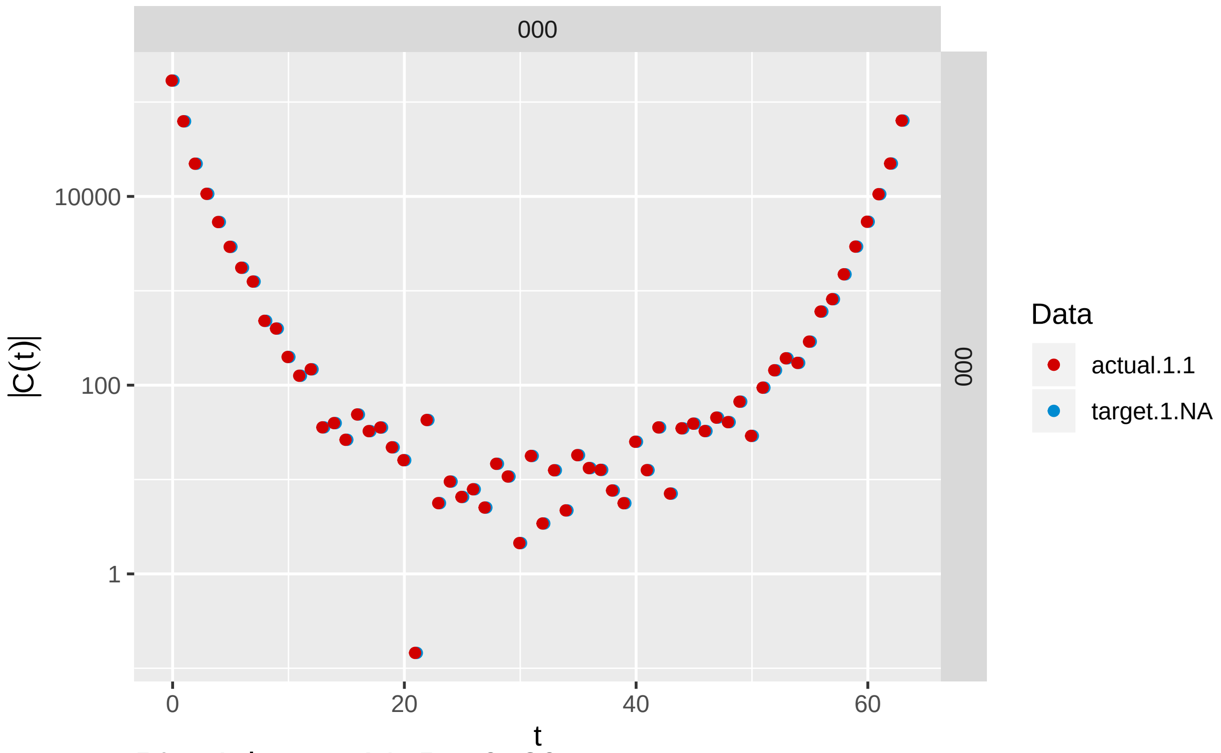

P² = 3, irrep = E, P = 1−11
cnfg = 2552
001
002
|C(t)|
1e+05
1e+02
1e−01
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 3, irrep = E, P = 11−1
cnfg = 2552
001
002
001
002
1e+04
1e+01
1e−02
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 3, irrep = E, P = 111
cnfg = 2552
001
002
001
002
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

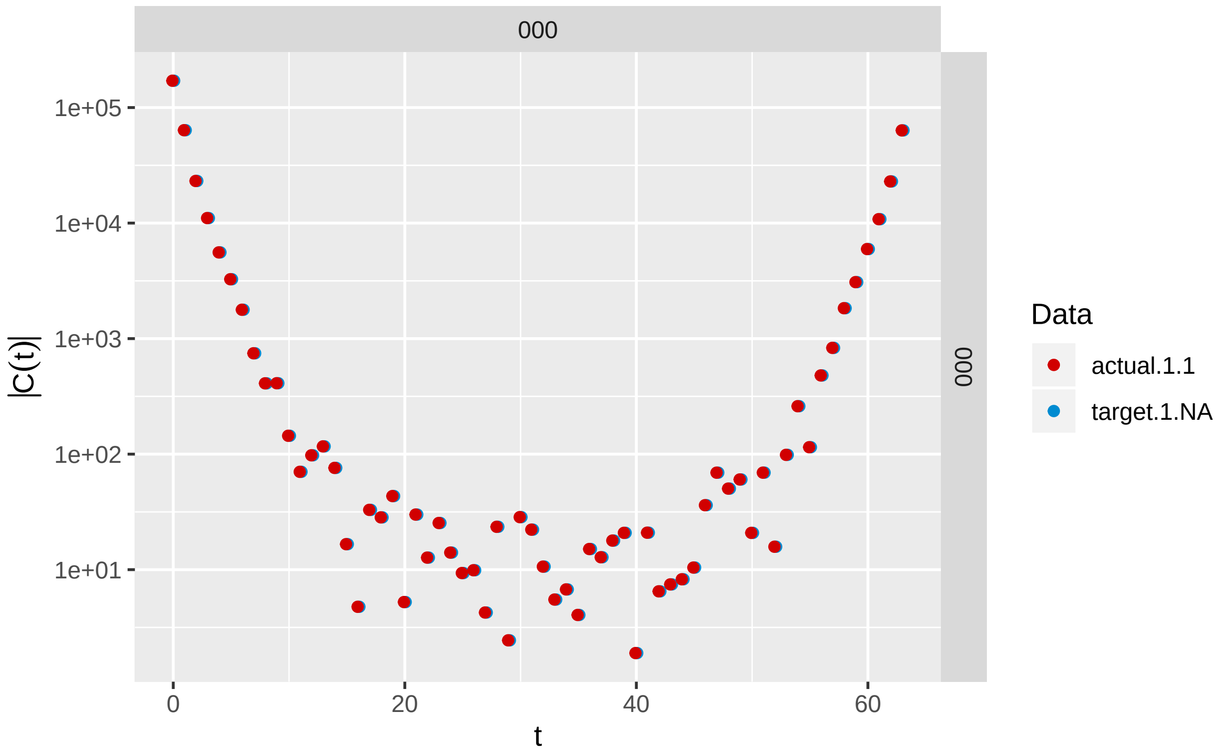
P² = 4, irrep = A1, P = −200

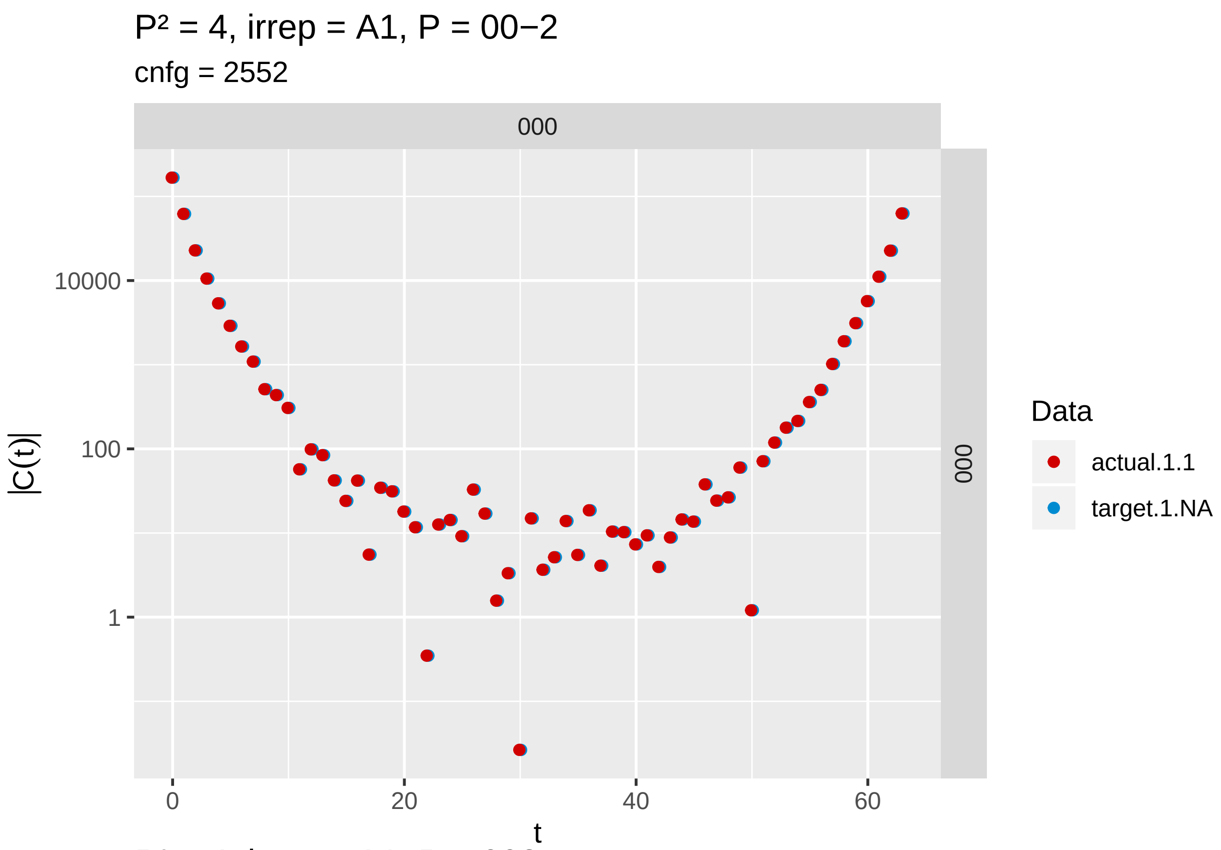
cnfg = 2552

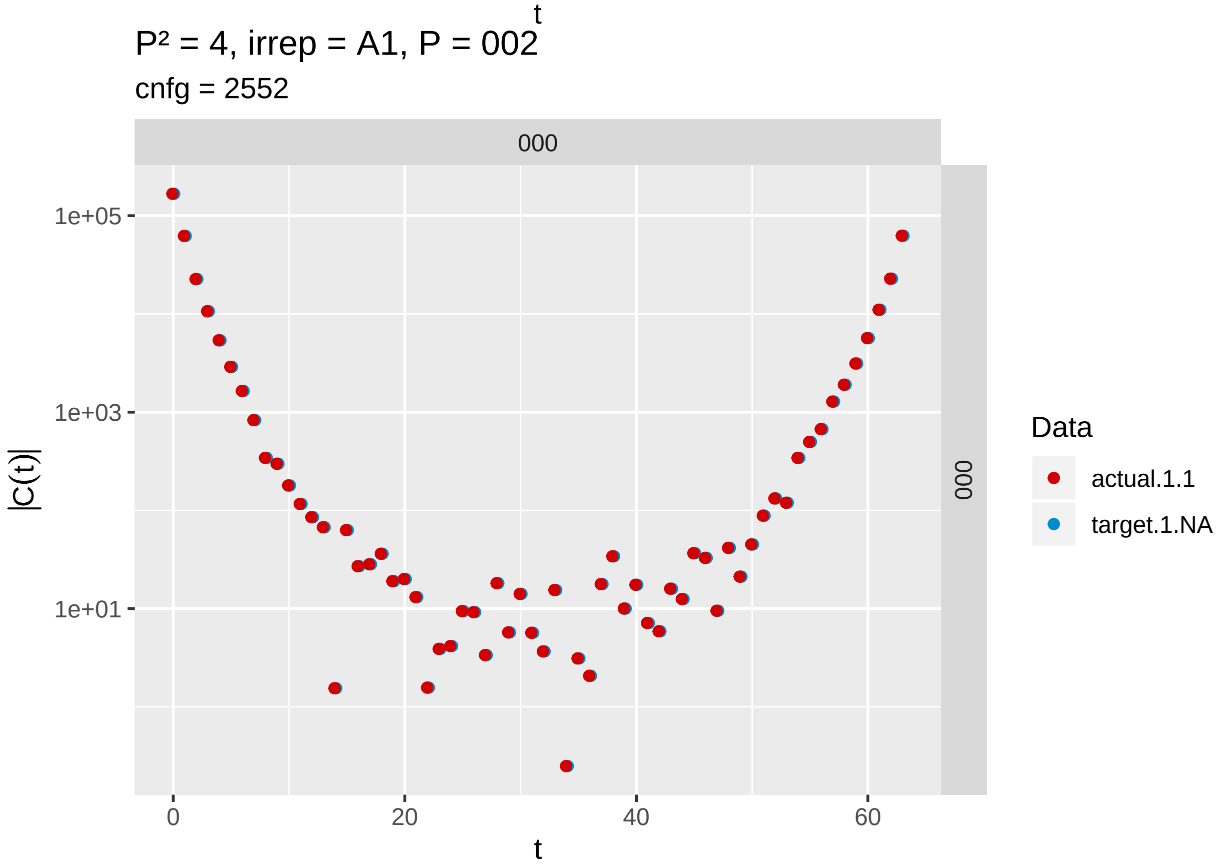


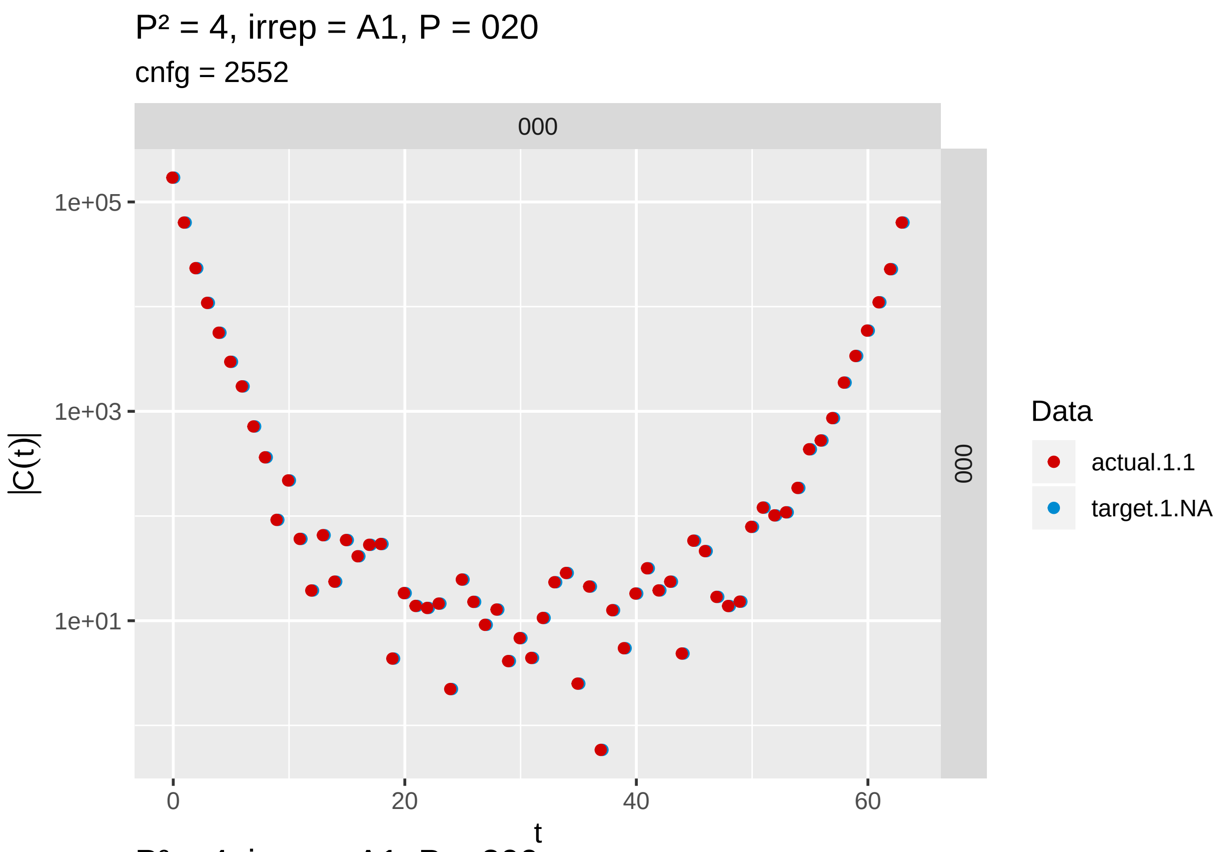
P² = 4, irrep = A1, P = 0−20

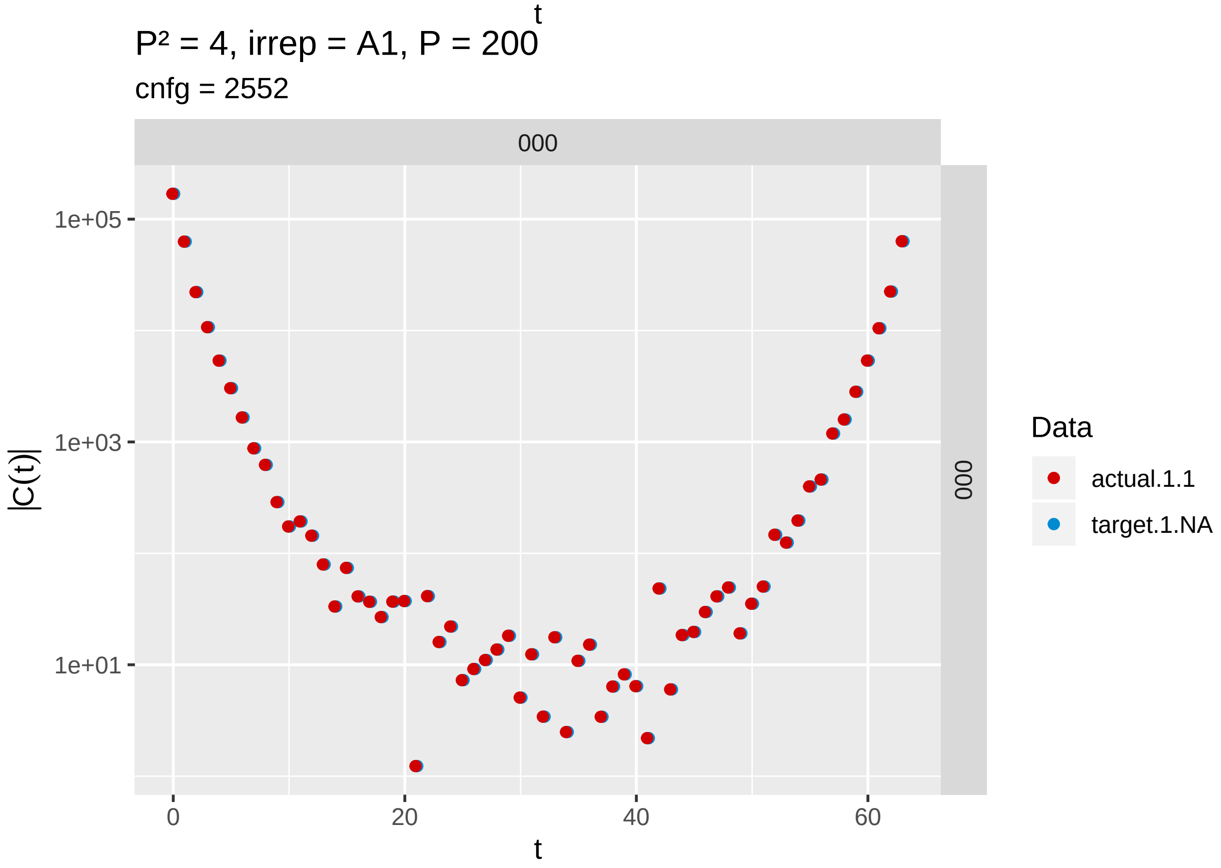
cnfg = 2552

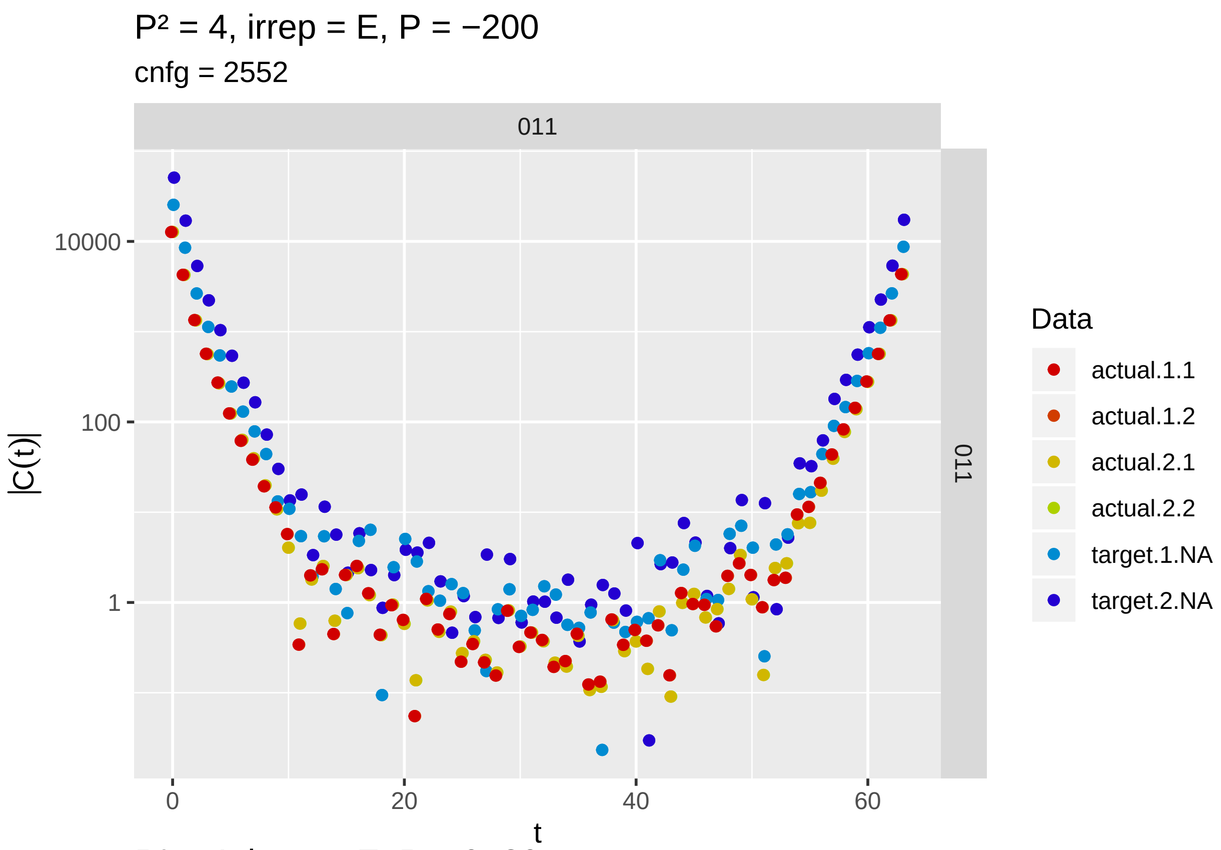


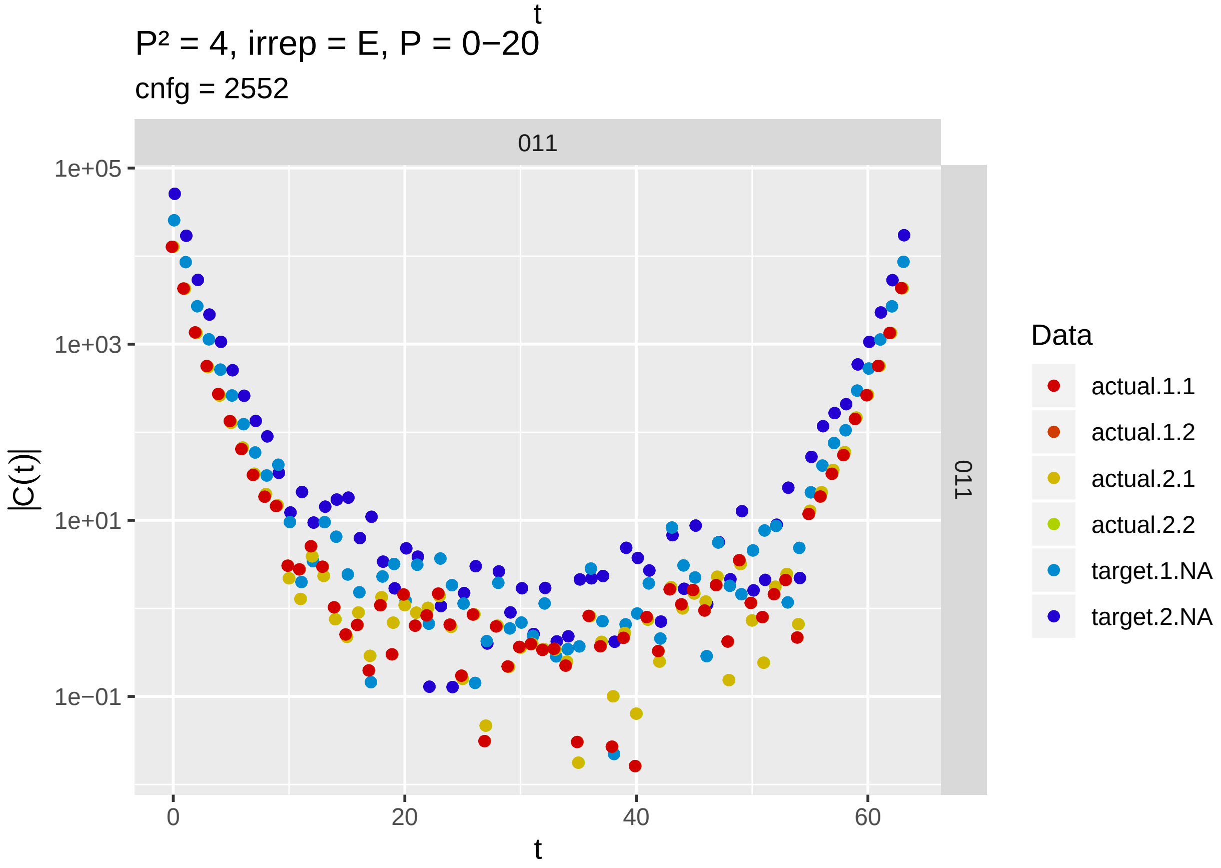

P² = 4, irrep = A1, P = 00−2
cnfg = 2552
000
000
|C(t)|
10000
100
1
0
20
40
60
t
Data
● actual.1.1
● target.1.NA


P² = 4, irrep = A1, P = 002
cnfg = 2552
000
000
|C(t)|
1e+05
1e+03
1e+01
0
20
40
60
t
Data
● actual.1.1
● target.1.NA


P² = 4, irrep = A1, P = 020
cnfg = 2552
000
000
1e+05
1e+03
1e+01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA

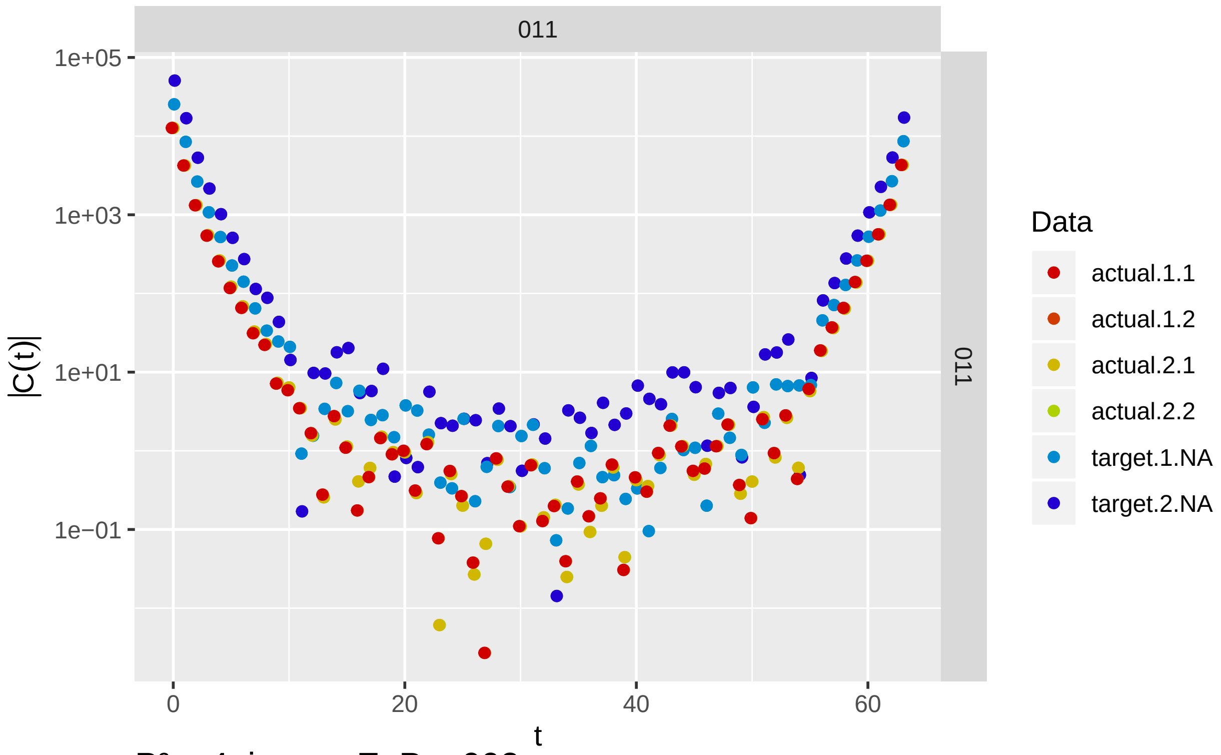

P² = 4, irrep = A1, P = 200
cnfg = 2552
000
000
1e+05
1e+03
1e+01
|C(t)|
0
20
40
60
t
Data
actual.1.1
target.1.NA


P² = 4, irrep = E, P = −200
cnfg = 2552
011
011
|C(t)|
10000
100
1
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 4, irrep = E, P = 0−20
cnfg = 2552
011
011
|C(t)|
1e+05
1e+03
1e+01
1e−01
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA

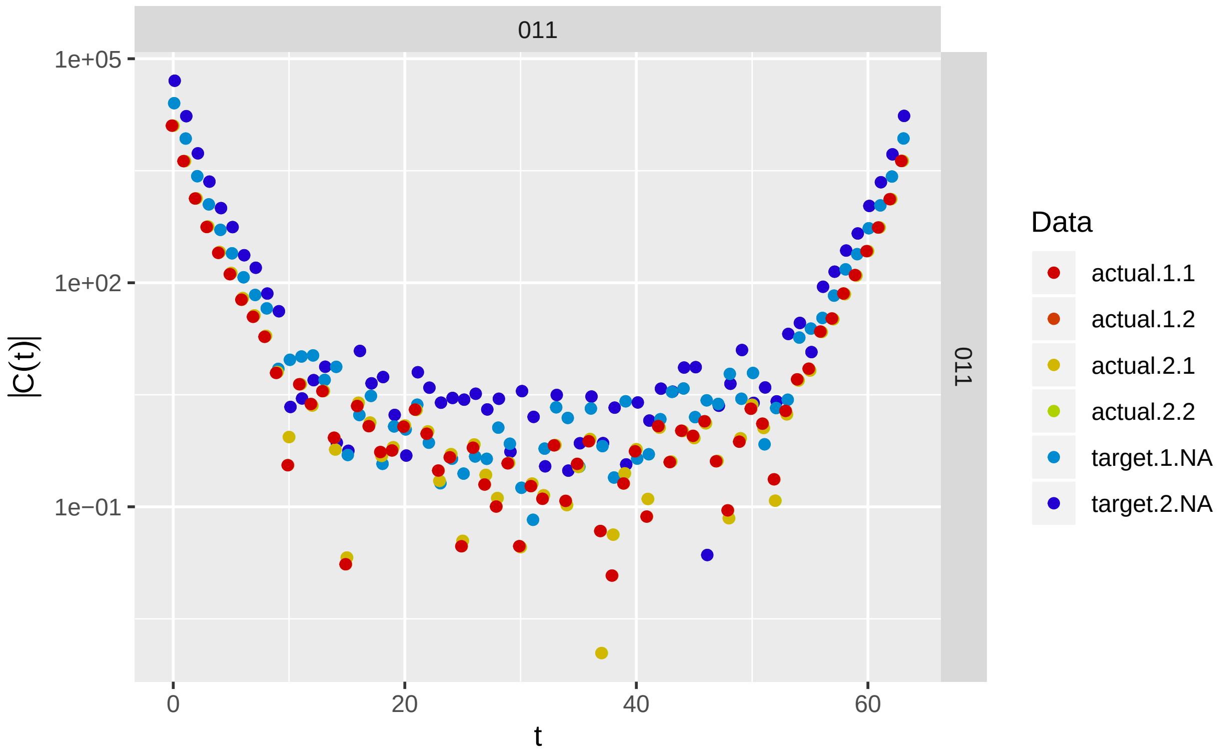
P² = 4, irrep = E, P = 00−2

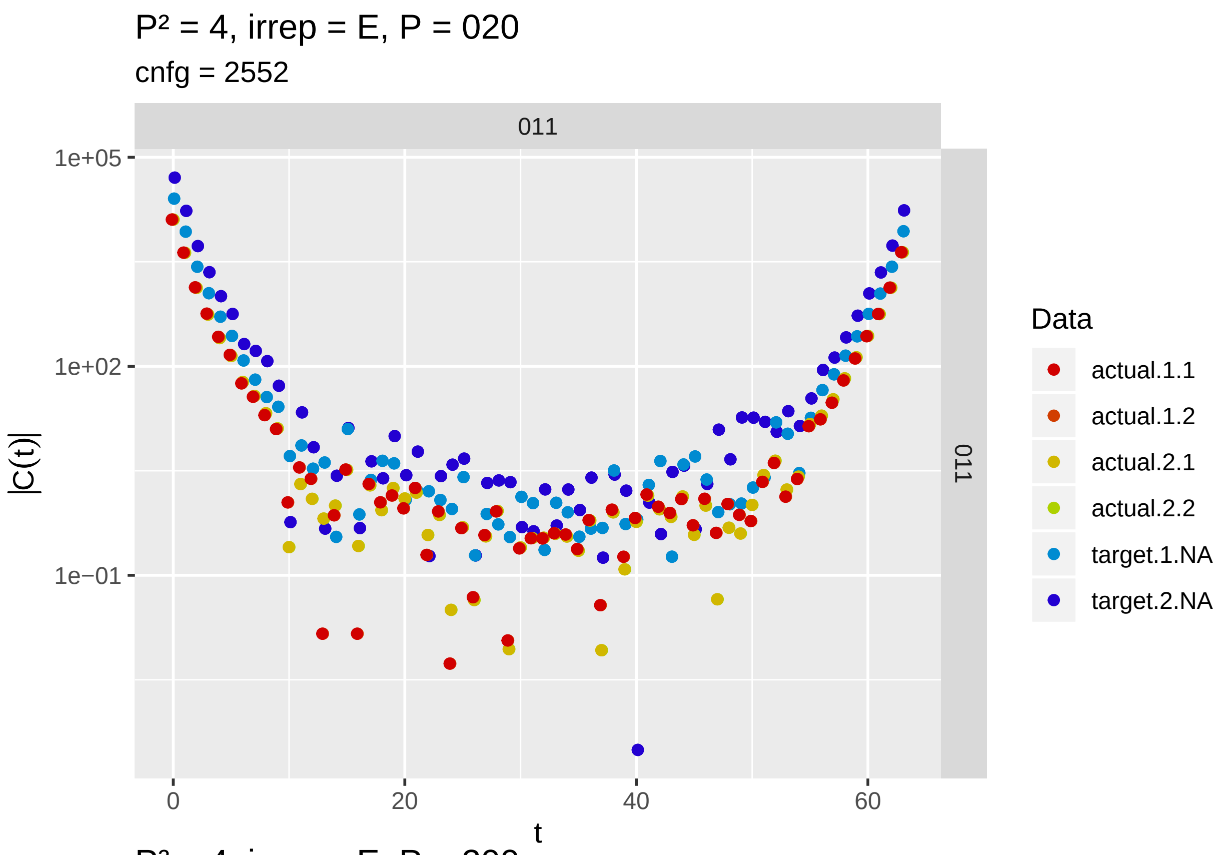
cnfg = 2552

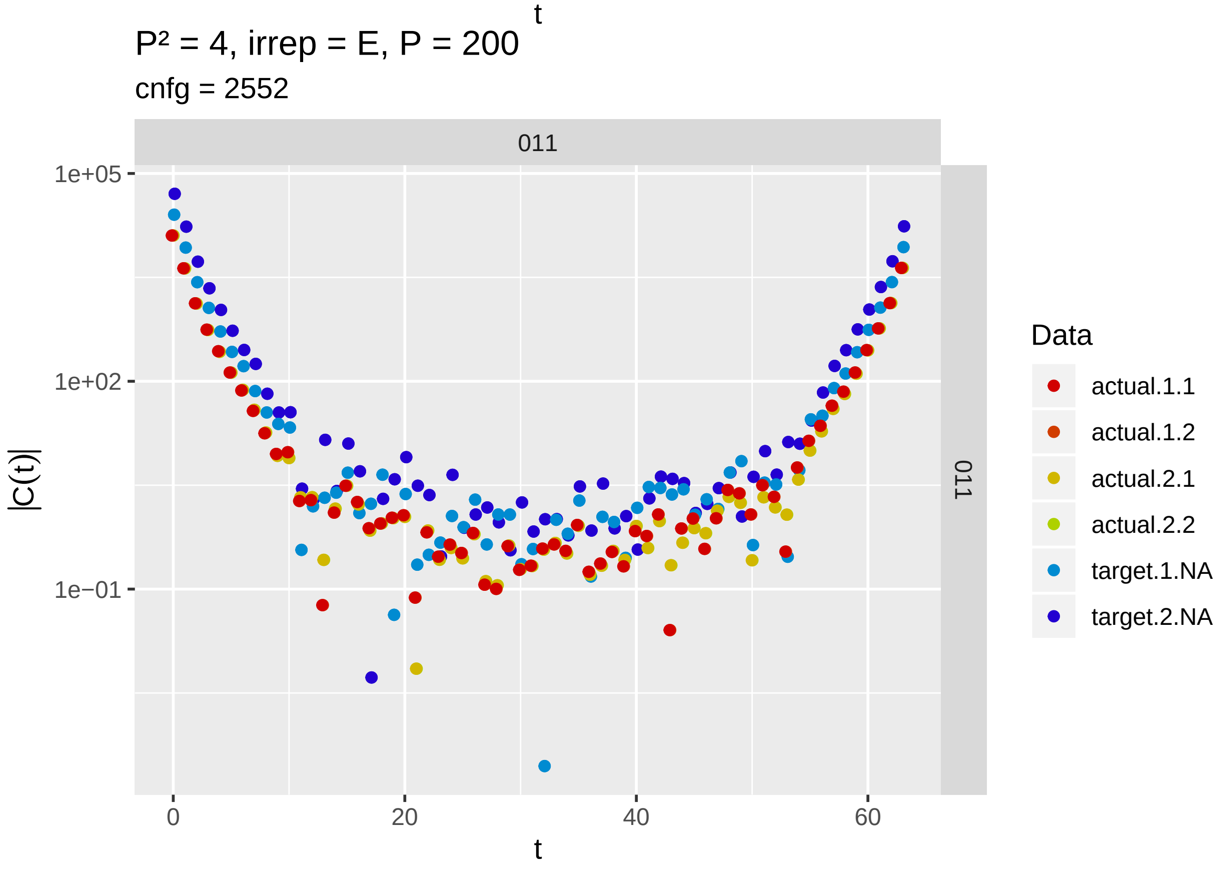


P² = 4, irrep = E, P = 002

cnfg = 2552




P² = 4, irrep = E, P = 020
cnfg = 2552
011
011
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA


P² = 4, irrep = E, P = 200
cnfg = 2552
011
011
1e+05
1e+02
1e−01
|C(t)|
0
20
40
60
t
Data
actual.1.1
actual.1.2
actual.2.1
actual.2.2
target.1.NA
target.2.NA