OSQP Settings Sequence Data Frame

Data frame containing a sequence of OSQP settings for `tsbalancing()` specified with argument `osqp_settings_df`. The package includes two predefined OSQP settings sequence data frames that are presented below.

Data frame `default_osqp_sequence`

Fast and effective sequence of OSQP settings that should be suitable for accurately solving most time series balancing problems. It is the default value of `tsbalancing()` argument `osqp_settings_df`.

```
#>   max_iter    sigma  eps_abs  eps_rel eps_prim_inf eps_dual_inf polish scaling
#> 1    4,000 1.00e-09 1.00e-06 1.00e-06     1.00e-07     1.00e-07   TRUE       0
#> 2   10,000 1.00e-15 1.00e-12 1.00e-12     1.00e-13     1.00e-13   TRUE       0
#> 3   10,000 1.00e-15 1.00e-12 1.00e-12     1.00e-13     1.00e-13   TRUE       0
#> 4   10,000 2.22e-16 2.22e-16 2.22e-16     2.22e-16     2.22e-16   TRUE       0
#>   prior_scaling require_polished
#> 1          TRUE             TRUE
#> 2          TRUE             TRUE
#> 3         FALSE            FALSE
#> 4          TRUE            FALSE
```

Data frame `alternate_osqp_sequence`

Alternative slower sequence of OSQP settings that could help achieve more precision, when needed, especially when combined with argument `full_sequence = TRUE`.

```
#>    max_iter    sigma  eps_abs  eps_rel eps_prim_inf eps_dual_inf polish scaling
#> 1    10,000 1.00e-15 1.00e-12 1.00e-12     1.00e-13     1.00e-13   TRUE       0
#> 2    10,000 1.00e-15 1.00e-12 1.00e-12     1.00e-13     1.00e-13   TRUE      10
#> 3    10,000 1.00e-12 1.00e-09 1.00e-09     1.00e-10     1.00e-10   TRUE       0
#> 4    10,000 1.00e-12 1.00e-09 1.00e-09     1.00e-10     1.00e-10   TRUE      10
#> 5    10,000 1.00e-09 1.00e-06 1.00e-06     1.00e-07     1.00e-07   TRUE       0
#> 6    10,000 1.00e-09 1.00e-06 1.00e-06     1.00e-07     1.00e-07   TRUE      10
#> 7    10,000 1.00e-06 1.00e-03 1.00e-03     1.00e-04     1.00e-04   TRUE       0
#> 8    10,000 1.00e-06 1.00e-03 1.00e-03     1.00e-04     1.00e-04   TRUE      10
#> 9    10,000 2.22e-16 2.22e-16 2.22e-16     2.22e-16     2.22e-16   TRUE       0
#> 10   10,000 2.22e-16 2.22e-16 2.22e-16     2.22e-16     2.22e-16   TRUE      10
#> 11   10,000 1.00e-15 1.00e-12 1.00e-12     1.00e-13     1.00e-13   TRUE       0
#> 12   10,000 1.00e-12 1.00e-09 1.00e-09     1.00e-10     1.00e-10   TRUE       0
#> 13   10,000 1.00e-09 1.00e-06 1.00e-06     1.00e-07     1.00e-07   TRUE       0
#> 14   10,000 1.00e-06 1.00e-03 1.00e-03     1.00e-04     1.00e-04   TRUE       0
#> 15   10,000 2.22e-16 2.22e-16 2.22e-16     2.22e-16     2.22e-16   TRUE       0
#>    prior_scaling require_polished
#> 1          FALSE             TRUE
#> 2          FALSE             TRUE
#> 3          FALSE             TRUE
#> 4          FALSE             TRUE
#> 5          FALSE             TRUE
#> 6          FALSE             TRUE
```

```
#> 7          FALSE             TRUE
#> 8          FALSE             TRUE
#> 9          FALSE             TRUE
#> 10         FALSE             TRUE
#> 11          TRUE             TRUE
#> 12          TRUE             TRUE
#> 13          TRUE             TRUE
#> 14          TRUE             TRUE
#> 15          TRUE             TRUE
```

Details

With the exception of `prior_scaling` and `require_polished`, all columns of the data frame must correspond to a OSQP setting. Default OSQP values are used for any setting not specified in this data frame. Visit https://osqp.org/docs/interfaces/solver_settings.html for all available OSQP settings. Note that the OSQP `verbose` setting is actually controlled through `tsbalancing()` arguments `quiet` and `display_level` (i.e., column `verbose` in a *OSQP settings sequence data frame* would be ignored).

Each row of a *OSQP settings sequence data frame* represents one attempt at solving a balancing problem with the corresponding OSQP settings. The solving sequence stops as soon as a valid solution is obtained (a solution for which all constraint discrepancies are smaller or equal to the tolerance specified with `tsbalancing()` argument `validation_tol`) unless column `require_polished = TRUE`, in which case a polished solution from OSQP (`status_polish = 1`) would also be required to stop the sequence. Constraint discrepancies correspond to $\max(0, l - Ax, Ax - u)$ with constraints defined as $l \leq Ax \leq u$. In the event where a satisfactory solution cannot be obtained after having gone through the entire sequence, `tsbalancing()` returns the solution that generated the smallest total constraint discrepancies among valid solutions, if any, or among all solutions, otherwise. Note that running the entire solving sequence can be *enforced* by specifying `tsbalancing()` argument `full_sequence = TRUE`. Rows with column `prior_scaling = TRUE` have the problem data scaled prior to solving with OSQP, using the average of the free (nonbinding) problem values as the scaling factor.

In addition to specifying a custom-made *OSQP settings sequence data frame* with argument `osqp_settings_df`, one can also specify `osqp_settings_df = NULL` which would result in a single solving attempt with default OSQP values for all settings along with `prior_scaling = FALSE` and `require_polished = FALSE`. Note that it is recommended, however, to first try data frames `default_osqp_sequence` and `alternate_osqp_sequence`, along with `full_sequence = TRUE` if necessary, before considering other alternatives.

Recommended Approach

Start with the default `tsbalancing()` solving sequence (`osqp_settings_df = default_osqp_sequence` and `full_sequence = FALSE`). Then, if more precision is needed, try with:

1. `full_sequence = TRUE`
2. `osqp_settings_df = alternate_osqp_sequence`
3. `osqp_settings_df = alternate_osqp_sequence` and `full_sequence = TRUE`

In practice, specifying `full_sequence = TRUE` should be enough when more precision is needed (at the expense of execution time, obviously). Only in rare occasions should you need to use the `alternate_osqp_sequence` data frame, which will often be even more costly in terms of execution time especially when combined with `full_sequence = TRUE`.

Guiding Principles

The following is a summary of the lessons learned while developing `tsbalancing()` and experimenting with the OSQP solver. They are the guiding principles that lead to both *OSQP settings sequence data* *frames* presented earlier. Note these observations apply to ***time series balancing problems*** as solved by `tsbalancing()` and may not directly apply to other types of *quadratic problems*.

- Data preconditioning options available in OSQP (with the `scaling` setting) are not sufficient for some (badly scaled) problems. External (prior) data scaling (`prior_scaling = TRUE`) is sometimes necessary for OSQP to converge at a decent pace and generate *precise enough* solutions in a reasonable number of iterations.
- Prior data scaling often reduces execution time (the required number of iterations to achieve the specified precision) and greatly increases the likelihood of polished solutions.
- Polished solutions are always very precise, even when prior data scaling is performed (i.e., the solution in the original scale will usually still be *precise enough*).
- While polished solutions from prior-scaled data are usually more precise than unpolished solutions from non prior-scaled data, the most precise solutions correspond to polished solutions from non prior-scaled data.
- Smaller `sigma` and tolerance (`eps_*`) settings result in more precise solutions but take longer to run (require more iterations).
- Enough precision is usually obtained after 10,000 iterations with small values for the `sigma` and tolerance (`eps_*`) settings.
- The default OSQP values for `alpha` and the various settings associated to ρ (`*rho*`) are sufficient (they work well). Reducing the `sigma` and tolerance (`eps_*`) settings and performing prior data scaling is sufficient to obtain precise solutions in a reasonable number of iterations.
- Keeping the same scale between the `sigma` and tolerance (`eps_*`) settings as the default OSQP values works well and is used in both *OSQP settings sequence data frames*, i.e.:
  - `eps_abs = eps_rel = 1,000 * sigma`
  - `eps_prim_inf = eps_dual_inf = 100 * sigma`
  - (and consequently) `eps_abs = eps_rel = 10 * eps_prim_inf = 10 * eps_dual_inf`
- The *machine epsilon* (`.Machine$double.eps`) for the `sigma` and tolerance (`eps_*`) settings, which basically forces the maximum number of iterations, is used as a *last resort* in both *OSQP settings sequence data frames*.

Summary - default sequence (data frame `default_osqp_sequence`)

- Geared towards achieving both fast and precise solutions.
- First try to get (fast) polished solutions on prior-scaled data before attempting solving the original (non-scaled) problem data.
- Make a final attempt on prior-scaled data with the *machine epsilon* for the `sigma` and tolerance (`eps_*`) settings.

Summary - alternative sequence (data frame `alternate_osqp_sequence`)

- Geared towards achieving precise solutions at the expense of execution time.
- Somewhat similar to a *brute force* or *try all* approach.
- Small `sigma` and tolerance (`eps_*`) settings that gradually increase, with the *machine epsilon* as a last attempt.
- Polished solutions are required for every step of the sequence (the best unpolished solution is returned if no polished solution could be obtained at the end of the entire sequence).
- Maximum of 10,000 iterations for every step of the sequence
- First try to get polished solutions from non prior-scaled data (the most precise solutions), then try with prior-scaled data.