CycDesigN

CycDesigN

Printed: July 2022

Table of Contents

This page intentionally left blank

Part I

1 CycDesigN / Introduction

CycDesigN 8.0

A Package for the Computer Generation of Experimental Designs

© CycSoftware Ltd

CycDesigN is a computer package for the generation of optimal or near-optimal experimental designs. The package has been written in Visual C++ and runs under Windows.

A number of design algorithms are used to construct designs; With the exception of spatial designs, all use simulated annealing to produce optimal or near-optimal designs as measured by, where available, theoretically derived upper bounds for the average efficiency factor.

Apart from the wide range of design types on offer in ***CycDesigN,*** there are many other options and facilities such as the specification of new levels for the entries in the design layout, randomization procedures appropriate to each design type, a log file containing details of the ***CycDesigN*** session, files for the design layout and factor indexing, and an extensive help facility.

CycDesigN provides the most comprehensive design generation package yet available for experimenters; particularly those involved in field, glass house, laboratory, clinical and taste testing trials. The design types covered incorporate the most recent developments in research into the construction of experimental designs.

CycDesigN also includes a facility to prepare Genstat (and SAS) code for analysis of core design types..

New in version 8.0

1. A completely new algorithm to construct partially replicated (p-rep) block and row-column designs. In general, designs will have treatments replicated r or r+1 times. This allows a very wide choice of designs. However, for some parameter values equal replicated designs can be constructed.
2. Spatial p-rep row-column designs can also be constructed which aims to have good spatial separation of the duplicated treatments and the standards of each type..

Part II

2 Menu items

2.1 Setup/Working directory

Change the directory where you wish to keep the output from ***CycDesigN***. You can also use the toolbar W icon to change the working directory.

2.2 View/Log file

Select a log file for viewing.You can also use the toolbar L icon to view the log file.

2.3 View/Design file

Select a design file for viewing. You can also use the toolbar D icon to view the design file.

This file records the design parameters and the generated design. A printed copy of this file would be useful for laying out the design for the field experiment. To facilitate this, block designs with a large number of blocks may be printed with the **blocks as rows** rather than columns (default), making it more convenient to print out onto A4 sheets.

2.4 Weights/Factorial

Factorial designs are generated with the aim of optimizing the estimation of main effects and interactions. By default more weight is given to estimating the main effects. You can change these weights to give more weight, for example, to the estimation of the interactions.

For example, consider a three-replicate **resolvable block design** for 4 treatments in blocks of size 2. Suppose the 4 treatments comprise two factors each at 2 levels. Then with default weight of 1.0 for the main effects and 0.25 for the interaction a design will be obtained with average efficiency factors for the main effects of 1.0 and 0.6667 and for the interaction of 0.3333. If the interaction weight is increased to 0.5 then a (balanced incomplete block) design with all average efficiency factors equal to 0.6667 will be generated.

In the factorial weights dialogue window clicking *Reset* will restore the default weights.

2.5 Weights/Row-column

The optimization algorithm aims to maximize the **average efficiency factor** E of a **row-column design** while at the same time trying to ensure that the average efficiencies factors Er and Ec respectively of the **row component** and **column component** designs are as large as possible. This is achieved by using an objective function that is a weighted linear combination of the three average efficiency factors. The default weights give more weight to E than to Er and Ec, and for most situations they are recommended. For small designs, in particular, changing the weights may lead to designs more suited to your needs. Note that, these weights are not used in the construction of partially replicated row-column designs, and unequally replicated non-resolvable row-column designs.

For example, consider the construction of a three-replicate resolvable row-column design for 16 treatments with 4 rows and 4 columns. With weights 1, 0 and 0 a design with E = 0.5696, Er = 0.7548 and Ec = 0.7263 can be obtained. With weights 1, 2 and 2 we get the (lattice square) design with E = 0.5556 which has optimal row and column components Er = Ec = 0.7692. Hence, by increasing the weights on the component designs we have improved component designs but with some loss in E.

In the weights dialogue window clicking *Reset* will restore the default weights.

2.6 Weights/Effects (crossover)

One or more objective functions (**efficiency factors**) are calculated, depending on the model chosen and on whether the treatments comprise a single factor or a combination of a number of factors. The algorithm attempts to maximize a single objective function that is a weighted combination of the individual objective functions. These weights can be changed in the *Weights* menu.

2.7 Design

Under this menu item you can run **resolvable**, **non-resolvable**, **partially replicated** and **crossover** designs.

2.8 Analysis

Select this option to run the ***CycAnalysis*** module to prepare GenStat or SAS code to analyse any of the designs you have generated.

See **CycAnalysis** to see how you can prepare analysis code for the design you have just generated.

2.9 Help/Help

View this help file. You can also use the toolbar ? icon to view the help file. There is also context help where you can select the ? at the top right hand side of the dialog box to get a brief explanation of the inputs required.

2.10 Help/Visit CycDesigN webpage

A link to **http://www.vsni.co.uk/software/cycdesign/**, which can be browsed within the program.

2.11 Help/About

Look at the credits for the ***CycDesigN*** program.

Part III

3 Algorithm

3.1 Interchange algorithm

Consider first the construction of all designs apart from spatial designs. At the setting-up stage, a search for a suitable starting design is carried out. Pairs of treatments are then interchanged and the effect this interchange has on the objective function evaluated. Initially a random *descent* is used, where treatments are interchanged until no further improvements in the objective function are possible. At this point we may have reached a local optimum. Hence, at the next stage simulated *annealing* techniques are used. This means that interchanges that do not improve the objective function can also be accepted, but with decreasing probability as the algorithm proceeds.

For the construction of spatial designs, a non-spatial design is first generated using the annealing methods described above. This design then becomes the starting design for an interchange algorithm that aims to get a good spatial separation of the treatments. While local optima are not a concern with this approach, improvements in the spatial objective function can often result if, as happens automatically in the algorithm, the starting design is perturbed at random and the spatial algorithm run again. This process will continue until terminated by the user.

The current *status* of either algorithm is shown in the **design generation** pane. Interchange treatments are chosen at random, with the random number seed specified in the **seed** window.

3.2 Average efficiency factors

3.2.1 Average efficiency factor

Different **block designs** of the same size can be assessed on the basis of the amount of information available from both within and between blocks. Efficiency factors, obtained from the within block analysis, provide criteria for comparing these designs. The average efficiency factor is obtained by comparing the average variance of the estimates of pairwise treatment differences in the presence and absence of blocking. It can be calculated as the harmonic mean of the canonical efficiency factors of the design. Similarly for **row-column designs**, with average variances compared in the presence and absence of rows and columns.

3.2.2 Upper bound

In the search for efficient designs it is useful to have an upper bound for the average efficiency factor as a measure against which to assess the scope for possible improvement. The algorithm can be stopped, for instance, when a design is found which is sufficiently close to the bound.

3.2.3 Upper bound percent

Gives the average efficiency factor as a percent of the upper bound.

3.3 Seed

The design generation algorithm uses a random number seed to initiate the **treatment interchanges**. This seed can be taken from the computer clock or specified by the user. In either case the value of the seed is printed in the log file.

3.3.1 Seed number

Enter a positive integer in this box for the seed that starts random number generation. This will be the seed for random numbers until it is changed, thus allowing generation of the same design.

3.3.2 Seed from user

You can enter your own random number seed.

3.3.3 Seed from clock

The random number seed is taken from the computer clock as the seconds past a fixed time and date.

3.4 Other

3.4.1 Viewing the design

Gives additional information about the current best design.

For **block designs**, and for the first stage of construction of **row-column designs**, the **average efficiency factor** and its **upper bound** are given. The number of concurrences are also given. For example, 36 concurrences of 1 means that 36 pairs of treatments occur together in exactly one block (column).

For resolvable row-column designs and non-resolvable row-column designs with equal treatment replications, the **average efficiency factor** and upper bound are given for the row and column component designs as well as for the row-column design.

For non-resolvable row-column designs with unequal treatment replications and for **p-rep** row-column designs, the **average efficiency factor** and upper bound are only given for the row-column design.

In all cases the concurrences of both row and column components are given.

For a **spatial design** in addition to the information above:
(i) Neighbour balance properties and the type and number of treatment self-adjacencies are listed for all resolvable and/or row-column designs
(ii) The spread of treatment replications over rows and columns is evaluated for all non-resolvable and p-rep row-column designs

3.4.2 Time

Gives the cumulative time taken in seconds to generate designs.

3.4.3 Status

For **single location p-rep designs**, status initially shows **Standards not optimal** and then **Standards optimal.** For **nested designs** status first shows that the algorithm is running then that **optimal nesting** is achieved. For all other non-spatial designs status shows whether the design generation algorithm is running, has reached the optimal theoretical efficiency, or has stopped. For spatial designs, status shows that the design generation algorithm is running, and will continue running until the user clicks Next>.

3.4.4 Next

Pressing this button starts the algorithm to produce the design

Generating the design

Part

IV

4 Generating the design

4.1 Resolvable/Non-resolvable designs

Resolvable and non-resolvable designs can be generated.

The algorithms automatically run the **cyclic** and **alpha** algorithms, switching to the more general algorithm, formerly called "*Other*" for further improvements in the average efficiency factor.

4.1.1 Unequal block Designs

For resolvable block designs where the number of treatments v is not a multiple of the block size k then, in each replicate, some of the blocks will be of size k while the remaining blocks will be of size k-1.

Example

4.1.2 Non-resolvable design

In a resolvable **block design** groups of incomplete blocks of size k are formed so that each treatment is replicated exactly once in each group. If the design cannot be resolved into replicate groups then it is a non-resolvable block design .

A non-resolvable **row-column design** has v treatments set out in a two dimensional array of k rows by s columns. The treatments can be replicated equally or unequally with the sum of the treatment replication numbers adding to ks. However, a non-resolvable design with a mixture of unreplicated and replicated treatments would be best run as an augmented p-rep designs.

Example

4.1.3 Resolvable design

In a resolvable **block design** for $v = ks$ treatments, groups of s incomplete blocks of size k are formed so that each treatment is replicated exactly once in each group. If v is not a multiple of s, the s incomplete blocks in each group will have **unequal block sizes**. For resolvable **row-column designs** the $v = ks$ plots in each replicate are arranged in k rows and s columns. Resolvable designs are needed if it is required to allow for possible differences between blocks, or between rows and columns, within replicates. It is often useful in practice to perform or measure an experiment one replicate at a time. Accuracy will also be increased if the experimental material can be arranged so that replicates are relatively homogeneous. Gains in efficiency can further be achieved by resolvable designs when inter-block information is recovered in the analysis.

If the design cannot be resolved into replicate groups then it is non-resolvable.

Example

4.1.4 Block/row-column design parameters

This button provides access to another window where you can enter the parameters for a block design or a row-column design.

4.1.4.1 Block design parameters

In a non-resolvable **block design** the *v* treatments are set out in *b* blocks each of size *k*. The treatments can be replicated equally or unequally with the sum of the treatment replication numbers adding to *ks*.

For **resolvable designs** if $v = ks$ for some integer *s* then in each replicate the treatments are set out in *s* blocks of size *k*. If *v* is not divisible by *k* the blocks will be of unequal size, usually of size *k* and *k*-1.

4.1.4.2 Row-column design parameters

In a non-resolvable **row-column design** the *v* treatments are set out in *k* rows and *s* columns. The treatments can be replicated equally or unequally with the sum of the treatment replication numbers adding to *ks*.

For **resolvable designs**, each replicate has the *v* treatments set out in *k* rows and *s* columns. Since each treatment occurs once in each replicate, the parameters must satisfy the condition that

$$v = ks$$

4.1.4.3 Row component

In a **row-column design** the **block design** given by the rows of the design is called the row component.

4.1.4.4 Column component

In a **row-column design** the **block design** given by the columns of the design is called the column component.

4.1.5 Block/Row-column design

This button toggles between a **block design** and a **row-column design**.

4.1.5.1 Block design

In a block design the *v* treatments are set out in *b* blocks each of size *k*. or in the case of unequal block sizes, some blocks will be of size *k*+1. Dividing the material into relatively homogenous blocks is an important technique for improving the precision of an experiment.

Example

4.1.5.2 Row-column design

In a row-column design the experimental units are grouped in two directions, i.e. two blocking factors are used with one factor representing the rows of the design and the other factor representing columns. It is hoped that there will be a gain in the accuracy of estimating treatment comparisons from eliminating the effects of the row and column factors. The v treatments are now set out in an array of k rows and s columns.

Example

4.1.6 Single factor/factorial

This button provides access to **single factor** and **factorial** designs.

4.1.6.1 Single factor

This treatment structure should be chosen for experiments that involve the allocation of a single set of treatments to plots. For example, diets in a nutritional experiment, seedlots in a variety trial, or nitrate levels in a fertilizer experiment. It is assumed that all treatments are of equal interest.

4.1.6.2 Number of treatment groups

A **single set of treatments** can have a nested treatment structure. For example, seedlots in a variety trial can come from a number of provenances; hence seedlots are nested within provenances. The number of nested treatments in each group must be specified; these numbers need not be equal, but must sum to v. For example, the first six seedlots might be from the first provenance, the next four seedlots from the next provenance and so on. In general, designs are generated which have the nested treatments occurring as equally as possible in blocks, or in rows and columns.When the best arrangement of the treatment groups is found, the **status** of the running design will change from "Running" to **Optimal nesting** and further designs generated will be optimally nested.

Example

4.1.6.3 Factorial

More complex treatment structures arise when the experiments involve more than a **single set of treatments**. For example, when treatments involve combinations of different amounts of nitrate and potash in a fertilizer experiment. Now the effects of differing amounts of nitrate and potash could be examined separately; the main effects of the factors. In addition the effectiveness of nitrate at different levels of potash could be studied, and vice versa; the interactions between factors. Designs are generated with the aim of optimizing the estimation of main effects and interactions. The package allows the generation of factorial designs for **resolvable designs** and **non-resolvable designs**, provided treatments are equally replicated.

Example

4.1.6.4 Number of factors

A factorial experiment involves combinations of a number of treatment factors, such as nitrate and potash in a fertilizer experiment. The number of factors has to be specified as well as the number of levels of each factor. There may be, for instance, three separate levels of the nitrate factor. The total number of factor combinations is equal to the number of treatments v, so that the product of the factor levels must be equal to v.

4.1.7 Latinization

A **resolvable design** is not latinized if the blocks within each replicate are randomized separately for each replicate. Such a randomization process is appropriate for an experimental layout where the replicates are separate entities. Then the blocks of one replicate have no relation to those of another.

For a resolvable block design, the design is latinized with m contiguous columns and p groups of long block replicates. For a resolvable row-column design, the design is latinized with m contiguous columns, n contiguous rows and p groups of long column replicates. If $m = 0$ then the columns are not latinized, and if $n = 0$ the rows are not latinized.

Example

4.1.7.1 Blocks not latinized

The blocks of a **resolvable block design** are not latinized if they are randomized separately for each replicate. Such a randomization process is appropriate for an experimental layout where the replicates are separate entities. Then the blocks of one replicate have no relation to those of another.

4.1.7.2 Rows not latinized

The rows of a **resolvable row-column design** are not latinized if they are randomized separately for each replicate. Such a randomization process is appropriate for an experimental layout where the replicates are separate entities. Then the rows of one replicate have no relation to those of another.

4.1.7.3 Columns not latinized

The columns of a **resolvable row-column design** are not latinized if they are randomized separately for each replicate. Such a randomization process is appropriate for an experimental layout where the replicates are separate entities. Then the columns of one replicate have no relation to those of another.

4.1.7.4 Latinized by blocks

A **resolvable block design** is latinized if the blocks of one replicate are next, or contiguous, to the blocks of another replicate. This is so that variation between blocks and over replicates can also be allowed for in the analysis. The blocks extending over all replicates can then be regarded as s long blocks each of rk treatments. If r is less than or equal to s the aim is to ensure that each treatment occurs no more than once in each long block. More generally, the latinized property is that long blocks will be such that each treatment occurs as equally as possible in each long block. In partially-latinized designs the replicates are set out in groups, and the latinized property then applies to the long blocks within each replicate group.

Example

4.1.7.5 Number of contiguous blocks

The **latinized property** property relates to each replicate group long blocks. This property can be applied to sets of *t* contiguous blocks resulting in *t*-latinized designs. A **resolvable block design** is *t*-latinized if each treatment occurs as equally as possible in each set of *t* long blocks in each replicate group. The number of contiguous blocks is set equal to *t*.

4.1.7.6 Partially latinized designs

In a partially-latinized design the replicates are broken up into groups of replicates, with the replicates in each group set out under each other.. The number of replicates in each group must be specified; these numbers need not be equal, but must sum to *r*. The configuration of these groups, and the latinization chosen, can be seen by clicking the *layout button*.

4.1.7.7 Replicate layout of block designs

For a latinized resolvable block design, the *r* replicates are contiguous and form long blocks of *rk* plots. Then the **latinized property** applies to the long blocks within each replicate group. The replicate layout gives the configuration of these groups. The number of groups must be specified.

Example

4.1.7.8 Latinized by columns

A **resolvable row-column design** is latinized by columns if the columns of one replicate are next, or contiguous, to the columns of another replicate. This is so that variation between columns and over replicates can also be allowed for in the analysis. The columns extending over all replicates can then be regarded as *s* long columns each of *rk* treatments. If *r* is less than or equal to *s* the aim is to ensure that each treatment occurs no more than once in each long column.. More generally, the latinized property is that long columns will be such that each treatment occurs as equally as possible in each long column. In partially-latinized designs the replicates are set out in $c < r$ column groups, and the latinized property then applies to the long columns within each of the *c* column groups..

Example

4.1.7.9 Latinized by rows

A **resolvable row-column design** is latinized by rows if the rows of one replicate are next, or contiguous, to the rows of another replicate. This is so that variation between rows over replicates can also be allowed for in the analysis. The rows extending over all replicates can then be regarded as k long rows each of rs treatments. If r is less than or equal to k the aim is to ensure that each treatment occurs no more than once in each long row. More generally, the latinized property is that long rows will be such that each treatment occurs as equally as possible in each long row. In partially-latinized designs the replicates are set out in groups, and the latinized property then applies to the long rows across groups of replicates.

To construct a row latinized with $c < r$ long row groups it will be necessary to swap rows and columns, run it as a column latinized design with c long column groups, and then re-orientate the resulting design by 90 degrees.

Example

4.1.7.10 Number of contiguous columns

The **latinized property** relates to each group of long columns. This property can be applied to groups of t contiguous columns resulting in **t-latinized designs**. A **resolvable design** is t-latinized with respect to columns if each treatment occurs as equally as possible in each set of t long columns in each replicate group. The number of contiguous columns is set equal to t.

4.1.7.11 Number of contiguous rows

The **latinized property** relates to each group of long rows. This property can be applied to groups of t contiguous rows resulting in **t-latinized designs**. A **resolvable design** is t-latinized with respect to rows if each treatment occurs as equally as possible in each set of t long rows across groups of replicates. The number of contiguous rows is set equal to t.

4.1.7.12 Replicate layout of row-column designs

For a latinized design the r replicates are contiguous and form long columns of rk plots. In a partially-latinized design the replicates are broken up into separate groups of replicates, with the replicates in each group set out under each other. Then the **latinized property** applies to the long columns within each replicate group and/or to the long rows across groups of replicates. The replicate layout gives the configuration of these groups. The number of groups must be specified.

Example

4.1.7.13 Number of column groups equal to number of replicates

In a resolvable row-column design latinized by columns, if the number of column groups is equal to the number of replicates r then each group will comprise a single replicate. The replicates are not, therefore, contiguous and thus there is no latinization in the column direction. That is, the design is not latinized and should be run as such.

If a design latinized by both columns and rows is specified with *r* column groups then again there is no latinization in the column direction. Hence the design required is a row latinized design. In CycDesigN a row latinized design can only be constructed if column latinization has also been specified. For this reason you will need to swap rows and columns, run it as a column latinized design with one column group, and then re-orientate the resulting design by 90 degrees.

Example

4.1.8 Stages

Row-column designs can be generated in **one stage** or **two stages**.

4.1.8.1 One stage

The construction of the row-column design is carried out simultaneously. That is, the design is obtained from a single run of the algorithm. This approach is recommended for small to moderate sized designs. For large **resolvable designs**, the **two stage** method of construction can produce efficient designs more quickly, giving a very good column design.

4.1.8.2 Two stages

The construction of a **row-column design** for *v* treatments in *k* rows and *s* columns is carried out in two stages. First, a **block design** is constructed for *v* treatments in blocks of size *k*. These blocks then become the columns of the row-column design. At the second stage the row-column design is obtained by fixing columns, and allowing **treatment interchanges** to take place within these columns. Note that for large designs the two stage method of construction can be considerably quicker than the one stage. The first stage of construction is completed either when the **average efficiency factor** of the design is equal to the **upper bound**, or when the user intervenes by clicking the *Next button*.

4.1.9 Type of design algorithm

In the search for good designs the algorithms used are dependent on the type of design required.

In the case of single factor non-resolvable designs the search begins with finding a good **cyclic design**, and the algorithm then automatically moves on to try to find more efficient non-cyclic designs.

Similarly, for single factor resolvable designs, first a good alpha design is found and then the search automatically proceeds to find more efficient non-alpha designs.

Finally, for **single location p-rep designs** a good design based on an alpha design is found and then the search automatically proceeds to try to find more efficient non-alpha designs.

4.1.9.1 Cyclic design

Cyclic designs are **block designs** for *v* treatments, with *k* treatments per block and *r* replications of each treatment. They consist of combinations of one or more cyclic sets, where each set is obtained by cyclic development of an initial block. Cyclic designs exist for any combination of parameters that satisfy the relationship that $vr = bk$, where b is the number of blocks. As cyclical methods of construction are involved, they can be generated very quickly. However, if you let the search continue, especially for small to moderate sized designs, the more general algorithm which follows the initial stage will often give further improvements.

Example

4.1.9.2 Alpha design

Alpha designs provide a general series of **resolvable designs** for $v = ks$ treatments. For these **block designs** there is no limitation on block size other than the unavoidable constraint that v/k must be an integer. As cyclical methods of construction are involved, they can be generated very quickly. As with cyclic designs, if you let the search for designs continue, the more general algorithm will of show improvements to the initial stage especially for small to moderate sized designs.

Example

4.1.9.3 More general designs

After the **cyclic design** or **alpha design** stage, further searching using a general algorithm will often lead to further improvements, especially for small to moderate sized designs

4.1.9.4 BIB

A balanced incomplete block (BIB) design has every pair of treatments occurring together in exactly λ blocks. A necessary, though not sufficient, condition for the existence of a BIB is that λ satisfies $\lambda\,(v\text{-}1) = r(k\text{-}1)$, where v is the number of treatments, k the block size and r the number of treatment replications. A BIB design is optimal and CycDesigN will instantly generate all known BIB designs with r<10.

The **dual design** of a BIB is optimal and will also be generated instantly by CycDesigN.

4.1.9.5 Dual design

The dual design of a **block design** is obtained by interchanging treatment and block labels in the (primal) design. The **average efficiency factor** of a dual design is a monotonic function of the average efficiency factor of the primal design. When the number of treatments is greater than the number of blocks the use of the dual design will generate designs more quickly. This approach is used in CycDesigN.

4.2 Partially replicated designs

A partially replicated (p-rep) design is a block or row-column design for v treatments at c locations, such that each treatment appears at most twice at each location. The number of replications is r, which in general means that some treatments are replicated r times and others r+1 times. For some parameter choices equal replicated designs can be constructed.

In addition to the treatments, standards can be incorporated into the designs. Different types of standards can also be specified.

Spatial designs can also be constructed for p-rep row-column designs.

Example

4.2.1 Standards

A number of standard types can be specified. For each type, a number of standards can be chosen. The standards are set out in each location such that they occur as equally as possible in the blocks of a p-rep block design or in the rows and columns of a row-column design. For example, if there are 6 standards (of a given type) and 4 rows then that standard will occur once in two of the rows and twice in the other two rows. In the output, standards of type 1 are denoted by S1, those of type 2 by S2 and so on.

4.2.2 Design parameters

Pressing the design parameters button displays a dialog box to input the parameters.

On the button are the current parameters as [v, t, c, r, k, s, d]

where v is the number of treatments
t is the number of standard types
c is the number of locations
r is the number of treatment replications, across all locations
k is the number of units per block, or the number of rows in a row-column design
s is the number of blocks, or the number of columns in a row-column design
d is the number of duplicates per location

For row-column designs the restriction $c(k\text{-}1)(s\text{-}1) - (v+t\text{-}1) > 0$ is placed on these parameters to ensure that the constructed design is connected. For block designs the restrictions is $cs(\text{k-}1) - (v+t\text{-}1) > 0$.

4.2.3 Number of replications

The number of replications is r, which in general means that some treatments are replicated r times across all the locations while others are replicated r+1 times. The choice of r will be in the range $\max(c\text{-}3,0) < r < 2c$. For example, if the number of locations is 5 then r must be a value between 3 and 9.

For some parameter choices equal replicated designs can be constructed.

4.2.3.1 Equal replication

If v is the number of treatments, c the number of locations, r the number of treatment replications, across all locations and z the total number of standard treatments per location, summed over each standard type, then an equal replicated design, where all treatments are replicated r times, will be obtained if c divides vr and

$$ks = (vr/c) + z.$$

where for block designs k is the number of units per block and s the number of blocks, and for row-column designs k and s are the number of rows and columns respectively.

For example, if $v = 50$, $c = 5$, $r = 3$ and $z = 6$ then $ks = 36$. An equal replicated design will be constructed if $k = 4$, $s = 9$ or $k = 3$, $s = 12$ or $k = 6$, $s = 6$.

An equal replication design for a single location will be called an augmented p-rep design,.

4.2.3.2 Augmented p-rep designs

An augmented p-rep design is a design for a single location (c=1) where all treatments are unreplicated. Such designs satisfy

$$v + z = ks$$

where v is the number of treatments and z the total number of standards across all locations. Also for block designs k is the number of units per block and s the number of blocks, and for row-column designs k and s are the number of rows and columns respectively.

It is important to note that no treatments are duplicated in such designs. That is, **all treatments are unreplicated**. Hence, in order to obtain a connected design it will be necessary to have a relatively large number of standards. As a rough guide the total number of standards z needs to be greater than $y = k + s + t - 3$ where t is the number of standard types. The larger z is relative to y the more likely it is that connected designs will be obtained. For example, suppose v = 179, t = 1 and z = 37. Then an augmented p-rep design may be possible if $ks = 216$. One possible choice would be $k = 6$ and $s = 36$ but since z < 40 no connected design can be constructed. However, such designs will be available if $k = 9$ and $s = 24$ or $k = 12$ and $s = 18$.

It usually takes time to find good row-column augmented p-rep designs when v is large, say $v >$ 400. The initial starting design will usually be obtained quickly but further improvements may take a few minutes. On occasions the algorithm may get stuck at local optima so that if no further designs are generated after, say, 5 minutes then it would be best to re-run the design with a different seed. One example is given by $v = 670$, $t = 2$ standards with 50 and 30 standards respectively and $k = 50$ and $s = 15$. Now z = 80 and y = 74 so connected designs should be possible. More often than not good designs will be generated after 2 or 3 minutes, but sometimes it may fail to make any progress (so run again with a different seed).

No such problems occur with block designs and good augmented p-rep designs should be obtained quickly, as long as *z* is large enough.

Example

4.2.4 Number of blocks, or columns

Once the number of treatments *v*, the number of locations *c*, the number of replications *r*, the total number of standards *z* and the number of units per block (or rows) *k* have been input then the number of blocks (or rows) *s* can then be chosen from a range of values. This range is between, say, s_{min} and s_{max}.

If $s = s_{min}$ then a design will be obtained with the greatest number of treatments replicated *r* times. In some cases, this could mean than all treatments are replicated *r* times; that is an equal replicated design will be constructed. As *s* increases more and more treatments will be replicated *r*+1 times, and less replicated *r* times. The number replicated *r*+1 times will be maximized when $s = s_{max}$.

For example, if $v = 36$, $c = 4$, $r = 4$, $z = 5$ and $k = 6$ then $s_{min} = 7$ and $s_{max} = 8$. With $s = 7$, 32 treatments will be replicated 4 times and 4 replicated 5 times. While if $s = 8$, 8 treatments will be replicated 4 times and 28 replicated 5 times.

As another example, suppose $v = 60$, $c = 5$, $r = 3$, $z = 9$ and $k = 5$ then $s_{min} = 9$ and $s_{max} = 11$. Choosing $s = 9$ will produce an equal replicated design with all treatments replicated 3 times. With $s = 10$ and 11 then 35 and 10 treatments respectively will be replicated 3 times, and the remainder replicated 4 times.

4.3 Crossover designs

A crossover design is used in experiments that involve the application of sequences of treatments (such as drugs or other stimuli) to a group of subjects over a number of successive time periods.

4.3.1 Crossover experiments

A crossover experiment involves the application of sequences of treatments to several subjects over a number of time periods. The observation made on each subject at the end of a time period may depend on the **direct effect** of the treatment applied in the current period, and the **carry-over effects** of the treatments applied in one or more previous periods. Various **models** have been proposed to explain the nature of the carry-over effects. Crossover designs are also known as change-over designs or carry-over designs.

Example

.

4.3.2 Overview of the input

The steps involved in generating a crossover design are:

1. Specify the **design parameters** by selecting the *Design parameter*s *button*.

2. Further options included in allow you to specify the number of **replications** of each treatment, the **model,** the **correlation structure**, whether its **single or multi-factor**, the **weights** given to the pairwise variances of the **direct effects**, and whether to **augment** an existing design.

3. Click the ***Optimize*** button to optimize the allocation of treatments to **subjects** or to both **subjects and periods**

4. Choose the random number by pressing the ***Seed*** button. The seed can be generated from the computer **clock** by checking the Clock box, or you can specify the seed by entering a number into the Seed box.

5. **Generate designs** by clicking the **Next** button in the at the bottom of the dialog after you have selected your design parameters, model, etc. to generate the design. The algorithm is an **interchange algorithm** in which pairs of treatments are interchanged at random to see whether they improve the design.

6. Click the ***Next*** button when you want to stop the algorithm.

7. The *Output* dialog appears and options allow you to view the pairwise variance of the **direct effects**. You can specify the name of the **log** and **design** files (the default is *design*).

4.3.3 Design parameters

A crossover experiment involves the application of sequences of treatments to each of a number of subjects or groups of subjects over a number of time periods. The numbers of treatments, periods and subjects are the design parameters. Each treatment will be replicated a number of times in the experiment. By default these **replications** are set to be as equal as possible, but the user can change them.

4.3.4 Models

The crossover design depends on the model selected to explain the nature of the carry-over effects. The following models are available: **additive**, **self-adjacency**, **proportionality**, **placebo**, **no carry-over into self**, **treatment decay**, **interaction** and **second-order**. In addition a model with no carry-over effects can be chosen.

4.3.5 Correlation structure

In a crossover experiment sequences of treatments are given to subjects. It can be assumed that the responses obtained on different subjects are uncorrelated. However, it is possible that the responses on the same subject could be correlated. Such correlations should be considered in both the design and analysis of the crossover experiment. There two types of correlation structures that are available; **linear variance** and **exponential variance**.

4.3.6 Factorial experiments

In a **crossover experiment** the treatments can be combinations of one or more treatment factors. The default is a single factor. With more than one factor, we have a factorial experiment. For instance, in a clinical trial the 4 treatments could be combinations of two factors, such as drug formulation and dosage, each at two levels.

4.3.7 Augmenting designs

Sometimes in a clinical (or other) trial it may be necessary to modify or extend an ongoing trial. The algorithm provides the facility to augment the crossover design by fixing a number of periods and then restricting treatment interchanges to the remaining periods. For example, after a few periods have been completed, one of the treatments may be dropped from the trial, so that it is then necessary to re-allocate the other treatments to the remaining periods of the trial. Alternatively, the trial can be extended by the addition of further periods.

Examples

Consider the following example of a crossover design for 4 treatments, 5 periods and 8 subjects using the additive model.

```
                           Subject
   Period     1     2     3     4     5     6     7     8

      1       3     4     2     1     2     3     1     4
      2       4     1     1     3     3     4     2     2
      3       4     3     1     3     1     2     4     2
      4       1     2     4     2     4     1     3     3
      5       2     2     3     4     4     1     3     1
```

After running the first 2 periods, it is decided to drop treatment 3 from the remainder of the experiment. Clicking the Augment design button will open a window where you can enter the number of fixed periods as 2 and the number of periods (in the augmented design) as 5. The above design will have been stored as a text file in your working directory, and this file will now be specified in the input file name box. Clicking Next> will open a replications window where you enter the number of replications of each treatment in the 3 remaining periods of the design. There are 24 cells to be filled so that the sum of these replications must be 24. In this case the default replications are 6 for each treatment but these can now be changed to 8 for treatments 1, 2 and 4 and to 0 for treatment 3 (since this treatment is to be dropped from the experiment). An augmented design can now be generated, such as the one below.

```
                          Subject
```

```
Period     1      2      3      4      5      6      7      8

   1       3      4      2      1      2      3      1      4
   2       4      1      1      3      3      4      2      2
   3       1      1      4      2      1      2      4      2
   4       2      2      4      2      4      1      4      1
   5       4      2      1      4      4      1      2      1
```

Note that the first two periods of these two designs are the same, as they should be as these two periods have already been run. In the remaining 3 periods the 24 cells have been allocated equally to the treatments 1, 2 and 4. This allocation has been made with the aim of maximising the average efficiency factors of the direct and carry-over effects.

As a second example, consider the following crossover design for 8 treatments, 4 periods and 6 subjects again using the additive model.

```
                      Subject
Period     1      2      3      4      5      6

   1       4      1      3      2      8      5
   2       1      4      2      7      5      6
   3       7      4      6      5      3      8
   4       2      3      7      8      6      1
```

After running 3 periods it is decided to extend the experiment with a further 2 periods. In Augment design the number of fixed periods is set to 3 and the number of periods to 6. The 8 treatments have now to be re-allocated to 18 cells in periods 4 to 6. We can do this by replicating 6 treatments twice and the other two three times. Note that in the 3 fixed rows in the design above treatments 4 and 5 are replicated three times, so we shall only replicate these treatments twice in the augmented design. Hence, we shall choose to replicate treatments 7 and 8, say, three times and the other treatments twice. An example of such an augmented design is

```
                     Subject
Period     1      2      3      4      5      6

   1       4      1      3      2      8      5
   2       1      4      2      7      5      6
   3       7      4      6      5      3      8
   4       7      3      5      8      4      1
   5       2      8      7      3      6      4
   6       5      2      8      6      1      7
```

The 3 fixed periods are the same in both designs, as they should be. In the augmented design treatments 1, 2, 3 and 6 are replicated four times while treatments 4, 5, 7 and 8 are replicated five times. In this example unequal replication cannot be avoided.

4.3.8 Optimizing subjects and periods

If the number of treatments is equal to the number of subjects it may be desirable for each subject to receive each of the treatments exactly once. More generally, a crossover design may be required in which each subject receives each treatment as equally as possible. An option in the algorithm allows the user to optimize the allocation of treatments to subjects to achieve this requirement; there is no guarantee that this will happen automatically. As an example consider the design of a control-treatment trial with one control and four treatments, to be set out in 4 periods with 8 subjects. It is required that each subject receives the control and three of the four treatments. Replicating the control 8 times and each of the treatments 6 times and then optimizing subjects can achieve this. If it is also desirable for each treatment to occur as equally as possible in each period, then there is an option to optimize the allocation of treatments to both subjects and periods. There may be some loss in efficiency, especially from imposing both of these constraints.

4.3.9 Objective function

In an interchange algorithm, design criteria are required that can be used as the basis for choosing efficient designs. A simple criterion, and one that is generally used, is to minimize the average variance of the estimated pairwise differences between treatments. However, in most of the carry-over models there is more than one set of treatments to consider. For instance, in the **additive model**, a separate criterion can be calculated for both the direct and carry-over effects. One, two or three criteria can be defined depending on the model chosen. In the algorithm these different criteria are combined into a single criterion by taking some suitable linear combination of them. Since the numerical values of each criterion may be very different, they are standardized and called the **efficiency factors**. These efficiency factors are shown in the **design generation** pane. The objective function is a weighted combination of the efficiency factors of the design, and the aim is to find a design that maximizes this function.

4.3.10 Design generation

Once the design parameters, models and other options (if any) are chosen, design generation commences when the *Next* button is clicked. The chosen model is shown as the heading to design generation window, and the efficiency factors displayed for the generated designs. Design generation is terminated when the *Next* button is clicked.

4.3.11 No design found

Sometimes the algorithm will be unable to generate a crossover design. This will probably be because you have imposed conditions on the design that are too restrictive. Your design may be too small for the **model** chosen. Increase the number of periods or subjects or both. Or choose a different model. If you have chosen to **optimize both subjects and periods** then try **optimizing subjects** only, or leave both optimizing boxes unchecked. If you have chosen your **treatment replications** or **weights** parameters, try changing these values. Generally, if no design is generated, try relaxing some of the conditions you have imposed.

4.4 Spatial designs

A spatial **block design** or r**ow-column design** takes into account the separation of different treatments in blocks (rows and columns). For example in a spatial design if two treatments are close together in a block (row or column) in one replicate, they are likely to be more separated in other replicates. Also a spatial design tries to minimize the number of times pairs of treatments are in adjacent plots. Three criteria are used in the construction of spatial designs, namely **neighbour balance**, **spatial neighbours** and **treatment spans**.

4.4.1 Construction

For all design types, a non-spatial design is initially obtained. This is called the starting design. A spatial design is then constructed using an **interchange algorithm** that calculates a **spatial score** and an **average efficiency factor** E. The starting design is then perturbed, at random, and the algorithm run again to see if the spatial score and the average efficiency can be improved. This continues until the user intervenes to stop the program.

If the starting design is optimal or near-optimal then any further treatment interchanges will generally result in a reduction in the average efficiency factor. That is, there is a trade-off between the need for a spatial separation of treatments and E. For some designs a compromise is made so that some degree of separation is sacrificed so that E doesn't decrease too much.

4.4.2 Design types

Spatial designs can be constructed for resolvable block and row-column designs, non-resolvable row-column designs, and single location and multi-location row-column p-rep designs. For some design types there are restrictions on the range of design parameters that can be used, or on the treatment structure of the designs.

4.4.3 Neighbour balance (NB)

One criterion used in the construction of spatial designs is the neighbour balance (NB) score. It aims to minimize the number of times pairs of treatments are in adjacent plots. Consider the rows of a design. If a pair of treatments is next to each other in adjacent columns in more than one row then it contributes to the NB score. In general if there are n row neighbour occurrences of a pair of treatments this will contribute n(n-1)/2 to the score. For instance, if a pair of treatments appears at most once as row neighbours the score is 0, if the pair appear twice the score is 1 and if the pair appears three times the score is 3. The NB score is then obtained by summing over all treatment pairs, and is used in the overall objective function.

4.4.4 Spatial neighbours

A spatial design also tries to minimize the number of treatment self-adjacencies. In the diagram below consider the treatment, X say, in the cell marked with an asterisk *. Self-adjacencies are said to occur if treatment X also occurs in one of the cells marked with the numbers 1 to 6.

		3		**3**		

	4	2		2	4	
6	5	1		1	5	6
			*			
6	5	1		1	5	6
	4	2		2	4	
		3		3		

The definition of these self-adjacencies is given in the table

Type of self-adjacency	Description
1	Single diagonal
2	Knight move in column (NS) direction (two plots in column direction, one plot in row direction)
3	Stretched knight move in column (NS) direction (three plots in column direction, one plot in row direction)
4	Double diagonal
5	Knight move in row (EW) direction (two plots in row direction, one plot in column direction)
6	Stretched knight move in row (EW) direction (three plots in row direction, one plot in column direction

If the design is binary in both row and column components then self-adjacencies cannot occur in the same row and column of the cell marked with *. If a component is non-binary then such a self-adjacency is possible but highly unlikely in the designs considered.

4.4.5 Treatment spans

The third spatial criterion used in the construction of spatial designs is the minimum block, row and column treatment spans. A treatment span is the difference between the first appearances of that treatment in a row, say, of the design and its last appearance. For example if the replications of a treatment are in row 3, 4, 7 and 9 then the row treatment span for that treatment is 7 (9-3+1). The minimum treatment span will then be equal to the smallest of these treatment spans.

The spatial algorithm will aim to have the minimum span equal to or exceed a certain value. What that value is depends on the design type.

4.4.6 Spatial score

In the run dialog window for spatial designs a spatial score is output. This is a function that involves the **NB score** and a score based on both the **spatial neighbours** and **treatment spans**. The form of this function depends on the type of design generated. The important thing to note is that the lower the score the greater the spatial separation in the design.

4.4.7 Resolvable block designs

For non-latinized designs the only measure of spatial separation used is the **neighbour balance** (NB) score; it is the only meaningful measure. For latinized designs, in addition to the NB score, the objective function includes the single diagonal and N-S knight move **spatial properties**.

Spatial resolvable block designs are not available if the treatment structure is **nested** or **factorial**.

4.4.8 Resolvable row-column designs

For spatial non-latinized row-column designs the neighbour balance (NB) score is calculated in both the row and column directions. The objective function used is the sum of these two scores. Resolvable latinized row-column designs can be **latinized by columns**, or in addition **latinized by rows**. Further there can be **contiguous columns** and **contiguous rows**, and **partially latinized designs** with contiguous groups of replicates. Spatial designs can be constructed for all these different types of latinization. The objective functions involve the NB score in both direction as well as single diagonal, N-S and E-W knight move neighbours, as appropriate.

Spatial resolvable row-column designs are not available if the treatment structure is **nested** or **factorial**.

Example

4.4.9 Non-resolvable row-column designs

Spatial non-resolvable row-column designs are probably most suitable in practice when the number of rows (k) is much less than the number of treatments (v). This is because of the need for relatively homogenous plots in each column and since in many trials plots tend to be long and narrow. There are less restrictions on the number of columns (s) but often they are also less than or equal to v.

For these reasons, one focus in the construction of the spatial design has been on the row **neighbour balance** (NB) score, i.e. on the number of times pairs of treatments are adjacent in rows. At the same time the program aims to minimize the **spatial neighbours** of the designs in terms of single diagonal, knight move neighbours and so on.. For **treatment spans** the program will aim to ensure that each treatment spans at least KP rows and SP columns, where KP = [2k/3] and SP = [2s/3].

If a spatial design is chosen with s < k then the NB score will refer to the occurrence of treatments that are adjacent in columns.

For very small designs there could be quite large variations in the spatial spread of treatments depending on the random number seed chosen for design generation. It is suggested, therefore, that in such situations a few designs are generated using different seeds and the design with the best spatial spread chosen..

Spatial non-resolvable row-column designs are not available if the treatment structure is **nested** or **factorial**

Example

4.4.10 Partially replicated row-column designs

For a spatial p-rep row-column design the algorithm aims to have good spatial separation of both standards and duplicated treatments.

There can be a number of standard types each with a specified number of standards. For each standard type, the algorithm aims to set out the standards so that the columns spans are not less than $[s/n(i)]$ where $n(i)$ is the number of standards of type i. Similarly, the rows spans are not less than $[k/\mathrm{n}(i)]$. The parameters k and s are the number of rows and columns in each location.

For the duplicated treatments the algorithm will aim to ensure that the treatments spans for rows and columns will be kp and sp respectively. If $k<=s$, $kp=[(k-1)/2]$ and $sp=[(s+1)/2]$, otherwise $kp=[(k+1)/2]$ and $sp=[(s-1)/2]$. If kp or sp, or both, are equal to 1 then the value is increased to 2. If designs with spans of size kp cannot be found then kp is reduced by one until designs can be found; similarly for sp.

Example

Randomizing
the design

Part V

5 Randomizing the design

5.1 Randomize

The package first generates an optimal or near optimal design which can then be randomized for use in practice. This involves a randomization of both blocking and treatment factors in a way that preserves the design properties. For example in a **resolvable block design**, the order of the replicates, blocks within replicates and plots within blocks can be randomized. Further, the order of the treatment numbers is randomized. The randomization process changes with different designs. For example, with **latinized designs** the randomized order of columns in each replicate must be the same so that the latinized property is preserved. Also designs with **nested** or **factorial treatment** structure require a randomization that preserves this structure.

Example

5.2 Number of randomizations

Sometimes the same optimal or near optimal design is required for a number of experiments. For example, variety trials of the same material at a number of sites. It is better then to use different randomizations of the design for each experiment. The number of randomizations required can be specified.

5.3 Seed for randomization

The randomization process uses a random number seed to initiate the blocking and treatment factor randomizations. This seed can be taken from the computer clock or specified by the user. In either case the value of the seed is printed in the log file.

5.4 Blocks in columns/rows

Sometimes it is convenient to have a copy of a generated design which can be printed on a A4 a sheet or sheets of paper. For this purpose a block design can be output with the blocks in rows rather than the default, which is in columns.

This convenient copy of the design generated is provided in the **design file**.

5.5 Pairwise variances

In a crossover design, if this option is chosen then the variances (as multiples of the plot variance) of the differences between each pair of estimated direct effects will be given in the log file.

5.6 Back

Aborts the generation of designs. Nothing is recorded in the log file.

5.7 Next

Randomizes the best generated design, if you selected randomize, and always produces log and design files which record the design parameters and results.

If you chose to use **CycAnalysis**, then pressing Next > takes you to the CycAnalysis module, automatically producing log and design files.

5.8 Log file

It is important to have a record of the design generation session. The log file gives details on the design parameters and options selected, the random number seeds used, the design properties, the block and treatment randomizations used and the final design layout. In essence the log file contains enough information to allow the user to reproduce any generated design. You can specify the name of the file or you can use the default name.

5.9 Design file

The design file gives details on the design type and parameters chosen and the final design layout. You can specify the name of the file or you can use the default name (design.des).

5.10 CycAnalysis

This button toggles between [Yes] and [No]. If [Yes] is chosen the program randomizes the design (if requested) and opens the the ***CycAnalysis*** module with the design just generated already loaded.

Preparing the design for analysis

Part VI

6 Preparing the design for analysis

6.1 Loading the design

If *CycAnalysis [Yes]* in the *Output* dialog box has been selected then the main CycAnalysis window will appear with the design details already loaded. The *CycAnalysis* dialog box shows the design generated from the design session, allows changes to be made to the treatment and blocking factors and generates code for the analysis of the design using Genstat or SAS. A display in the main Dialog box gives details of your previous design session In this display, you will see the design parameters of the previous session and the default labels for the spreadsheet columns. For example, in the spreadsheet file for a block design with the three blocking factors you will see the labels "rep", "block" and "plot" while the treatment factor is labelled "treatment". For complex designs further design information will be provided, and it will probably be necessary to scroll down the screen to see all details.

If *CycAnalysis [No]* in the *Output* dialog box has been selected then it will be necessary to run CycAnalysis separately at some subsequent occasion. This can be done by clicking *Analysis* in the menu bar and choosing *CycAnalysis*. No details of a design will appear in the CycAnalysis dialog box, as no design has been loaded. Clicking on *Current design: None* opens up an "Open file" dialog box where you can select your previously generated design, with an .aux extension.

6.2 Renaming treatments

CycDesigN by default names the treatments as 1, 2, 3, etc. Clicking on *Rename treatments* opens a dialog box that allows you to name then with alphanumeric names which may be more relevant to your experiment. For example in an experiment with three treatments, 1, 2 and 3 can be replaced by Sodium, Potash and Nitrate. Similarly, nested treatments can also be renamed.

In larger experiments where treatment labels are required overwriting them can be tedious. Thus, more convenient alternative would be to set up a separate text file (*.txt) containing these new labels and then to load them directly into the *Edit treatment* dialog box using the *Load labels from file* under the *File* menu item. Alternatively you can store the labels in some other file, such as an Excel spreadsheet, and use cut and paste editing facilities under the *Edit* menu item.

After making these changes the spreadsheet file will now have the new treatment labels that you required.
.

6.3 Expanding the plot factor

When running the experiment there may be a number of observations, or yields, to be collected on each plot. For example, in the forestry trial, there may be three trees in each plot. While the analysis will be carried out on the mean yield of the three trees, it will often be more convenient to record the individual yields on each tree. CycAnalysis can expand the spreadsheet to allow multiple, but equal, responses in each plot.

You can click on *Expand plot* factor in the CycAnalysis dialog box and overwrite *Name* by *Tree* and increase the *Items/block* number to 3 in the *Expand the plot factor* dialog box that appears. The resulting spreadsheet will now have a new column labelled *Tree*, with column entries labelled *1*, *2* and

3 for each plot, increasing the spreadsheet rows by a a factor three.

The *Expand plot factor* has now been replaced by *Collapse plot factor*, which allows you to undo the expanding operation. That is, in our example, to delete the *Tree* factor.

6.4 Renaming the blocking and plot factors

Just as it was possible to rename treatment factors and to edit the treatment labels, the same can be done for the blocking and plot factors.

Clicking *Rename blocking/plot factors* will open a dialog box that allows you to rename and relabel any or all of the blocking factors. For example, rep, row and col can be renamed and relabelled.

6.5 Code generation

For the major design types, CycAnalysis will generate the Genstat or SAS code that can subsequently be used as part of the analysis of your design, assuming you use one of these statistical software packages. The design types available are non-resolvable and resolvable block and row column designs, resolvable row-column design latinized by columns or rows (but not both), nested designs, factorial designs, both p-rep block and row-column designs with or without standards, spatial designs and the additive crossover design.

In the *CycAnalysis* dialog box you can select either Genstat or SAS as the code to be generated Clicking on the *Genstat code* button will change it to *SAS code* and vice-versa.

The appropriate code is generated once *Next>* has been clicked. A window containing this code will then automatically open. Note that it may take a few seconds for this file to be created.

Pressing < *Undo* will return you to the original design loaded undoing any changes you have made.

Part VII

7 Examples

7.1 Resolvable designs

7.1.1 Blocks as columns example

A resolvable incomplete block design for v = 9 treatments, r = 2 replicates and k = 3 plots per block is usually written as:

	Replicate 1			Replicate 2		
Blocks	1	2	3	1	2	3
	1	4	7	1	2	3
	2	5	8	4	5	6
	3	6	9	7	8	9

or

	Replicate 1		
Blocks	1	2	3
	1	4	7
	2	5	8
	3	6	9

	Replicate 2		
Blocks	1	2	3
	1	2	3
	4	5	6
	7	8	9

where in both cases the blocks are written in columns.

7.1.2 Blocks as rows example

A resolvable block design for v = 21 treatments, r = 3 replicates and k = 7 plots per block with blocks written as rows is

Replicate 1

18	9	4	15	11	7	2
19	16	10	6	8	13	3
20	1	14	17	21	12	5

Replicate 2

4	2	10	16	1	15	14

```
21   9   6  13  12   5  19
 3  20  18   8  17  11   7

        Replicate 3
 8  10  14   1  16   7   4
20  11  17  21  18   9  13
 6  12   2  19   3   5  15
```

7.1.3 Unequal block sizes example

Consider the construction of resolvable block design where the following values of the design parameters are input: $v = 23$ treatments, $r = 3$ replicates and $k = 5$ plots per block. This will then be a design with unequal block sizes having, in each replicate, three blocks of size 5 and two blocks of size 4. An example of such a resolvable block design is

```
      Replicate 1            Replicate 2            Replicate 3
19  10  17  21   2      16  10  21  2   4      23  19  20   1   5
 8  13  15  14   1       5  22  20  8  13       4  11   8   7   3
 5   9   3   4  20      18   6   7  9  19      17   6  15  22   1
 7  23  22  18   6       1  12  23  3  15       2   9  16  13  21
12  16  11              17  14  11             12  18  14
```

where the blocks are written in columns. It can be seen that each treatment occurs once in each replicate.

7.1.4 Alpha design example

An alpha design for $v = 12$ treatments, $k = 4$ plots per block and $r = 3$ replicates is

```
Replicate            1               2               3
Block            1   2   3       1   2   3       1   2   3

                 1   2   3       1   2   3       1   2   3
                 4   5   6       5   6   4       6   4   5
                 7   8   9       9   7   8       8   9   7
                10  11  12      10  11  12      11  12  10
```

where the blocks are written in columns. Note that there are $s = 3$ blocks per replicate.

7.1.5 Resolvable examples

A resolvable incomplete block design for $v = 9$ treatments, $r = 4$ replicates and $k = 3$ plots per block is:

```
      Replicate 1   Replicate 2   Replicate 3   Replicate 4
Block   1   2   3     1   2   3     1   2   3     1   2   3

        1   4   7     1   2   3     1   2   3     1   2   3
        2   5   8     4   5   6     5   6   4     6   4   5
        3   6   9     7   8   9     9   7   8     8   9   7
```

where the blocks are written in columns. Note that the b = 12 blocks have been split into four replicate groups each of three blocks, with each treatment occurring once in each replicate.

A resolvable row-column design for v = 12 treatments, r = 3 replicates and with k = 4 rows and s = 3 columns per replicate is:

```
    Replicate 1         Replicate 2         Replicate 3

     4    8    6      9    7    8     11    9    3
    10   11   12      5    2   12      6    4    5
     1    2    3     10    6    3      1   12    7
     7    5    9      1   10    4      8    2   10
```

Each treatment occurs once within each replicate.

7.1.6 Nested treatments example

A resolvable row-column design for v = 15 treatments, k = 3 rows, s = 5 columns and r = 3 replicates, where the treatments are nested in four groups of sizes 5, 3, 4 and 3 is

```
Replicate 1
               3 (1)   9 (3)   4 (1)  13 (4)   6 (2)
              11 (3)  15 (4)   8 (2)   2 (1)  10 (3)
               7 (2)   5 (1)  14 (4)  12 (3)   1 (1)

Replicate 2
              12 (3)   8 (2)   1 (1)  13 (4)  10 (3)
               3 (1)   9 (3)  14 (4)   6 (2)   2 (1)
              15 (4)   4 (1)  11 (3)   5 (1)   7 (2)

Replicate 3
              14 (4)   2 (1)   5 (1)   7 (2)   9 (3)
              10 (3)   6 (2)  12 (3)   4 (1)  15 (4)
               3 (1)  11 (3)   8 (2)  13 (4)   1 (1)
```

Note that the levels of the four nested groups are included in brackets after the treatment numbers.

7.1.7 Factorial experiment example

A resolvable row-column design for v = 12 treatments, k = 3 rows, s = 4 columns and r = 3 replicates, where the treatments comprise a 2 x 6 factorial structure is

```
Replicate
  1            2 6      1 3      1 5      2 1
               2 3      2 5      1 4      1 2
               1 1      2 4      2 2      1 6

  2            1 4      2 6      2 3      1 1
               2 1      1 5      1 2      2 4
               1 3      2 2      1 6      2 5

  3            2 4      1 2      2 6      1 3
               1 5      2 3      1 1      2 2
```

```
                1 6      2 1      2 5      1 4
```

In this example each treatment combination is represented by a pair of numbers; the two numbers giving the levels of the first and second factor respectively

7.1.8 Treatment groups example

Consider the following 3-replicate resolvable row-column design for $v = 12$ seedlots in $k = 3$ rows and $s = 4$ columns, where the first 6 seedlots are from one provenance and the remaining from another provenance; i.e. 2 treatment groups each of size 6.

```
   Replicate 1            Replicate 2            Replicate 3

   8   4   7   5          2   4   8  11          1  10   4   8
   6  10   2  11         10  12   5   3         12   5   7   2
  12   3   1   9          9   6   1   7         11   6   9   3
```

The arrangement of the two provenances in this design is as follows

```
   Replicate 1            Replicate 2            Replicate 3

  2   1   2   1          1   1   2   2          1   2   1   2
  1   2   1   2          2   2   1   1          2   1   2   1
  2   1   1   2          2   1   1   2          2   1   2   1
```

Note that in each row of each replicate there are two seedlots from each provenance, while in the columns there are two seedlots from one provenance and one seedlot from the other.

7.1.9 Latinized block example

Consider the following resolvable block design for $v = 12$ treatments, $k = 3$ plots per block and $r = 3$ replicates

```
                         Blocks
                      1    2    3    4
Replicate
        1             1    2    3    4
                      5    6    7    8
                      9   10   11   12

        2             2    1    4    3
                      7    8    5    6
                     12   11   10    9

        3             3    4    1    2
                      8    7    6    5
                     10    9   12   11
```

With the replicates set out next to each other, the design can also be viewed as a block design with four long blocks of size 9. In this case no treatment occurs more than once in each long block. This is an example of a Latinized…[1,1] design.

7.1.10 Column latinized example

Consider the following resolvable row-column design for v = 12 treatments, k = 3 rows, s = 4 columns per block and $r = 5$ replicates, and with 2 long column groups comprising 3 and 2 replicates respectively..

```
                      Columns
                  1    2    3    4
 Replicate
        1         1    8    3    2
                 10   12    5    6
                  4    9   11    7

        2         2    1   10    4
                  9    6    8    3
                 12    5    7   12

        3         8   11   12   10
                  3    4    2    5
                  6    7    1    9

         ........................

        4        10    2    6   11
                  7    5    4    8
                  3    9   12    1

        5         8   12    7    2
                  4    6    9   10
                  5   11    1    3
```

The 3 replicates above the dotted line are in the first long column group and the remaining 2 replicates in the second group. The replicates within each group constitute a latinized design.

7.1.11 Row latinized example

Consider constructing a row latinized design for 15 treatments with 5 rows, 3 columns and 4 replicates. Two steps are involved. First is to construct a Latinized...[1,0,1] row-column design for 4 replicates, each with 3 rows and 5 columns. An example of such a design latinized by columns is

```
                       Columns
                  1    2    3    4    5
 Replicate
        1        15    3   14    6   12
                 10    7    9    5    8
                 13    4   11    1    2

        2         9   12    3   15    4
```

```
                       7    5   13   14   10
                       1   11    8    2    6

      3                2    6   10    9   15
                       5   14    1    4   11
                       3   13   12    8    7

      4                4   15    2    3   13
                      14    8    6   10    5
                      12    1    7   11    9
```

Then the design is rotated through 90 degrees to give the required row latinized design with 5 rows and 3 columns, as follows:

```
                                 Replicates

           1                 2                  3                 4

      15   10   13       9    7    1        2    5    3        4   14   12
       3    7    4      12    5   11        6   14   13       15    8    1
      14    9   11       3   13    8       10    1   12        2    6    7
       6    5    1      15   14    2        9    4    8        3   10   11
      12    8    2       4   10    6       15   11    7       13    5    9
```

7.1.12 Latinized row-column example

The effects of long rows and long columns can be eliminated in the following resolvable row-column design for $v = 12$ treatments in $k = 3$ rows and $s = 4$ columns; in this case the four replicates have been set out in a 2 by 2 array.

```
Replicate                                  Replicate
    1      6  7  3  4        11 10 12  1       3
           5  2 12 11         8  7  3  9
           1  8 10  9         2  5  4  6

    2      4  5  1  8         9  2  6 10       4
          10  3 11  6        12  1  7  4
           2  9  7 12         5  3 11  8
```

The long columns in each group are latinized, as are the long rows across the two groups of replicates. This is an example of a Latinized…[1,1,2] design.

7.1.13 Partially latinized example

Consider a resolvable block design for $v = 12$ treatments, $k = 3$ plots per block and $r = 4$ replicates. Suppose the first two replicates are contiguous, as are the third and fourth replicates. Also the first two replicates are separate from the other two.

The following design is partially latinized.

	Block				Block				
	1	2	3	4	1	2	3	4	
Replicate									Replicate
1	6	7	3	4	11	10	12	1	3
	5	2	12	11	8	7	3	9	
	1	8	10	9	2	5	4	6	
2	4	5	1	8	9	2	6	10	4
	10	3	11	6	12	1	7	4	
	2	9	7	12	5	3	11	8	

No treatment occurs more than once in the long blocks of 6 plots per block in replicates 1 and 2; similarly for replicates three and four. This is an example of a Latinized…[1,2] design.

7.1.14 Partially t-latinized block example

The effects of long blocks can be eliminated in the following resolvable block design for $v = 18$ treatments in $r = 5$ replicates each with $s = 6$ blocks of size $k = 3$; in this case the 5 replicates have been set out in 2 groups of replicates, with the first group consisting of 3 replicates and the second 2 replicates. The blocks have been written as columns.

```
              Blocks                Blocks
          1  2  3  4  5  6      1  2  3  4  5  6

Replicate                                          Replicate
    1     4  2 12  8 14  5      8 12 15  6  1 16       4
         17 13  3 16  9 18     14 13 17  9 18  2
          1 10  6 11  7 15      4 11  3 10  7  5


    2     7  3  4 10 16 11      2 16  4 18 14  9       5
         12  9 18  4 13  6      1  7 11 13 17 12
          8  5  2 15 17  1      3 10  5  8  6 15

    3    15 18  5  9 10  2
         14  6  7 13  3  4
         11 16 17  1  8 12
```

Note that in the first group of three replicates, each consecutive pairs of long blocks constitutes a replicate of the treatments. In the second group no treatment occurs more than once in the successive long blocks within that group. The design is a 2-latinized column design with no row latinization in two separate groups of replicates, and is denoted by Latinized..[2,0,2].

7.1.15 Randomization example

Suppose a latinized design for v = 12 treatments, $r = 2$ replicates, k = 3 rows and s = 4 columns has been generated as:

```
  Replicate 1

3   7  12  10
6   1  11   9
8   5   2   4

  Replicate 2

9  11   5   8
2   3  10   1
7   4   6  12
```

The randomization for replicates is (2 1) meaning, for example, that replicate 2 from the unrandomized design has been randomized to replicate 1. The long column randomization (common across both replicates because the design is latinized) is (3 4 1 2) meaning, for example, that column 1 from the unrandomized design has been randomized to column 3. The row randomizations for the two replicates are (3 1 2) and (1 3 2) meaning, for example, that row 3 in replicate 1 from the unrandomized design has been randomized to row 2 in replicate 2 of the randomized design. Finally the treatment randomization is (4 5 6 8 9 7 11 12 1 2 10 3) meaning, for example, that treatment 8 from the unrandomized design has been randomized to treatment 4. The full randomization thus becomes:

```
   Replicate 1

  3   8   6   1
  2   4   5   7
 11   9  10  12

   Replicate 2

  8  11  12   6
 10   1   4   2
  7   5   3   9
```

7.2 Non-resolvable designs

7.2.1 Cyclic design example

A cyclic design for v = 8 plots, k = 4 plots per block and r = 12 replications obtained from the two initial blocks (1 2 4 7) and (1 2 5 6) is

```
Block     1  2  3  4  5  6  7  8  9 10 11 12

          1  2  3  4  5  6  7  8  1  2  3  4
          2  3  4  5  6  7  8  1  2  3  4  5
```

```
4  5  6  7  8  1  2  3  5  6  7  8
7  8  1  2  3  4  5  6  6  7  8  1
```

where the blocks are written in columns. Note that the first initial block generates a set of eight blocks, while the second initial block generates a partial set of four blocks.

7.2.2 Non-resolvable block example

A block design for $v = 9$ treatments, $b = 12$ blocks, $k = 3$ plots per block and $r = 4$ replicates of each treatment is:

```
Block   1   2   3   4   5   6   7   8   9  10  11  12

        1   1   1   1   2   2   2   3   3   3   4   7
        2   4   5   6   4   5   6   4   5   6   5   8
        3   7   9   8   9   8   7   8   7   9   6   9
```

where the blocks are written in columns.

7.2.3 Non-resolvable examples

A non-resolvable block design for $v = 9$ treatments in $b = 12$ blocks of $k = 3$ treatments per block is

```
Block      1   2   3   4   5   6   7   8   9  10  11  12

           6   7   9   5   1   5   5   9   5   1   2   9
           8   8   1   7   4   2   8   8   6   6   4   7
           3   2   2   4   3   3   1   4   9   7   6   3
```

where the blocks are written in columns. Each treatment is replicated r = 4 times in the design.

A non-resolvable row-column design for v = 9 treatments in a two-dimensional array with k = 6 rows and s = 6 columns is

```
2   8   7   6   9   4
4   2   5   9   3   1
1   3   6   2   7   9
7   9   4   1   5   8
8   7   1   5   6   3
3   5   2   8   4   6
```

Each treatment is replicated r = 4 times in the design.

7.2.4 Non-resolvable row-column example

A row-column design for $v = 10$ treatments in $k = 5$ rows and $s = 6$ columns is:

```
1   4   3   10   5    2
8   7   6    4   1   10
7   1   5    9   8    3
```

3	2	4	8	6	9
10	9	7	5	2	6

Each treatment is replicated *r* = 3 times.

7.2.5 Spatial design example

A non-resolvable row-column design for *v* = 28 treatments, *k* = 8 rows and *s* = 21 columns and 6 replications. It has 1 double diagonal (treatment 26 in rows 2 and 4), 2 stretched EW knight's moves and a neighbour score of 0 (NB=0) - this means no two treatments appear as row neighbours more than once.

For all the treatments in the design the replications span at least 15 columns and 6 rows. For example treatment 19 has replications in column 1 and 19, a treatment span of 19 columns. A treatment with the smallest column span would be treatment 13 (column 5 to column 19).

19	8	14	26	20	12	6	2	24	9	18	27	15	1	7	25	4	22	21	5	10
15	4	28	10	9	17	21	1	27	20	6	26	23	14	3	13	16	11	19	12	24
17	22	18	23	13	14	7	28	3	11	2	4	9	5	6	10	24	20	25	26	16
21	24	3	19	27	25	11	4	5	16	12	13	8	26	22	23	15	7	2	17	18
2	25	16	6	15	18	26	13	19	7	10	21	20	28	11	12	1	3	8	22	14
10	1	5	22	24	28	8	12	15	17	23	3	18	16	27	14	19	9	13	4	6
20	9	12	21	7	3	16	10	14	28	1	22	25	13	2	8	17	5	27	23	11
7	26	27	11	1	2	23	9	25	8	5	24	17	19	4	21	18	6	28	15	20

7.3 Partially replicated designs

7.3.1 Example of a p-rep row-column design

A p-rep row-column design for *v* = 21 test treatments in *c* = 3 locations and *z*=4 standards per location is

location 1

13	**S1**	17	8	7	9	**2**	14
20	**21**	**19**	**11**	16	**S1**	**10**	**3**
19	1	**S1**	4	**5**	18	15	12
3	**5**	**11**	**S1**	**10**	**2**	6	**21**

location 2

14	11	**16**	5	**S1**	13	**9**	15
16	**18**	**17**	20	4	10	**S1**	**14**
2	**7**	**S1**	**17**	21	**12**	8	19
S1	6	1	**12**	**7**	**9**	3	**18**

location 3

18	**S1**	**6**	16	**20**	9	12	**8**
8	**20**	21	**6**	**15**	19	**S1**	10
5	2	**13**	3	**S1**	4	**15**	**1**
13	**1**	17	4	7	**S1**	11	14

The 4 standards in each location are denoted by **S1**. There are *d* = 7 duplicates per location with each treatment occurring twice in one of the locations and once in the other two. The duplicates are given in bold.

7.3.2 Example of an augmented p-rep row-column design

Consider the following augmented p-rep row column design for 30 treatments with 2 standard types, each with 10 standards, in 5 rows and 10 columns.

```
 411S122 5S1S2 130S2
 2S11319 2S2 3S2S115
 4
1410S2S2S1 726S1 817
S2 921S12520S112S2 6
S1S22827S216182923S1
```

Each of the 30 treatments is unreplicated. The 2 standard types are denoted by S1 and S2 and are shown in red. The allocation of the standards is optimal in the sense that each standard type occurs twice in each row and once in each column.

7.3.3 Example of a p-rep spatial design

Consider the following 3 location p-rep row column design for 42 treatments in 8 rows and 8 columns with each treatment replicated 4 times. In addition there are 8 standards and 14 duplicates in each location. In each location a standard S1, shown in red, occurs once in each row and column. For duplicates, the minimum row treatment spans in each location is equal to *kp* = [(*k*-1)/2] = 3, while for columns it is *sp* = [(*s*+1)/2] = 4.

Location 1

42	39	17	23	**7**	**S1**	**10**	25
S1	**6**	**26**	5	**20**	**11**	30	33

28	3	**S1**	**29**	37	**9**	13	**40**
20	**22**	19	41	**S1**	**42**	**35**	18
10	**11**	12	32	36	**26**	**S1**	**31**
14	15	1	**S1**	**16**	2	**6**	8
21	**7**	**9**	**35**	38	34	4	**S1**
31	**S1**	**40**	24	22	27	**29**	**16**

Location 2

27	**1**	**28**	**32**	**S1**	**33**	23	8
S1	18	7	**39**	**14**	40	11	9
41	2	**21**	**37**	31	6	19	**S1**
14	20	3	**S1**	**25**	4	**27**	**32**
12	**S1**	16	**41**	13	36	**15**	30
39	35	26	24	38	**17**	S1	**28**
37	5	**S1**	15	**12**	42	34	**25**
33	29	**17**	10	**1**	**S1**	**21**	22

Location 3

32	39	**36**	21	**S1**	**13**	**8**	5
S1	25	**19**	**24**	**30**	**2**	14	**3**
40	**4**	**8**	**38**	41	11	**S1**	10
23	29	**S1**	6	5	**34**	**36**	**19**
15	31	17	**3**	9	33	35	**S1**
2	**S1**	22	12	28	**23**	**4**	**18**
30	20	7	**S1**	1	37	**24**	16
42	**34**	**13**	**18**	27	**S1**	26	**38**

7.4 Crossover designs

7.4.1 Example

An example of a crossover design for 4 treatments, 4 subjects and 5 periods is as follows:

	Subject			
Period	1	2	3	4
1	**4**	**3**	**1**	**2**
2	**4**	**1**	**1**	**4**
3	**3**	**4**	**2**	**1**
4	**2**	**2**	**3**	**3**
5	**1**	**2**	**4**	**3**

For instance, **subject 1** gets treatment **4** in the first **period**, the same treatment again in the second **period**, and then gets treatments **3**, **2** and **1** in the remaining **periods**. The observation on **subject 1** in **period 5**, for example, depends on the direct effects of the treatment applied in that period (treatment **1**) and some function of the carry-over effect of the treatment applied in **period 4** (treatment **2**). The form of this function depends on the **model** chosen. When a treatment precedes itself, as in **period 2** for **subject 1**, it is called a self-adjacency.

7.4.2 Augment examples

Consider the following example of a crossover design for 4 treatments, 5 periods and 8 subjects using the additive model.

```
                            Subject
Period     1     2     3     4     5     6     7     8

  1        3     4     2     1     2     3     1     4
  2        4     1     1     3     3     4     2     2
  3        4     3     1     3     1     2     4     2
  4        1     2     4     2     4     1     3     3
  5        2     2     3     4     4     1     3     1
```

After running the first 2 periods, it is decided to drop treatment 3 from the remainder of the experiment. Clicking the Augment design button will open a window where you can enter the number of fixed periods as 2 and the number of periods (in the augmented design) as 5. The above design will have been stored as a text file in your working directory, and this file will now be specified in the input file name box. Clicking Next> will open a replications window where you enter the number of replications of each treatment in the 3 remaining periods of the design. There are 24 cells to be filled so that the sum of these replications must be 24. In this case the default replications are 6 for each treatment but these can now be changed to 8 for treatments 1, 2 and 4 and to 0 for treatment 3 (since this treatment is to be dropped from the experiment). An augmented design can now be generated, such as the one below.

```
                           Subject
Period     1     2     3     4     5     6     7     8

  1        3     4     2     1     2     3     1     4
  2        4     1     1     3     3     4     2     2
  3        1     1     4     2     1     2     4     2
  4        2     2     4     2     4     1     4     1
  5        4     2     1     4     4     1     2     1
```

Note that the first two periods of these two designs are the same, as they should be as these two periods have already been run. In the remaining 3 periods the 24 cells have been allocated equally to the treatments 1, 2 and 4. This allocation has been made with the aim of maximizing the average efficiency factors of the direct and carry-over effects.

As a second example, consider the following crossover design for 8 treatments, 4 periods and 6 subjects again using the additive model.

	Subject					
Period	1	2	3	4	5	6
1	4	1	3	2	8	5
2	1	4	2	7	5	6
3	7	4	6	5	3	8
4	2	3	7	8	6	1

After running 3 periods it is decided to extend the experiment with a further 2 periods. In Augment design the number of fixed periods is set to 3 and the number of periods to 6. The 8 treatments have now to be re-allocated to 18 cells in periods 4 to 6. We can do this by replicating 6 treatments twice and the other two three times. Note that in the 3 fixed rows in the design above treatments 4 and 5 are replicated three times, so we shall only replicate these treatments twice in the augmented design. Hence, we shall choose to replicate treatments 7 and 8, say, three times and the other treatments twice. An example of such an augmented design is

	Subject					
Period	1	2	3	4	5	6
1	4	1	3	2	8	5
2	1	4	2	7	5	6
3	7	4	6	5	3	8
4	7	3	5	8	4	1
5	2	8	7	3	6	4
6	5	2	8	6	1	7

The 3 fixed periods are the same in both designs, as they should be. In the augmented design treatments 1, 2, 3 and 6 are replicated four times while treatments 4, 5, 7 and 8 are replicated five times. In this example unequal replication cannot be avoided.

7.5 Analysis

7.5.1 Index file example

The first few rows of the spreadsheet file might be as follows

```
1   1   1   1    3
1   1   1   2    5
1   1   1   3    4
1   1   1   4   10
1   1   2   1    1
```

If the design is a block design then we have for randomisation 1 treatments 3, 5, 4 and 10 in the 4 plots of block 1 of replicate 1, and treatment 1 in plot 1 of block 2 of replicate 1. If it is a row-column design then treatments 3, 5, 4 and 10 are in columns 1, 2, 3 and 4 respectively of row 1 of replicate 1, and treatment 1 is in column 1 of row 2 of replicate 1.

7.5.2 Renaming levels example

A latinized block design for v = 9 treatments, r = 3 replicates and k = 3 plots per block and treatment levels defined as 1, 3, 5, 7, 9, 11, 13, 15 and 17 is

```
                   Blocks
Replicate         1   2   3

   1             15   9  17
                  1  13   7
                  5  11   3

   2              7  15  13
                 17   3   1
                 11   5   9

   3              3   7   5
                 13  17  15
                  9   1  11
```

where the blocks are written as columns.

References

Part

VIII

8 References

CycDesigN is linked closely to the book
J.A. John and E.R. Williams (1995). *Cyclic and Computer Generated Designs*. London: Chapman & Hall.

The majority of the design classes discussed in this book can be constructed. For example, cyclic, alpha and factorial designs with one or two dimensional blocking structures, resolvable and non-resolvable designs and latinized designs. Much of the theory used to generate designs can also be found in this book.

Since 1995 there have been many further developments in the construction of designs. Topics and relevant papers are listed below:

***t*-Latinized designs**
J.A. John and E.R. Williams (1998). t-Latinized designs. *Australian & New Zealand Journal of Statistics* 40, 111-118.

Partially-latinized designs
J.A. John and E.R. Williams (1999). Partially-latinized designs. *Statistics and Computation* 9, 203-207.

α_n-Designs
J.A. John, K. Ruggiero and E.R. Williams (2002). α_n-Designs. *Australian & New Zealand Journal of Statistics* 44, 457-465.

Spatial designs
H-P. Piepho, V. Michel and E.R. Williams (2018). Neighbor balance and evenness of distribution of treatment replications in row-column designs. *Biometrical Journal* 60, 1173-1189.
E.R. Williams and H-P Piepho (2019). Error variance bias in neighbour balance and evenness of distribution designs. *Australian & New Zealand Journal of Statistics* 61, 466-473.

Partially replicated (p-rep) designs
E.R. Williams, H-P Piepho and D. Whitaker (2011). Augmented p-rep designs. *Biometrical Journal* 53, 19-27.
E.R. Williams, J.A.John and D. Whitaker (2014). Construction of more flexible and efficient p-rep designs. *Australian & New Zealand Journal of Statistics* 56, 89-96.

An excellent book on **crossover designs**, and in particular their application in clinical trials, is
B. Jones and M.G. Kenward (2003). *Design and Analysis of Cross-Over Trials*. London: Chapman & Hall.

For a description of the methods used to construct **crossover designs** in CycDesigN see
J.A. John, K.G. Russell and D. Whitaker (2004). CrossOver: An algorithm for the construction of efficient cross-over designs. Statistics in Medicine 23, 2645-2658.

See also the references contained in that paper and
E.R. Williams and J.A. John (2007). Construction of crossover designs with correlated errors. *Australian & New Zealand Journal of Statistics* 49, 61-68.

Index