OPERATING-CHARACTERISTIC GUIDED DESIGN OF GROUP-SEQUENTIAL TRIALS

ANIKO SZABO

Abstract. Group-sequential designs are commonly used for clinical trials to allow early stopping for efficacy or futility. While the design of a single-stage randomized trial is guided by a target power for an alternative hypothesis of interest, the addition of interim analyses is driven by technical choices that are less understandable for clinicians. For example, the commonly used Lan-DeMets methodology requires specification of the timing of analyses and error spending functions. Since the rationale and effect of these technical choices is often unclear, the operating characteristics of the final design are explored under various values of the parameter of interest, and the design is then adjusted until desired properties are obtained.

In this work we develop methods for constructing designs that achieve the desired operating characteristics without the need to specify error spending functions or the timing of analyses. Specifically, we consider designing a study for the mean difference θ of a normally distributed outcome with known variance. The null hypothesis $H_0 : \theta = \theta_0$ is tested versus $H_a : \theta = \theta_A$, with power π at a significance level α. The interim analyses are designed so that for a pre-specified sequence θ_{Ak} the study stops for efficacy at stage k with probability π if $\theta = \theta_{Ak}$. If stopping for futility is also considered, then the requirement to stop for futility at stage k with probability π_F if $\theta = \theta_{0k}$ for pre-specified sequence θ_{0k} can also be added. We show that under some monotonicity restrictions, such designs exist for any choice of the timing of interim analyses. Specific designs can be selected by imposing additional optimality requirements, such as minimizing the expected sample size under the target alternative θ_A, or the average sample size under a weighted selection of the alternatives.

1. Setup and notations

All the calculations are set up assuming a setting with a normally distributed outcome with a known variance, but many common settings can be reduced to / approximated by this special case.

Let $X_i \sim N(\theta, \sigma^2)$ be a sequence of independent identically distributed measurements. We are designing a study to test the following hypotheses:

$$H_0 : \theta = \theta_0 \quad \text{versus} \quad H_A : \theta \geq \theta_0 \tag{1}$$

with type I error α and power π at a pre-specified value $\theta = \theta_A$.

The test statistic for H_0 based on n values is $Z(n) = (\bar{X}_n - \theta_0)\sqrt{n}/\sigma = \sqrt{\mathcal{I}_n}(\bar{X}_n - \theta_0)$, where $\mathcal{I}_n = [\sigma^2/n]^{-1}$ is the Fisher information after n observations. Then for any value of θ

$$Z(n) \sim N\big(\sqrt{\mathcal{I}_n}(\theta - \theta_0), 1\big), \tag{2}$$

and for $n_1 \leq n_2$

$$Cov(Z_{n_1}, Z_{n_2}) = \sqrt{\mathcal{I}_{n_1}/\mathcal{I}_{n_2}}. \tag{3}$$

This information-based formulation holds (approximately) for a variety of potential designs and outcomes, including two-arm randomized and cross-over studies with normal or binary outcomes.

The study will use a group-sequential design with K analyses, conducted at relative times $t_1, \ldots, t_K = 1$. The k^{th} analysis will be conducted after $n_k = N \times t_k$ observations have been obtained, where N is the maximum sample size of the study. Based on the test statistic $Z_k = Z(n_k)$ at analysis $k = 1, \ldots, K-1$, the trial can be stopped early for efficacy (reject H_0) if $Z_k \geq u_k$, stopped early for futility (fail to reject H_0) if $Z_k \leq l_k$, or continued to the next stage. At the final, K^{th} analysis, H_0 would either be rejected if $Z_K \geq u_K = l_K$, or we would fail to reject H_0 otherwise. As a special case, setting $l_k = -\infty$ will result in a study when interim stopping for futility is not considered.

The decision boundaries $u_k, l_k, k = 1, \ldots, K$, and the maximum information target ($\sim$ sample size) $\mathcal{I}_{\max}$ have to be selected so that the overall type I error and power of the study are maintained at their design

Date: September 28, 2020.

TABLE 1. Notations

Characteristic	Notation	Event
No or non-binding futility boundary		
Rejection *at* stage k	$\mathcal{R}_k$	$\{Z_1 < u_1, \ldots, Z_{k-1} < u_{k-1}, Z_k \geq u_k\}$
Continuation *at* stage k	$\mathcal{C}_k$	$\{Z_1 < u_1, \ldots, Z_{i-1} < u_{k-1}, Z_k < u_k\}$
Non-binding futility boundary		
Acceptance *at* stage k	$\mathcal{A}_k$	$\{l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, Z_i \leq l_k\}$
Binding futility boundary		
Rejection *at* stage k	$\mathcal{R}_k^*$	$\{l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, Z_k \geq u_k\}$
Continuation *at* stage k	$\mathcal{C}_k^*$	$\{l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, l_k < Z_k < u_k\}$
Acceptance *at* stage k	$\mathcal{A}_k$	$\{l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, Z_i \leq l_k\}$

TABLE 2. Target operating characteristics

Characteristic	Event	θ	Probability target
No or non-binding futility boundary			
Overall type I error	$\bigcup_{i=1}^{K} \mathcal{R}_i$	θ_0	$\leq \alpha$
Overall power	$\bigcup_{i=1}^{K} \mathcal{R}_i$	θ_A	$\geq \pi$
Stop by stage k for efficacy	$\bigcup_{i=1}^{k} \mathcal{R}_i$	θ_{Ak}	$\geq \pi_E$
Non-binding futility boundary			
Stop by stage k for futility	$\bigcup_{i=1}^{k} \mathcal{A}_i$	θ_{0k}	$\geq \pi_F$
Binding futility boundary			
Overall type I error	$\bigcup_{i=1}^{K} \mathcal{R}_i^*$	θ_0	$\leq \alpha$
Overall power	$\bigcup_{i=1}^{K} \mathcal{R}_i^*$	θ_A	$\geq \pi$
Stop by stage k for efficacy	$\bigcup_{i=1}^{k} \mathcal{R}_i^*$	θ_{Ak}	$\geq \pi_E$
Stop by stage k for futility	$\bigcup_{i=1}^{k} \mathcal{A}_i$	θ_{0k}	$\geq \pi_F$

values, but there are many valid choices. In this *operating characteristic driven* approach we propose to select the boundaries based on the cumulative probabilities of stopping for efficacy or futility at each stage.

Theorem 1. *A design satisfying the operating characteristic requirements of Table 2 exists, if the following conditions are satisfied:*

C1. $\pi_E \leq \pi$
C2. $\theta_{A1} \geq \theta_{A2} \geq \cdots \geq \theta_{A,K-1} \geq \theta_A$

Additionally, if a futility boundary is needed:

C3. $\pi_F \leq 1 - \alpha$
C4. $\theta_{01} \leq \theta_{02} \leq \cdots \leq \theta_{0,K-1} \leq \theta_0$

Proof. We will consier the cases of no, non-binding, and binding futility boundaries separately.

I. no futility boundary We will use proof by induction over the number of stages K. When $K = 1$, only the overall type I error and power need to be considered, and the well-known single-stage design achieving the overall information

$$\mathcal{I}_{\text{fix}} = \frac{(z_\alpha + z_{1-\beta})^2}{(\theta_A - \theta_0)^2}$$

will satisfy the requirements with $u_K = z_\alpha$.

Now suppose we can construct the desired design for $K-1$ stages, and we try to add an additional stage. For the first $K-1$ stages select the boundary $u_1, \ldots, u_{K-1}$ and information times $\mathcal{I}_{n_1}, \ldots, \mathcal{I}_{n_{K-1}}$ to achieve over these $K-1$ stages stage-specific stopping probabilities π_E at $\theta_{A1}, \ldots, \theta A, K-2$, overall power π_E at $\theta_{A,K-1}$, and overall type I error $\alpha_{K-1} = \alpha - \Delta\alpha_K$, where $0 < \Delta\alpha_K < \alpha$ is an arbitrary value. With these choices, the requirements for all the stage-specific probabilities for the K-stage design are satisfied. We need to select I_{n_K} and u_K to satistfy the overall power and type I error restrictions.

Consider the function $a(u \mid \mathcal{I}_N) = Pr\big(\{\bigcup_{i=1}^{K-1} \mathcal{R}_i\} \bigcup \{Z_1 < u_1, \ldots, Z_{K-1} < u_{K-1}, Z(N) \geq u\} \mid \theta_0\big)$, that gives the type I error if the boundary is set at u for the last stage for any given $I_N > I_{n_{K-1}}$. Under H_0, $Z(N) \sim N(0,1)$, so $a(z_\alpha \mid \mathcal{I}_N) \geq \alpha$. On the other hand, $a(\infty|\mathcal{I}_N) = \alpha - \Delta\alpha_K < \alpha$ by the design of the first $K-1$ stages. Since a is monotone in u, for any I_N there exists a cutoff $u_K(\mathcal{I}_N)$ for which $a(u_K(\mathcal{I}_N)) = \alpha$, ie for which the overall type I error is maintained at the desired level.

Next consider the function $b(\mathcal{I}_N) = Pr\big(\{\bigcup_{i=1}^{K-1} \mathcal{R}_i\} \bigcup \{Z_1 < u_1, \ldots, Z_{K-1} < u_{K-1}, Z(N) \geq u_K(\mathcal{I}_N) \mid \theta_A\big)$ that gives the power to reject H_0 if the final analysis is set at information $\mathcal{I}_N > I_{n_{K-1}}$. If $b(\mathcal{I}_{K-1}) \geq \pi$, then we can set $\mathcal{I}_N = \mathcal{I}_{K-1}$. Otherwise, note that $b(\infty) = 1$, and b is monotone in $\mathcal{I}_N$. Thus we can select a value n_K such that $b(\mathcal{I}_{n_K}) = \pi$, which will provide the desired design.

II. Non-binding futility boundary The overall power, type I error, and stage-specific efficacy power requirements do not depend on a non-binding futility boundary, so we can start with a design constructed without a futility boundary. We then need to calculate the lower bounds l_k to satisfy the futility stopping probabilities. For stage 1, we need $Pr(Z_1 < l_1|\theta_{01}) = \pi_F$, where $Z_1 \sim N(\sqrt{I_1}(\theta_{01} - \theta_0), 1)$. Thus $l_1 = z_{1-\pi_F} + \sqrt{I}(\theta_{01} - \theta_0)$ satisfies the target power. Given conditions C3-C4, $l_1 \leq z_\alpha$, while $u_1 \geq z_\alpha$, so $l_1 \leq u_1$.

At stage k, consider the function $c(l) = Pr(l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, Z_k < l \mid \theta_{0k})$. Since $\theta_{0k} \leq \theta_0$, $c(z_\alpha) \geq 1 - \alpha \geq \pi_F$, while $c(-\infty) = 0$, so a value l_k can be selected for which $c(l_k) = \pi_F$.

III. Binding futility boundary The construction follows the induction-based approach of part I. During the inductive step in addition to I_K and u_K, we need to find a value for l_{K-1} that will satisfy the futility boundary restriction. This can be done as shown in part II before proceeding with the selection of the parameters of the last stage. A potential complication compared to the efficacy-only situation, is that by adding l_{K-1} we might overspend the type II error, $Pr(\bigcup_{i=1}^{K-1} \mathcal{A}_i|\theta_A) > 1 - \pi$. In this case we would not be able to select a value for $\mathcal{I}_N$ such that $b^*(\mathcal{I}_N) \geq \pi$, because even $b^*(\infty) < \pi$. One of the possible solutions is to adjust the timing of the $(K-1)st$ stage to reduce its beta-spending. As $\mathcal{I}_{K-1}$ is increased beyond the value that provided power π_E at significance level $\alpha - \Delta\alpha_K$, if we maintain the significance level, $P(\mathcal{R}^*_{K-1}|\theta_{A,K-1})$ will increase, so the power will be maintained. However $Pr(\mathcal{A}_{K-1}|\theta_A)$ will decrease, so we can find a value of I_{K-1} for which $Pr(\bigcup_{i=1}^{K-1} \mathcal{A}_i|\theta_A) = 1 - \pi$. Starting from this modified design with $K-1$ stages, we can now construct the required K-stage design.

□

In the proof of the theorem we have obtained the following uniqueness result:

Corollary 1. *Under the assumptions of Theorem 1, an alpha-spending sequence $\Delta\alpha_k > 0$, $k = 1, \ldots, K$ such that $\sum_{k=1}^{K} \Delta\alpha_k = \alpha$ uniquely determines a design with the required operating characteristics.*

2. Main function: calculate design

"../R/gsDesignOC.R" 4≡

```
#'Find optimal group-sequential design
#'
#'The \code{gsDesignOC} function finds the analysis times, boundaries, and sample size
#'for a group-sequential trial
#'
#'@export
#'@param thA numeric; effect size under the alternative hypothesis
#'@param thA.seq monotone numeric vector of values getting closer to thA from outside the
#'th0-thA range (ie, if thA > th0, they should form a decreasing sequence with all values > thA). #'For t
#'@param th0 numeric; effect size under the null hypothesis
#'@param th0.seq monotone numeric vector of values getting closer to th0 from outside the
#'th0-thA range (ie, if thA > th0, they should form an increasing sequence with all values < th0).
#'The study will stop for futility at or before the k-th stage with probability \code{power.futility}.
#'Its length should be equal to that of \code{thA.seq}. The default value of \code{NULL} implies no
#'futility monitoring.
#'@param sig.level numeric; the one-sided significance level. The default is 0.025.
#'@param power numeric; the desired study power. The default is 0.9.
#'@param power.efficacy numeric; the probability of stopping for efficacy at stage k under \code{thA.seq}
#'@param power.futility numeric; the probability of stopping for futility at stage k under \code{th0.seq}
#'@param futility.type character string, one of \code{c("none", "non-binding","binding")} or their
#'unique abbreviations. Specifies whether a futility boundary should not be used at all ("none"), or if i
#'is used, then whether the effect of the futility boundary should be included
#'in the efficacy power/type I error calculations ("binding"), or not ("non-binding").
#'@param mix.theta numeric vector; specifies the set of alternatives under which the design is optimized.
#' Defaults to \code{thA}.
#'@param mix.w numeric vector of length equal to that of \code{mix.theta}. The weights of the
#'elements of \code{mix.theta} in the optimality criterion. It will be normalized to a sum of 1.
#'Defaults to equal weights.
#'@param control list of parameters controlling the estimation altorithm to replace the default
#'values returned by the \code{glsControl} function.
#'@return an object of class \code{gsDesignOC}
#'@author Aniko Szabo
#'@references Szabo, A, Tarima, S (?) Operating-characteristic guided design of group-sequential trials.
#'@keywords nonparametric design
#'@examples
#'
#'gsDesignOC(0.3, thA.seq = 0.5, th0.seq = -0.3, power.efficacy=0.8, power.futility=0.8, power=0.9,
#'           futility.type = "non-binding")
#'
#'@name gsDesignOC

gsDesignOC <- function(thA, thA.seq, th0=0, th0.seq=NULL,
                       sig.level = 0.025,
                       power=0.9, power.efficacy=power, power.futility = power,
                       futility.type=c("none","non-binding","binding"),
                       mix.theta = thA,
                       mix.w = rep(1, length(mix.theta)),
                       control=ocControl()){
  ⟨ Check inputs 5 ⟩

  # create skeleton gsDesignOC object
  res <- list(thA=thA, thA.seq=thA.seq,
              th0=th0, th0.seq=th0.seq,
              sig.level = sig.level,
              power=power, power.efficacy = power.efficacy, power.futility = power.futility,
```

```
            futility.type=futility.type)
  class(res) <- "gsDesignOC"

  n.stages <- length(thA.seq) + 1
  if (n.stages == 1){
    alpha.seq <- sig.level
  } else if (control$optim.method == "direct"){
  ⟨ Find optimal design using direct search 6a ⟩
  } else if (control$optim.method == "dynamic"){
  ⟨ Find optimal design using recursive search 6c ⟩
  } else {
    stop(sprintf("Optimization method %s is not available. Choose 'dynamic' or 'direct'.", control$optim.
  }

  res <- calc.bounds(x=res, alpha.seq)

  return(res)
}
```

◇

File defined by 4, 8, 9, 10.
Uses: calc.bounds 10.

⟨ *Check inputs* 5 ⟩ ≡

```
  if (any(diff(c(thA.seq, thA)) * sign(th0-thA) <= 0))
    stop("'thA.seq' should approach thA in a monotone sequence outside the th0-thA range")

  if (!is.null(th0.seq)){
    if (length(th0.seq) != length(thA.seq))
      stop("If specified, 'th0.seq' should have the same length as 'thA.seq'")
    if (any(diff(c(th0.seq, th0)) * sign(thA-th0) <= 0))
      stop("'th0.seq' should approach th0 in a monotone sequence outside the th0-thA range")
  }

  if (length(mix.w) != length(mix.theta))
      stop("'mix.w' should have the same length as 'mix.theta'")

  if (power.efficacy > power)
    stop("The value of 'power.efficacy' should not exceed the value of 'power'.")

  if (power.futility > 1-sig.level)
    stop("The value of 'power.futility' should not exceed the value of 1-'sig.level'.")

  futility.type <- match.arg(futility.type)
  if (futility.type != "none"){
    if (is.null(th0.seq)) stop("`th0.seq` should be specified if a futility bound is requested")
  }

  controlvals <- ocControl()
  if (!missing(control))
    controlvals[names(control)] <- control
```

◇

Fragment referenced in 4.

2.1. **Direct optimization.**

⟨ *Find optimal design using direct search* 6a ⟩ ≡

```
⟨ Define optimization function 6b ⟩

if (n.stages == 2){
  oo <- optimize(.cp, interval=c(-5,5))

  y.res <- oo$minimum
  alpha.seq <- sig.level * exp(c(y.res,0)) / sum(exp(c(y.res,0)))
} else {
  y.start <- -log(seq(n.stages, 2, by=-1))

  oo <- optim(y.start, fn=.cp, method="Nelder-Mead")

  y.res <- oo$par
  alpha.seq <- sig.level * exp(c(y.res,0)) / sum(exp(c(y.res,0)))
}
```

◇

Fragment referenced in 4.

Using the `calc.bounds` function, we can construct a design matching all the type I error/power requirements given a sequence of stage-specific positive alpha-spending values $\Delta\alpha_k$ that add up to α. To allow unconstrained optimization, we can reparametrize it using the multinomial logit:

$$y_k = \log \frac{\Delta\alpha_k}{\Delta\alpha_K}, \quad k = 1, \ldots, K-1,$$

with reverse transition

$$\Delta\alpha_k = \frac{\alpha e^{y_k}}{\sum_{i=1}^{K} e^{y_i}}, \quad k = 1, \ldots, K,$$

where $y_K := 0$.

⟨ *Define optimization function* 6b ⟩ ≡

```
.cp <- function(y.vec){

  y <- c(y.vec, 0)  # add y_K
  alpha.seq <- sig.level * exp(y) / sum(exp(y))

  res <- calc.bounds(x=res, alpha.seq = alpha.seq)
  dp <- oc(res, EN_theta=mix.theta, mix.w=mix.w)
  Q <- dp$ave.EN
  Q
}
```

◇

Fragment referenced in 6a.
Uses: `calc.bounds` 10, `oc` 8.

2.2. **Recursive optimization.**

⟨ *Find optimal design using recursive search* 6c ⟩ ≡

```
⟨ Define recursive optimization function 7a ⟩
  alpha.seq <- .opt.spend(x=res)
```

◇

Fragment referenced in 4.

The recursive optimization also implements the construction process in the proof of Theorem 1, but will select the optimal $\Delta\alpha_K$ when adding the last stage.

For the first $K-1$ stages we will select the boundary $u_1, \ldots, u_{K-1}$ and information times $\mathcal{I}_{n_1}, \ldots, \mathcal{I}_{n_{K-1}}$ to achieve over these $K-1$ stages stage-specific stopping probabilities π_E at $\theta_{A1}, \ldots, \theta A, K-2$, overall power π_E at $\theta_{A,K-1}$, and overall type I error $\alpha_{K-1} = \alpha - \Delta\alpha_K$.

The `.opt.spend` function will return the optimal alpha-spending sequence.

⟨ *Define recursive optimization function* 7a ⟩ ≡

```
.opt.spend <- function(x){
  n.stages <- length(x$thA.seq) + 1

  if (n.stages == 1){
    return(x$sig.level)  # we spend it all
  }

  ⟨ Find optimal DeltaAlpha to spend 7b ⟩
}
```

◇

Fragment referenced in 6c.

The search for $0 < \Delta\alpha_K \leq \alpha$ will be parametrized via $y = \log(\Delta\alpha_K/\alpha)$ with $y \leq 0$ to restrict it to the correct range, and better spread out small values.

⟨ *Find optimal DeltaAlpha to spend* 7b ⟩ ≡

```
#create skeleton for a design with one fewer stages
xx <- list(thA=tail(x$thA.seq, 1), thA.seq=head(x$thA.seq, -1),
           th0=x$th0, th0.seq=head(x$th0.seq,-1),
           power=x$power.efficacy, power.efficacy = x$power.efficacy, power.futility = x$power.futilit
           futility.type=x$futility.type)
class(xx) <- "gsDesignOC"
⟨ Define function to calculate EN given DeltaAlpha 7c ⟩
res.spend <- optimize(en, interval=c(-5, 0))
opt.spending <- attr(res.spend$objective, "spending")
return(opt.spending)
```

◇

Fragment referenced in 7a.

⟨ *Define function to calculate EN given DeltaAlpha* 7c ⟩ ≡

```
en <- function(y){
  delta.alpha <- exp(y) * x$sig.level
  cum.alpha <- x$sig.level - delta.alpha
  # figure out the optimal spending within the first K-1 stages
  xx$sig.level <- cum.alpha
  prev.spend <-.opt.spend (xx)
  # calculate design using these spending values
  xx2 <- calc.bounds(x, alpha.seq = c(prev.spend, delta.alpha))
  dp <- oc(xx2, EN_theta = mix.theta, mix.w = mix.w)
  en.res <- dp$ave.EN
  attr(en.res, "spending") <- c(prev.spend, delta.alpha)
  en.res
}
```

◇

Fragment referenced in 7b.
Uses: `calc.bounds` 10, `oc` 8.

3. Key support functions

"../R/gsDesignOC.R" 8≡

```
#'Operating characteristics of a potential design
#'
#'Calculates the average expected information under a weighted set of values for the alternative hypothes
#' the probability of stopping for futility or effecacy under the design alternatives. If the futility bo
#' is non-binding, then the lower boundary is not included in the calculation of the expected sample size
#' efficacy stopping probabilities, but it is included in the futility stopping calculations.
#'
#'@export
#'@param x object of class \code{gsDesignOC}
#'@param EN_theta numeric vector of parameter values for which expected sample size calculation is perfor
#'@param mix.w numeric vector of positive weights for each value of \code{EN_theta}. Defaults to equal we
#'@return a list with the following elements:
#'\describe{
#'\item{ave.EN}{numeric; weighted average of expected sample sizes under the requested alternatives}
#'\item{EN.vec}{numeric vector; expected sample sizes under each of the requested alternatives}
#'\item{thA.cumcross}{numeric vector; cumulative probability of stopping for efficacy at or before stage
#'  under thA.seq[k], including thA at the end.}
#'\item{th0.cumcross}{numeric vector}{numeric vector; cumulative probability of stopping for futility at
#'  under th0.seq[k], including th0 at the end.}
#'}
#'@author Aniko Szabo
#'@keywords design
#'@examples
#'g <- gsDesignOC(0.3, thA.seq = 0.5, th0.seq = -0.3, power.efficacy=0.8, power.futility=0.8, power=0.9,
#'           futility.type = "non-binding", mix.theta=c(0.5, 0.3, 0))
#'oc(g, EN_theta = c(0.5, 0.3, 0))
#'
#'

oc <- function(x, EN_theta=x$thA,  mix.w = rep(1, length(EN_theta))){

  if (length(mix.w) != length(EN_theta))
    stop("`EN_theta` and `mix.w` should have the same length")
  if (!all(mix.w > 0))
    stop("`mix.w` should have only positive elements")

  n.stages <- length(x$thA.seq) + 1
  n.EN <- length(EN_theta)
  n.A <- length(c(x$thA.seq, x$thA))
  n.0 <- length(c(x$th0.seq, x$th0))

  # crossing probabilities for futility
  ggF <- gsDesign::gsProbability(k = n.stages,
                                 theta = c(EN_theta, x$thA.seq, x$thA, x$th0.seq, x$th0),
                                 n.I = x$info,
                                 a = x$lower,
                                 b = x$upper)

  # crossing probabilities for efficacy and EN
  if (x$futility.type == "non-binding") {
    #ignore lower boundary (except last stage) for EN & efficacy stopping
    ggE <- gsDesign::gsProbability(k = n.stages,
                                   theta = c(EN_theta, x$thA.seq, x$thA, x$th0.seq, x$th0),
                                   n.I = x$info,
                                   a = c(rep(-20, n.stages-1), x$lower[n.stages]),
```

```
                                        b = x$upper)
   } else {
    ggE <- ggF
   }

  # expected sample size calculations
  en_vec <- ggE$en[1:n.EN]

  en <- c(en_vec %*% mix.w / sum(mix.w))

  # cumulative stopping probability calculations
  p.up <- ggE$upper$prob[, n.EN + (1:n.A), drop=FALSE]
  cump.up <- apply(p.up, 2, cumsum)
  thA.cumcross <- diag(cump.up)

  p.low <- ggF$lower$prob[, n.EN + n.A + (1:n.0), drop=FALSE]
  cump.low <- apply(p.low, 2, cumsum)
  th0.cumcross <- diag(cump.low)

  list(ave.EN = en,
       EN.vec = en_vec,
       thA.cumcross = thA.cumcross,
       th0.cumcross = th0.cumcross)
}
```
◇

File defined by 4, 8, 9, 10.
Defines: `oc` 6b, 7c.

"../R/gsDesignOC.R" 9≡

```
#'Control values for gsDesignOC
#'
#'The values supplied in the function call replace the defaults and a list with all possible
#'arguments is returned. The returned list is used as the \code{control} argument to the
#'\code{gsDesignOC} function.
#'
#'@export
#'@param optim.method character; defines the optimization method used: \code{"dynamic"} uses a recursive
#' \code{\link{optim}} to simultaneously search over all possible alpha-spending sequences.
#'@return a list with components for each of the possible arguments
#'@author Aniko Szabo
#'@keywords design
#'@examples
#'
#'ocControl(optim.method = "direct")

ocControl <- function(optim.method = c("direct", "dynamic")){
  optim.method <- match.arg(optim.method)
  list(optim.method = optim.method)
}
```
◇

File defined by 4, 8, 9, 10.

4. Boundary achieving the desired operating characteristics

Given the alpha-spending sequence and using the probability targets defined in Table 2, we can derive the information times $\mathcal{I}_k$, and boundaries u_k and l_k using the recursive construction process described in the proof of Theorem 1.

"../R/gsDesignOC.R" 10≡

```
#'Calculate the information times and efficacy/futility boundary values given the alpha-spending sequence


#'@param x a list with desired operating characteristics. Should have \code{th0}, \code{thA}, \code{thA.s
#' \code{power}, \code{power.efficacy}, and \code{futility.type} components; and \code{th0.seq} and \code
#' components if lower boundary is requested. Object of class \code{gsDesignOC} (potentially incomplete)
#' these components.
#'@param alpha.seq numeric vector of stage-specific alpha-spending; values should add up to \code{x$sig.l
#'@return The value of \code{x} augmented with the following components:
#'\describe{
#'\item{info}{numeric vector; information times for the analyses}
#'\item{lower}{numeric vector; lower Z-score boundary for the futility decisions}
#'\item{upper}{numeric vector; upper Z-score boundary for the efficacy decisions}
#'\item{spending}{numeric vector}{the value of alpha-spending sequence \code{alpha.seq}}
#'}
#'@internal

calc.bounds <- function(x, alpha.seq){

  n.stages <- length(x$thA.seq) + 1
  alpha.cum <- cumsum(alpha.seq)
  ub <- rep(20, n.stages)
  lb <- rep(-20, n.stages)
  ivec <- rep(NA, n.stages)

⟨ Define exceedance probability functions 11b ⟩
⟨ Construct first stage 11a ⟩

  if (n.stages > 1){
    for (k in 2:n.stages){
    ⟨ Construct next stage 12a ⟩
    }
  }

  if (x$futility.type == "non-binding"){
    ⟨ Add non-binding lower bound 14a ⟩
  }

  x$upper <- ub
  x$lower <- lb
  x$info <- ivec
  x$spending <- alpha.seq
  return(x)
}
```

◇

File defined by 4, 8, 9, 10.
Defines: `calc.bounds` 4, 6b, 7c.

The first stage is based on the fixed sample-size formula for alternative θ_{A1}, power π_E, and significance level $\Delta\alpha_1$.

⟨ *Construct first stage* 11a ⟩ ≡

```
ivec[1] <- ztest.I(delta=x$thA.seq[1]-x$th0, sig.level=alpha.seq[1], power=x$power.efficacy)
ub[1] <- qnorm(alpha.seq[1], lower=FALSE)
```

◇

Fragment referenced in 10.
Uses: `ztest.I` 14b.

The cumulative rejection / acceptance probabilities depend on the type of the futility boundary.

⟨ *Define exceedance probability functions* 11b ⟩ ≡

⟨ *Define upper bound exceedance* 11c ⟩
⟨ *Define function to find u(I)* 11d ⟩
⟨ *Define lower bound exceedance* 13a ⟩
⟨ *Define function to find l* 13b ⟩

◇

Fragment referenced in 10.

4.1. **Upper bound exceedance.** The functions a and b in the proof of Theorem 1 have a similar form, so we will define the following function:

$$e(u \mid \mathcal{I}) = Pr\Big(\bigcup_{i=1}^{k-1} \{l_1 < Z_1 < u_1, \ldots, l_{i-1} < Z_{i-1} < u_{i-1}, Z_i \geq u_i\} \bigcup \{l_1 < Z_1 < u_1, \ldots, l_{k-1} < Z_{k-1} < u_{k-1}, Z(\mathcal{I}) \geq u\} \mid \theta \Big).$$

If l_i is set to $-\infty$, then that is equivalent to not having a lower boundary. We will need to solve equations of the form $e(u) = c$, so we will actually define a function for the difference of the LHS and RHS.

⟨ *Define upper bound exceedance* 11c ⟩ ≡

```
exc <- function(u, I, stage,  theta, target){
  gg <- gsDesign::gsProbability(k=stage,
                                theta=theta,
                                n.I =c(head(ivec, stage-1),I),
                                a = c(head(lb, stage-1), -20),
                                b = c(head(ub, stage-1), u))
   sum(gg$upper$prob[1:stage]) - target
}
```

◇

Fragment referenced in 11b.

To find $u(I)$ we need to solve $a(u|\mathcal{I}) = \sum_{i=1}^{k} \Delta\alpha_i = \tilde{\alpha}_k$. Since $a(z_\alpha \mid \mathcal{I}_N) \geq \alpha$ and $a(\infty|\mathcal{I}_N) = \tilde{\alpha}_{k-1}$, the solution for fixed $\mathcal{I}_N$ is between z_α and 20 (which is essentially ∞).

⟨ *Define function to find u(I)* 11d ⟩ ≡

```
uI <- function(I, stage){
  res <- uniroot(exc, interval=c(qnorm(x$sig.level, lower=FALSE), 20),
                 I = I, stage=stage, theta=x$th0, target = alpha.cum[stage],
                 extendInt = "downX")
 res$root
}
```

◇

Fragment referenced in 11b.

4.2. **Construct next stage.**

⟨ *Construct next stage* 12a ⟩ ≡

```
⟨ Determine target power and alternatives 12b ⟩
⟨ Find lower bound for previous stage 13c ⟩
⟨ Find I for next stage 12c ⟩
```

◇

Fragment referenced in 10.

To find I_k, we need to solve $b(\mathcal{I}) = \pi_E$, if $k < K$ and $= \pi$ for $k = K$.

⟨ *Determine target power and alternatives* 12b ⟩ ≡

```
if (k == n.stages){
  power.target <- x$power
  .th <- x$thA
} else {
  power.target <- x$power.efficacy
  .th <- x$thA.seq[k]
}
```

◇

Fragment referenced in 12a.

Note that $b(I_{k-1}) \leq \pi_E$ and $b(\mathcal{I}_{fix}(\theta_{Ak} - \theta_0, \tilde{\alpha}_k, \pi_E)) \leq \pi_E$, we can start the search at the maximum of these two values.

⟨ *Find I for next stage* 12c ⟩ ≡

```
exc_i <- function(ii)exc(uI(ii, k), ii, stage=k, theta=.th,
                                  target = power.target)

minI <- max(ztest.I(delta = .th - x$th0, power=power.target, sig.level=alpha.cum[k]),
            ivec[k-1])
curr.exc <- exc_i(minI)

if (curr.exc > 0){
  # already past target power
  ivec[k] <- minI
  ub[k] <- lb[k] <- uI(minI, k)
} else {
  resI <- uniroot(exc_i, interval=c(minI, 2*minI), extendInt = "upX")
  ivec[k] <- resI$root
  ub[k] <- lb[k] <- uI(resI$root, k)
}
```

◇

Fragment referenced in 12a.
Uses: `ztest.I` 14b.

4.3. **Add lower bound.** The lower bound will usually be computed only when the information and upper bound for a stage has been determined.

⟨ *Define lower bound exceedance* 13a ⟩ ≡

```
exc_low <- function(l, stage, theta, target, I=ivec[stage]){
  gg <- gsDesign::gsProbability(k=stage,
                                theta=theta,
                                n.I =c(head(ivec, stage-1), I),
                                a = c(head(lb, stage-1), l),
                                b = ub[1:stage])
   sum(gg$lower$prob[1:stage]) - target
}
```

◇

Fragment referenced in 11b.

⟨ *Define function to find l* 13b ⟩ ≡

```
lI <- function(stage, theta, I=ivec[stage]){
  res <- uniroot(exc_low, interval=c(-20, qnorm(x$sig.level, lower=FALSE)),
                 stage=stage, theta=theta, target = x$power.futility, I=I,
                 extendInt = "upX")
 res$root
}
```

◇

Fragment referenced in 11b.

We need to find the lower bound for stage $k-1$, before computing the size of the next stage. If the boundary is not binding, we need to overwrite $l_k := u_k$ that was indicating the "last" stage so far to $l_k = -\infty$.

⟨ *Find lower bound for previous stage* 13c ⟩ ≡

```
if (x$futility.type == "binding"){
  lb[k-1] <- lI(stage=k-1, theta=x$th0.seq[k-1])
  ⟨ Check beta-spending and adjust previous stage if needed 13d ⟩
} else {
  lb[k-1] <- -20
}
```

◇

Fragment referenced in 12a.

With a binding boundary it is possible that the addition of l_{K-1} overspends the type II error under θ_{Ak}, and the target power for the next stage is not achievable anymore. In this case, the information for the $(K-1)$st stage needs to be increased.

⟨ *Check beta-spending and adjust previous stage if needed* 13d ⟩ ≡

```
typeII.overspent <- exc_low(lb[k-1], stage=k-1, theta=.th, target = 1 - power.target)
if (typeII.overspent > 0){
  exc_low_i <- function(ii)exc_low(lI(I=ii, theta=x$th0.seq[k-1], stage=k-1), ii, stage=k-1, theta=.th,
                                   target = 1-power.target)
  resI_low <- uniroot(exc_low_i, interval=c(ivec[k-1], 2*ivec[k-1]), extendInt = "downX")
  ivec[k-1] <- resI_low$root
  ub[k-1] <- uI(I = resI_low$root, stage = k-1)
  lb[k-1] <- lI(I = resI_low$root, theta = x$th0.seq[k-1], stage = k-1)
}
```

◇

Fragment referenced in 13c.

For a non-binding boundary, the lower bound is added only after the entire upper bound has been calculated.
⟨ *Add non-binding lower bound* 14a ⟩ ≡

```
for (k in 1:(n.stages-1)){
  lb[k] <- lI(stage=k, theta=x$th0.seq[k])
}
```

◇
Fragment referenced in 10.

5. Utility help-functions

"../R/Utility.R" 14b≡

```
#'Fixed sample-size information for one-sided test
#'
#'The \code{ztest.I} function finds the information required of a one-sided Z-test with given power
#'and significance level
#'
#'@param delta numeric; targeted standardized effect size
#'@param sig.level numeric; one-sided significance level
#'@param power numeric; target power
#'@return numeric value with the (fractional) information achieving the target power
#'@author Aniko Szabo
#'@keywords internal
#'@examples
#'
#'ztest.I(delta=1, sig.level=0.05, power=0.9)

ztest.I <- function(delta, sig.level, power){
  za <- qnorm(sig.level, lower=FALSE)
  zb <- qnorm(power)
  I <- (za+zb)^2 /delta^2
  I
}
```

◇
Defines: `ztest.I` 11a, 12c.