



# Spreading Code Sequence Design via Mixed-Integer Convex Optimization

Alan Yang | Tara Mina | Grace Gao

Stanford University

## Correspondence

Grace Gao

Email: [gracegao@stanford.edu](mailto:gracegao@stanford.edu)

## Abstract

For a satellite navigation system, binary spreading codes with good autocorrelation and cross-correlation properties are critical for ensuring precise synchronization and tracking with minimal intrasystem interference. In this paper, we demonstrate that multiple instances of the spreading code design problem found in the literature may be cast as binary-constrained convex optimization problems. This approach enables new optimization methods that can exploit the convex structure of the problem. We demonstrate this approach using a block coordinate descent (BCD) method, which applies a convexity-exploiting branch-and-bound method to perform the block updates. With minimal tuning, the BCD method was able to identify Global Positioning System codes with better mean-squared correlation performance compared with the Gold codes and codes derived from a recently introduced natural evolution strategy.

## Keywords

block coordinate descent, memory codes, mixed-integer programming, spreading codes

## 1 | INTRODUCTION

The soon-to-be-launched Navigation Technology Satellite-3 (NTS-3) platform provides a vehicle for exploring future Global Positioning System (GPS) technologies (Chapman et al., 2020), and its upcoming launch heralds a new era for satellite navigation. NTS-3 is equipped with a reprogrammable signal generator that may be used to test new spreading code families (Air Force Research Laboratory, 2024). As a result, there has been renewed interest in designing new spreading codes with desirable properties, such as low autocorrelation and cross-correlation (Boukerma et al., 2021; Mina & Gao, 2022; Yang et al., 2023a).

The spreading code design problem is a challenging combinatorial optimization problem that typically requires the minimization of a nonconvex function over thousands of binary-valued variables. In this work, we pose several existing variations of the spreading code sequence design problem in the literature as equivalent mixed-integer convex optimization problems (MICPs), based on our recent ION GNSS+ 2023 conference paper (Yang et al., 2023b). A MICP involves the minimization of a convex function, in which some of the variables are subject to integer (in this case, binary) constraints. We believe that most, if not all, spreading code design problems may be posed as MICPs and that this insight may be used to develop

convexity-exploiting algorithms for optimizing spreading codes used by future global navigation satellite system (GNSS) applications.

In prior work, Yang et al. (2023a) proposed a MICP formulation for the problem of minimizing a mean of squared autocorrelation and cross-correlation values. The current work shows how this approach may be extended to handle odd autocorrelation and cross-correlation (NAVSTAR GPS Directorate, 2022; Winkel, 2011), alternative objective functions, and design constraints.

Spreading code optimization typically involves minimizing a convex objective function of codes' autocorrelation and cross-correlation values. For example, the mean of squared autocorrelation and cross-correlation sidelobes, which is sometimes referred to as the integrated sidelobe level (ISL) (He et al., 2012), is a convex quadratic function of those values. Other modifications of the mean-of-squares objective include evenly balancing the autocorrelation and cross-correlation performance (Mina & Gao, 2022) as well as considering the mean squared correlation values that exceed a particular correlation threshold (Soualle et al., 2005; Winkel, 2011). Other objective functions that have been proposed in the literature include the maximum absolute correlation or peak sidelobe level (PSL) (He et al., 2012) and the mean of higher powers of the autocorrelation and cross-correlation values (Winkel, 2011).

Several common constraints on spreading codes and their autocorrelation and cross-correlation values may also be formulated as convex constraints, which may be easily incorporated into our framework. For example, balance constraints ensure that each code contains similar counts of +1 and -1 entries (Winkel, 2011). Another example is the autocorrelation sidelobe zero (ASZ) property (Wallner et al., 2007), which ensures that the autocorrelation at a shift of one chip, or bit, is minimal. Finally, an upper bound on the absolute autocorrelation and cross-correlation magnitude may be enforced.

To illustrate how our framework may be used for spreading code design, we consider a simple block coordinate descent (BCD) method, which repeatedly solves the MICP over a randomly selected subset (or block) of binary variables, with the others variables held fixed (Yang et al., 2023a; Yuan et al., 2017). The block updates involve solving a binary-constrained convex optimization problem, which, in practice, can be performed by using a branch-and-bound-based solver, such as Gurobi (Gurobi Optimization, LLC, 2022). In our experiments, we compared codes found by the BCD method against randomly generated codes, the Gold codes (Gold, 1967), and codes optimized via a recently introduced natural evolution strategy (NES) (Mina & Gao, 2022), with respect to a modification of the mean-of-squares objective that evenly balances autocorrelation and cross-correlation performance (Mina & Gao, 2022). We found that with minimal tuning, the set of 31 length-1023 codes found using BCD gave the best performance.

This paper is organized as follows. Prior work is reviewed in Section 2. The general spreading code optimization problem is posed in Section 3, as well as our proposed formulation, which unifies various objective functions and constraints. Solution methods are discussed in Section 4, and a numerical example involving GPS codes is given in Section 5. Finally, Section 6 concludes.

## 2 | PRIOR WORK

GPS currently employs algorithmically generated codes. These codes do not require explicit storage in memory; instead, they can be produced on-the-fly, using,

for example, linear shift feedback registers. The legacy GPS L1 coarse/acquisition (C/A) signal utilizes the Gold codes (Gold, 1967), whereas the modernized L1C signal leverages modified Weil codes (Rushanan, 2007).

During the initial developmental phase of GPS in the 1970s, the ability to algorithmically generate codes was paramount. In particular, algorithmically generable codes provide a large family of low-correlation codes to choose from, without requiring exploration of the exponentially large binary code space. Furthermore, the use of algorithmically generated codes enables code storage with limited digital memory capacities. However, memory capacity is no longer a limitation for modern receivers, which can readily store local copies of entire sets of spreading codes. We refer to codes that are stored explicitly as *memory codes*, to distinguish them from algorithmically generated codes.

Coupled with the increase in computational power that has occurred over the past few decades, there has been a renewed interest in designing memory codes via optimization. The European Union’s Galileo satellite constellation, for example, uses memory codes for its E1 signal (Wallner et al., 2007, 2008), which were designed via a genetic algorithm. While reducing memory requirements is still advantageous for smaller-sized chipsets (van Diggelen, 2014, 2020), many of today’s low-end chipsets process the Galileo signals that use memory codes, including the world’s smallest GNSS chipset in 2022 produced by u-blox (u-blox, 2022).

Spreading code optimization is a challenging combinatorial optimization problem, as the objective function is generally nonconvex and the number of binary variables can be large, on the order of 10,000 or more. As a result, most prior work has focused on either population-based methods or heuristics based on an exhaustive search. The former methods include genetic algorithms (Wallner et al., 2007, 2008) and NESs (Mina & Gao, 2022), whereas the latter methods typically involve a variant of BCD. BCD methods repeatedly assess a subset (or block) of binary variables and search for an arrangement of those variables that can improve the objective value. BCD methods have been used to optimize binary spreading codes for code-division multiple access applications, particularly for radar (Alaee-Kerahroodi et al., 2019; Cui et al., 2017; Huang & Lin, 2020; Lin & Li, 2020). However, those works considered small block sizes (typically one), as the number of combinations that must be assessed to update a block of size  $B$  is  $2^B$ . However, the use of larger block sizes can lead to codes with better performance, in fewer iterations (Yang et al., 2023a).

This study generalizes the work of Yang et al. (2023a), in which the code design problem was formulated as an equivalent mixed-integer quadratic program (MIQP). With this formulation, the block updates may be performed by solving a small MIQP using a branch-and-bound-based solver such as Gurobi (Gurobi Optimization, LLC, 2022), instead of an exhaustive search. This approach allows larger block sizes to be used.

### 3 | SPREADING CODE OPTIMIZATION

#### 3.1 | Preliminaries

The goal is to identify a set (or family) of  $m$  spreading codes  $(x^0, \dots, x^{m-1})$ , where each  $x^i \in \{\pm 1\}^n$  is a length- $n$  binary sequence. We represent  $x \in \{\pm 1\}^{n \times m}$  as a matrix, where  $x^i$  is the  $i$ -th column of  $x$ , for  $i = 0, \dots, m - 1$ .

**Cross-Correlation.** For any two  $i, j \in 0, \dots, m-1$ , we define the cross-correlation between the two codes  $x_i$  and  $x_j$  as a length- $n$  vector  $(x^i \star x^j)$ , given by the following:

$$(x^i \star x^j)_k = \sum_{s=0}^{n-1} x_s^i x_{(s+k) \bmod n}^j, \quad k = 0, \dots, n-1 \quad (1)$$

Here,  $(x^i \star x^j)_k$  is the inner product between  $x^i$  and a  $k$ -circularly shifted version of  $x^j$ . The cross-correlation may also be defined in terms of negative shifts, *i.e.*, the cross-correlation of  $x^i$  with  $x^j$  at shift  $-k$  is equal to  $(x^i \star x^j)_{n-k}$  for any  $k$ . Note that the periodic cross-correlation in Equation (1) also exhibits the following symmetry:  $(x^i \star x^j)_k = (x^j \star x^i)_{n-k}$ .

In this work, we exclusively consider the periodic cross-correlation, although an extension to the aperiodic case is straightforward (see, *e.g.*, the work by Song et al. (2016)). The cross-correlation levels are sometimes referred to as cross-correlation sidelobes.

**Autocorrelation.** We refer to the cross-correlation of  $x^i$  with itself as the autocorrelation of  $x^i$ . Note that for any binary sequence  $y \in \{\pm 1\}^n$ , the zero-shift autocorrelation is as follows:

$$(y \star y)_0 = \sum_{s=0}^{n-1} y_s^2 = n \quad (2)$$

which is a constant that does not depend on  $y$ .

**Spreading Code Design.** The goal here is to choose  $x$  such that  $(x^i \star x^j)_k$  is suitably close to zero for all  $i, j$ , and  $k$ , with the exception of zero-shift autocorrelation values. In this work, we achieve this goal by minimizing a convex function of the relevant autocorrelation and cross-correlation values. To formalize this, we consider the pairwise cross-correlation function  $h: \{\pm 1\}^{n \times m} \rightarrow \mathbb{R}^{n \times m \times m}$ , which we define as follows:

$$h(x)_i^{j,k} = (x^i \star x^j)_k, \quad i, j = 0, \dots, m-1, \quad k = 0, \dots, n-1 \quad (3)$$

Here, the first dimension of  $h(x)$  is indexed by a subscript, and the last two dimensions are indexed by superscripts. The function  $h$  maps the spreading code family  $x$  to all pairwise cross-correlations, at every shift. Minimizing every component of  $h(x)$  involves a set of competing objectives, as reducing one cross-correlation value may come at the expense of increasing a different cross-correlation value.

For ease of notation, we do not exclude the zero-shift autocorrelation values in the definition of  $h$  unless otherwise specified, because they are constant. In addition, we note that by symmetry, the autocorrelation may be represented by just  $\lfloor n/2 \rfloor$  unique terms.

**Mixed-Integer Convex Optimization Problem.** Here, we aim to achieve a trade-off by minimizing a convex scalar function of the autocorrelation and cross-correlation values, given by  $g: \mathbb{R}^{n \times m \times m} \rightarrow \mathbb{R} \cup \{\infty\}$ . In addition, we may have a convex regularization or constraint function  $r: \{\pm 1\}^{n \times m} \rightarrow \mathbb{R} \cup \{\infty\}$ . The functions  $g$  and  $r$  may represent constraints by taking a value of infinity for disallowed values of the cross-correlation and code, respectively. A comprehensive background on convex functions and convex optimization may be found in the book by Boyd

and Vandenberghe (2004). We may write the spreading code design problem as the following optimization problem:

$$\begin{aligned} & \text{minimize} && g(h(x)) + r(x) \\ & \text{subject to} && x \in \{\pm 1\}^{n \times m} \end{aligned} \quad (4)$$

In general, this combinatorial optimization problem is challenging. We note that the composition of  $h$  and  $g$  is nonconvex even though  $g$  is convex, because  $h$  is nonconvex.

In Section 3.2, we show how Equation (4) may be converted into the minimization of a convex function subject to binary constraints, following the approach of Yang et al. (2023a). This conversion is achieved by introducing auxiliary variables and constraints and replacing  $h$  with an equivalent linear function of auxiliary variables. Treating the problem as the minimization of a convex function over the nonconvex set of binary variables enables search heuristics based on convex optimization. Those methods, which are discussed in Section 4, utilize the fact that when the binary constraints are removed, the problem in Equation (4) becomes a convex optimization problem, which can be efficiently solved (Boyd & Vandenberghe, 2004).

### 3.2 | Spreading Code Optimization as Convex Optimization Over Binary Variables

In this section, we review the method of Yang et al. (2023a). In that work, the cross-correlation function in Equation (1) was represented as a linear function of additional auxiliary variables, subject to additional linear inequality constraints. This approach allowed the problem in Equation (4) to be cast as a minimization of a convex function of the variables, subject to binary constraints.

**Product of Binary Variables.** First, let us consider two binary variables:  $u, v \in \{\pm 1\}$ . The product  $uv$  is not a convex function of  $u$  and  $v$ . However, suppose that we introduce a new auxiliary variable  $w$ . Then, we have  $w = uv$  for any of the four possible combinations of  $u$  and  $v$ , if and only if  $w$  satisfies the four linear inequality constraints (Yang et al., 2023a, 2023b):

$$w \leq -u + v + 1 \quad (5)$$

$$w \leq -v + u + 1 \quad (6)$$

$$w \geq -1 - u - v \quad (7)$$

$$w \geq -1 + u + v \quad (8)$$

We refer to these constraints as *linking constraints*, as they relate the auxiliary variable  $w$  to the binary variables  $u$  and  $v$ . Note that the linking constraints define a convex set, corresponding to the intersection of four halfspaces.

**Cross-Correlation of Binary Variables.** Recall that the cross-correlation between two binary sequences  $x^i$  and  $x^j$ , given by Equation (1), is a sum of products of binary variables. Thus, we may introduce auxiliary variables

$z^{i,j} \in \{\pm 1\}^{n \times n}$  that each satisfy the linking constraints in Equations (5)–(8). This approach leads to the following linking constraints:

$$z_{s,k}^{i,j} \leq -x_s^i + x_{(s+k) \bmod n}^j + 1 \quad (9)$$

$$z_{s,k}^{i,j} \leq -x_{(s+k) \bmod n}^j + x_s^i + 1 \quad (10)$$

$$z_{s,k}^{i,j} \geq -1 - x_s^i - x_{(s+k) \bmod n}^j \quad (11)$$

$$z_{s,k}^{i,j} \geq -1 + x_s^i + x_{(s+k) \bmod n}^j \quad (12)$$

that is,  $z_{s,k}^{i,j} = x_s^i x_{(s+k) \bmod n}^j$ . Then, subject to these constraints, we may represent the cross-correlation in Equation (1) using the following linear function:

$$(x^i \star x^j)_k = \sum_{s=0}^{n-1} z_{s,k}^{i,j} \quad k = 0, \dots, n-1 \quad (13)$$

We will write the auxiliary variables as a four-dimensional array  $z \in \mathbb{R}^{n \times n \times m \times m}$ , where the first two components are indexed by subscripts and the last two components are indexed by superscripts. Note that the number of unique auxiliary variables is the number of unique pairs of binary variables, which is  $\binom{nm}{2}$ .

We may now define a linearized pairwise cross-correlation function  $f: \mathbb{R}^{n \times n \times m \times m} \rightarrow \mathbb{R}^{n \times m \times m}$  as follows:

$$f(z)_i^{j,k} = \sum_{s=0}^{n-1} z_{s,k}^{i,j}, \quad i, j = 0, \dots, m-1, \quad s, k = 0, \dots, n-1 \quad (14)$$

Similar to Equation (3), the first dimension of  $f(x)$  is indexed by a subscript, and the last two dimensions are indexed by superscripts.

We may now replace the nonconvex function  $h$  in Equation (4) with the linearized function  $f$ . Let  $C$  denote the convex set such that  $(x, z) \in C$  implies that all of the linking constraints in Equations (9)–(12) are satisfied for all  $i, j, k, s$ . Then, the spreading code optimization problem in Equation (4) may be written as the following equivalent problem:

$$\begin{aligned} & \text{minimize} && g(f(z)) + r(x) \\ & \text{subject to} && x \in \{\pm 1\}^{m \times n}, \quad (x, z) \in C \end{aligned} \quad (15)$$

The problem in Equation (15) is a MICP, as we have integer-valued constraints on only some of the variables ( $x$ ) and the problem is convex when the binary constraints are removed. The latter conclusion follows from the fact that the composition of a linear function  $f$  with a convex function  $g$  is convex (Boyd & Vandenberghe, 2004, Section 3.2.2).

### 3.2.1 | Extensions of the Cross-Correlation Function

The problem formulation in Equation (15) may be directly extended to the case in which the cross-correlation function  $f$  in Equation (14) involves any linear

function of auxiliary variables, instead of simple sums. That is, we may consider the following function:

$$\tilde{f}(z)_i^{j,k} = \sum_{s=0}^{n-1} a_{s,k}^{i,j} z_{s,k}^{i,j}, \quad i, j = 0, \dots, m-1, \quad s, k = 0, \dots, n-1 \quad (16)$$

which corresponds to replacing the cross-correlation function in Equation (1) with a modified cross-correlation function:

$$\varphi(x^i, x^j)_k = \sum_{s=0}^{n-1} a_{s,k}^{i,j} x_s^i x_{(s+k)_{\text{mod } n}}^j \quad k = 0, \dots, n-1 \quad (17)$$

When  $a_{s,k}^{i,j} = 1$ , we recover the original cross-correlation function, *i.e.*,  $\varphi(x^i, x^j)_k = (x^i \star x^j)_k$ .

Modified cross-correlation functions of this form appear in certain code design applications. One example involves *odd* cross-correlation. Odd cross-correlation represents the cross-correlation observed when a data bit transition is present. In this case, the binary data stream is superimposed on the spreading code, as is the case with the GPS L1 C/A signal (NAVSTAR GPS Directorate, 2022; Winkel, 2011). As a result, the code flips in sign between successive cycles. The odd cross-correlation may be expressed via the modified cross-correlation function  $\varphi$ , with the following:

$$a_{s,k}^{i,j} = \begin{cases} +1, & \text{for } (s+k)_{\text{mod } n} \geq 0 \\ -1, & \text{otherwise} \end{cases} \quad (18)$$

### 3.3 | Objective Functions for Spreading Code Design

In this section, we review several objective functions that have been proposed in the literature for designing spreading codes. For each, we show how the objective may be represented by a suitable choice of the function  $g$  for the optimization problem in Equation (15).

**Mean of Squares.** We first consider the mean of squared autocorrelation and cross-correlation values, given as follows:

$$g(c) = \frac{1}{n \left( m + \binom{m}{2} \right)} \sum_{i=0}^{m-1} \sum_{j=i}^{m-1} \sum_{k=0}^{n-1} (c_k^{i,j})^2 \quad (19)$$

The argument  $c \in \mathbb{R}^{n \times m \times m}$  represents the auto- and cross-correlation values, where  $c_k^{i,j} = (x^i \star x^j)_k$ . Note that we sum from  $j=i$  to  $m-1$  because of the symmetry property with periodic cross-correlation, as discussed in Section 3.1, where  $(x^i \star x^j)_k = (x^j \star x^i)_{n-k}$ . In some contexts, this objective is referred to as the ISL (He et al., 2012). There are  $\binom{m}{2}$  cross-correlation and  $m$  autocorrelation vectors, each of length  $n$ . The mean-of-squares objective  $g(h(x))$  is a nonconvex quartic function of  $x$ . However, with the use of the linearized cross-correlation,  $g(f(x))$  becomes a convex quadratic function of the auxiliary variables  $z$ , and Equation (15) becomes a MIQP.

**Balanced Mean of Squares.** In the mean-of-squares objective, there are more cross-correlation terms than autocorrelation terms, which can lead to codes with cross-correlation values that are relatively larger than the autocorrelation values. To address this, Mina and Gao (2022) proposed minimizing the larger of the

mean of squared autocorrelations and the mean of squared cross-correlations. We refer to this as the balanced mean of squares, which corresponds to the following choice of  $g$ :

$$g(c) = \max \left\{ \frac{1}{n \binom{m}{2}} \sum_{i=0}^{m-1} \sum_{j=i+1}^{m-1} \sum_{k=0}^{n-1} (c_k^{i,j})^2, \frac{1}{m(n-1)} \sum_{i=0}^{m-1} \sum_{k=1}^{n-1} (c_k^{i,i})^2 \right\} \quad (20)$$

Because  $g$  is the maximum of two convex quadratic functions, it is also convex. Moreover, the optimization problem in Equation (15) becomes a mixed-integer second-order cone program (MISOCP) (Boyd & Vandenberghe, 2004; Gurobi Optimization, LLC, 2022).

**Peak Sidelobe Level.** Another choice of objective is the PSL, which is the maximum absolute value of the autocorrelation and cross-correlation values (He et al., 2012). This objective corresponds to the function  $g$  given as follows:

$$g(c) = \max_{i,j,k} |c_k^{i,j}| \quad (21)$$

Because  $g$  is convex and piecewise linear in this case, the problem in Equation (15) becomes a mixed-integer linear program.

**Mean of the  $p$ -th Power Absolute Correlation Levels.** Instead of minimizing a mean of squares, we may consider minimizing a mean of absolute autocorrelation and cross-correlation values taken to the power of some positive value  $p$ , based on the generalized correlation objective defined by Winkel (2011). Although Winkel (2011) defined a generalized correlation objective for even values of  $p$ , by considering the absolute correlation magnitude, we can choose odd values of  $p$  as well, leading to the following choice of  $g$ :

$$g(c) = \frac{1}{n \left( m + \binom{m}{2} \right)} \sum_{i=0}^{m-1} \sum_{j=i}^{m-1} \sum_{k=0}^{n-1} |c_k^{i,j}|^p \quad (22)$$

Similar to the PSL objective, this objective may be used to more heavily penalize large correlation peaks by selecting larger values of  $p$  (e.g.,  $p \geq 4$ ). Furthermore, this objective can be easier to minimize than the PSL objective, because it is smooth. When  $p = 2$ , we recover the mean of squares. Here, the optimization problem in Equation (15) may be represented as a MISOCP.

### 3.4 | Constraints and Regularization

In this section, we discuss some possible constraints on the spreading codes, which may be incorporated into the functions  $g$  or  $r$  in the optimization problem in Equation (15) as either convex constraints or convex penalty functions.

**Balance Constraints.** For a particular code sequence, the difference between the number of +1 and -1 entries is called the *balance* of the sequence (Winkel, 2011). In many cases, it is desirable for the number of +1 and -1 entries in each code to be close to the same or to have a balance that is close to 0. For example, for the GPS L1C signal, the length-10,230 spreading codes are designed to have a balance of zero (Rushanan, 2007). Incorporating balance constraints ensures that the statistical properties of each code are similar to those of random noise. Moreover,

reducing the mean absolute level of the codes reduces the power needed to transmit the signal.

Balance constraints may be implemented by adding linear inequality constraints to the optimization problem in Equation (15). Formally, we set  $r$  to be an indicator function:

$$r(x) = \begin{cases} 0 & a \leq \sum_{k=0}^{n-1} x_k^i \leq b, \quad i = 0, \dots, m-1 \\ \infty & \text{otherwise} \end{cases} \quad (23)$$

where  $a$  and  $b$  are fixed parameters. Here,  $a$  and  $b$  represent the maximum allowable excess  $-1$  and  $+1$  entries, respectively.

A scalar penalty term may also be used instead of constraints. For example, we may use a quadratic imbalance penalty:

$$r(x) = \rho \sum_{i=0}^{m-1} \left( \sum_{s=0}^{n-1} x_s^i \right)^2 \quad (24)$$

where  $\rho > 0$  is a penalty parameter. When  $n$  is even and  $\rho$  is sufficiently large, the optimal solution will have codes with equal numbers of  $+1$  and  $-1$ .

**Autocorrelation Sidelobe Zero.** A binary sequence  $y \in \{\pm 1\}^n$  that satisfies the (ASZ) property exhibits minimal autocorrelation at a shift of 1 (and equivalently, at a shift of  $-1$ ). This corresponds to the condition  $(y \star y)_1 = 0$  in the even-length case and  $|(y \star y)_1| = 1$  in the odd-length case. The ASZ property can improve the ability of a receiver to accurately identify the originating satellite of a received signal. If a receiver detects similar correlations at shifts of 0 and 1, it may erroneously lose the signal lock. The Galileo constellation employs even-length spreading codes that have been designed to satisfy the ASZ property (Wallner et al., 2007).

The ASZ condition is described by a set of linear equality constraints. In the even-length case, we may take  $g$  to be the following indicator function:

$$g(c) = \begin{cases} \tilde{g}(c) & c_1^{i,i} = 0, \quad i = 0, \dots, m-1 \\ \infty & \text{otherwise} \end{cases} \quad (25)$$

where  $\tilde{g}: \mathbb{R}^{n \times m \times m} \rightarrow \mathbb{R} \cup \{\infty\}$  is a convex function, such as the mean-of-squares objective in Equation (19). In the odd-length case, we may set the right-hand sides of the equality constraints to either  $+1$  or  $-1$ . The ASZ constraints may also be converted into a penalty term, similar to the balance constraints. For example, we may take the following:

$$g(c) = \tilde{g}(c) + \rho \sum_{i=0}^{m-1} (c_1^{i,i})^2 \quad (26)$$

where  $\tilde{g}$  is convex and  $\rho > 0$  is a penalty parameter.

**Peak Correlation Level Constraint.** Minimizing the mean of squares in Equation (19) or its variants, such as the balanced mean of squares in Equation (20), can lead to codes with autocorrelation and cross-correlation values that are small on average. However, the codes may have a relatively large peak absolute correlation, in comparison with codes designed to minimize the PSL objective (Equation (21)).

Conversely, the PSL objective seeks to minimize the worst-case correlation. The Gold codes, for example, have been proven to exhibit favorable PSL properties. However, PSL minimization can be overly conservative, resulting in codes with

average correlation values that exceed those of a code family specifically optimized to have a small mean of squares.

One strategy for achieving a trade-off is to minimize the mean of squares, subject to a maximum absolute correlation constraint. That is, we may take the following:

$$g(c) = \begin{cases} \tilde{g}(c) & -c^{\max} \leq c_k^{i,j} \leq c^{\max}, \quad (i, j, k) \in \mathcal{J} \\ \infty & \text{otherwise} \end{cases} \quad (27)$$

where  $\tilde{g}$  is convex,  $c^{\max}$  is a fixed parameter that represents the maximum allowable PSL value, and  $\mathcal{J}$  is the index set on which the constraints are enforced. Typically,  $\mathcal{J}$  would exclude the zero-shift autocorrelation values, as those are constant and equal to  $n$ . Thus, we would take the following:

$$\mathcal{J} = \{(i, j, k) \mid i \neq j \text{ or } (i = j \text{ and } k = 0)\} \quad (28)$$

## 4 | SOLUTION METHOD

In principle, the binary-constrained convex optimization problem in Equation (15) may be solved to optimality by using global optimization methods such as branch-and-bound (Brucker et al., 1994; Lawler & Wood, 1966), branch-and-cut (Padberg & Rinaldi, 1991; Stubbs & Mehrotra, 1999), direct enumeration, or a combination of the aforementioned methods. In each iteration, a convex optimization problem derived from Equation (15) is solved, with the binary constraints removed and possibly additional variables and convex constraints added. Efficient commercial solvers exist that utilize a combination of these solution methods, along with other proprietary techniques. For example, Gurobi (Gurobi Optimization, LLC, 2022) is a state-of-the-art commercial solver that is widely used in academia and industry. Noncommercial open-source alternatives also exist for MICPs, such as the SCIP Optimization Suite (Bestuzheva et al., 2021).

Owing to the large problem sizes typical of spreading code design problems, direct application of those methods is generally impractical (Yang et al., 2023a). Instead, local solution methods and heuristics may be used. For example, heuristics based on the alternating direction method of multipliers (ADMM) may also be used (Diamond & Boyd, 2016). ADMM-based methods alternate between solving a convex optimization problem, projection (here, rounding to  $\pm 1$ ), and enumeration (Diamond et al., 2017). In this work, we consider a BCD method that combines a local search with global optimization methods (Yang et al., 2023a; Yuan et al., 2017).

### 4.1 | Block Coordinate Descent

BCD is an iterative method that repeatedly solves the optimization problem (Equation (15)) over subsets of the variables, or blocks, with the other variables held fixed at their previous values. Once a block is selected in each iteration, the optimization problem is solved exactly, but only over the selected variables in the current block. Let  $B$  be the block size. In each iteration, the block optimization problem may be solved, in principle, via an exhaustive search, by checking all  $2^B$  combinations of the  $B$  selected binary variables and choosing the combination that minimizes the objective value. In this work, we instead solve the MICP in Equation (15) over the selected variables in the block using Gurobi. For modestly

sized  $B$ , we find that Gurobi can solve the block optimization problem faster than a brute-force search. The values of the variables in the block are then set to their minimizing values, and a new block of variables is selected in the next iteration.

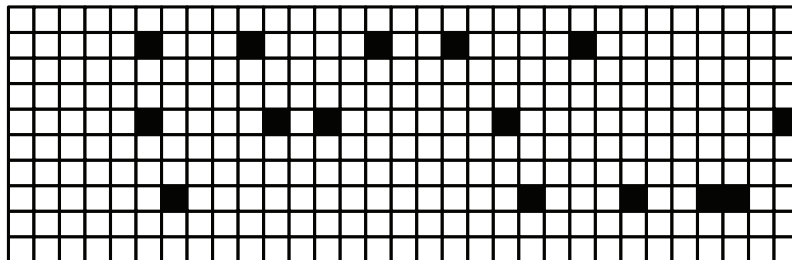
This process is repeated until a predetermined termination condition is satisfied. For example, we may set a maximum number of iterations or terminate when a satisfactory level of solution quality is met. Because the block optimization problems are performed exactly, BCD is a descent method. That is, the objective function is guaranteed to be non-increasing between iterations.

**Block Selection Strategy.** We consider a random block selection strategy in our experiments. In each iteration, we select  $B$  of the  $nm$  binary variables at random. However, we limit the selected variables to be members of only  $K$  of the codes, in order to reduce the difficulty of solving the block optimization problem. We have applied this limitation because the number of auxiliary variables that must be considered grows quadratically with  $K$  (Yang et al., 2023a). In this work, we select blocks by first choosing  $K$  codes at random and then randomly choosing  $B/K$  variables from each of the  $K$  selected codes.

In general, the block size  $B$  and number of codes  $K$  should be chosen such that the block optimization problem can be solved quickly, for example, using Gurobi. For problems with very long code lengths, for example, with  $n$  on the order of 10,000, it may not be possible to choose blocks such that the block optimization problems can be effectively solved, owing to the complexity of the resulting MICPs. In this case, it may be more efficient to simply solve the block optimization problems using an exhaustive search.

Figure 1 illustrates the block selection scheme for an example with  $m = 10$  codes, each of length  $n = 50$ . Each cell represents one of the binary variables. Variables that have been selected as part of the current block are shaded in black, and variables that are held fixed are not shaded. In this example, the block size is  $B = 15$ , and the variables to be chosen are restricted by choosing 5 variables at random, from only 3 of the codes.

**Handling Constraints.** The BCD method may be modified to handle the balance and ASZ constraints described in Subsection 3.4. One potential issue is that the block optimization problem may be infeasible, if the initial code family used to initialize BCD does not satisfy the constraints. However, as long as the initial code family satisfies the constraints, all following BCD iterations will also satisfy the constraints. An initial code that satisfies the balance constraints may be generated by taking any code family and flipping the signs of a subset of code entries, such that the sum of each code is zero (in the even-length case) or  $+1$  or  $-1$  (in the odd-length case). The method proposed by Yang et al. (2024) may be used to generate an initial code family that satisfies the ASZ property.



**FIGURE 1** Illustration of block selection in BCD for the case of  $m = 10$  codes, with code length  $n = 50$ , block size  $B = 15$ , and block variables selected from  $K = 3$  codes. Black cells represent selected variables, and white cells represent variables that are not selected.

## 4.2 | Theoretical Results

The code design optimization problem in Equation (4) is an NP-hard combinatorial optimization problem that, in the worst case, requires an assessment of  $2^{nm}$  possible combinations of binary code entries. Because the objective function is non-increasing between iterations, the BCD method is guaranteed to converge. However, it is not guaranteed to converge to the global optimum. Theoretical lower bounds on the optimal objective value are available for certain choices of the objective function. For example, in the absence of constraints, the Welch bound provides a lower bound on the PSL (Welch, 1974).

## 5 | NUMERICAL EXAMPLE

We consider the design of a spreading code family of size  $m = 31$ , with code length  $n = 1023$ , based on the legacy GPS L1 C/A code family (Misra & Enge, 2012). In our experiments, we consider the mean-of-squares and balanced mean-of-squares objectives, as well as the use of balance constraints. We compare our results against an NES method (Mina & Gao, 2022), Gold codes, and random codes.

In the NES method considered, a probability distribution is modeled over the spreading codes, and the distribution is iteratively updated to improve the objective value. The updates are performed via a gradient-based method, applied to a stochastic estimate of the gradient. For a code length of 1023, there are a total of 1025 Gold codes. To select the Gold codes to use for comparison, we generated 10,000 random subsets of size 31 and selected the subset with the best objective. To compare against random codes, we generated 10,000 codes with entries chosen uniformly at random and selected the code with the best objective. For both the Gold codes and random codes, we took the best set of codes out of 10,000 codes to reduce the impact of random sampling.

For each case, we applied BCD with block size  $B = 15$  and the block selection strategy described in Subsection 4.1. That is, 3 of the 31 codes were chosen at random, and 5 variables were chosen from each of the selected codes. BCD was initialized using a code family with entries chosen uniformly at random.

**Comparison Against NES and Gold Codes.** We minimized the balanced mean-of-squares objective using the BCD method and compared the results against the Gold codes and the NES method proposed by Mina and Gao (2022). Figure 2 shows the objective value per iteration. The BCD method finds a code family that outperforms the Gold codes after 59 iterations and a code family that outperforms the one found by NES after 296 iterations.

We also note that for the mean-of-squares objective in Equation (19), the BCD method found a code family with an objective value of 909.85, thus outperforming the Gold codes, which have a mean-of-squares objective value of 953.87. The NES method did not consider the mean-of-squares objective value.

**Balanced vs. Nominal Mean of Squares.** We also compared codes optimized for the balanced mean-of-squares objective (Equation (20)) against codes optimized for the nominal mean-of-squares objective (Equation (19)). Figure 3 presents histograms of the obtained absolute autocorrelation and cross-correlation values for all shifts. The compared codes were both found using the same BCD method described above. We see that the code optimized for the balanced objective has similar distributions of absolute autocorrelation and cross-correlation values,

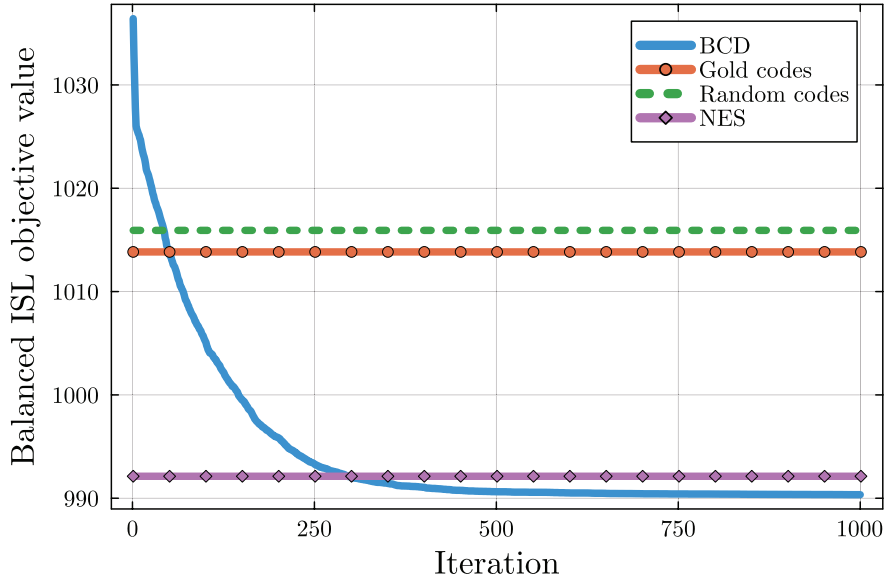


FIGURE 2 Balanced ISL objective value as a function of BCD iteration

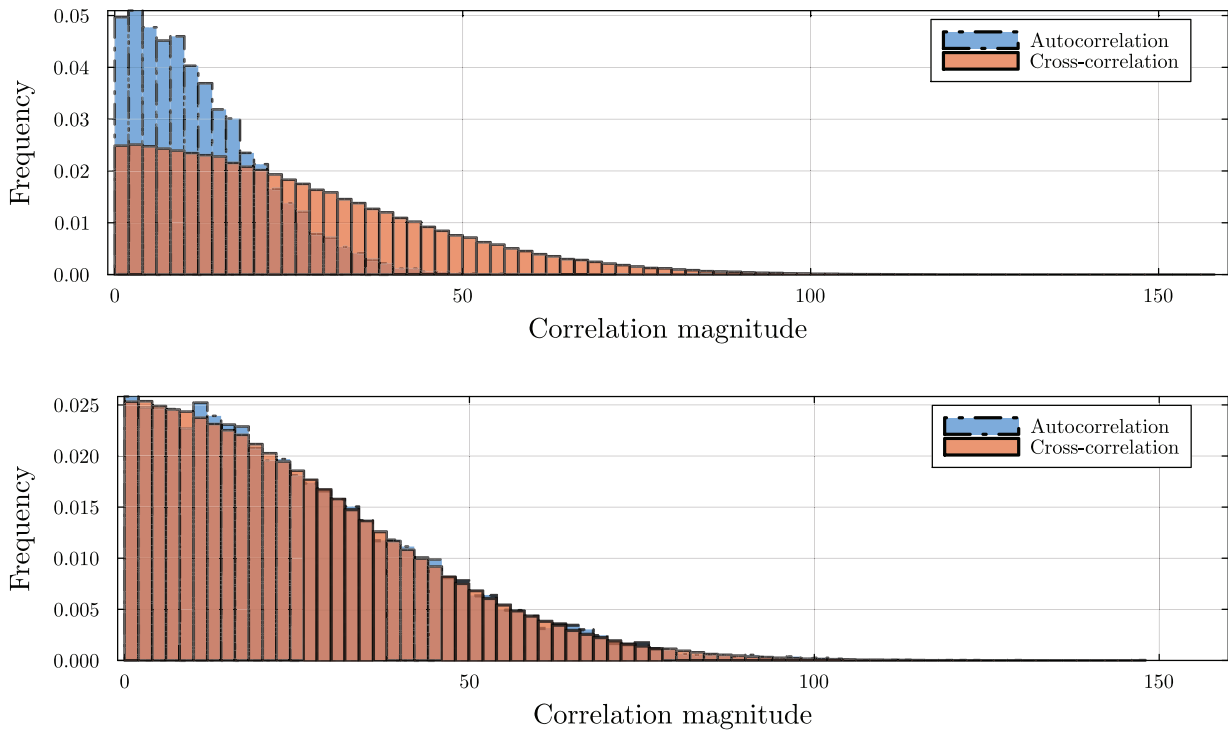
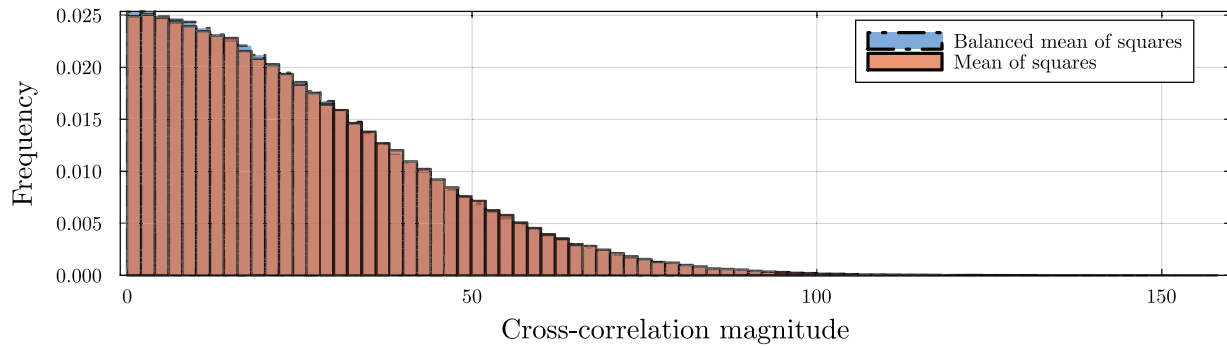


FIGURE 3 Frequency of absolute autocorrelation and cross-correlation values when minimizing the mean of squares (top) vs. balanced mean of squares (bottom) using BCD with a subset size of 15  
The problem involved finding  $m = 31$  codes, each of length 1023.

as expected. In contrast, the code optimized for the nominal mean of squares has larger values for cross-correlation than autocorrelation.

The code optimized for the balanced objective has a slightly lower average cross-correlation value (1012.17 vs. 992.10), but as shown in Figure 4, the distributions of cross-correlation values are nearly identical for the two codes. Whereas



**FIGURE 4** Frequency of absolute cross-correlation values when minimizing the mean of squares vs. balanced mean of squares using the same BCD method

the balanced objective can be used to trade off autocorrelation performance for cross-correlation performance when  $m$  is small (e.g., 3) (Mina & Gao, 2022), the results suggest that this trade-off may not be achievable for  $m = 31$ .

**Balance Constraints.** Finally, we compared the BCD method with and without balance constraints, for both the mean-of-squares and balanced mean-of-squares objectives. Because the code length is odd, the balance constraints ensure that the sum of each code is either  $+1$  or  $-1$ .

For the balanced mean-of-squares objective, BCD found a code with an objective value of 990.27 when no balance constraints were enforced. The resulting code family has a maximum sum of 83, which is a significant imbalance. When balance constraints were enforced, the BCD method found a code family with a balanced mean-of-squares value of 991.18, which is only slightly larger.

For the mean-of-squares objective, BCD found a code family with an objective value of 909.85 without balance constraints and 911.34 with balance constraints. The code family optimized without balance constraints had a maximum sum of 47.

In both cases, imposing the balance constraints lead to optimized codes with slightly larger but comparable objective values.

## 6 | CONCLUSION

In this paper, we showed that various versions of the spreading code optimization problem may be formulated as MICPs, demonstrating that simple-to-tune algorithms such as BCD may be effectively used to find competitive spreading codes. Without the need for *ad hoc* solution methods, which often require extensive tuning, our problem formulation may simplify the process of tuning and testing objective functions and constraints to suit the needs of specific applications.

## ACKNOWLEDGMENTS

This material is based upon work supported by the Air Force Research Lab under grant number FA9453-20-1-0002.

## REFERENCES

- Air Force Research Laboratory. (2024). *Navigation Technology Satellite-3 (NTS-3)*. Retrieved April 11, 2024, from <https://afresearchlab.com/technology/nts-3>
- Alaee-Kerahroodi, M., Modarres-Hashemi, M., & Naghsh, M. M. (2019). Designing sets of binary sequences for MIMO radar systems. *IEEE Transactions on Signal Processing*, 67(13), 3347–3360. <https://doi.org/10.1109/TSP.2019.2914878>

- Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S. J., ... Witzig, J. (2021). *The SCIP optimization suite 8.0*. arXiv. <https://doi.org/10.48550/arXiv.2112.08872>
- Boukerma, S., Rouabah, K., Mezaache, S., & Atia, S. (2021). Efficient method for constructing optimized long binary spreading sequences. *International Journal of Communication Systems*, 34(4), e4719. <https://doi.org/10.1002/dac.4719>
- Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge University Press. <https://doi.org/10.1017/CBO9780511804441>
- Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Math*, 49, 107–127. [https://doi.org/10.1016/0166-218X\(94\)90204-6](https://doi.org/10.1016/0166-218X(94)90204-6)
- Chapman, D., Hinks, J., & Anderson, J. (2020). Way, way out in front—Navigation Technology Satellite-3: The vanguard for space-based PNT. *Inside GNSS*, 15(4), 32–35. <https://insidignss.com/way-way-out-in-front-navigation-technology-satellite-3-the-vanguard-for-space-based-pnt/>
- Cui, G., Yu, X., Foglia, G., Huang, Y., & Li, J. (2017). Quadratic optimization with similarity constraint for unimodular sequence synthesis. *IEEE Transactions on Signal Processing*, 65(18), 4756–4769. <https://doi.org/10.1109/TSP.2017.2715010>
- Diamond, S., & Boyd, S. (2016). CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83), 1–5. <https://www.jmlr.org/papers/volume17/15-408/15-408.pdf>
- Diamond, S., Takapoui, R., & Boyd, S. (2017). A general system for heuristic solution of convex problems over nonconvex sets. *Optimization Methods and Software*, 33(1), 165–193. <https://doi.org/10.1080/10556788.2017.1304548>
- Gold, R. (1967). Optimal binary sequences for spread spectrum multiplexing. *IEEE Transactions on Information Theory*, 13(4), 619–621. <https://doi.org/10.1109/TIT.1967.1054048>
- Gurobi Optimization, LLC. (2022). *Gurobi optimizer reference manual*. <https://www.gurobi.com>
- He, H., Li, J., & Stoica, P. (2012). *Waveform design for active sensing systems: A computational approach*. Cambridge University Press. <https://doi.org/10.1017/CBO9781139095174>
- Huang, W., & Lin, R. (2020). Efficient design of Doppler sensitive long discrete-phase periodic sequence sets for automotive radars. *Proc. of the 2020 IEEE 11th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, Hangzhou, China, 1–5. <https://doi.org/10.1109/SAM48682.2020.9104358>
- Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research*, 14(4), 699–719. <https://doi.org/10.1287/opre.14.4.699>
- Lin, R., & Li, J. (2020). On binary sequence set design with applications to automotive radar. *Proc. of the ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Barcelona, Spain, 8639–8643. <https://doi.org/10.1109/ICASSP40776.2020.9054143>
- Mina, T. Y., & Gao, G. X. (2022). Designing low-correlation GPS spreading codes with a natural evolution strategy machine-learning algorithm. *NAVIGATION*, 69(1). <https://doi.org/10.33012/navi.506>
- Misra, P., & Enge, P. (2012). *Global Positioning System: Signals, measurements, and performance* (revised 2nd ed.). Ganga-Jamuna Press. <https://www.navtechgps.com/global-positioning-system-signals-measurements-and-performance-revised-second-edition-paperback/>
- NAVSTAR GPS Directorate. (2022). *NAVSTAR GPS space segment/navigation user segment interfaces (IS-GPS-200, Revision N)* (tech. rep.). MilComm & PNT Directorate, Space Systems Command. <https://www.gps.gov/technical/icwg/IS-GPS-200N.pdf>
- Padberg, M., & Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1), 60–100. <https://doi.org/10.1137/1033004>
- Rushanan, J. J. (2007). The spreading and overlay codes for the L1C signal. *NAVIGATION*, 54(1), 43–51. <https://doi.org/10.1002/j.2161-4296.2007.tb00394.x>
- Song, J., Babu, P., & Palomar, D. P. (2016). Sequence set design with good correlation properties via majorization-minimization. *IEEE Transactions on Signal Processing*, 64(11), 2866–2879. <https://doi.org/10.1109/TSP.2016.2535312>
- Soualle, F., Soellner, M., Wallner, S., Avila-Rodriguez, J.-A., Hein, G.W., Barnes, B., Pratt, T., Ries, L., Winkel, J., Lemenager, C., & Erhard, P. (2005). Spreading code selection criteria for the future GNSS Galileo. *Proc. of the European Navigation Conference GNSS*, Munich, Germany, 19–22. <https://www.bibsonomy.org/publication/f204b5741d355adce688b91f35538e84>
- Stubbs, R., & Mehrotra, S. (1999). A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86, 515–532. <https://doi.org/10.1007/s101070050103>
- u-blox. (2022, June). *u-blox launches world's smallest GPS module*. Retrieved April 22, 2024, from <https://www.u-blox.com/en/u-blox-launches-worlds-smallest-gps-module>
- van Diggelen, F. (2014). Who's your daddy? Why GPS will continue to dominate consumer GNSS. *Inside GNSS*, 9(2), 30–41. <https://www.insidignss.com/auto/marapr14-vanDiggelen.pdf>

- van Diggelen, F. (2020). High-sensitivity GNSS. In Morton, Y. T. J., van Diggelen, F., Spilker, J. J., Parkinson, B. W., Lo, S., and Gao, G. (Eds.), *Position, navigation, and timing technologies in the 21st century: Integrated satellite navigation, sensor systems, and civil applications* (Vol. 1, 445–479). Wiley Online Library. <https://doi.org/10.1002/9781119458449.ch18>
- Wallner, S., Avila-Rodriguez, J.-A., Hein, G. W., & Rushanan, J. J. (2007). Galileo E1 OS and GPS L1C pseudo random noise codes – Requirements, generation, optimization and comparison. *Proc. of the 20th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2007)*, Fort Worth, TX, 1549–1563. <https://www.ion.org/publications/abstract.cfm?articleID=7359>
- Wallner, S., Avila-Rodriguez, J.-A., Won, J.-H., Hein, G., & Issler, J.-L. (2008). Revised PRN code structures for Galileo E1 OS. *Proc. of the 21st International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS 2008)*, Savannah, GA, 887–899. <https://www.ion.org/publications/abstract.cfm?articleID=8011>
- Welch, L. (1974). Lower bounds on the maximum cross correlation of signals (corresp.) *IEEE Transactions on Information Theory*, 20(3), 397–399. <https://doi.org/10.1109/TIT.1974.1055219>
- Winkel, J. O. (2011). *Spreading codes for a satellite navigation system* (Patent No.: US 8,035,555 B2). <https://patents.google.com/patent/US8035555B2/en>
- Yang, A., Mina, T., & Gao, G. (2023a). Binary sequence set optimization for CDMA applications via mixed-integer quadratic programming. *IEEE International Conference on Acoustics, Speech, & Signal Processing*, Rhodes Island, Greece, 1–5. <https://doi.org/10.1109/ICASSP49357.2023.10095359>
- Yang, A., Mina, T., & Gao, G. (2023b). Spreading code sequence design via mixed-integer convex optimization. *Proc. of the 36th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS+ 2023)*, Denver, CO, 1341–1351. <https://doi.org/10.33012/2023.19318>
- Yang, A., Mina, T., & Gao, G. (2024). *Spreading code optimization for low-Earth orbit satellites via mixed-integer convex programming*. arXiv. <https://doi.org/10.48550/arXiv.2404.12629>
- Yuan, G., Shen, L., & Zheng, W.-S. (2017). *A hybrid method of combinatorial search and coordinate descent for discrete optimization*. arXiv. <https://doi.org/10.48550/arXiv.1706.06493>

**How to cite this article:** Yang, A., Mina, T., & Gao, G. (2025). Spreading code sequence design via mixed-integer convex optimization. *NAVIGATION*, 72(3). <https://doi.org/10.33012/navi.706>