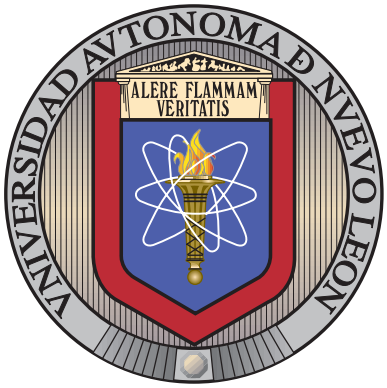
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado



Implementación de un algoritmo para el control de un sistema de heliostatos autónomos

por

Ing. Job Ordaz Castillo

como requisito parcial para obtener el grado de

maestría en ciencias de la ingeniería

con orientación en Energías Térmicas y Renovables

23 de julio de 2025

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Implementación de un algoritmo para el control de un sistema de heliostatos autónomos

por

Ing. Job Ordaz Castillo

como requisito parcial para obtener el grado de

maestría en ciencias de la ingeniería

con orientación en Energías Térmicas y Renovables

23 de julio de 2025

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN
Facultad de Ingeniería Mecánica y Eléctrica
Posgrado

Los miembros del Comité de Evaluación de Tesis recomendamos que la Tesis "Implementación de un algoritmo para el control de un sistema de heliostatos autónomos", realizada por el estudiante Job Ordaz Castillo, con número de matrícula 1838207, sea aceptada para su defensa como requisito parcial para obtener el grado de Maestría en Ciencias de la Ingeniería con Orientación en Energías Térmica y Renovable.

El Comité de Evaluación de Tesis

Dr. Héctor Daniel García Lara
Director

Dr. Santos Méndez Díaz
Co-Director

Dr. Oscar Alejandro de la Garza de León
Revisor

Dr. Daniel de la Rosa Urbalejo
Revisor

Dr. Carlos Alejandro Ramírez Paredes
Revisor

Vo.Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

Institución 190001

Programa 507507

Acta Núm. 4512

Ciudad Universitaria, a 26 de junio de 2025.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Primeramente, agradezco al Creador, fuente de vida, sabiduría y fuerza, por acompañarme en cada paso de este camino y por permitir que este proyecto llegara a buen término.

A mis padres, por su amor incondicional, su ejemplo de trabajo y perseverancia, y por haber creído en mí incluso en los momentos en que yo dudaba. Gracias por enseñarme a no rendirme y por ser mi mayor inspiración.

A mi asesor de tesis, el Dr. Héctor Daniel García Lara, por su guía, paciencia y apoyo constante durante todo el proceso de investigación. Su experiencia y consejos fueron fundamentales no solo para el desarrollo de este trabajo sino tambié para crecer como ser humano y puedo decir que me llevo una gran amistad esperando seguir colaborando en el futuro.

A mi novia la Lic. Nadxhe Matus, por estar a mi lado en los momentos más intensos de esta etapa a pesar de la distancia, por su comprensión, su apoyo emocional, y por recordarme lo que realmente importa cuando el cansancio nublaba la mente.

A mis compañeros de la maestría en especial a Luis, Patricia, Marian, Lilia, Erick, con quienes compartí largas jornadas, desvelos, debates y risas. Cada uno aportó algo valioso en este proceso, y me alegra haber coincidido con personas tan divertidas y comprometidas.

A mis profesores, quienes a lo largo del posgrado me brindaron no solo cono-

cimiento técnico, sino también motivación y perspectiva. En especial agradezco al Dr. Arturo Morales Fuentes y el Dr. Santos Méndez Díaz, por sus consejos, ideas y opiniones siempre constructivas.

A mi familia y amigos, por estar ahí con palabras de aliento, con comprensión ante mis ausencias y con apoyo en los momentos que más lo necesitaba.

A mi amigo el Ing. Víctor RM, por sus consejos, su disposición constante para ayudar y por compartir siempre su experiencia de vida de forma desinteresa.

A la FIME y a la Universidad Autónoma de Nuevo León, por brindarme el espacio, los recursos y la formación académica necesaria para el desarrollo de este trabajo.

Finalmente, agradezco al Consejo Nacional de Ciencia y Tecnología (CONACYT) por el apoyo brindado a través de la beca Nacional (Tradicional) 2023 - 1, sin la cual esta formación académica no habría sido posible.

Este trabajo se realizó en los Laboratorios de Investigación e Innovación en Tecnología Energética (LIITE) del Grupo de Energías Térmica y Renovable (GETR) de la FIME-UANL.

Resumen

Ing. Job Ordaz Castillo.

Candidato para obtener el grado de Maestría en Ciencias de la Ingeniería con orientación en Energías Térmicas y Renovables.

Universidad Autónoma de Nuevo León.

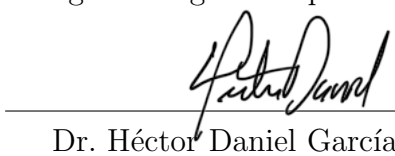
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Implementación de un algoritmo para el control de un sistema de heliostatos autónomos.

Número de páginas: 164.

Objetivos y método de estudio: El método de estudio empleado combina técnicas experimentales y computacionales que incluyen diseño electrónico basado en microcontroladores, implementación de sensores y procesamiento digital de imágenes para obtener datos solares precisos. El objetivo principal de este trabajo es desarrollar e implementar un algoritmo eficiente para el control preciso de un sistema de heliostatos autónomos, optimizando su seguimiento solar mediante técnicas avanzadas de control electrónico. Se diseñó un sistema basado en microcontroladores ESP32, complementado con un método adicional para adquisición y análisis de datos solares mediante el uso de un gnomon digital con procesamiento de imágenes en MATLAB. Este método combinado permitió validar y afinar la precisión del control del heliostato.

Contribuciones y conlusiones: La contribución más significativa de este estudio radica en la implementación exitosa de un algoritmo de control robusto y económico que mejora considerablemente la precisión del posicionamiento de heliostatos autónomos. La incorporación de sensores y la metodología complementaria basada en el análisis digital del gnomon permitieron reducir los errores operativos.

Firma del asesor: 

Dr. Héctor Daniel García Lara

Capítulo 1

Introducción

La energía solar es una fuente sumamente abundante y se clasifica principalmente en dos vertientes: fotovoltaica y térmica. La primera, conocida como energía solar fotovoltaica (PV), convierte la luz solar en electricidad mediante celdas fotovoltaicas que aprovechan el efecto fotoeléctrico. Sin embargo, se sabe que su eficiencia disminuye con el aumento de la temperatura, lo que impacta negativamente en el voltaje y conlleva la necesidad de implementar estrategias de enfriamiento. Como solución a este problema, se han desarrollado soluciones híbridas que combinan la generación fotovoltaica con la producción de energía térmica [1]. Estas configuraciones híbridas (PV/T) logran una mayor eficiencia energética al integrar la generación de electricidad con la recolección de calor residual. Jia et. al. destacan cómo estos enfoques pueden operar bajo diversas condiciones ambientales, utilizando diferentes fluidos de trabajo y configuraciones diversas [2].

La integración y el desarrollo de tecnologías de energía solar, como los sistemas PV/T, representan un paso adelante hacia la optimización del uso de esta fuente renovable. Salari et. al. resaltan los avances en esta área, proponiendo soluciones innovadoras que combinan eficazmente la generación de electricidad con la recolección térmica, maximizando así el potencial de la energía solar [3]. Estas tecnologías ofrecen una alternativa sostenible y eficiente para satisfacer las necesidades energéticas. También contribuyen significativamente a la reducción de emisiones de

gases de efecto invernadero, alineándose con los objetivos globales de sostenibilidad y protección ambiental.

Por otro lado, la energía solar térmica se enfoca en la captación y uso de calor solar. Dentro de esta categoría, encontramos dos divisiones principales: sistemas sin concentración y sistemas de concentración. Los sistemas sin concentración incluyen colectores solares planos, que absorben directamente la radiación solar para calentar un fluido, siendo comúnmente utilizados en aplicaciones residenciales para el calentamiento de agua y calefacción de espacios.

En cambio, los sistemas de concentración solar utilizan espejos o lentes para focalizar la luz solar en un receptor, lo que permite alcanzar temperaturas más altas. Entre las tecnologías de concentración, se destacan los concentradores parabólicos, y los sistemas de torre central, también conocidos como campos de heliostatos. Los concentradores parabólicos, enfocan la luz solar en un tubo receptor para calentar un fluido, que luego puede utilizarse en la producción de energía o en procesos industriales. Los campos de heliostatos, por su parte, concentran la radiación solar en un receptor situado en la cima de una torre, esto resulta ser adecuado para aplicaciones industriales o la generación de electricidad a gran escala.

De acuerdo con Gwiazda et. al. los heliostatos son componentes cruciales en las plantas de torre central, representando entre el 40-50 % del costo total de estas instalaciones [4]. Dado su papel esencial en la eficiencia y rentabilidad de las plantas termosolares, el desarrollo tecnológico en el campo de los helióstatos es de vital importancia. Estos dispositivos requieren ajustes precisos y un control constante para asegurar una reflexión solar óptima hacia el receptor. Por ello, la calibración inicial, que puede realizarse mediante métodos manuales o automáticos utilizando tecnologías avanzadas, es fundamental para maximizar su desempeño y efectividad.

Tras la calibración, es vital que los heliostatos sigan el movimiento solar diario mediante sistemas de control, ajustando su posición en tiempo real. La gestión de estos sistemas varía entre el control centralizado, con una computadora principal

para todo el campo, y el control distribuido, donde cada heliostato opera de forma independiente. Los algoritmos de optimización, como la teoría de optimización convexa, búsqueda heurística o algoritmos genéticos, se utilizan en la disposición de los heliostatos para maximizar la eficiencia energética y reducir costos. Estos se evalúan según criterios como área efectiva disponible, eficiencia de sombra asociada, eficiencia de bloqueos asociado, la captación de energía, la fracción solar, costo nivelado de la energía (LCOE) entre otros.

1.1 Concepto de heliostato

Según la Real Academia Española, un heliostato se puede describir como un dispositivo que, gracias a un servomecanismo, permite que un espejo siga el trayecto diario del Sol [5]. A continuación, se detallará más a fondo cómo funciona este aparato, de igual forma se muestra un diagrama simple en la Figura 1.1.

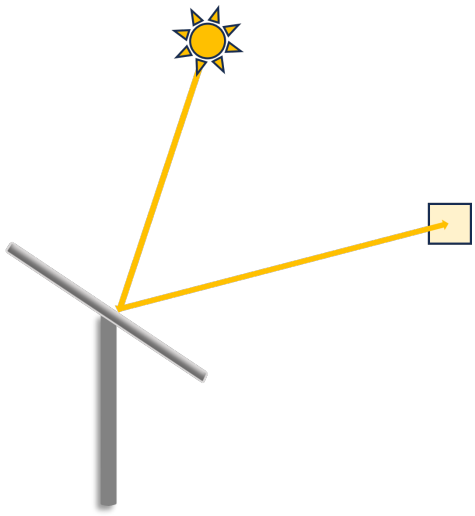


Figura 1.1: Diagrama simple de heliostato

Los heliostatos constan de una o más superficies reflectantes. Son comúnmente utilizados en plantas solares de torre central, donde se disponen en una formación circular alrededor de la torre. Sin embargo, su uso no se restringe a este tipo de instalaciones. También se encuentran en hornos solares y sistemas de iluminación

solar. En estos últimos casos, es más habitual encontrarlos de manera individual o en menor cantidad que en las plantas solares de torre central [6].

El seguimiento del sol realizado por los helióstatos y sus espejos se logra mediante dos sistemas de movimiento que facilitan la orientación en dirección del azimut y de la elevación. Es importante entender que los términos azimut y elevación se refieren a los ejes que determinan la ubicación de un cuerpo celeste en el cielo en un momento específico y desde un lugar concreto. Un sistema permite el ajuste en el azimut, que es el ángulo medido desde el norte geográfico en el plano horizontal, y se logra mediante un movimiento giratorio del helióstato sobre su base fijada al suelo. El otro sistema permite el ajuste en el ángulo de elevación, que se refiere a la altura angular sobre el plano horizontal, tal y como es descrito por Collado et al. [7].

1.2 Antecedentes

1.2.1 Historia del heliostato

La historia del heliostato se remonta a tiempos antiguos, con leyendas atribuyendo a Arquímedes el uso de "vidrio ardiente"para concentrar la luz del sol contra enemigos. Sin embargo, el inventor reconocido del heliostato moderno es el científico holandés Willem Jacob's Gravesande (1688-1742), quien conceptualizó un aparato capaz de girar un espejo a la velocidad adecuada para asegurar un rayo solar constante. El término "heliostato"deriva del griego, significando "Sol estático", refiriéndose a la habilidad de mantener el sol en una posición fija.

Inicialmente, los heliostatos se utilizaban en laboratorios de investigación para estudios sobre fotones y el espectro electromagnético, así como para proveer iluminación constante en microscopios. Con el tiempo, la tecnología del heliostato ha evolucionado significativamente desde su concepción en el siglo XVIII. Los diseños

modernos incorporan servomotores, controladores electrónicos, y actuadores para el seguimiento solar preciso, además de considerar factores como el área de reflexión adecuada, la estructura de soporte, y la estabilidad frente a cargas de viento y deformaciones.

Esta evolución ha sido impulsada por el deseo de maximizar la captación de radiación solar con la mayor eficiencia y al menor costo posible. A lo largo de la historia, los desarrollos en la tecnología del heliostato se han clasificado en diferentes generaciones, reflejando los avances en diseño, eficiencia y aplicación de nuevos materiales para mejorar su rendimiento.

1.2.2 Desarollo

La primera oleada de innovaciones en heliostatos comenzó en la década de 1970, enfocándose en la investigación y desarrollo para perfeccionar los sistemas de seguimiento solar y sus estructuras de soporte. La generación subsiguiente puso su atención en la investigación de nuevos materiales para las superficies reflectantes. La tercera generación trajo avances notables al resolver desafíos clave anteriores, concentrándose en el desarrollo de superficies reflectantes más ligeras y con mayor capacidad reflectiva que pudieran resistir fuertes vientos.

Desde 1975, se intensificó el desarrollo de tecnologías para la concentración solar térmica, evidenciado por un crecimiento en las publicaciones especializadas. Ledesma destacó en su estudio los avances en durabilidad, costo-beneficio y eficiencia de los heliostatos. Griffith examinó las propiedades modales de los heliostatos mediante análisis de elementos finitos (MEF), logrando un sistema con facetas de rotación independiente [8]. Chacón avanzó en el diseño asistido por ordenador (CAD) de heliostatos, incorporando análisis de elementos finitos para el soporte de la superficie reflectante. Kribus se enfocó en el diseño de sistemas de control para mejorar el seguimiento solar. Torres Roldan propuso un heliostato polar simplificado e in-

novador para su integración en edificios y entornos urbanos, ofreciendo una nueva perspectiva en términos de variación temporal [9]. Además, Behar proporcionó una revisión exhaustiva de las plantas de energía solar con receptor central, subrayando el papel crucial de los heliostatos en los sistemas de concentración solar [10].

1.3 Hipótesis

Con base en lo expuesto anteriormente se plantea lo siguiente:"Es posible desarrollar e implementar estrategias de control que afinen el seguimiento y redireccionamiento en los sistemas de concentración termosolar de tipo torre central, además de disminuir la complejidad del mismo, logrando una reducción de los costos de implementación en comparación con sistemas convencionales."

1.4 Objetivos

1.4.1 Objetivo General

Diseñar e implementar un algoritmo de calibración y redireccionamiento en un prototipo autónomo de concentración solar de bajo costo que logre ser competitivo en comparación con los sistemas convencionales.

1.4.2 Objetivos específicos

- Establecer las ecuaciones que regirán el modelo del algoritmo de calibración y redireccionamiento del sistema.
- Definir métrica de evaluación del sistema de redirección.

- Desarrollar el pseudocódigo integral para implementación en heliostato y seguidor solar.
- Realizar la simulación del sistema para definir los parámetros de evaluación y definir la matriz experimental.
- Implementar el algoritmo validado en los prototipos existentes y caracterizar su eficiencia y rendimiento.
- Analizar los resultados obtenidos, comparando los comportamientos del sistema a partir de las simulaciones.
- Concluir el trabajo de investigación resaltando los resultados procesados e indicando las fuentes de oportunidad para posibles trabajos futuros.

1.5 Justificación

En 2015, los estados miembros de la ONU, junto con organismos no gubernamentales (ONGs), establecieron 17 objetivos de desarrollo sostenible (ODS) con una meta de cumplimiento para el año 2030, siendo el objetivo 7 el desarrollo de energía asequible y no contaminante [11]. De acuerdo con la Agencia Internacional de Energía la demanda de calor industrial hasta 2030 aumentará cada año el 1,7 %, cabe mencionar que dicha demanda del sector industrial representa el 32 % del consumo de energía térmica mundial, esto significa que el consumo final de energía térmica en el sector industrial es mayor que el consumo de electricidad a nivel mundial.

Otro dato de importancia a considerar es que de ese 32 % de energía, solo el 9 % proviene de energías renovables. El uso de energía solar para producir calor reduce significativamente la emisión de contaminantes a la atmósfera, además de la reducción de recursos económicos que hacen más rentables a las empresas [12]. Por tal motivo, se han desarrollado diversos proyectos e investigaciones sobre el uso de tecnologías de concentración de energía termosolar que permiten la reducción de

las emisiones y coadyuvan en la atención de la demanda energética. Entre las diversas alternativas estudiadas por la industria e investigadores, se tiene un especial interés por el desarrollo, mejora e implementación de sistemas de concentración de torre central, siendo de gran interés la reducción de costos de implementación, y operación. Actualmente, la mejora continua de estos sistemas ha cobrado más relevancia, el principal motivo es que las tecnologías solares térmicas son adecuadas para suministrar calor a un gran número de procesos como el secado, la ebullición, la esterilización o el blanqueo con necesidades de temperatura de hasta 400°C. Esto es importante si se tiene en cuenta que la industria es uno de los sectores económicos más difíciles de descarbonizar, dados los largos ciclos de inversión en nuevas infraestructuras energéticas [13].

Capítulo 2

Estado actual del conocimiento

En un campo de helióstatos, compuesto por espejos controlados que reflejan y concentran la luz solar en un colector central, se requiere de un sistema de control eficiente. Durante la operación del campo, cada helióstato se alinea individualmente de tal manera que la normal a la superficie biseca el ángulo entre la posición del sol y las coordenadas del punto objetivo en el receptor. Sin embargo, se considera poco realista lograr una alineación precisa de hasta 1 miliradián para todos los helióstatos en relación con los puntos objetivo en el receptor sin un sistema de calibración, debido a diversas fuentes de error de seguimiento. Este desafío resalta la importancia de un sistema de calibración para mejorar la precisión de la redirección y lograr distribuciones de flujo deseadas, así como para reducir o eliminar el desbordamiento, como presentan Sattler y otros, quienes ofrecen una visión general de las fuentes de error de seguimiento y los requisitos básicos de un sistema de calibración ideal [14].

Los errores de seguimiento pueden ocurrir por varias razones. En un estudio exhaustivo realizado por Jones y Stone, se examinaron y detallaron las fuentes de error de seguimiento de helióstatos en la planta Solar Two, incluyendo la flexión debido a la gravedad, el desplazamiento del punto de pivote, la refracción atmosférica, entre otros. Este análisis profundo ayuda a entender los desafíos específicos que deben abordarse para mejorar la precisión en el campo de helióstatos [15].

Adicionalmente, Díaz-Félix y otros evaluaron las distribuciones globales de error de seguimiento en el campo de helióstatos, identificando errores específicos como el desplazamiento angular en la posición de referencia de los mecanismos de seguimiento, el nivelado imperfecto del pedestal del helióstato y la falta de perpendicularidad entre los ejes de seguimiento [16]. Estos hallazgos, junto con las observaciones de Gross y Balz sobre los desacuerdos entre los sistemas de coordenadas utilizados por ingenieros solares, civiles y topográficos, ilustran la complejidad de los errores de seguimiento y subrayan la necesidad de abordar de manera integral estos errores [17].

Además, un estudio detallado realizado por Freeman y otros, enfocado en los errores que afectan el diseño de sistemas de calibración y control, evaluó los impactos de errores individuales. Los errores con un fuerte impacto incluyen aquellos en sensores y actuadores, así como errores de instalación, resaltando la importancia de considerar cuidadosamente estos factores en el diseño de sistemas de calibración para mitigar su impacto y optimizar el rendimiento del sistema de helióstatos [18].

Existen dos enfoques principales: el control en lazo abierto y el control en lazo cerrado, cada uno con ventajas y desventajas específicas.

Control en Lazo Abierto: El control en lazo abierto opera sin retroalimentación, utilizando modelos predefinidos para posicionar los heliostatos. Según Zhang, Li y Zhao, este tipo de control se caracteriza por su simplicidad, lo que lo hace fácil de implementar y reduce tanto costos como esfuerzos de mantenimiento [19]. Además, al no depender de retroalimentación, ofrece una respuesta rápida, ideal para condiciones operativas estables. Wang, Cao y Liu destacan que este enfoque es particularmente efectivo en entornos donde las condiciones ambientales son constantes y predecibles [20]. Por su parte, Smith y Lee analizan cómo la ausencia de retroalimentación en este sistema limita su capacidad para adaptarse a condiciones operativas cambiantes, especialmente en instalaciones grandes donde las pequeñas desviaciones pueden acumularse [21].

Ventajas:

- Simplicidad y bajo costo: Este enfoque es fácil de implementar, reduciendo costos y simplificando el mantenimiento [19].
- Respuesta rápida: Al no depender de retroalimentación, puede reaccionar inmediatamente en tiempo real [20].
- Estabilidad en condiciones ideales: Funciona bien cuando las condiciones ambientales son constantes y predecibles [20].

Desventajas:

- Benítez y Fernández señalan que una de las principales limitaciones del control en lazo abierto es su incapacidad para responder a perturbaciones externas, como ráfagas de viento o nubosidad [22]. González añade que la falta de un mecanismo de corrección automática limita la capacidad del sistema para ajustarse ante desviaciones, lo que afecta su eficiencia [23]. También López menciona que este tipo de sistemas dependen de una calibración inicial precisa, lo que requiere ajustes periódicos para mantener la precisión [24].
- Sensibilidad a perturbaciones: No puede corregir cambios inesperados, como ráfagas de viento o nubes [22].
- Falta de corrección automática: No detecta ni corrige desviaciones en la orientación, disminuyendo la eficiencia [23].
- Dependencia de la calibración inicial: Requiere calibraciones frecuentes para mantener su precisión [24].

Control en Lazo Cerrado: El control en lazo cerrado emplea retroalimentación constante para ajustar dinámicamente la posición de los heliostatos. Según Zhang et al., este enfoque ofrece una orientación precisa gracias a la retroalimentación continua [19]. Wang y sus colaboradores subrayan que, al adaptarse a las

variaciones climáticas y a otras perturbaciones externas, este tipo de control asegura un rendimiento óptimo, incluso en condiciones operativas adversas [20]. Smith y Lee x también destacan que los sistemas en lazo cerrado pueden ofrecer mayor precisión en configuraciones complejas, aunque advierten que requieren una calibración cuidadosa para evitar errores acumulativos [20].

Ventajas:

- Precisión y estabilidad: Permite una orientación precisa gracias a la retroalimentación continua [19].
- Adaptabilidad a cambios: Compensa perturbaciones externas, como variaciones climáticas, asegurando una eficiencia óptima [20].
- Reducción de errores: Detecta y corrige automáticamente las desviaciones en tiempo real [22].

Desventajas:

A pesar de sus ventajas, López indica que los sistemas en lazo cerrado tienden a ser más complejos y costosos, debido a la necesidad de sensores avanzados y algoritmos sofisticados [24]. González advierte que, si no están correctamente calibrados, pueden generar oscilaciones e inestabilidad en la respuesta [23]. Además, señala que la fiabilidad de estos sistemas depende directamente de la precisión de los sensores, lo que representa un punto crítico de fallo potencial.

- Complejidad y costo: Su implementación requiere sensores avanzados y algoritmos sofisticados, lo que incrementa los costos [24].
- Complejidad y costo: Su implementación requiere sensores avanzados y algoritmos sofisticados, lo que incrementa los costos [23].
- Posible inestabilidad: Si no están correctamente calibrados, pueden generar oscilaciones o inestabilidad.

- Dependencia de sensores: Su fiabilidad depende de la precisión de los sensores, que pueden fallar [23].

La elección entre un sistema de control en lazo abierto o cerrado depende de las necesidades específicas del sistema y del entorno operativo. Para proyectos en los que se prioriza la simplicidad y el bajo costo, el control en lazo abierto puede ser suficiente. Sin embargo, para aplicaciones que requieren alta eficiencia y operan en condiciones variables, el control en lazo cerrado resulta ser una mejor opción. La combinación de estrategias, como lo sugieren diversos autores, podría ser una solución híbrida efectiva para aprovechar las ventajas de ambos enfoques [19]-[21].

Una de las estrategias más utilizadas en sistemas de redirección de helióstatos de lazo1 abierto es el uso de algoritmos que calculan y pueden predecir la trayectoria solar durante el día. Carvajal presenta un diseño de helióstato con seguimiento solar para redirigir la radiación incidente hacia un concentrador parabólico, basado en cálculos cenitales y azimutales en tiempo real con intervalos de tiempo cortos [25]. El algoritmo central contiene un número considerable de parámetros iniciales para su ejecución.

2.1 Clasificación

De acuerdo a su sistema de gestión y operación, los helióstatos se dividen en **helióstatos centralizados** y **autónomos**. Los helióstatos centralizados están conectados a un sistema de control central que envía señales a cada actuador de los helióstatos pertenecientes al mismo arreglo de concentración de energía solar térmica (Figura 2.1). Este control puede basarse en el seguimiento del movimiento del sol, lo que requiere un rastreador solar compartido para todos los helióstatos en el grupo. Sin embargo, también es posible emplear un enfoque que se base en cálculos astronómicos para predecir la posición del sol en momentos separados del día, eliminando así la necesidad del rastreador solar.

Figura 2.1: Campo de heliostatos centralizados (Global Nevada Corp 2011).

Por otro lado, los helióstatos autónomos tienen su propio sistema de control independiente, que opera independientemente de otros helióstatos en la misma plataforma, así como del sistema de recolección central 2.2. Al igual que los helióstatos centralizados, los autónomos pueden usar un sistema de seguimiento solar o basarse en cálculos astronómicos para determinar la posición relativa del Sol y la Tierra.

Figura 2.2: Heliostato autónomo (Pfahl 2013).

En el primer caso, cada helióstato autónomo requiere su propio rastreador solar, junto con un mecanismo de sincronización (ya sea mecánico o electrónico) para controlar los actuadores. El concepto de autonomía aplicado a los helióstatos implica incorporar una unidad de control local que modifica varios aspectos de su operación, cálculos, seguridad y suministro de energía, entre otros. Esto asegura que los helióstatos reflejen consistentemente la radiación solar directa hacia un objetivo

fijo, logrando el seguimiento solar sin depender de dispositivos externos [26]. Un helióstato autónomo puede realizar las siguientes funciones por sí mismo:

- Realizar posicionamiento, funciones de guía de ejes y cálculos y determinar el posicionamiento solar y el control de enfoque, eliminando la necesidad de asistencia de control central (cálculos autónomos).
- Asegurar la toma de decisiones sobre su propia seguridad, protección e integridad, gracias al conocimiento que adquiere de las condiciones meteorológicas externas o malfuncionamientos internos (autonomía de seguridad).
- Lograr ciclos operativos gracias al control local equipado. Los ciclos pueden ser identificados de forma remota o declarados con antelación [27].

Por estas razones, se exige alta precisión durante el seguimiento solar, de lo contrario, surgirán desbordamientos. Se refieren a una disparidad presentada entre la orientación teórica y real y el eje óptico del helióstato a lo largo de un año. Este tipo de desviación proviene de una alineación incorrecta de los activadores o un fallo en el cálculo de la posición solar. Las desviaciones pueden ser abordadas individualmente para cuantificarlas y corregirlas, logrando esto al obtener discrepancias para diferentes ángulos de incidencia solar. Esto, a su vez, se logra dirigiendo el enfoque hacia una cámara que calcula y evalúa la desviación del punto focal dado [28].

2.2 Fundamento del estudio solar

Es crucial entender previamente cómo se comporta el sol para maximizar el uso de la energía solar que nos llega. Esto implica identificar específicamente la zona de estudio para analizar el movimiento del sol a través del cielo y sus variaciones durante las diferentes temporadas del año. Los elementos que pueden disminuir la fuerza de la radiación solar incluyen factores tanto atmosféricos como geográficos. Entre los

atmosféricos, se incluyen las nubes, partículas en suspensión y la contaminación en general. En cuanto a los geográficos, afectan la rotación diaria de la Tierra, así como su latitud, longitud y su órbita alrededor del Sol. Considerando el ciclo de 365 días que toma la Tierra para orbitar alrededor del Sol en el plano eclíptico, se observa que la posición aparente del Sol en el cielo cambia durante el año, un fenómeno resultado de lo que se conoce como declinación solar. La declinación solar se refiere al ángulo aproximado y variable entre el plano del círculo máximo que traza el Sol sobre la esfera celeste a lo largo de un año. y el plano perpendicular a este, conocido también como plano ecuatorial. (Figura 2.3).

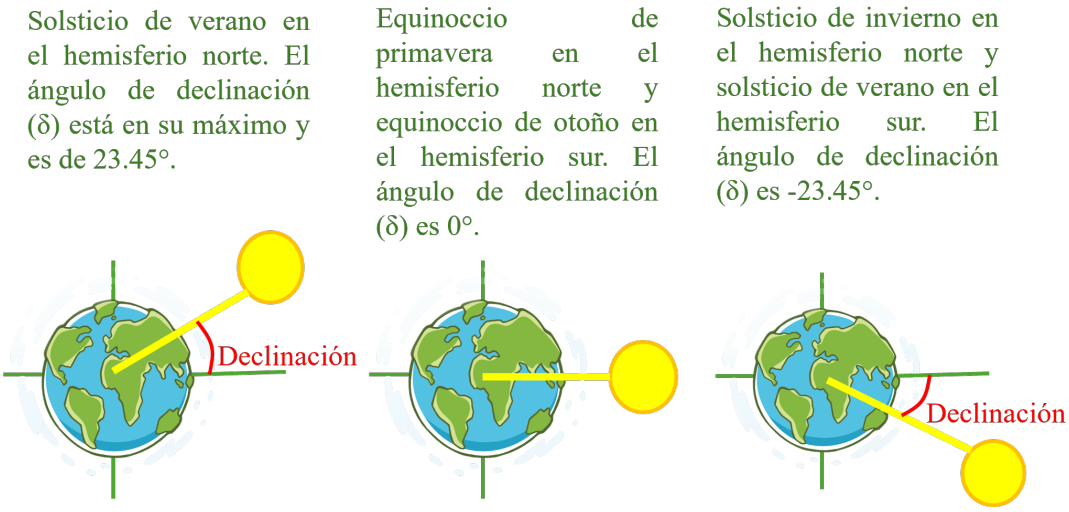


Figura 2.3: Declinación de la tierra.

Aunque se utiliza una declinación solar promedio para fines prácticos, es importante mencionar que este ángulo varía entre -23.45º y 23.45º a lo largo del año. Estos valores extremos ocurren durante los solsticios de verano (22 de junio) y de invierno (22 de diciembre), conocidos también como el afelio y perihelio, respectivamente. En los equinoccios de primavera (21 de marzo) y de otoño (23 de septiembre), la declinación solar es de 0 grados, como se muestra en la Figura 2.4.

En el ámbito de la geometría solar, se emplean tres ángulos fundamentales para describir la posición del sol en relación con un observador terrestre: el **azimuth** solar, la **altura o elevación solar** y el **cenit** (Figura 2.5). Según Duffie y Beckman [29], el

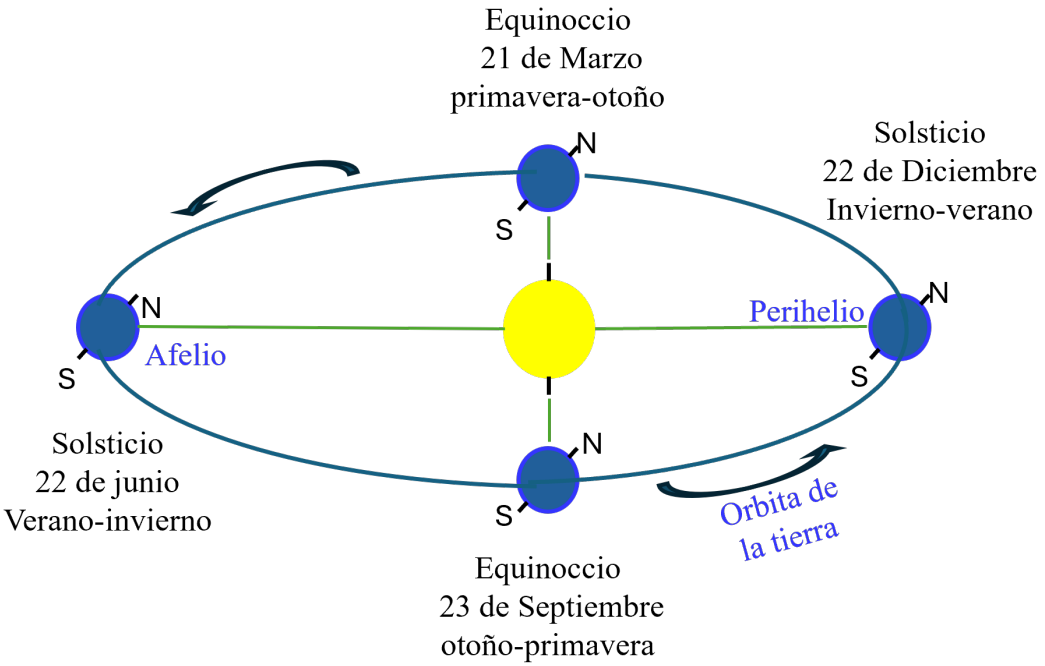


Figura 2.4: Planos de sección de la tierra.

azimuth solar se define como el ángulo medido en el plano horizontal desde el norte verdadero, en sentido horario, hasta la proyección del vector solar sobre dicho plano. Este parámetro es crucial para determinar la orientación óptima de los sistemas de captación solar.

La altura solar es el ángulo entre el sol y el horizonte, medido en un plano perpendicular al horizonte, y representa qué tan elevado está el sol en el cielo. Este valor varía a lo largo del día debido a la rotación de la Tierra y depende de la declinación solar y de la posición geográfica del observador.

Por último, el cenit se refiere al punto directamente sobre el observador en la esfera celeste, y el ángulo de cenit se define como el complemento de la altura solar, es decir, la inclinación del vector solar respecto a la vertical local. Estos conceptos geométricos son fundamentales para calcular la radiación solar incidente en superficies inclinadas y para optimizar el diseño de sistemas solares térmicos.

2.2.1 Trayectoria solar en área metropolitana de Monterrey Nuevo León, México

Para una estimación aproximada de la posición del Sol con respecto a una ubicación específica, se recurre a la carta solar estereográfica de Fisher-Mattioni, más comúnmente conocida como carta solar. Esta es una representación gráfica que, de acuerdo con una latitud determinada y eligiendo una hora y fecha específicas, permite obtener el ángulo de elevación solar y el ángulo de azimuth.

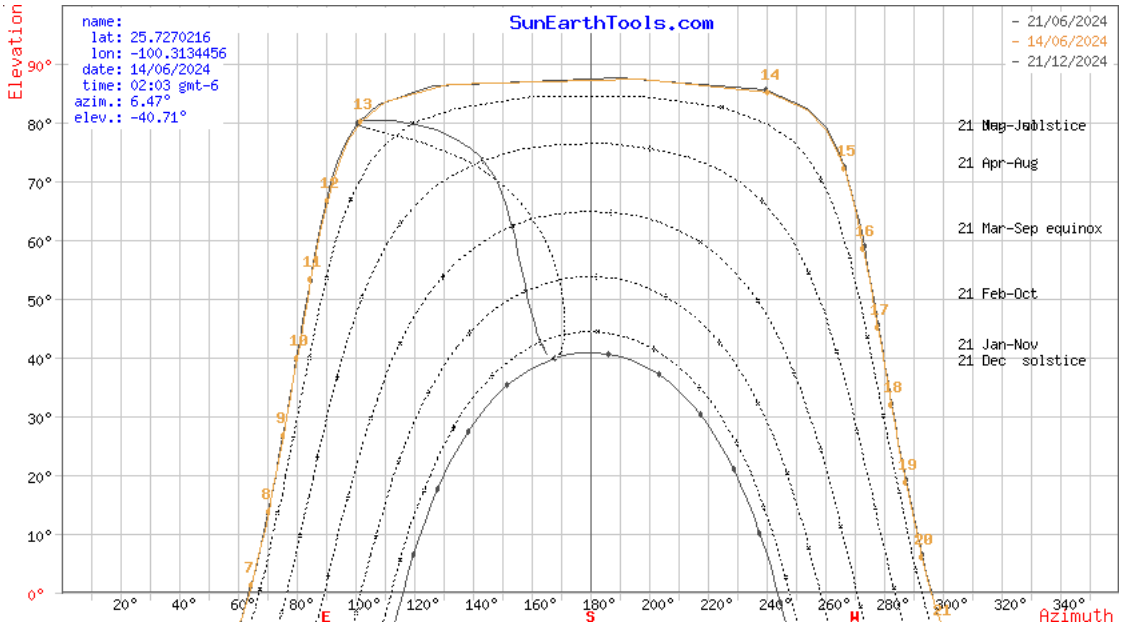


Figura 2.5: Carta Solar de Ciudad Universitaria UANL (Azimut vs Elevación) obtenido de SunEarthTools.com.

2.3 Elementos generales de un heliostato

Cuando hablamos de un heliostato, nos referimos al conjunto de espejos o espejo modular que reflejan la radiación solar hacia una planta termosolar. Este sistema se compone de varios elementos (Figura 2.6) esenciales:

Pedestal: Estructura que soporta el helióstato, asegurando su fijación al suelo

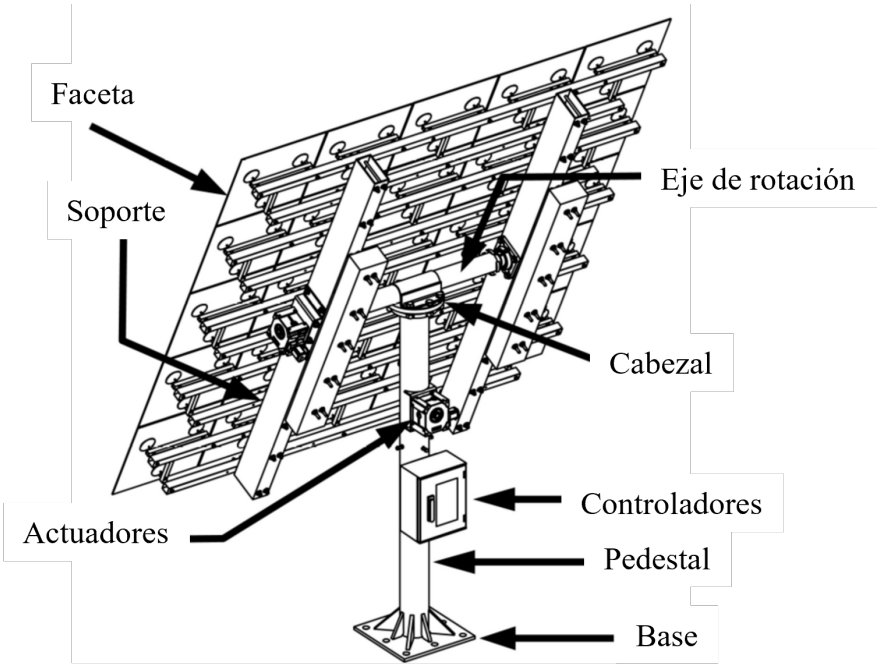


Figura 2.6: Elementos principales de un heliostato (Cock 2018).

y su estabilidad frente a condiciones meteorológicas adversas. El tipo más común es el pedestal tubular [28].

Cabezal: Sistema que permite la rotación y/o inclinación de la superficie reflectante, facilitando el ajuste del ángulo de reflexión.

Estructura de soporte o bastidor: Encargada de mantener las facetas reflectantes en su lugar con precisión, soportando el peso y las cargas de viento. Suelen usar materiales resistentes como aluminio, acero al carbono, plata, entre otros [30].

Superficie reflectante: Parte crucial del helióstato, su eficiencia depende del índice de reflexión del material utilizado. Está compuesta por módulos independientes llamados facetas, generalmente hechos de vidrio con una capa de metal reflectante como aluminio o plata aunque ultimamente se observan tendencias en la aplicación de otros tipos de recubrimiento.

Facetas: En los heliostatos, la superficie reflectante actúa como el componente óptico que refleja la luz solar [28]. Estos elementos son los responsables de concentrar

la radiación solar en un área lo más pequeña posible, especialmente en las plantas de torre central. Sin embargo, los hornos solares que emplean heliostatos para mantener un haz de luz unidireccional utilizan facetas planas y paralelas.

Se han reportado diversos materiales para los módulos de espejos. Entre ellos se incluye el vidrio, que es una opción popular debido a su bajo costo, gran durabilidad, alta aceptación en la industria y su reflectancia inherente. Otras opciones consideradas han sido espejos de metal pulido y de plástico, aunque son menos comunes debido a su reflectancia inadecuada para esta aplicación [28]. Para las facetas de vidrio, se busca una composición que optimice la transmitancia de la radiación solar en la primera superficie, y se utilizan depósitos de metales como aluminio y plata sobre la placa de vidrio para la película reflectora. Cuando el espejo tiene dos capas de vidrio con el material reflectante en medio, se les llama espejos laminados.

Las facetas de espejos laminados proporcionan un alto rendimiento óptico, gran fuerza y rigidez. Además, con un buen diseño de esta estructura, se puede lograr una reducción en los costos totales de la estructura de soporte [28].

Actuadores: Son los dispositivos que proporcionan movimiento al heliostato. Se instala un actuador por cada grado de libertad; estos pueden ser de rotación o desplazamiento lineal, según el diseño. Además, pueden estar adaptados con un sistema de reducción de velocidad para ofrecer mayor precisión y fuerza al posicionar la superficie reflectante, asegurando que la imagen del Sol se mantenga reflejada en el receptor. También están diseñados para soportar las cargas del viento a través del pedestal [31].

Algunos de los componentes más comunes que proporcionan movimiento a los heliostatos son: motores con reducción mecánica, actuadores lineales y sistemas de accionamiento hidráulico.

El sistema azimutal montado se considera uno de los principales. Por otro lado, el sistema de actuadores lineales, además de ser más económico en comparación con otros, se considera el sustituto ideal del motor del eje azimutal. Sistemas como los

actuadores hidráulicos se utilizan más en heliostatos grandes debido a su elevado costo. Por último, los sistemas alternativos son un ejemplo de métodos que evitan los costos de montaje en el pedestal [28].

Sistema de control: Un sistema de control automático es crucial en una central solar termoeléctrica, ya que gobierna los actuadores de cada heliostato en el conjunto. Este sistema puede ser centralizado o independiente para cada heliostato. Además, existen controles para heliostatos tanto de lazo abierto como de lazo cerrado.

2.4 Generalidades de operación

2.4.1 Pérdidas

Los principales factores que causan pérdidas ópticas son la precisión del seguimiento, la gravedad y el viento. Sin embargo, se ha analizado que el impacto del viento en las pérdidas ópticas es bajo. Se observa que la cantidad de heliostatos afectados por el viento es limitada, afectando principalmente a los situados en el perímetro del campo de heliostatos. Además, el viento rara vez alcanza sus velocidades máximas y las cargas de viento son mayores solo en ciertos ángulos de ataque [28]. No hay pérdidas significativas en el receptor debido al viento, ya que el punto focal solo oscila alrededor de su eje en una posición media, permitiendo que la mayor parte de la energía solar incidente se mantenga efectiva.

2.4.2 Errores y sus causas

Los errores de seguimiento en los heliostatos pueden ocurrir por diversas razones, y hay numerosas publicaciones que describen estas fuentes de error. Según Mancini, a medida que los errores se acumulan, pueden causar un desplazamiento de

la imagen del heliostato, alejándose del punto de mira previsto con el tiempo [31]. Un estudio exhaustivo realizado por Jones y Stone (1999) detalló las fuentes de error de seguimiento en la planta Solar Two [15].

Más recientemente, Díaz-Félix y otros (2014) evaluaron las distribuciones globales de error de seguimiento en el campo de heliostatos. Entre los errores mencionados en estos estudios se incluyen la flexión debido a la gravedad, el desplazamiento del punto de pivote, la refracción atmosférica, el desplazamiento angular en la posición de referencia de los mecanismos de seguimiento, el nivelado imperfecto del pedestal del heliostato y la falta de perpendicularidad entre los ejes de seguimiento [16]. Gross y Balz (2020) señalaron que los errores de seguimiento también pueden deberse a desacuerdos entre los sistemas de coordenadas utilizados por ingenieros solares, civiles y topográficos, así como a errores de programación durante el proceso topográfico de un sitio CSP (Concentrating Solar Power) [17].

Un estudio detallado realizado por Freeman y otros (2014) evaluó los impactos de errores individuales en el diseño de sistemas de calibración y control. Este estudio encontró que los errores con mayor impacto se deben a errores en los sensores y actuadores (como la elección de la relación de engranajes, el retroceso del heliostato y la baja resolución del codificador), así como a errores de instalación (inclinación del pedestal, ángulo de referencia de azimut y elevación, y terreno no nivelado) [18].

Considerando el error de seguimiento global (es decir, la suma de los errores individuales de los heliostatos) y las tolerancias de seguimiento marginal, queda claro que el desbordamiento y la orientación inexacta del heliostato son problemas principales para las plantas de receptor central (CRS). Según Jones y Stone, la energía captada por el receptor de la planta Solar Two fue entre un 10 y un 20 % menor de lo anticipado, y se sospechó que esto se debió al error de seguimiento. Esta sospecha fue confirmada por observaciones de heliostatos ocasionalmente desalineados y un flujo de calor excesivo en el protector térmico directamente arriba y abajo del receptor [32].

La deriva en los campos de heliostatos se refiere a la desviación no intencionada de los espejos de sus posiciones óptimas, lo que resulta en una disminución de la concentración solar en el receptor. Esta desviación puede deberse a diversos factores, siendo los errores geométricos uno de los más críticos. Estos errores pueden originarse en la imprecisión en la fabricación de componentes, deformaciones estructurales causadas por cargas mecánicas y térmicas, y fallos en la alineación y calibración de los espejos.

Estudios recientes han demostrado que incluso pequeños errores en la geometría de los heliostatos pueden impactar significativamente en la eficiencia del sistema. Las desviaciones geométricas pueden llevar a una pérdida de hasta el 20 % en la eficiencia de concentración solar, lo cual resalta la necesidad de abordar estos errores con precisión y rigor [18]. Sattler et al. destacaron que la acumulación de errores menores a lo largo del tiempo puede provocar una deriva considerable, afectando la capacidad del sistema para mantener un enfoque preciso en el receptor solar[14].

Para mitigar estos problemas, se han desarrollado varias estrategias de corrección y compensación. Técnicas avanzadas de control y calibración, como el uso de sensores de alta precisión y algoritmos de ajuste dinámico, han demostrado ser efectivas para reducir los efectos de la deriva. Sin embargo, la implementación de estas soluciones requiere una comprensión profunda de los mecanismos subyacentes de los errores geométricos y su impacto en la operación del sistema [33].

2.5 Aplicaciones

2.5.1 Plantas solares de torre central

Una planta solar de torre central consiste en un receptor ubicado en la parte superior de una torre, el cual está rodeado por un conjunto de heliostatos. Estos

heliostatos son responsables de reflejar los rayos solares hacia el receptor [34]. Es crucial mantener una alta precisión en el seguimiento solar para maximizar el aprovechamiento de la energía solar incidente y reducir las pérdidas del haz reflejado [9].

La producción de energía, como se muestra en la Figura 2.7, se realiza cuando el receptor (2) recibe la radiación solar reflejada por los heliostatos y la convierte en energía térmica, elevando un fluido de trabajo a temperaturas entre 300 y 700 ºC. En las plantas generadoras de electricidad, este fluido de trabajo transfiere el calor al vapor, que luego se utiliza para generar electricidad mediante turbinas [4]. El fluido de trabajo también se emplea para almacenar energía térmica en tanques, permitiendo su uso posterior.

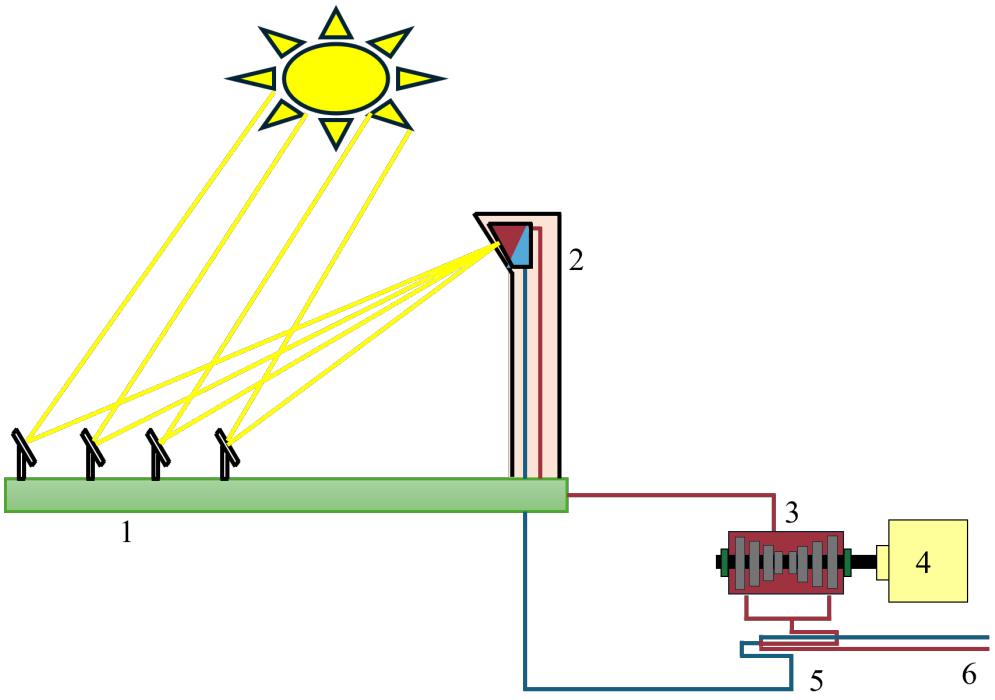


Figura 2.7: Diagrama de proceso en una planta solar de torre central. 1. Heliostato, 2. Receptor en Torre Central, 3. Turbina, 4.Generador, 5.Intercambiador de calor, 6. Entrada y salida de agua del tanque de almacenamiento.

2.5.2 Hornos solares

Un horno solar se considera un sistema óptico que concentra la luz solar sobre sus superficies para calentar cuerpos a altas temperaturas. Este sistema se basa en una doble reflexión de la radiación solar en el punto focal.

Para una central de horno solar, se requiere un heliostato o un conjunto de ellos que reflejen los rayos solares hacia un disco parabólico que concentra los rayos en el foco. Con el uso de un concentrador parabólico estacionario, se logran concentraciones de miles de soles debido a la pequeña área de concentración (Figura 2.8).

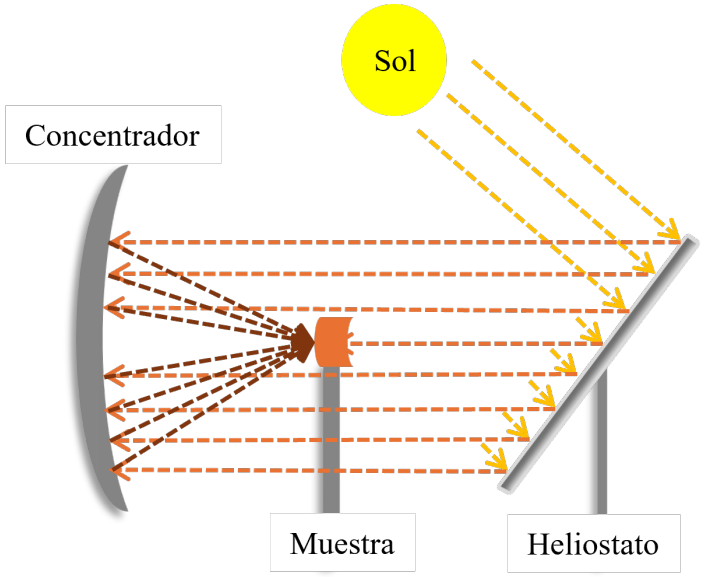


Figura 2.8: Diagrama general de un horno solar.

Una central de horno solar se compone principalmente de un atenuador, que consiste en un conjunto de láminas horizontales que regulan la cantidad de luz orientándolas en menor o mayor medida; un concentrador, que concentra los rayos provenientes de los heliostatos con una forma paraboloide en conjunto con espejos; y el foco receptor, posiblemente el elemento más importante, ya que en él incide toda la radiación concentrada, la cual depende en gran medida de la longitud focal [30].

2.5.3 Calor de proceso

Otra aplicación de este tipo de torres es el uso de la energía térmica obtenida en procesos termoquímicos, como la producción de combustibles sintéticos, la fabricación de cemento y la recuperación de petróleo, actividades industriales que requieren altas temperaturas de trabajo [35].

Una de las ventajas de estos sistemas solares de torre central es la alta concentración solar, logrando temperaturas de hasta 1000 ^{o}C. Esta disposición de los heliostatos permite un buen volumen de almacenamiento energético y una alta eficiencia en la producción de energía eléctrica mediante turbinas de vapor [35]. Esto ha permitido desarrollar plantas solares de torre central que proporcionan entre 50 y 300 MW de potencia eléctrica [36].

Actualmente, la mejora continua de estos sistemas ha cobrado más relevancia, el principal motivo es que las tecnologías solares térmicas son adecuadas para suministrar calor a un gran número de procesos como el secado, la ebullición, la esterilización o el blanqueo con necesidades de temperatura de hasta 400°C.

Esto es importante si se tiene en cuenta que la industria es uno de los sectores económicos más difíciles de descarbonizar, dados los largos ciclos de inversión en nuevas infraestructuras energéticas.

2.6 Costos

El diseño e instalación de un campo de heliostatos representa una de las inversiones más significativas en las plantas solares de torre central. Este costo elevado se debe principalmente a la complejidad de los estudios previos, el diseño del campo y los equipos necesarios. Además, los costos de operación, mantenimiento y análisis del ciclo de vida también influyen en la viabilidad económica de estos sistemas.

2.6.1 Implementación

La implementación de un campo de heliostatos implica actividades técnicas y logísticas complejas que abarcan desde estudios iniciales hasta la instalación y puesta en marcha del sistema. Entre los aspectos más costosos se encuentran el diseño y disposición del campo, los estudios topográficos y de nivelación del terreno, y la fabricación de los heliostatos. Según Behar et al. [10], la etapa de implementación puede representar hasta el 40 % del costo total de una planta solar de torre central.

El análisis del arreglo óptimo del campo, conocido como layout, es crucial para maximizar la captación solar y minimizar las sombras y bloqueos entre heliostatos. Estudios como el de Paul y Kedare [6] destacan que esta fase requiere modelos computacionales avanzados y simulaciones intensivas, lo que incrementa los costos de diseño.

Los estudios topográficos y de nivelación del terreno también tienen un impacto significativo en los costos. Según Sepúlveda [35], las características del terreno pueden requerir movimientos de tierra extensivos o estructuras de soporte personalizadas, lo que eleva el gasto total. Este es un aspecto donde los avances en algoritmos de control y optimización podrían ofrecer soluciones más económicas.

2.6.2 Operación

Los costos de operación incluyen el consumo energético de los actuadores, los sistemas de control y el monitoreo continuo del campo solar. Zhang et al. identifican que los sistemas de control basados en bucles cerrados son más eficientes, lo que ayuda a reducir el gasto energético [19]. Sin embargo, estos sistemas requieren hardware y software avanzados, lo que puede elevar los costos iniciales.

El monitoreo y ajuste continuo de los heliostatos es fundamental para mantener su alineación con el receptor. Métodos modernos, como los sistemas autónomos des-

critos por López [24], permiten un control más preciso y reducen los costos asociados al consumo energético y al personal técnico.

2.6.3 Mantenimiento

El mantenimiento preventivo y correctivo es esencial para garantizar el funcionamiento continuo de los heliostatos. Según Gross y Balz [17], los costos de mantenimiento representan aproximadamente el 15-20 % del costo anual de operación de un campo de heliostatos. Este gasto incluye inspecciones regulares, calibración de espejos y reparación de componentes dañados por condiciones climáticas.

En la actualidad, el uso de tecnologías como drones para la calibración y evaluación del estado de los heliostatos, como se menciona en Lombard y Smit [33], ha reducido significativamente los costos y tiempos de mantenimiento, mejorando la eficiencia general del sistema.

2.6.4 Análisis de ciclo de vida

El análisis de ciclo de vida (LCA por sus siglas en inglés Life Cycle Assessment) permite evaluar los costos y beneficios económicos y ambientales de un campo de heliostatos desde su fabricación hasta su disposición final. Sepúlveda [35] señala que la inversión inicial es alta debido a los materiales y procesos de fabricación, pero los beneficios energéticos a largo plazo compensan estos costos.

Además, el LCA destaca que las mejoras en el diseño y la implementación, como la reducción de materiales y la simplificación de los sistemas de control, pueden disminuir la huella de carbono y los costos operativos en un 20 % en comparación con diseños tradicionales [22].

2.7 Evaluación del desempeño de un campo de heliostatos

La evaluación del desempeño de un campo de heliostatos es fundamental para determinar su viabilidad económica, eficiencia técnica y sostenibilidad ambiental. Existen diversas herramientas y métricas utilizadas para analizar y comparar el funcionamiento de estos sistemas en escenarios específicos. Entre estas, destacan el Costo Nivelado de Energía (LCOE), la Fracción Solar, y las métricas técnicas como la eficiencia óptica, la tasa de interceptación y la uniformidad del flujo. A continuación, se presentan estas herramientas, acompañadas de las ecuaciones necesarias para su cálculo.

2.7.1 Costo Nivelado de Energía (LCOE)

El *Costo Nivelado de Energía* (LCOE, por sus siglas en inglés Level Cost Of Energy) se define como el costo promedio de generación de energía a lo largo de la vida útil de una instalación. Se calcula mediante la siguiente ecuación:

$$\text{LCOE} = \frac{\sum_{t=1}^{n} \frac{C_t}{(1+r)^t}}{\sum_{t=1}^{n} \frac{E_t}{(1+r)^t}} \tag{2.1}$$

Donde:

- C_t: Costos totales en el año t, incluyendo operación, mantenimiento, y reemplazo.
- E_t: Energía generada en el año t.
- r: Tasa de descuento.
- n: Vida útil del sistema.

Este indicador permite comparar el costo de generación de diferentes tecnologías, siendo menor el valor más competitivo económicamente [37].

2.7.2 Fracción Solar

La *Fracción Solar* mide el aporte relativo de la energía solar al sistema energético total y se define como:

$$\mathrm{FS} = \frac{Q_{\mathrm{solar}}}{Q_{\mathrm{total}}} \tag{2.2}$$

Donde:

- Q_{solar}: Energía térmica aportada por el campo de heliostatos.
- Q_{total}: Demanda total de energía térmica del sistema.

Este indicador es especialmente útil para evaluar el grado de impacto de la energía solar en instalaciones híbridas o sistemas industriales [38].

2.7.3 Métricas de Evaluación de Efectividad

Para una evaluación integral del desempeño de un campo de heliostatos, es esencial considerar diversas métricas técnicas que reflejan la eficiencia y eficacia del sistema.

2.7.3.1 Eficiencia Óptica

La *Eficiencia Óptica* mide la proporción de la radiación solar incidente que es efectivamente reflejada y concentrada hacia el receptor central. Esta métrica con-

sidera pérdidas debidas a factores como el sombreado entre heliostatos, el bloqueo mutuo y la reflectividad de los espejos. Una alta eficiencia óptica es indicativa de un diseño y operación óptimos del campo solar [37]. La eficiencia óptica ($\eta_{\text{óptica}}$) se expresa como:

$$\eta_{\text{óptica}} = \eta_{\text{sombra}} \cdot \eta_{\text{bloqueo}} \cdot \eta_{\text{reflejo}} \cdot \eta_{\text{atmosférica}} \tag{2.3}$$

Donde:

- η_{sombra}: Factor de pérdida por sombreado entre heliostatos.
- η_{bloqueo}: Pérdidas debidas al bloqueo de rayos solares.
- η_{reflejo}: Reflectividad de los espejos.
- $\eta_{\text{atmosférica}}$: Pérdidas por dispersión atmosférica.

2.7.3.2 Tasa de Interceptación

se refiere al porcentaje de la energía solar reflejada por los heliostatos que es capturada por el receptor. Esta tasa depende de la precisión en el seguimiento solar y la alineación de los heliostatos. Una tasa de interceptación elevada asegura que la mayor parte de la energía reflejada sea aprovechada, minimizando pérdidas. La tasa de interceptación (I_f) se calcula como:

$$I_f = \frac{E_{\text{receptor}}}{E_{\text{reflejada}}} \tag{2.4}$$

Donde:

- E_{receptor}: Energía solar recibida por el receptor.
- $E_{\text{reflejada}}$: Energía solar reflejada por los heliostatos.

2.7.3.3 Uniformidad del Flujo

La *Uniformidad del Flujo* en el receptor es esencial para garantizar una distribución homogénea de la energía térmica. Desbalances en esta uniformidad pueden provocar puntos calientes, generando tensiones térmicas que afectan la integridad del receptor y reducen la eficiencia global del sistema. Estrategias de apuntamiento y diseño del campo de heliostatos son fundamentales para mantener esta uniformidad [38]. La uniformidad del flujo (U_f) se define por:

$$U_f = 1 - \frac{\sum_{i=1}^{N} |q_i - q_{\text{prom}}|}{2 \cdot q_{\text{prom}} \cdot N} \tag{2.5}$$

Donde:

- q_i: Flujo térmico en el receptor en la posición i.
- q_{prom}: Flujo térmico promedio en el receptor.
- N: Número total de posiciones medidas en el receptor.

2.7.4 Impacto Ambiental y Huella de Carbono

Además de las métricas técnicas y económicas, es vital evaluar el *Impacto Ambiental* y la *Huella de Carbono* asociada al campo de heliostatos. Aunque la energía solar es una fuente limpia, la fabricación, instalación y mantenimiento de los heliostatos conllevan emisiones y alteraciones al entorno. Evaluaciones de ciclo de vida permiten cuantificar estas emisiones y compararlas con las de otras tecnologías, proporcionando una visión completa de la sostenibilidad del sistema [39].

Capítulo 3

Marco Teórico

3.1 Síntesis de la revisión bibliográfica

Durante la operación del campo, cada heliostato se alinea individualmente de tal manera que la normal de la superficie en general biseca el ángulo entre la posición del sol y las coordenadas del punto de mira en el receptor. Debido a diversas fuentes de error de seguimiento, lograr una alineación precisa de a lo más 1 miliradián para todos los heliostatos con respecto a los puntos de mira en el receptor sin un sistema de calibración puede considerarse irrealista. Por lo tanto, un sistema de calibración es necesario no solo para mejorar la precisión del redireccionamiento para lograr distribuciones de flujo deseadas, sino también para reducir o eliminar el desbordamiento. Sattler y otros presentan una visión general de las fuentes de error de seguimiento y los requisitos básicos de un sistema de calibración ideal. [5] Un error de seguimiento puede ocurrir debido a varias razones, y existen diversas publicaciones que describen las fuentes de error. A medida que los errores se acumulan, pueden dar lugar al desplazamiento de la imagen del heliostato, que es una desviación del seguimiento ideal, donde la imagen se aleja del punto de mira previsto con el tiempo, según lo descrito por Mancini [6]. En un estudio exhaustivo realizado por Jones y Stone, se examinaron y describieron en detalle las fuentes de error de seguimiento del heliostato en la planta Solar Two. [7] Más recientemente, Díaz-Félix y otros evaluaron las

distribuciones globales de error de seguimiento en el campo de heliostatos. Algunos de los errores mencionados en estas dos publicaciones incluyen la flexión debida a la gravedad, el desplazamiento del punto de pivote, la refracción atmosférica, el desplazamiento angular en la posición de referencia de los mecanismos de seguimiento, el nivelado imperfecto del pedestal del heliostato y la falta de perpendicularidad entre los ejes de seguimiento, entre otros errores [8]. Gross y Balz señalan que los errores de seguimiento también se deben a desacuerdos entre los sistemas de coordenadas utilizados por ingenieros solares, civiles y topográficos, por ejemplo, durante el proceso topográfico de un sitio CSP (Cocentrating Solar Power) y debido a errores de programación [9]. Un estudio detallado y completo realizado por Freeman y otros, cuyo enfoque fue estudiar los errores que afectan el diseño de sistemas de calibración y control, evaluó los impactos de errores individuales. Según este estudio, los errores con un fuerte impacto se deben a errores en los sensores y actuadores (elección de la relación de engranajes, retroceso del heliostato, baja resolución del codificador), así como a errores de instalación (inclinación del pedestal, ángulo de referencia de azimut y elevación, terreno no nivelado) [10]. Cuando se considera el error de seguimiento global (es decir, la suma de los errores individuales de los heliostatos) y las tolerancias de seguimiento marginal mencionadas anteriormente, queda claro que el desbordamiento y la orientación inexacta del heliostato son dos problemas principales para las plantas de receptor central (CRS). Se debería apuntar a una precisión de seguimiento del heliostato de no más de mrad en general. Según Jones y Stone, la energía captada por el receptor de la planta Solar Two estuvo en el rango del 10 al 20 % menor de lo anticipado. En una evaluación, se sospechó que el rendimiento del heliostato se redujo debido al error de seguimiento. La sospecha fue confirmada por observaciones de heliostatos ocasionalmente desalineados y un flujo de calor excesivo en el protector térmico directamente arriba y abajo del receptor [11].

3.2 Metodología general de la tesis

1. **Revisión literaria:**

   Se realizará una revisión constante del estado del arte y la literatura disponible sobre el tema.

2. **Diseño y desarrollo:**

   Se llevará a cabo una cuidadosa definición de métricas de evaluación, centrándose en aspectos críticos como la eficiencia de concentración y la precisión del seguimiento solar, con el objetivo de medir exhaustivamente el rendimiento del sistema. Posteriormente, se desarrollará un pseudocódigo integral para la implementación práctica en el heliostato y el seguidor solar, abordando funciones clave desde la captación de radiación solar hasta la correcta orientación de los dispositivos de concentración.

3. **Simulación:**

   Consistirá en realizar simulaciones detalladas del sistema, con herramientas de software especializado, para refinar los parámetros operativos, superar desafíos identificados y obtener información crucial. Además, se obtendrán datos cuantificables como energía captada, esta información será esencial para definir con precisión los parámetros de evaluación frente a otros diseños reportados en la literatura.

4. **Validación mediante una maqueta experimental:**

   Se realizarán diversas pruebas experimentales variando ciertas configuraciones del sistema.

5. **Analizar los resultados:**

   Se analizarán los resultados obtenidos y compararán frente a otros sistemas reportados.

6. **Discusión de los resultados:**

   Se comparará el comportamiento del prototipo implementado bajo las diferen-

tes configuraciones de funcionamiento con relación a la simulación. Así mismo se realizará un análisis y estimaciones económicas con base en lo reportado por la simulación contra y lo encontrado en la literatura.

7. **Conclusiones:**
   Se resaltarán los resultados más relevantes para el cumplimiento de los objetivos de la investigación.

3.3 Software de simulación

En el diseño y análisis de campos de heliostatos, es fundamental emplear herramientas de simulación que permitan modelar y evaluar el comportamiento óptico y de control del sistema. Entre las herramientas más destacadas se encuentran *MATLAB* y *TracePro*, cada una con funcionalidades específicas que aportan al desarrollo integral del proyecto.

3.3.1 MATLAB

MATLAB es un entorno de programación y cálculo numérico ampliamente utilizado en ingeniería y ciencias aplicadas. Su versatilidad y capacidad para manejar matrices, funciones y datos lo convierten en una herramienta esencial para el desarrollo de algoritmos de control y simulaciones dinámicas en sistemas complejos.

En el contexto de los campos de heliostatos, *MATLAB* se puede utilizar para:

- Desarrollar algoritmos de seguimiento solar que optimizan la orientación de los heliostatos.
- Simular el comportamiento dinámico del sistema de control bajo diversas condiciones ambientales.

- Analizar datos y visualizar resultados mediante gráficas y representaciones tridimensionales.

La capacidad de *MATLAB* para integrarse con otros entornos y su amplia gama de toolboxes lo hacen especialmente útil para abordar problemas multidisciplinarios en el diseño de sistemas solares [40].

3.3.2 TracePro

TracePro es un software especializado en el diseño y análisis de sistemas ópticos e iluminación. Utiliza técnicas de trazado de rayos basadas en el método de Monte Carlo para simular la propagación de la luz a través de componentes ópticos complejos, permitiendo evaluar la eficiencia y el rendimiento de sistemas como los campos de heliostatos [41].

Las principales aplicaciones de *TracePro* en este ámbito incluyen:

- Modelado detallado de la geometría del campo de heliostatos y sus interacciones ópticas.
- Simulación de la distribución de la luz reflejada hacia el receptor central.
- Análisis de pérdidas ópticas debidas a factores como sombreado, dispersión y absorción.

La utilización del método de Monte Carlo en *TracePro* permite una simulación precisa de fenómenos ópticos complejos, proporcionando información detallada sobre el comportamiento de la luz en el sistema.

3.3.3 Integración de MATLAB y TracePro

La combinación de *MATLAB* y *TracePro* ofrece una plataforma robusta para el diseño y análisis de campos de heliostatos. Mientras que *MATLAB* facilita el desarrollo de algoritmos de control y el análisis de datos, *TracePro* proporciona una comprensión profunda del comportamiento óptico del sistema. Esta integración permite optimizar tanto el diseño geométrico como el rendimiento operativo del campo de heliostatos.

Capítulo 4

Metodología

En este capítulo se describe la metodología seguida para el diseño, implementación y validación del algoritmo propuesto para el control de heliostatos autónomos. La metodología se estructura en tres fases principales: diseño matemático del algoritmo, implementación en un entorno computacional y validación experimental mediante simulaciones y pruebas con un prototipo físico.

El algoritmo propuesto elimina la necesidad de datos tradicionales como tiempo, fecha, latitud y longitud, utilizando en su lugar una calibración inicial basada en la dirección de reflexión de la luz solar hacia un objetivo. Este enfoque reduce la complejidad del sistema y potencia su aplicabilidad en entornos donde los recursos computacionales y de datos son limitados.

A lo largo del capítulo se detalla el desarrollo del modelo matemático que sustenta el algoritmo, las herramientas computacionales utilizadas para su implementación, y los pasos seguidos para la construcción y configuración del prototipo heliostato y otros dispositivos auxiliares. Finalmente, se presentan las simulaciones realizadas para analizar el comportamiento del sistema bajo diversas condiciones y se discuten las observaciones obtenidas respecto a la precisión del apuntamiento y la deriva del sistema.

Este enfoque metodológico integral tiene como objetivo garantizar la validez de

los resultados obtenidos y sentar las bases para futuras optimizaciones en el diseño y control de heliostatos autónomos.

4.1 Diseño de algoritmo

El algoritmo propuesto elimina la necesidad de utilizar variables como el tiempo (fecha y hora) y los datos de geolocalización (latitud, longitud de la ubicación) para orientar la superificie reflejante (espejo) de los heliostatos en el campo solar. Con este enfoque, solo se requiere una calibración inicial, que consiste en redirigir manualmente la luz solar hacia el objetivo usando el heliostato que se está calibrando. Una vez logrado esto, se registra la orientación del espejo, caracterizado por su vector normal, denominado $\overrightarrow{N}$.

Para que el algoritmo de calibración funcione, es necesario conocer la dirección que tienen los rayos solares, caracterizado por un vector unitario $\overrightarrow{S}$ en ese momento específico. Esta información se obtiene mediante un dispositivo capaz de proporcionar la posición del sol en tiempo real (seguidores solares, tablas, etc), el desarrollo e implementación de dicho dispositivos será detallado más adelante.

El resultado del cálculo de la primera ecuación en esta etapa de calibración es la dirección de reflexión que redirige la luz hacia el objetivo en ese instante, caracterizado por el vector unitario de dirección $\overrightarrow{R}$. Es importante mencionar que dicha dirección $\overrightarrow{R}$ será constante a lo largo del día, para dicho heliostato, dado que la posición física del heliostato y el target permanencen constantes,es decir, durante el día lo único que cambia es la orientación del espejo, mas no su ubicación.

A continuación se presenta como se determinaron los modelos matemáticos para un instante cualquiera. Observando la posición del sol que corresponde a un vector dirección $\overrightarrow{S}$ unitario (Figura 4.1).

Si se toma el plano generado por los vectores dirección $\overrightarrow{R}$ y $\overrightarrow{S}$ se obtiene una

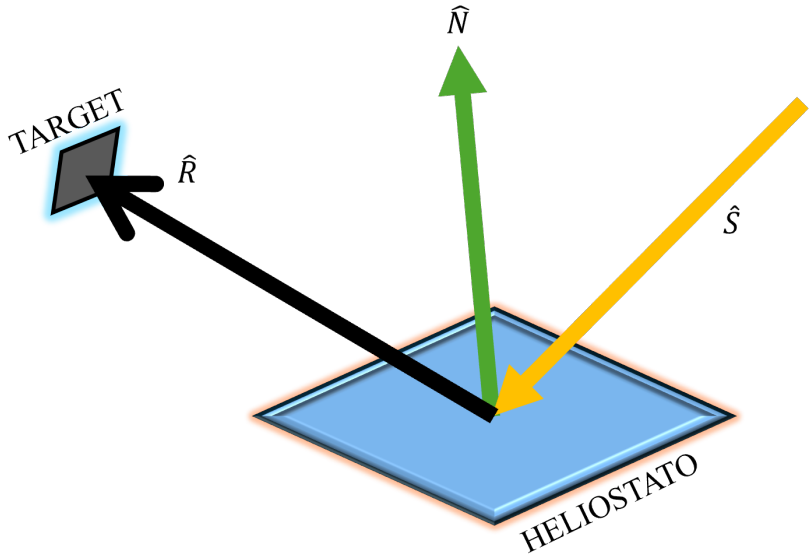


Figura 4.1: Representación de los vectores $\overrightarrow{R}$, $\overrightarrow{N}$ y $\overrightarrow{S}$ en el espejo del heliostato

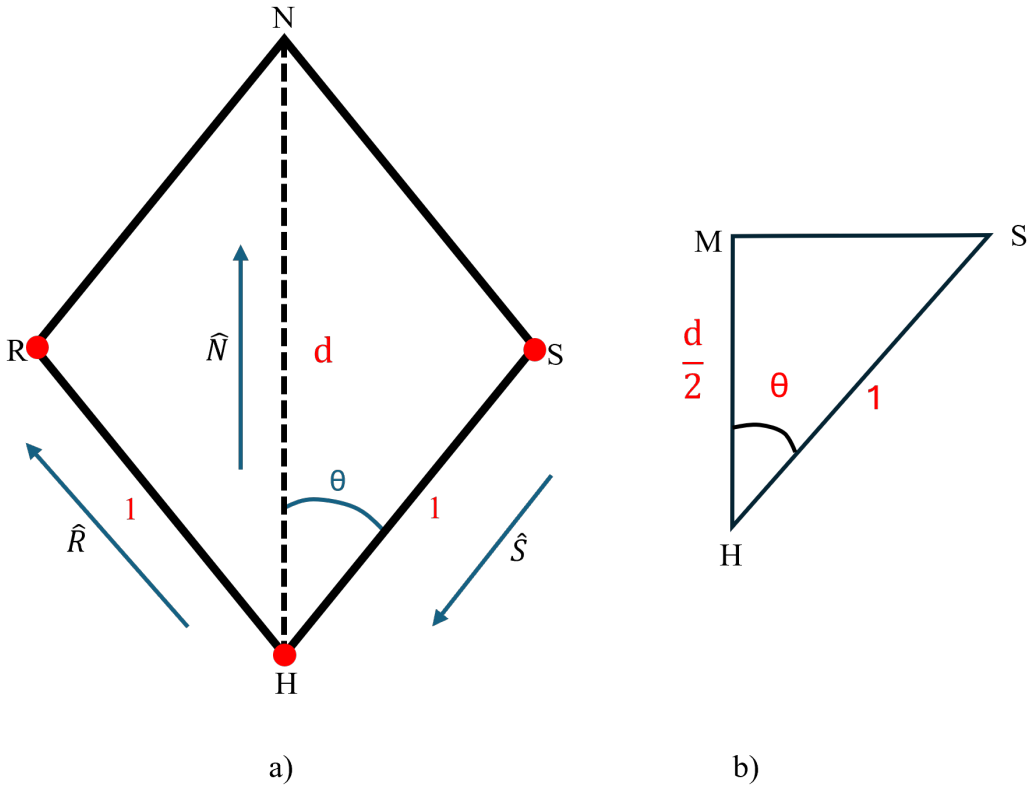


Figura 4.2: Geometría del plano RS

geometría como se muestra en la Figura 4.2. El rombo generado por la geometría tendría valores de lado 1, debido a que son vectores unitarios, con una diagonal mayor d, la cual bisecta el ángulo ∠ RHS, Por tanto la diagonal coincide precisamente con el vector $\overrightarrow{N}$ como se puede apreciar en la Figura 4.2a. En el triángulo HMS (Figura 4.2b) donde M es el punto medio de la diagonal $\overline{ON}$, con el uso de la expresión del coseno se obtiene la ecuación 4.1.

$$\cos(\theta) = \frac{d}{2} \tag{4.1}$$

Despejando d en la Ecuación 4.1 se obtiene la Ecuación 4.2 :

$$d = 2\cos(\theta) \tag{4.2}$$

Ahora bien, haciendo uso de la definición del producto punto entre dos vectores se obtiene la Ecuación 4.3:

$$\overrightarrow{N} \cdot (-\overrightarrow{S}) = |\overrightarrow{N}||(-\overrightarrow{S})|\cos(\theta) \tag{4.3}$$

Como se mencionó previamente los vectores son unitarios, entonces, simplificando la Ecuación 4.3 se obtiene la Ecuación 4.4.

$$\overrightarrow{N} \cdot (-\overrightarrow{S}) = \cos(\theta) \tag{4.4}$$

En la misma Figura 4.2b, se aprecia que $\overrightarrow{HR} = \overrightarrow{HN} + \overrightarrow{NR}$, se sabe que $\overrightarrow{HR} = \overrightarrow{R}$, $\overrightarrow{NR} = \overrightarrow{S}$ porque es un rombo y $\overrightarrow{HN} = d\overrightarrow{N}$ por lo tanto se establece el valor del vector $\overrightarrow{R}$ como se muestra en la Ecuación 4.5.

$$\overrightarrow{R} = d\overrightarrow{N} + \overrightarrow{S} \tag{4.5}$$

Lo que se busca es obtener una ecuación tal que $\overrightarrow{R}$ sea dependiente solo de $\overrightarrow{N}$ y $\overrightarrow{S}$ por tal motivo se usan susticiones para la variable d usando las ecuaciones 4.2 y 4.4 respectivamente teniendo como resultado a la Ecuación 4.6:

$$\overrightarrow{R} = 2(\overrightarrow{N} \cdot (-\overrightarrow{S}))\overrightarrow{N} + \overrightarrow{S} \tag{4.6}$$

Es importante destacar que el vector de dirección $\overrightarrow{R}$ calculado es una idealización, bajo la suposición de que el centro del espejo permanece fijo en todo el rango de

movimiento del heliostato. En estas condiciones, $\overrightarrow{R}$ no debe cambiar de dirección independientemente de la posición del espejo del heliostato (vector $\overrightarrow{N}$) o de la posición del sol (vector $\overrightarrow{S}$). Esto implica que a cada orientación de $\overrightarrow{N}$ le corresponde un único valor de $\overrightarrow{S}$ y viceversa, el cual debe satisfacer la Ecuación 4.6.

La segunda etapa del algoritmo es la de operación y cálculo del vector $\overrightarrow{N}$ dado que en esta fase el algoritmo el sistema debe calcular la posición idónea del espejo ($\overrightarrow{N}$) para que esta redireccione la luz solar en un instante cualquiera del día. Dado que es el mismo sistema de refencia, la Figura 4.2a , funge como herramienta para realizar el modelado. Se oberva que $\overrightarrow{HN}$ es quivalente a la suma de las direcciones $\overrightarrow{HR}$+$\overrightarrow{NR}$, y sustituyendo los valores de los vectores direción se determina la Ecuación 4.7.

$$\overrightarrow{N} = \frac{\overrightarrow{R} + (-\overrightarrow{S})}{|\overrightarrow{R} + (-\overrightarrow{S})|} \tag{4.7}$$

Nótese que en la Ecuación 4.7 el resultado se divide entre su magnitud para generar un vector $\overrightarrow{N}$ unitario y que además el vector $\overrightarrow{S}$ es negativo por la convención en la dirección de este. Sumado a lo anterior se debe tener presente que en la Ecuación previa se considerará al vector $\overrightarrow{R}$ como una constante debido a que este fue calculado en la primera etapa, por lo cual la variable independiente de la ecuación es el vector dirección $\overrightarrow{S}$ en cada instante.

4.2 Implementación de algoritmo

Una vez obtenidos los modelos matemáticos basados en las ecuaciones finales de $\overrightarrow{R}$ y $\overrightarrow{N}$ de la sección anterior (ecuaciones 4.6 y 4.7), se procede con su implementación en el entorno de MATLAB, esto para su validación. Para ello, se desarrollaron las funciones `calibration_R()` y `calculation_N()`:

- **Función** `calibration_R()`**:** Esta función corresponde a la primera etapa del algoritmo y se encarga de calcular el vector de dirección de reflexión $\overrightarrow{R}$. Re-

cibe como parámetros de entrada dos variables: $\overrightarrow{N}$ (orientación del espejo del heliostato) y $\overrightarrow{S}$ (vector dirección del un rayo de sol proporcionada de forma externa). La función devuelve un vector unitario $\overrightarrow{R}$, que representa la dirección en la que la luz solar se redirige hacia el objetivo en el momento de la calibración.

- **Función** `calculation_N()`**:** Forma parte de la segunda etapa del sistema. Toma como argumentos de entrada el vector $\overrightarrow{R}$ (calculado previamente en la etapa de calibración y considerado constante durante la operación) y el vector $\overrightarrow{S}$ (dirección vectorial del sol proporcionada externamente). Devuelve un vector unitario $\overrightarrow{N}$ que representa la posición óptima que el espejo del heliostato debe adoptar para redirigir la luz solar hacia el objetivo a lo largo del día.

Dado que el prototipo opera con dos grados de libertad (azimutal y de elevación) para definir la orientación del espejo, se desarrollaron dos funciones adicionales para facilitar la conversión entre ángulos y vectores tridimensionales:

- **Función** `angle_to_vector()`**:** Recibe dos valores angulares (azimut y altura) y retorna un vector unitario en tres dimensiones, correspondiente a la dirección espacial del espejo o del sol (Apéndice A).
- **Función** `vector_to_angle()`**:** Recibe un vector tridimensional y devuelve los valores de azimut y altura que componen dicho vector, permitiendo interpretar la orientación del espejo del heliostato o la posición solar (Apéndice B).

Esta implementación permite que el sistema gestione la orientación del espejo del heliostato en tiempo real, utilizando las ecuaciones matemáticas desarrolladas previamente para optimizar el seguimiento y redireccionamiento de la luz solar. Además, las funciones adicionales fueron diseñadas de manera que respeten la convención y el rango de los signos de los ángulos azimutales, garantizando que el sistema opere dentro del mismo marco de referencia y mantenga la coherencia en los cálculos angulares durante la operación del heliostato (Figura 4.3). En la Figura 4.4 se

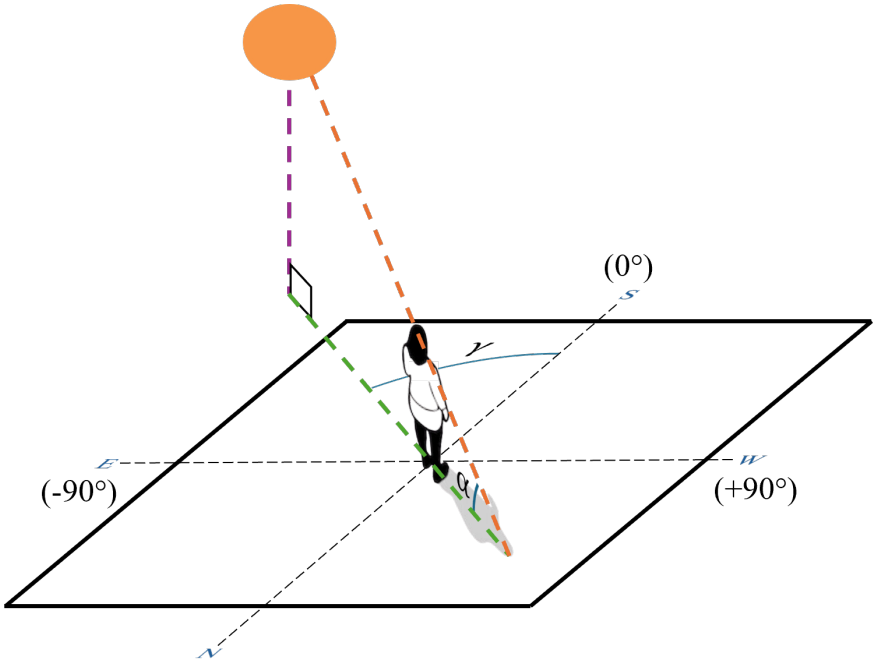


Figura 4.3: Sistema de referencia diseñado para la implementación.

muestra el diagrama de flujo que representa el funcionamiento general del sistema general al que de desea llegar. Para la ejecución del algoritmo de control se diseñaron e implementaron múltiples dispositivos y software de programación.

4.3 Herramientas experimentales

4.3.1 Prototipo Heliostato

El diseño y la implementación del heliostato se llevaron a cabo mediante un enfoque sistemático que abarcó desde la conceptualización inicial hasta la programación y el ensamblaje final. A continuación, se detallan los pasos seguidos en el proceso:

- **Diseño en CAD:** El primer paso en el desarrollo del heliostato fue su diseño utilizando software de modelado CAD (Figura 4.5). Cada componente fue di-

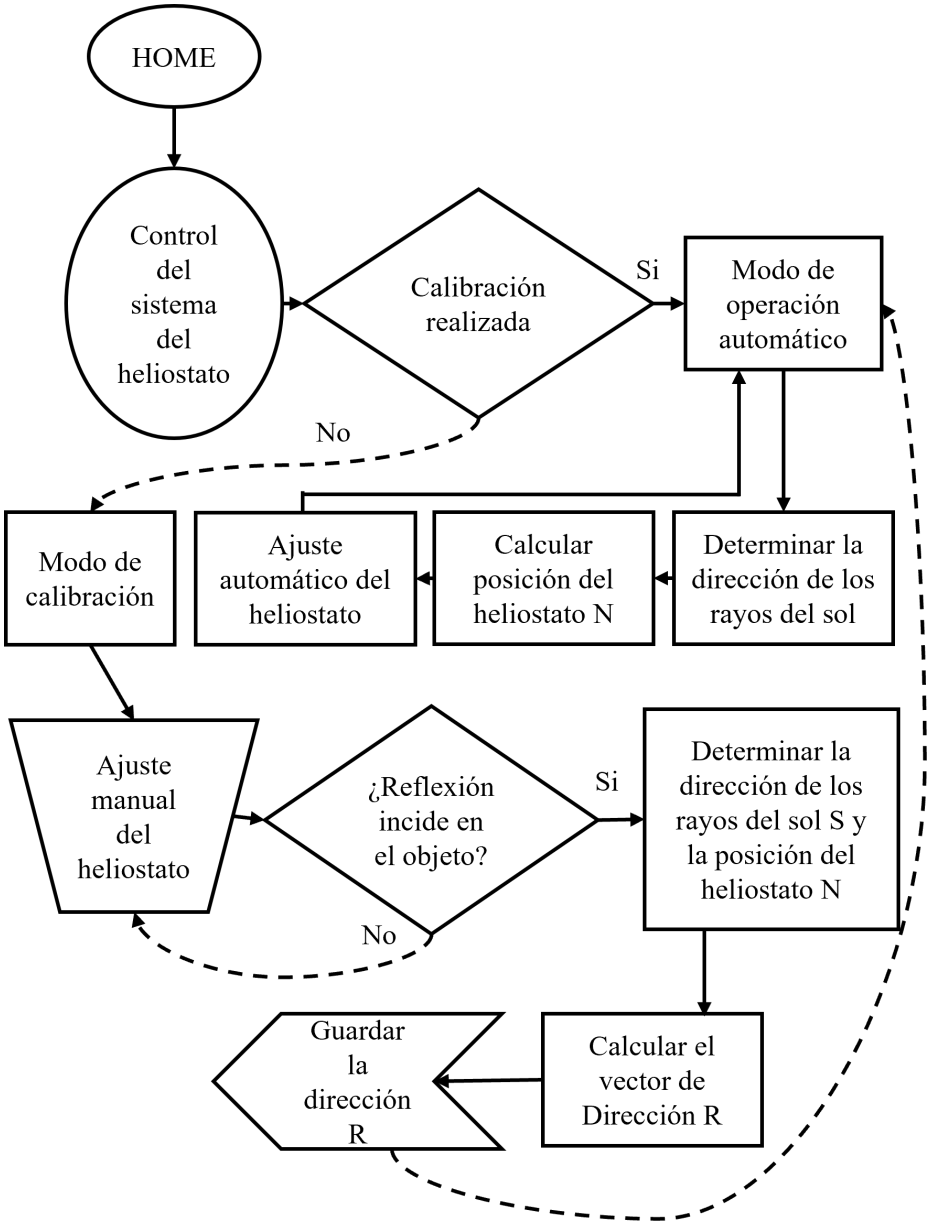


Figura 4.4: Diagrama de flujo del algoritmo de control desarrollado.

señado meticulosamente para asegurar su funcionalidad y compatibilidad con el resto del sistema. Una vez finalizado el diseño digital, las piezas se imprimieron utilizando una impresora 3D con filamento PLA, un material plástico que ofrece una buena combinación de bajo costo y facilidad de impresión. Las piezas impresas fueron ensambladas siguiendo el diseño CAD realizado en SOLIDWORKS 2023, asegurando que cada componente encajara perfectamente.

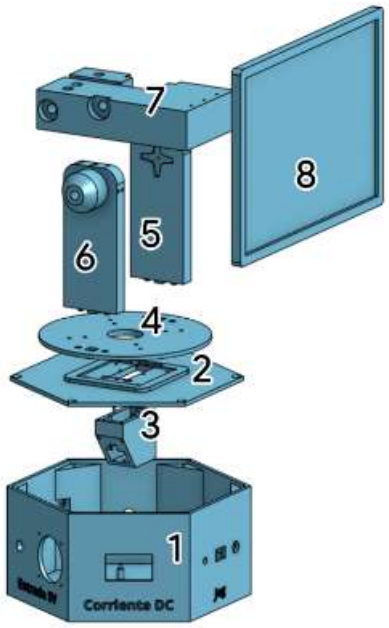


Figura 4.5: Elementos del prototipo heliostato.

1. *Base del heliostato:* La base hexagonal del heliostato contiene el microcontrolador y la tarjeta de conexiones, que son esenciales para su funcionamiento. En su interior, se encuentran los elementos responsables de la electrónica principal y las conexiones de corriente que permiten alimentar el sistema. Además, la base incluye botones para encender el heliostato y cambiar entre los diferentes modos de operación. También cuenta con entradas de voltaje, a través de las cuales se conecta a las fuentes de alimentación necesarias para su correcto desempeño.

2. *Placa de montaje:* La placa de montaje protege los componentes internos de control y potencia. Además, contiene el espacio destinado a un rodamiento giratorio que permite el movimiento del espejo en el eje X (azimutal). La parte superior también funciona como tapa para el rodamiento giratorio. En la parte inferior de la placa, se ubica la sujeción para la pieza 3.

3. *Montura de servomotor para eje X:* Esta pieza es una carcarsa para el servomotor encargado de realizar el movimiento en el eje X, y su función

es posibilitar el ensamble del servomotor con la pieza 2.

4. *Base móvil:* La base conecta y permite el movimiento rotativo del eje X mediante el uso de un rodamiento giratorio. Esta misma base también actúa como soporte para las piezas 5 y 6. En su parte superior, presenta orificios destinados al ensamblaje de dichas piezas, mientras que en el centro cuenta con un agujero para permitir el paso de los cables de los componentes externos, los cuales serán almacenados en la base. En su parte inferior, la base está diseñada para montar el rodamiento giratorio del heliostato.

5. *Montaje eje Y- Servo:* El soporte del eje Y está diseñado para incluir el rodamiento y permitir el ensamblaje de un tornillo roscado con la pieza 6, lo que facilita el movimiento preciso del eje Y. Se ensambla manualmente mediante un acoplamiento en forma de cruz y se atornilla al servomotor del eje Y. Además, su base contiene guías cuadradas para asegurar una conexión uniforme con la pieza 4, a la cual también se atornilla.

6. *Montaje eje axial Y:* Su objetivo es centrar y proporcionar un movimiento fluido con la menor fricción posible, utilizando un rodamiento modelo 6000z fijado mediante tornillos opresores. La extrusión central sirve para alinear el eje y ajustar la distancia deseada en la que se encuentra el eje axial Y.

7. *Brazo móvil:* tiene la función de transferir el movimiento en el eje Y y alojar el servomotor correspondiente. Además, está diseñado para permitir la fijación de la pieza 8, que sirve como base del espejo. Una de las secciones del brazo está diseñada para generar un contrapeso, lo que ayuda a reducir el torque necesario para mover el servomotor Y (de altura).

8. *Base del espejo:* Esta base se hizo para sostener el espejo utilizado en el heliostato, el espejo seleccionado tiene el tamaño de 260 mm x 180 mm.

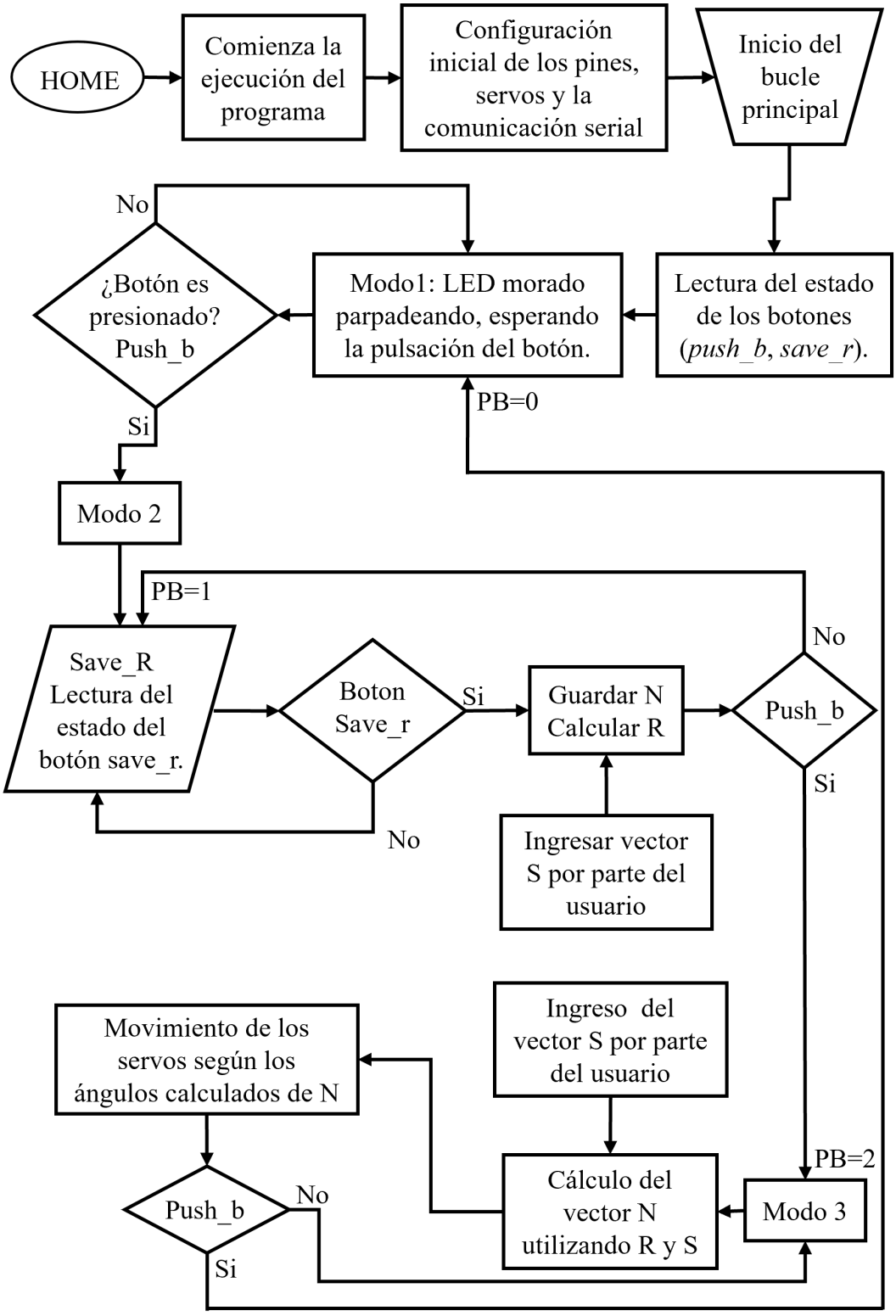


Figura 4.6: Diagrama de flujo del control del heliostato

- **Programación y Control:** El control del heliostato se realizó utilizando un microcontrolador ESP32 (Figura 4.7), conocido por su capacidad de manejo de múltiples tareas. La programación del ESP32 (Apéndice C) se enfocó en la gestión del movimiento de los servomotores a través del cálculo de los angulos de éstos mediante el uso del algoritmo diseñado y la interacción con el usuario (Figura 4.6). Para facilitar la navegación y el control manual, se incorporaron

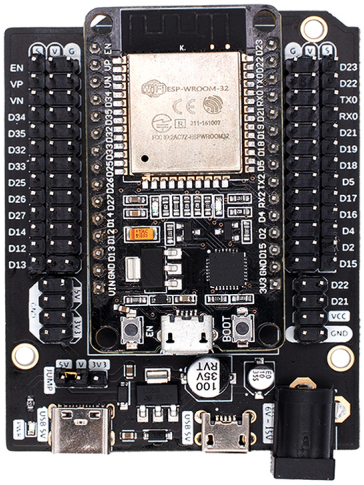


Figura 4.7: Microcontrolador ESP32.

botones que permiten al usuario seleccionar opciones del menú y ajustar el heliostato según sea necesario. Adicionalmente, se diseñó e imprimió un joystick que proporciona una interfaz de control manual intuitiva, permitiendo ajustes de los ángulos del heliostato en tiempo real (Véase Figura 4.8).

Figura 4.8: Joystick desarrollado para el control manual.

- **Ensamblaje Final y Pruebas:** Una vez ensambladas todas las piezas y completada la programación del microcontrolador, se procedió a la integración de todos los componentes. El sistema completo fue sometido a diversas pruebas para asegurar su correcto funcionamiento de tal forma en la que pueda ser comparado con el software de trazado de rayos y Matlab a través de los datos del NOAA (National Oceanic and Atmospheric Administration) (Figura 4.9).

Figura 4.9: Prototipo del heliostato ensamblado.

4.3.2 Seguidor solar

En el desarrollo del seguidor solar, se evaluaron diversas alternativas con el fin de determinar la mejor solución técnica y funcional para maximizar la eficiencia del sistema. La primera versión del seguidor se diseñó tomando como referencia la forma y dimensiones del heliostato, con el objetivo de aprovechar al máximo el diseño

mecánico existente y reducir complejidades en la integración de componentes (Figura 4.10).

Figura 4.10: Seguidor solar primera versión.

El seguidor solar utiliza un arreglo de fotorresistencias (LDR, Light Dependent Resistor) para detectar la intensidad de luz en diferentes direcciones. Las LDRs varían su resistencia en función de la cantidad de luz recibida: a mayor intensidad luminosa, menor resistencia, y viceversa (Figura 4.11). El sistema orienta automáticamente el panel frontal del prototipo hacia la posición del sol, minimizando la diferencia en las lecturas de luz entre las LDRs para garantizar un seguimiento preciso y eficiente durante todo el día.

Figura 4.11: Fotorresistencia (LDR).

El control del sistema se lleva a cabo mediante el mismo tipo de microcontrolador ESP32, que procesa las señales de las fotorresistencias y, según la diferencia detectada, envía órdenes a los motores para ajustar la posición del dispositivo. Para lograr el control de direccionamiento, el sistema emplea cuatro fotorresistencias (LDRs) dispuestas en un patrón cruzado o dentro de una carcasa que capta luz desde distintas direcciones. Las LDRs se agrupan en pares: Un par controla el movimiento horizontal (este-oeste). El otro ajusta el movimiento vertical (inclinación). El microcontrolador analiza las diferencias de luz entre los pares de LDRs. Si se detecta un desequilibrio, se genera una señal de error que activa los motores para corregir la posición y mantener la alineación con el sol.

El sistema se mantiene en constante ajuste a medida que el sol se desplaza en el cielo, repitiendo este proceso durante todo el día para asegurar la alineación. El punto de equilibrio se alcanza cuando las lecturas de todas las LDRs son similares o están dentro de un margen aceptable.

Sin embargo, durante la implementación, surgieron desafíos relacionados con las limitaciones del hardware (componentes electrónicos de baja calidad en la sensibilidad) disponible. Estos factores impidieron alcanzar la precisión deseada en el seguimiento solar, lo cual era crucial para el rendimiento óptimo del sistema

Buscando abordar este problema, se llevó a cabo un rediseño integral del seguidor solar. Este rediseño no solo implicó mejoras en la estructura física del dispositivo, sino también en el software de control. Se introdujeron nuevos mecanismos buscando un seguimiento más preciso y fiable (Figura 4.12).

Simultáneamente, se utilizó otro seguidor solar comercial. Este dispositivo fue sometido a una serie de modificaciones para adaptarlo a los requerimientos específicos del proyecto. Las modificaciones incluyeron ajustes en los componentes mecánicos (implementación de cruceta) y electrónicos (se retiraron múltiples sensores humedad, temperatura que no eran necesarios para el objetivo de dispositivo, reagrupación de los LDRs, así como en el software de control (reemplazo del código de programación),

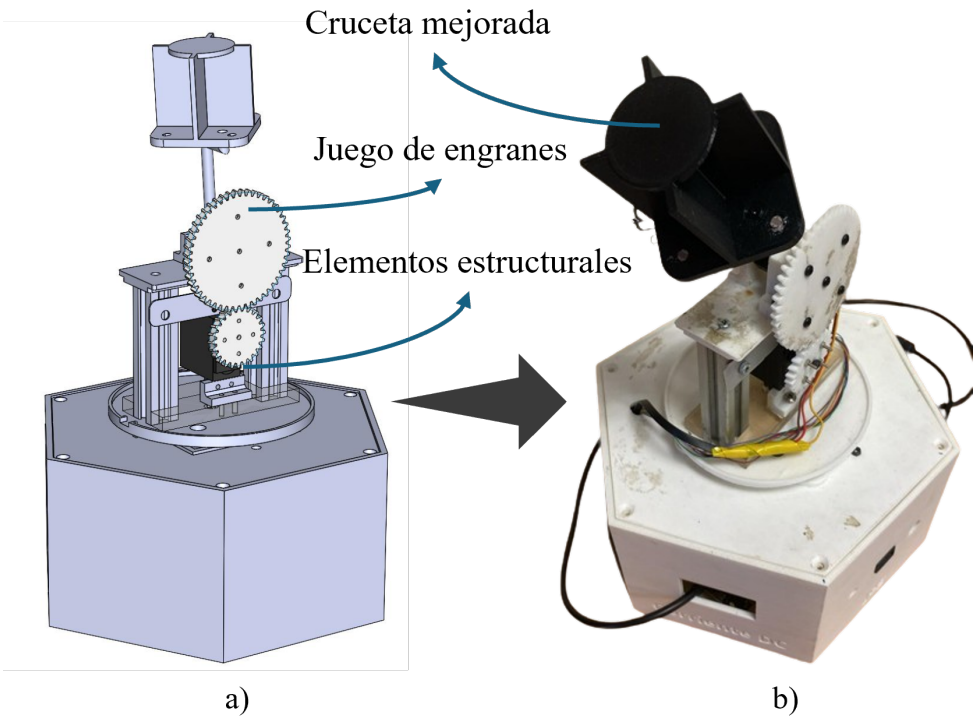


Figura 4.12: Seguidor solar versión 2. a) Ensamble en CAD, b) Implementación física.

y se realizaron pruebas de seguimiento solar para observar su comportamiento y fiabilidad (Figura 4.13).

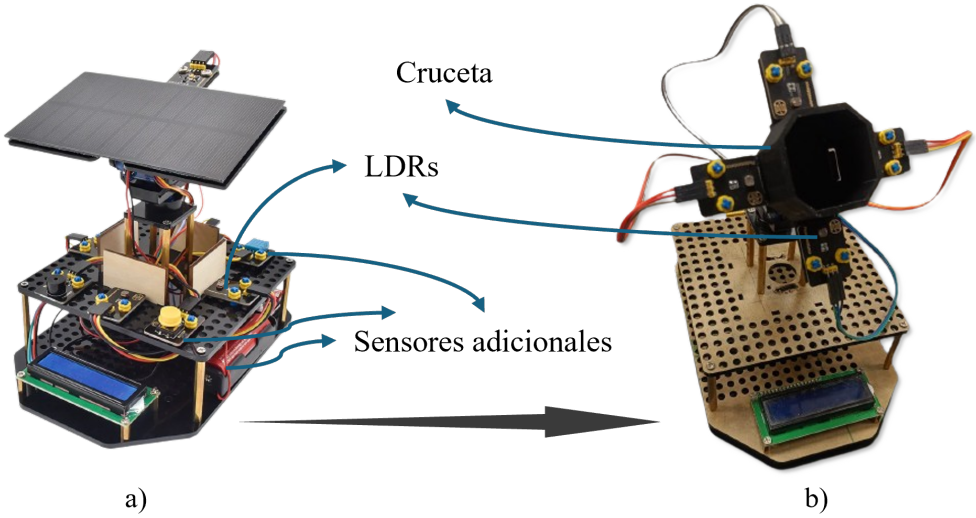


Figura 4.13: Seguidor solar KEYE. a)Comercial, b)Adaptado

4.3.3 Seguidor Solar óptico basado en sombras

Se desarrolló una innovadora alternativa utilizando procesamiento de imágenes de sombras para determinar la posición óptima del heliostato en todo momento. Esta tecnología ofrece un enfoque prometedor para mejorar la precisión y eficiencia en la medición del posicionamiento solar.

La metodología para el desarrollo del sistema de adquisición y procesamiento de imágenes se centra en el análisis de sombras proyectadas utilizando un gnomon como instrumento de referencia. Este sistema implementado en MATLAB emplea el espacio de color HSV para identificar puntos de referencia y calcular la longitud de las sombras capturadas en imágenes por una cámara. El objetivo principal es determinar los parámetros solares clave, como el azimut y la altura solar, de manera precisa y en tiempo real.

El sistema de adquisición de imágenes se diseñó con un gnomon cilíndrico de 8 mm de diámetro y 4 cm de altura, montado sobre una placa plana cuadrada de 20 cm de lado. Cada vértice de esta placa contiene un cuadrado adicional de 1 cm, lo que facilita la identificación de esquinas durante el postprocesamiento de imágenes. Este diseño fue fabricado mediante manufactura aditiva con impresora 3D, asegurando precisión y consistencia.

Para este trabajo se determina el **azimuth** como el ángulo que describe la dirección del sol con respecto al norte geográfico, considerando el eje sur como 0° en un sentido positivo horario para facilitar la comprensión visual. La **altura solar**, por su parte, es el ángulo de elevación del sol sobre el horizonte, variando entre 0° (cuando el sol está en el horizonte) y 90° (cuando el sol se encuentra en el cenit, directamente sobre el observador).

4.3.4 Modelado

Considerando los fundamentos previamente establecidos, se procede a analizar el dispositivo en una situación genérica con el propósito de determinar el modelo matemático correspondiente. Como se ilustra en la Figura 4.14, un rayo solar, representado como un vector dirección $\vec{S}$, incide sobre la superficie superior del gnomon. Esta interacción genera una sombra proyectada cuya característica de interés es el punto P (Figuras 4.14-4.15), definido como el extremo más alejado de la sombra proyectada. Este punto constituye un parámetro clave para el desarrollo del modelo.

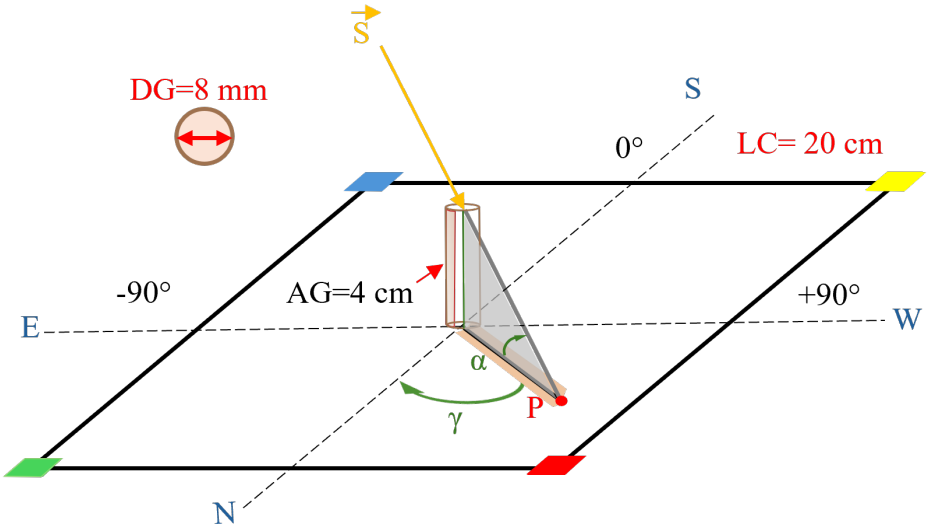


Figura 4.14: Sistema de referencia del gnomon.

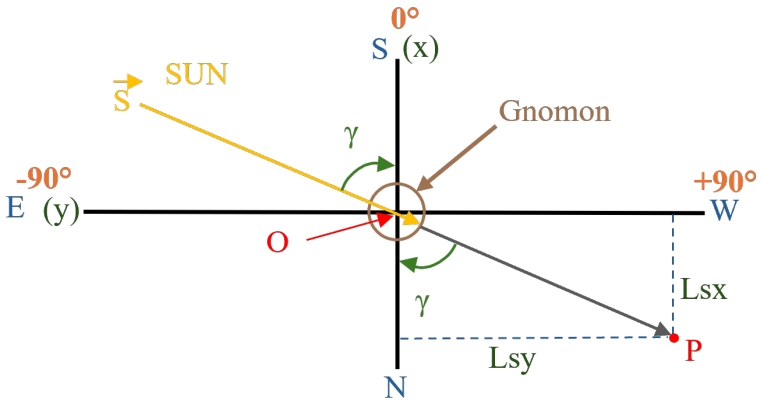


Figura 4.15: Vista superior del gnomon.

Es posible determinar el ángulo γ conociendo las coordenadas del punto P mediante la Ecuación 4.8.

$$\gamma = -\tan^{-1}\left(\frac{L_{Sy}}{L_{Sx}}\right) \tag{4.8}$$

donde L_{Sx} y L_{Sy} son las componentes de la línea de sombra en el plano horizontal. El signo negativo asegura que el cálculo se realice en el sistema de referencia definido previamente. La Figura 4.16 facilita la visualización para determinar el cálculo de "α" de la misma manera en que se calculó "γ" previamente, por lo tanto, se obtiene la Ecuación 4.9.

$$\alpha = \tan^{-1}\left(\frac{AG}{L_S}\right) \tag{4.9}$$

donde AG es la altura del gnomon y L_S es la longitud de la sombra proyectada en la base del gnomon. Una vez obtenidas las ecuaciones, lo siguiente es obtener el valor de las variables independientes, la cual es la distancia de la sombra proyectada en la base del gnomon, o bien las coordenadas del punto P en el sistema de referencia del plano.

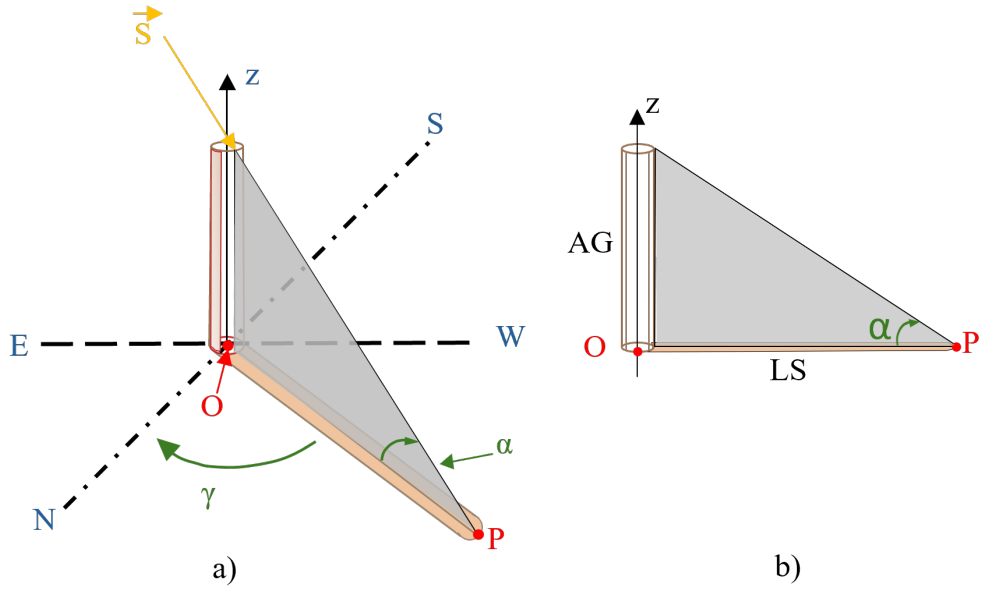


Figura 4.16: Análisis para cálculo de azimuth.

4.3.5 Implementación

El código MATLAB desarrollado gestiona la adquisición de imágenes, el procesamiento de sombras, y el cálculo de coordenadas en tiempo real (Apéndice D). La implementación se llevó a cabo mediante una integración de hardware y software como se muestra en la Figura 4.17 (Cámara Web, trípode para webcam, Laptop con Software de MATLAB, Gnomon fabricado con base plana de esquinas de colores).

Figura 4.17: Elementos del sistema de adquisión de imágenes.

Procesado de imagen

La correcta captura de imágenes es fundamental para garantizar la precisión del análisis de sombra. La ubicación ideal de la cámara sería directamente sobre la placa del gnomon para evitar cualquier distorsión; sin embargo, este posicionamiento podría generar problemas cuando el sol está cerca del mediodía solar, ya que la cámara podría proyectar su propia sombra sobre el gnomon, afectando la calidad de las imágenes capturadas. Para evitar este inconveniente, se optó por colocar la cámara en una posición fija que no interfiera con la proyección de la sombra en ningún momento del día.

No obstante, esta ubicación fija implica que las imágenes capturadas no presentan la perspectiva ideal y pueden incluir elementos adicionales fuera de la base cuadrada del gnomon. Por ello, es necesario aplicar un procesado de transformación de perspectiva para corregir la distorsión en las imágenes, así como un recorte preciso para aislar únicamente la región de interés. Para llevar a cabo estas operaciones, el algoritmo se basa en las esquinas de colores situadas en la placa, las cuales sirven como puntos de referencia para identificar y delimitar la zona de interés de manera precisa. Las etapas del proceso son las siguientes:

- **Captura de la fotografía.** Se seleccionó y configuró la cámara junto con el software de adquisición de imágenes para garantizar una captura óptima. Tras la configuración, se toma una imagen digital del gnomon y su sombra (Ver Figura 4.18), la cual se visualiza en tiempo real para verificar la calidad y composición adecuadas.

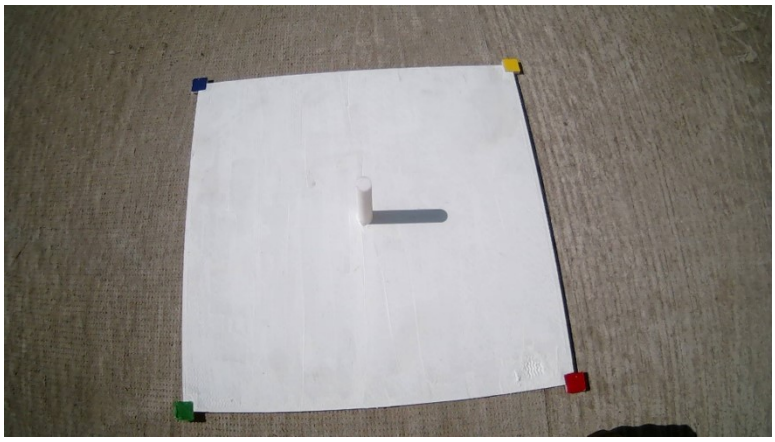


Figura 4.18: Imagen capturada por la cámara web.

- **Identificación de las esquinas.** Para determinar la posición y orientación del gnomon en la fotografía es clave identificar con precisión las esquinas y sus colores respectivos. Se llevó a cabo una transformación del espacio de color RGB a HSV, ya que este último ofrece una representación más intuitiva de los colores. Se definieron rangos de valores en el espacio HSV para cada color de

interés (rojo, verde, azul y amarillo), ajustados experimentalmente según las condiciones de iluminación. Utilizando máscaras binarias, se segmentaron los píxeles correspondientes a cada color, y se calcularon los centroides de las regiones segmentadas (usando el promedio de cada zona de color) para determinar las coordenadas de las esquinas de la base cuadrada. Estos centroides se superpusieron sobre la imagen original para evaluar cualitativamente la precisión del proceso tal como se observa en la Figura 4.19.

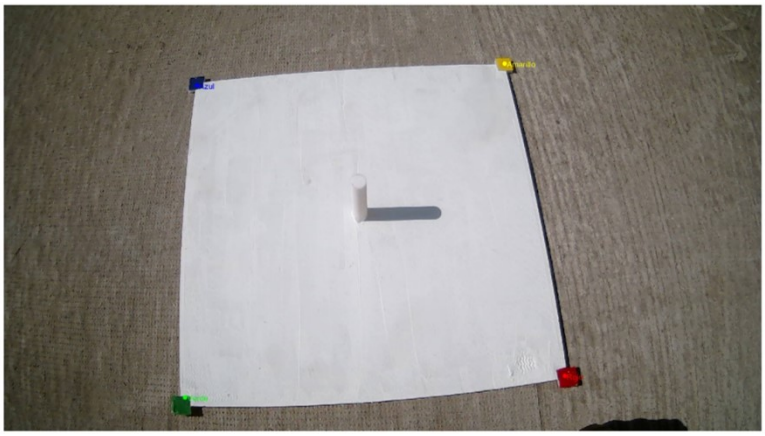


Figura 4.19: Detección de esquinas.

- **Corrección de Perspectiva.** La corrección de perspectiva se realizó utilizando una codificación cromática específica en las esquinas de la base cuadrada. Los colores rojo, verde, azul y amarillo (RGBY) se ubicaron en las esquinas de manera ordenada en sentido horario comenzando por la esquina superior izquierda. Esta disposición permitió identificar la orientación cardinal del gnomon de acuerdo con la convención previamente establecida.

  Para realizar la corrección de perspectiva, se emplearon transformaciones geométricas en MATLAB. Primero, se identificaron los puntos de origen y los puntos de destino en la imagen, los cuales definieron cómo debía ajustarse la perspectiva. Posteriormente, se calculó una transformación proyectiva entre estos puntos con el uso del comando **"tform"**, la cual fue utilizada para corregir la distorsión de la imagen.

Esta transformación fue ajustada de manera que los puntos identificados en la imagen original coincidieran con los puntos de referencia en la perspectiva corregida como se muestra en la Figura 4.20.

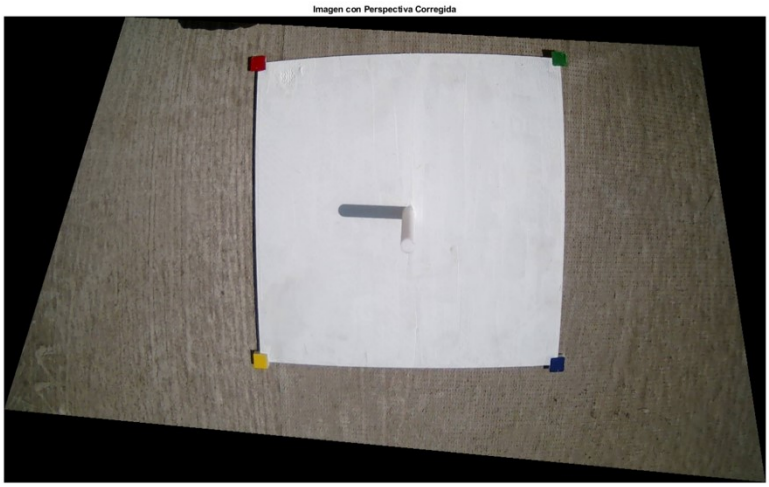


Figura 4.20: Corrección de perspectiva.

- **Recorte de la Imagen.** La identificación precisa del área sombreada proyectada por el poste central del gnomon representó un desafío debido a la presencia de múltiples regiones oscuras en la imagen al corregir la perspectiva. Para abordar esto, se realizó un recorte de la imagen delimitando un área de

Figura 4.21: Recorte de imagen.

interés que incluía únicamente el cuadro base del gnomon y sus esquinas de colores, utilizando las coordenadas de los centroides de las esquinas detectadas en la etapa anterior. Con esta información, se construyó un contorno rectangular que excluye el resto de la imagen (Figura 4.21).

- **Eliminación de Esquinas de Colores.** Para segmentar correctamente el área sombreada proyectada por el gnomon (como se muestra en la Figura 4.22), se convirtió la imagen a escala de grises, resaltando los contrastes lumínicos. Se delimitó un rectángulo dentro de la región blanca de la imagen para excluir las esquinas de colores y evitar interferencias en el análisis posterior. Utilizando la función **"imcrop"**, se extrajo la porción de la imagen correspondiente al área de interés.

Figura 4.22: Eliminación de esquinas.

- **Conversión a Escala de Grises y Eliminación de la Sombra Falsa.** Para identificar con precisión la sombra proyectada por el poste central del gnomon, se convirtió la imagen a escala de grises. Durante el análisis, se detectó una "sombra" adicional, denominada "Sombra falsa", generada por el propio gnomon debido al ángulo de captura de la fotografía (Figura 4.23). Para eliminar esta sombra artefactual, el algoritmo segmenta los píxeles correspondientes a esta sombra, enfocándose en aquellos ubicados por encima de una diagonal pre-

definida, creando así una "zona segura"libre de interferencias para el análisis posterior.

Figura 4.23: Máscara de sombra aplicada.

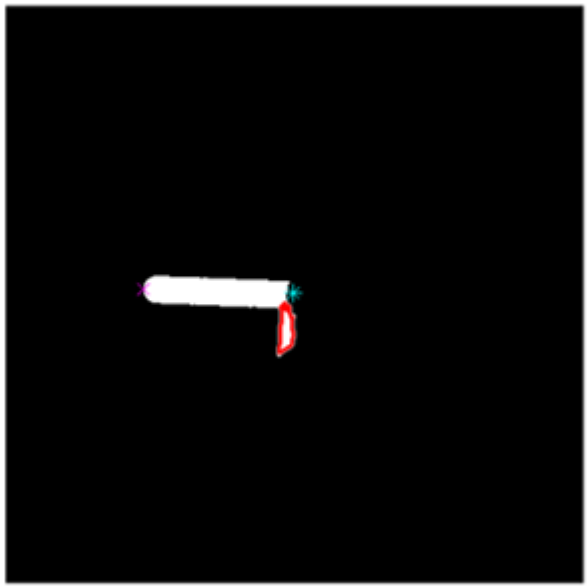


Figura 4.24: Zona segura e identificación del centro del Gnomon y final de sombra.

- **Identificación del Punto más Alejado del Centro del Gnomon.** En la Figura 4.23, se observa una serie de píxeles blancos alineados que parten del

centro del gnomon. El objetivo es encontrar el punto más alejado en esta línea y calcular su distancia al centro del gnomon. Esto se logra haciendo un barrido de todos los pixeles de sombra identifcados y determinar el más alejado. Esta distancia es fundamental para calcular las coordenadas del cenit y el azimut (Figura 4.24).

Para ello, se establece una relación entre la longitud de un lado del cuadro en píxeles y su longitud real en centímetros, permitiendo expresar la longitud de la sombra en unidades métricas. Es importante mencionar que, para un cálculo preciso, se resta medio diámetro del gnomon (4 mm) al valor de la longitud de la sombra, ajustando así el desfase del origen causado por la proyección de la sombra que se origina desde el borde superior del gnomon.

- **Cálculo de Azimut y Altura solar.** Utilizando las ecuaciones 4.8 y 4.9 desarrolladas previamente en el modelado matemático, se calculan las coordenadas y azimutales y de elevación del sol en función de la longitud de la sombra proyectada y la orientación del gnomon.

- **Exportación de Datos.** Finalmente, los resultados obtenidos para las coordenadas azimutales y de altura solar se exportan en un formato estándar, lo que facilita su posterior análisis e integración en otros modelos o sistemas para su uso (Figura 4.25). Se adaptó el código de tal manera que permite el procesamiento de las imágenes de manera iterativa, realizando capturas en intervalos de tiempo y períodos establecidos. Este proceso garantizó la obtención continua de datos relevantes para cada instante.

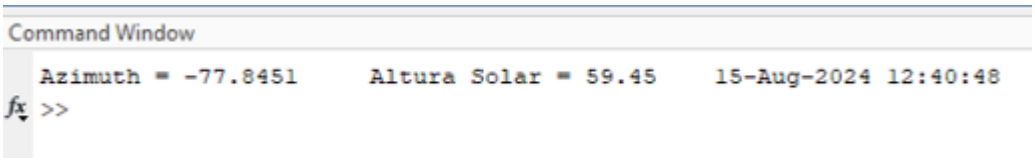


Figura 4.25: Datos exportados.

Durante las pruebas realizadas, se registraron los valores de tiempo, azimut y al-

tura solar, con el propósito de capturar las variaciones de la trayectoria solar a lo largo del tiempo. Estos datos se compararon posteriormente con los ángulos solares proporcionados por el modelo del NOAA (National Oceanic and Atmospheric Administration), con el fin de analizar su tendencia y similitud.

4.3.6 Integración de dispositivos para validación de algoritmo

Se llevaron a cabo pruebas experimentales como parte de la validación del algoritmo de redireccionamiento desarrollado. Estas pruebas consistieron en utilizar los dispositivos diseñados durante la metodología: un prototipo de heliostato, un gnomon y una base niveladora. El objetivo fue evaluar el desempeño del algoritmo en condiciones reales y validar sus cálculos con herramientas externas.

En primer lugar, se calibró el gnomon apoyándolo sobre la base niveladora. Se aseguró que su plano estuviera perfectamente alineado a 0 grados con respecto a la horizontal. Una vez nivelado, el dispositivo se conectó a una computadora para recibir datos en tiempo real sobre la posición solar. Paralelamente, el heliostato se colocó orientado en el eje norte-sur, con su espejo apuntando hacia el sur. Un "target-ectangular de 0.5 x 0.65 m se ubicó a una distancia de 5 metros hacia el sur y a una altura de 3 metros. Finalmente, el heliostato también se conectó a la computadora para proceder con su operación.

La prueba consistió en utilizar el heliostato para redirigir la energía solar hacia el "target". Inicialmente, en el modo de calibración, se fijó manualmente un punto de referencia en el "targetütilizando un joystick. Esta dirección inicial, representada por el vector $\overrightarrow{N}$, fue registrada junto con el vector de reflexión $\overrightarrow{R}$. Luego, se esperaron 10 minutos para permitir que la posición del sol cambiara de manera perceptible. Una vez registrada la nueva posición solar ($\overrightarrow{S}$), se activó el modo automático del algoritmo. En esta etapa, el heliostato recalculó en tiempo real el nuevo valor de

$\overrightarrow{N}$ requerido para mantener $\overrightarrow{R}$ constante, redirigiendo el haz solar al mismo punto inicial. Este proceso se repitió dos veces más, recalculando $\overrightarrow{R}$ en intervalos promedio de 10 minutos. Registro de datos Durante las pruebas, se registraron los siguientes parámetros:

- $\overrightarrow{N}$, posición inicial del heliostato en formato $[Az^{\circ}, Als^{\circ}]$.
- $\overrightarrow{S}$, posición solar inicial en formato $[Az^{\circ}, Als^{\circ}]$.
- SV, vector unitario asociado a $\overrightarrow{S}$.
- $\overrightarrow{R}$, dirección de reflexión en formato $[Az^{\circ}, Als^{\circ}]$.
- RV, vector unitario asociado a $\overrightarrow{R}$.
- $\overrightarrow{S2}$, nueva posición solar tras el intervalo de 10 minutos en formato $[Az^{\circ}, Als^{\circ}]$.
- $S2V$, vector unitario asociado a $\overrightarrow{S2}$.
- $\overrightarrow{N2}$, nueva posición del heliostato tras el intervalo en formato $[Az^{\circ}, Als^{\circ}]$.
- $N2V$, vector unitario asociado a $\overrightarrow{N2}$.

Los valores obtenidos del microcontrolador ESP32 se compararon rigurosamente con cálculos realizados en MATLAB. Las únicas variables compartidas entre ambos sistemas fueron $\overrightarrow{N}$ al inicio y todas las posiciones solares ($\overrightarrow{S}$) proporcionadas por el gnomon.

4.4 Simulaciones

4.4.1 Deriva de Campo 3x3

En esta sección se reporta toda la información generada acerca del análisis de la deriva en el prototipo de heliostato de laboratorio de termosolar de la facultad

causada por su diseño geométrico real. El algoritmo y el pseudocódigo de control empleados idealizan un movimiento con 2 grados de libertad (azimutal y cenital), originados en el centro del espejo, lo que provoca una deriva en el objetivo. Todo el análisis ha sido implementado utilizando MATLAB y TracePro.

Se realizó una caracterización detallada del prototipo, incluyendo las dimensiones del espejo y los mecanismos del heliostato. Se documentaron las dimensiones precisas para su uso en el modelado y simulación.

Tabla 4.1: Dimensiones del prototipo físico para la simulación

Articulación	**Longitud**
Base-eje	250 mm
Eje-espejo	150 mm

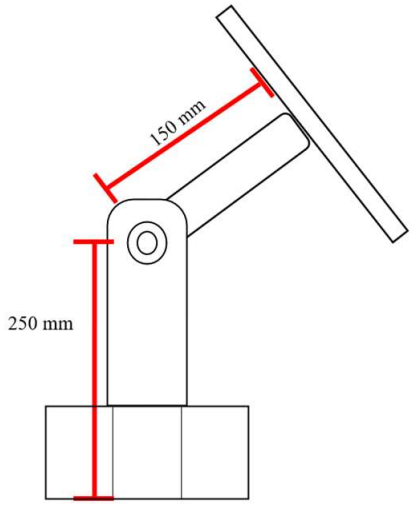


Figura 4.26: Dimensiones del Heliostato.

Modificación del Código: Con las dimensiones obtenidas, se modificó el código en MATLAB para introducir estos valores y establecer un sistema de referencia adecuado. Se determinó la distancia del heliostato al objetivo (target) y las dimensiones del target, posicionándolo en el origen a 2 metros de altura como se muestra en la Figura 4.27.

```
%Ajuste de inclinación para el espejo (update 2023)
Ajuste_altura = 90;         %Distancia de la rotula al espejo
Ajuste_EjeIncY = 0;         %Default =0; para tesis
Ajuste_AlturaEjeIncY = 247;   %Ajuste de altura al eje de inclinacion, 247 prototipo impreso
Ajuste_Brazo_espejo=90;    %Ajuste distancia entre el eje de inclinación y espejo


%Tamaño y resolución del sol
SunSize = 1;        %*Mas adelante se le hace un fix
SunGridRes = 50;    % default =100    al aumentar disminuye

%Información de heliostatos
ancho_espejo=0.26;
alto_espejo=0.18;
Area_espejo = ancho_espejo*alto_espejo; %Area real del espejo
numero_targets = 1; %numero de targets (nuestro caso es UNICO)
n_material = 1; %Eff 100% material: Perfect mirror
altura_target=2000;

%Prueba experimental
Coordenadas_del_target = [0,0,altura_target];     %Coordenadas del target
```

Figura 4.27: Parámetros geométricos del heliostato.

Configuración y diseño del campo de heliostatos: Se introdujo un campo de heliostatos, optando por una configuración cuadrada de 3x3, resultando en un total de 9 heliostatos (Ver Figura 4.28).

```
% Heliostatos en un "campo" cuadraro de LxL m2 con Nh x Nh heliostatos
L = 12; % distancia en metros
Nh = 3;  % numero de helistatos por lado

dist_campo_target=3000; %%mm
% Tamaño del sol a simular
SunSize=1000;
```

Figura 4.28: Parámetros geométricos del campo.

Se aplicó el algoritmo de seguimiento que utiliza datos externos de ángulos de Azimut y altura solar, correspondientes a cada día 21 del mes durante el año 2024 en intervalos de 5 minutos, obtenidos de datos georreferenciados de la facultad. Implementación del Algoritmo de Seguimiento: El algoritmo calcula los ángulos que cada heliostato debe realizar para redirigir el rayo de luz solar hacia el objetivo predeterminado. Se añadieron bloques de código para mostrar gráficamente el punto de impacto de la redirección y las coordenadas respecto al centro del espejo en

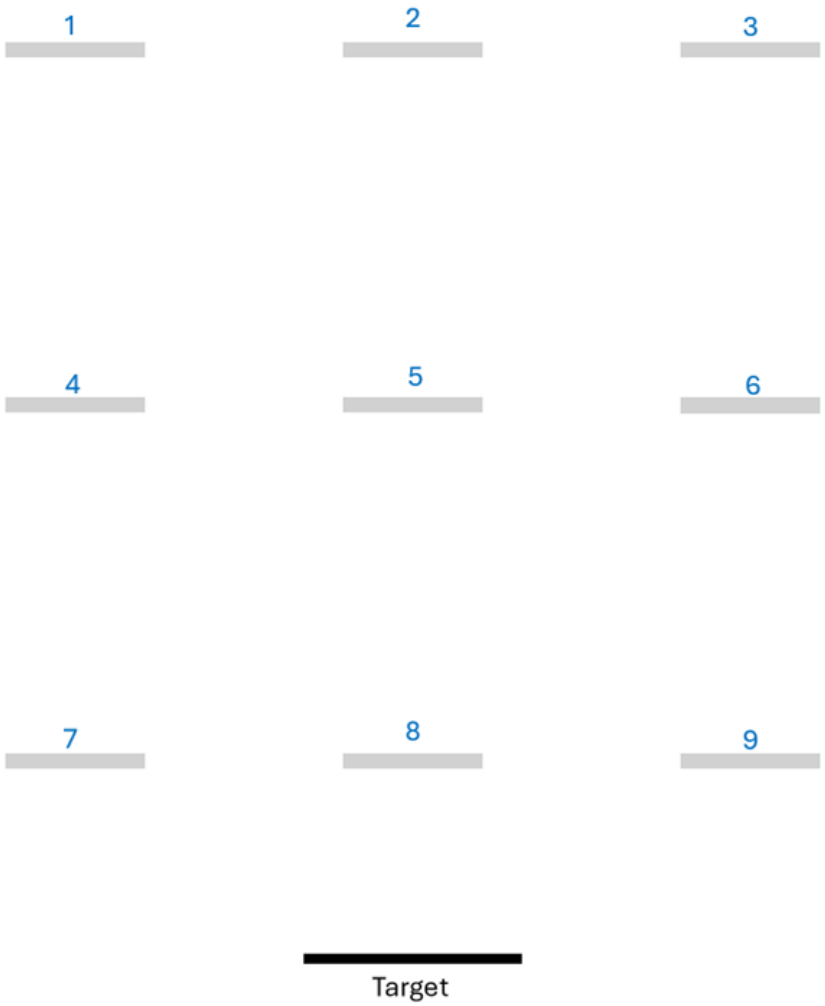


Figura 4.29: Diagrama representativo de la distribución del campo.

milímetros (Ver Figura 4.30).

Si desea conocer más a detalle el pseudocódigo desarrollado en MATLAB consulte el Apéndice E.

4.4.2 Desfase de Campo 3x3

El desfase en un campo de heliostatos se refiere a la desviación de cada espejo respecto a su posición ideal dentro del diseño original. Este error puede deberse a

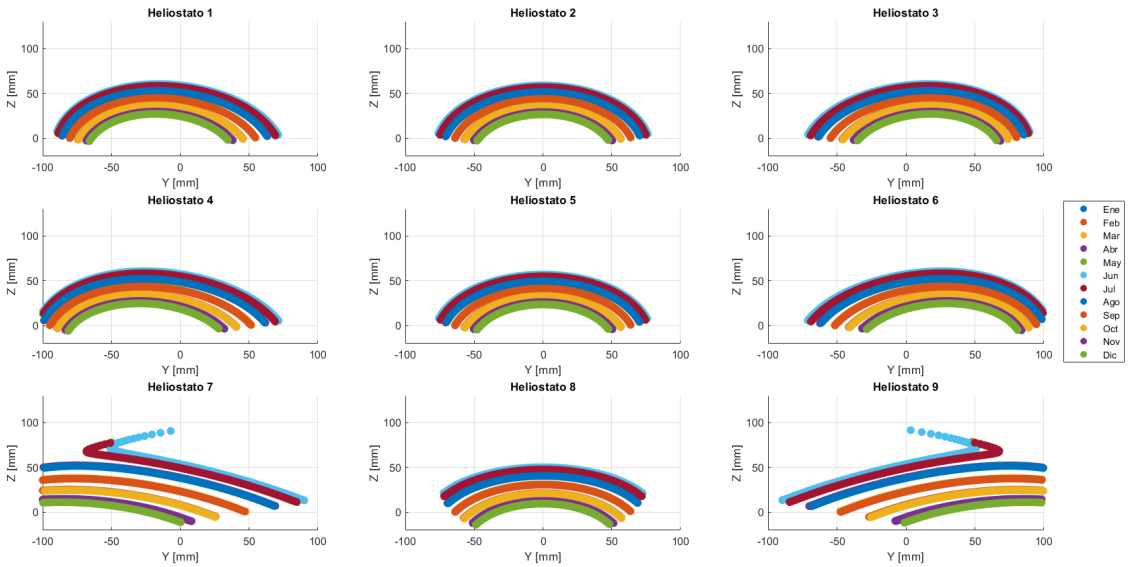


Figura 4.30: Redireccionamiento del campo durante todo el año de cada heliostato.

inexactitudes en la instalación, alineación inicial incorrecta o factores estructurales que afectan la disposición de los heliostatos en el terreno. En esta sección, se pretende analizar cómo estos desfases afectan la eficiencia de concentración sobre el objetivo, con especial enfoque en el derrame de energía.

Para evaluar el impacto del desfase, se realizaron simulaciones en MATLAB y TracePro considerando el mismo campo 3X3 de la sección anterior con las mismos instantes para evaluar. Se introdujeron errores en las coordenadas de cada heliostato respecto a su ubicación ideal, variando los desplazamientos en los siguientes rangos:

- Desplazamiento en eje X: 0,10,20,30,40,50,100,250,500,750,1000 mm.
- Desplazamiento en eje Y: 0, 1, 2, 5, 10, 15, 20, 25, 30, 40, 50,100, 200, 250, 300, 350, 400, 500, 750, 1000 mm.

En cada simulación (Apéndice F), se mantuvieron constantes el algoritmo de seguimiento (que utiliza datos externos de ángulos de Azimut y altura solar, correspondientes a cada día 21 del mes durante el año 2024 en intervalos de 5 minutos,

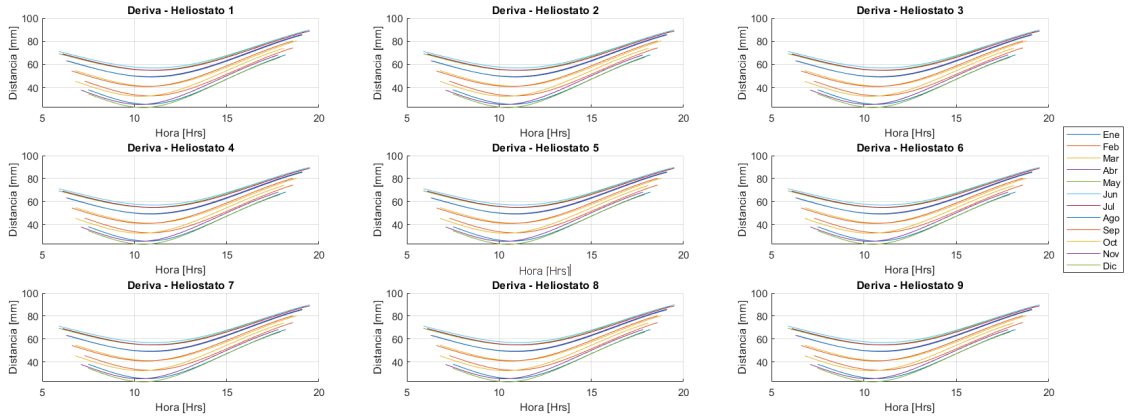


Figura 4.31: Deriva del campo durante el 2024.

obtenidos de datos georreferenciados de la facultad), los parámetros de radiación solar, posición del objetivo y condiciones de reflexión del espejo, permitiendo observar exclusivamente el efecto del desfase en la eficiencia de concentración.

Los resultados de cada caso se analizaron comparando el flujo energético incidente en el objetivo, identificando pérdidas de concentración en forma de derrames de energía. Se graficó la eficiencia en función del desplazamiento, destacando los umbrales en los que la desviación comienza a afectar significativamente la captación solar.

En proyectos termosolares, la correcta instalación de los heliostatos es fundamental para optimizar la eficiencia del campo. Errores en el posicionamiento pueden generar pérdidas considerables de energía, afectando el rendimiento global de la planta. Estudios previos han demostrado que incluso desplazamientos de pocos milímetros pueden reducir la concentración solar en un 10-20 %, dependiendo de la disposición del campo y de las tolerancias del sistema de seguimiento [42].

4.4.3 Rotación de Campo 3x3

Además del desfase en la posición de los heliostatos, otro factor crítico en la implementación de campos solares de torre central es la rotación involuntaria del campo. Esto puede ocurrir debido a errores en la nivelación del terreno, imprecisiones en la calibración de la estructura de montaje o deformaciones mecánicas que afectan la orientación global del conjunto de heliostatos.

Para evaluar el efecto de la rotación del campo sobre la eficiencia de concentración, se realizaron simulaciones en MATLAB y TracePro aplicando un giro uniforme a todo el conjunto de heliostatos dentro de un rango de:

- Ángulos de rotación: -2.25°, -2.00°, -1.75°, -1.50°, -1.25°, -1.00°, -0.75°, -0.50°, -0.25°, 0°, 0.25°, 0.50°, 0.75°, 1.00°, 1.25°, 1.50°, 1.75°, 2.00°, 2.25°.

Se simuló la respuesta del campo a los mismos instantes del día tal y como se realizaron en las 2 simulaciones anteriores, manteniendo constantes la posición del objetivo y las condiciones de radiación. En cada caso, se midió el flujo energético captado por el objetivo y se evaluó la eficiencia de concentración, con énfasis en derrames de energía debido a la desalineación progresiva del campo (Apéndice F).

En proyectos reales de CSP (Concentrated Solar Power), la rotación del campo puede ser consecuencia de errores en la instalación o de movimientos estructurales con el tiempo. En instalaciones grandes, incluso una rotación de 0.5° puede generar pérdidas de eficiencia significativas, comprometiendo el rendimiento del sistema. Este análisis permite cuantificar estos efectos y resaltar la importancia de un control preciso en la implementación de plantas solares de torre central [43].

Capítulo 5

Resultados

En esta sección se presentan los resultados obtenidos durante la ejecución de la metodología desarrollada y las pruebas realizadas. Las primeras pruebas experimentales se produjeron en el mes de julio 2023, sin embargo, se reporta información (Hora, ángulos de rotación e inclinación) de operación del heliostato durante un periodo de 4 horas comenzando a las 8:45 AM del día 13 de Julio 2023 con saltos de 30 min por instante evaluado. Se registraron los datos obtenidos en una simulación en Matlab-TracePro analizando las mismas variables de entrada (γ_h,α_h) provenientes del heliostato (operando en modo manual) con la misma tasa de muestreo del evento tal y como se muestra en la Tabla 5.1.

La Tabla 5.2 muestra el porcentaje de similitud entre los ángulos de salida para ambos procesos de la metodología en los mismos instantes. Adicionalmente durante la experimentación se tomaron fotografías de la luz incidente en el objetivo predeterminado. La comparación de las imágenes resultantes de la simulación y aquellas capturadas durante la experimentación desempeña un papel crucial para observar la funcionalidad (Figura 5.1).

Al confrontar las imágenes que representan la redirección de la luz sobre el objetivo con los mapas de irradiancia calculados mediante simulaciones, se busca evaluar la coherencia y validez del modelo de simulación empleado. Esta comparación

Tabla 5.1: Ángulos de salida del heliostato en la experimentación y simulación del 13 de julio 2023.

Hora	**Rotación** (γ_h) [°]		**Inclinación** (α_h) [°]	
	(Exp.)	**(Sim.)**	**(Exp.)**	**(Sim.)**
08:45	54.00	54.77	34.50	32.68
09:15	58.00	58.24	37.50	35.63
09:45	63.00	62.15	39.50	38.61
10:15	68.00	65.98	41.50	41.02
10:45	73.00	70.66	42.50	43.14
11:15	77.00	75.45	44.50	45.00
11:45	84.00	80.60	45.50	46.71
12:15	86.00	86.76	46.50	47.59
12:45	91.00	91.38	46.50	48.10

Tabla 5.2: Porcentaje de similitud de los ángulos entre la simulación y experimentación.

Hora	**Rotación** (γ_h) [%]	**Inclinación** (α_h) [%]
08:45	98.6	94.7
09:15	99.6	95.0
09:45	98.7	97.7
10:15	97.0	98.8
10:45	96.8	98.5
11:15	98.0	98.9
11:45	96.0	97.4
12:15	99.1	97.7
12:45	99.6	96.7

directa permite determinar que el grado de similitud observado en la Tabla 5.2 tiene una correlación coherente con la realidad experimental. De esta manera, se busca respaldar la fiabilidad del enfoque de simulación y ofrecer una mayor confianza en los resultados obtenidos, fortaleciendo así las conclusiones y aportes de este estudio. Es importante enfatizar que el centro negro en el objetivo tiene fines de referencia para la captura de imágenes y no contribuye positiva o negativamente en el resultado de las pruebas obtenidas.

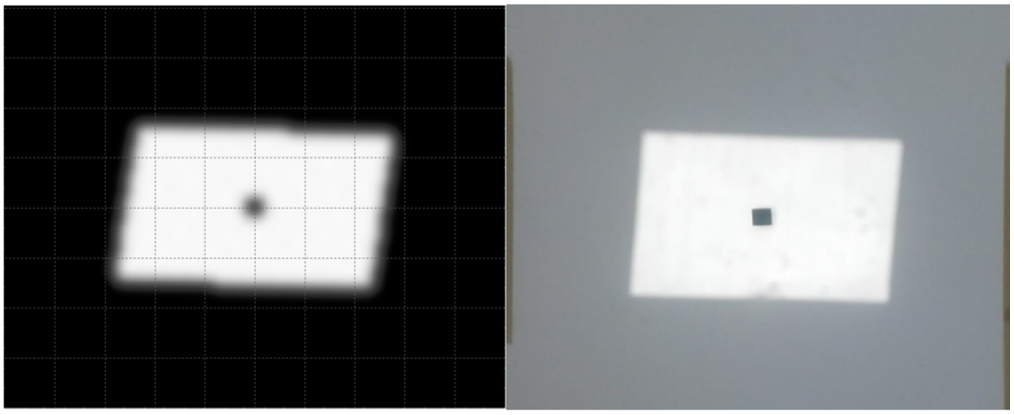


Figura 5.1: Imágenes obtenidas en ambos procesos metodológicos para el mismo instante: a) simulación en TracePro y b) prueba física experimental.

5.1 Gnomon

Con los datos obtenidos en la prueba del 27 de septiembre del 2024 (8:24 AM. – 16:12 PM en intervalos de 2 min.) realizado en la parte superior del edificio del polideportivo de la Facultad de Ingeniería Mecánica y Eléctrica (FIME) (con ubicación geográfica 25.7267675602177° Lat. N, -100.313586577701° Long. E. y -6 en zona de tiempo E.) se presenta la figura que muestra los datos recopilados y los datos del NOAA (Ver Figura 5.2.).

Se observaron diferencias constantes entre los valores obtenidos en las pruebas

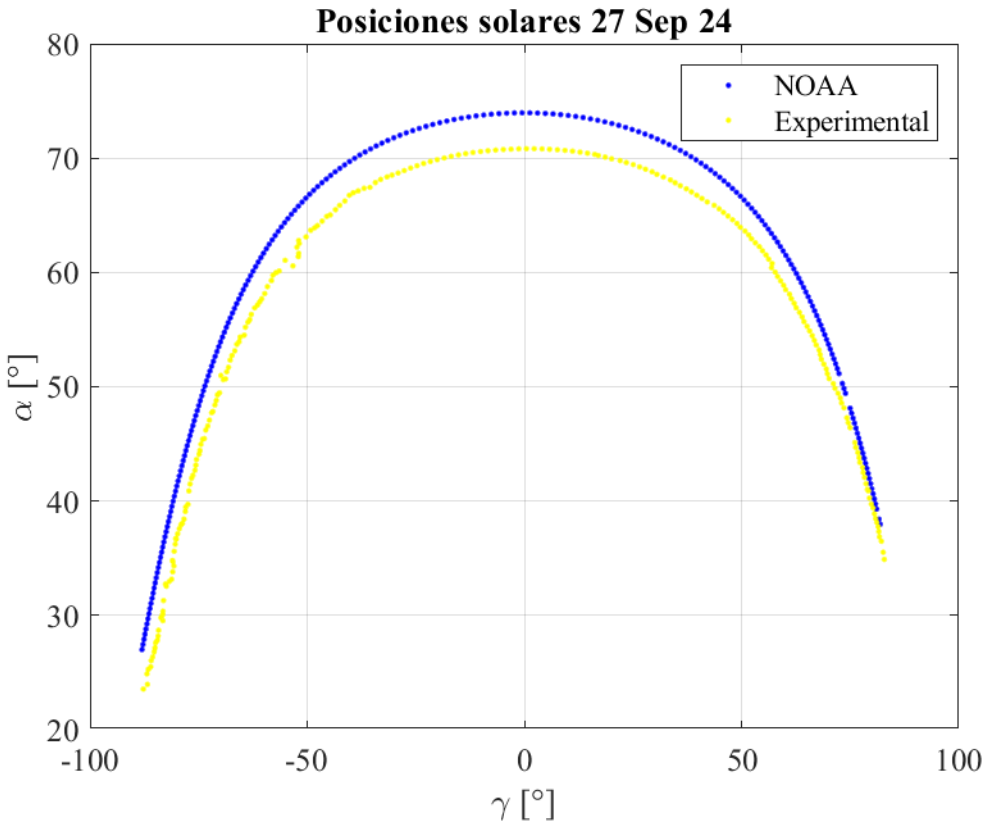


Figura 5.2: Posiciones solares obtenidas por el gnomon comparadas con el NOAA en los mismos instantes.

y los valores ideales calculados por el modelo del NOAA. Estas discrepancias pueden atribuirse principalmente a factores como la desalineación del sistema o un ángulo de desfase entre el marco de referencia utilizado y las condiciones ideales asumidas por el modelo. Es importante resaltar que el NOAA considera un sistema de referencia perfecto, con una orientación ideal hacia los polos y en un plano completamente horizontal. Sin embargo, en este trabajo, los ángulos se midieron con un marco de referencia que puede estar sujeto a declinaciones, inclinaciones o desfasamiento respecto a los polos, lo cual introduce variaciones en los resultados. En el post-procesado se realizó un ajuste en los datos de azimuth ya que es esta variable la que tiende a presentar desfase constante atribuido a la desalineación del gnomon .

5.2 Validación del algoritmo

El 14 de noviembre del 2024 se llevaron a cabo pruebas experimentales como parte de la validación del algoritmo de redireccionamiento desarrollados. Estas pruebas consistieron en utilizar los dispositivos diseñados durante la metodología: un prototipo de heliostato, un gnomon y una base niveladora. El objetivo fue evaluar el desempeño del algoritmo en condiciones reales y validar sus cálculos con herramientas externas.

5.2.0 Registro de datos

Durante las pruebas, se registraron los siguientes parámetros:

- N: posición inicial del heliostato (coordenadas polares ajustadas de la normal de la superficie reflejante).
- S: posición solar inicial.
- SV: vector unitario asociado a S.
- R: dirección de reflexión.
- RV: vector unitario asociado a R.
- $S2$: nueva posición solar tras el intervalo de 10 minutos.
- $S2V$: vector unitario asociado a $S2$.
- $N2$: nueva posición del heliostato tras el intervalo.
- $N2V$: vector unitario asociado a $N2$.

Los valores obtenidos del microcontrolador ESP32 se compararon rigurosamente con cálculos realizados en MATLAB. Las únicas variables compartidas entre

ambos sistemas fueron N al inicio y todas las posiciones solares (S) proporcionadas por el gnomon. Las comparaciones revelaron una alta concordancia entre los resultados obtenidos por el microcontrolador y MATLAB, confirmando la fiabilidad del algoritmo implementado.

5.2.0 Análisis de resultados de Comparación entre ESP32 y MATLAB.

Tabla 5.3: Comparación de resultados entre ESP32 y MATLAB.

Magnitud	**ESP32**	**MATLAB**
N	7.2800	7.2800
S	[29.0137, 37.8536]	[29.0137, 37.8536]
SV	[-0.6905, 0.3830, -0.6136]	[-0.6905, 0.3830, -0.6136]
R	[0.9473, 0.1819, 0.2637]	[0.9473, 0.1819, 0.2637]
RV	[31.2392, 36.4661]	[31.2392, 36.4661]
$S2$	[31.2392, 36.4661]	[31.2392, 36.4661]
$S2V$	[-0.6876, 0.4171, -0.5943]	[-0.6876, 0.4171, -0.5943]
$N2$	[8.1867, 27.4517]	[8.1867, 27.4517]
$N2V$	[0.8784, -0.1264, 0.4610]	[0.8784, -0.1264, 0.4610]

Se observó que las discrepancias entre los resultados fueron mínimas, con diferencias insignificantes en las cifras significativas de los vectores unitarios.

5.3 Análisis de la Deriva:

Se evaluó la efectividad del algoritmo durante todas las épocas del año, observando los resultados para cada heliostato. Se encontró que los heliostatos 2, 5 y 8, ubicados frente al espejo, mostraron menor deriva en comparación con los demás. Los

heliostatos 7 y 9 fueron los menos eficientes debido a su proximidad y desalineación con el centro.

Análisis usando trazado de rayos: Para complementar el estudio de la deriva en el prototipo de heliostato, se realizó una simulación del campo utilizando TracePro con el objetivo de observar y verificar el mismo comportamiento en la deriva. Para ello, se decidió simular cada espejo de manera individual con el argumento de poder observar la deriva individualmente. Considerando que el campo es simétrico con una configuración de 3x3 heliostatos, se seleccionaron tres espejos ubicados en la diagonal principal. Esta selección permitió aportar la información necesaria de cada zona del campo. El análisis se llevó a cabo evaluando un instante específico del año: el 20 de junio de 2024 a las 11:54 horas. Esta fecha y hora fueron elegidas para proporcionar una referencia temporal clara y específica. Los resultados de la simulación permitieron obtener mapas de irradiancia, los cuales se presentan en las Figuras 5.3,5.4 y 5.5.

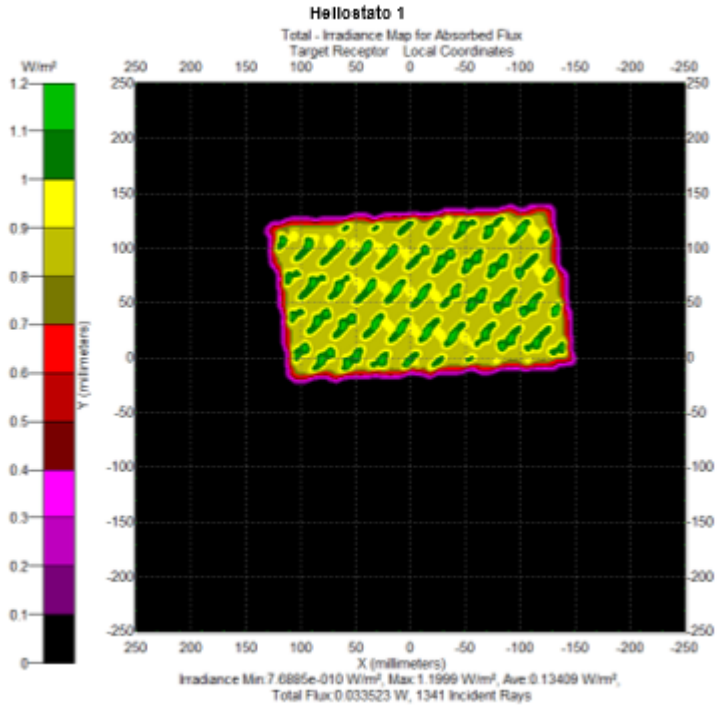


Figura 5.3: Mapa de irradiancia del heliostato 1.

Identificación de Causas de Deriva: Se concluyó que la deriva observada en el

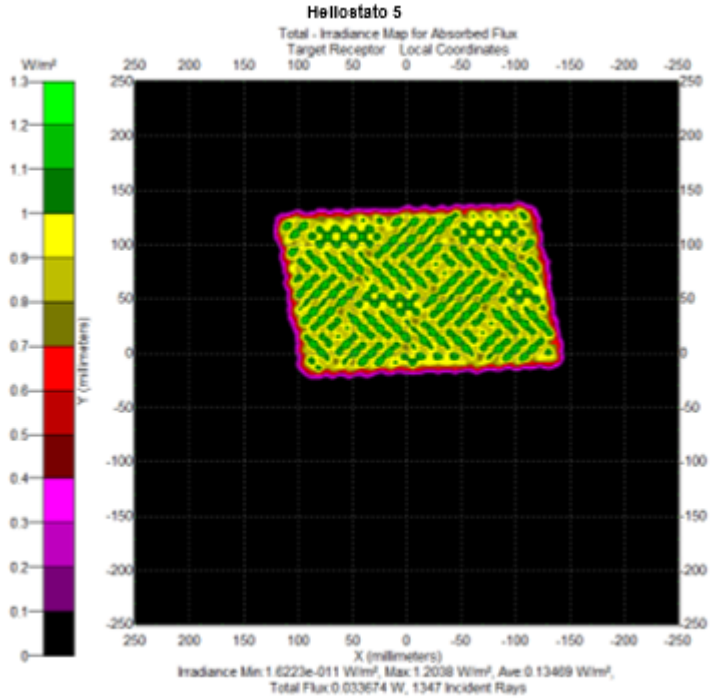


Figura 5.4: Mapa de irradiancia del heliostato 5.

comportamiento del heliostato se debe a la idealización del movimiento del espejo en el algoritmo de control. Este algoritmo actualmente considera que el movimiento se realiza desde el centro del espejo, lo cual no representa fielmente la realidad del sistema, que opera mediante un mecanismo de eslabones. Esta simplificación en el modelo matemático provoca errores en el apuntamiento y, en consecuencia, una deriva acumulada a lo largo del tiempo.

Propuesta de Mejora: Se propuso una optimización del algoritmo de control mediante la implementación de una serie de iteraciones adicionales que permitirán afinar el apuntamiento del heliostato. Estas iteraciones están diseñadas para ajustar de manera precisa la orientación del espejo, considerando las características del mecanismo de eslabones y su impacto en el movimiento final del espejo. El objetivo es minimizar los errores de apuntamiento y, por ende, reducir la deriva.

Implementación y Simulación: Una vez implementada la mejora en el algoritmo, se realizó una nueva simulación utilizando MATLAB y TracePro. Esta simulación

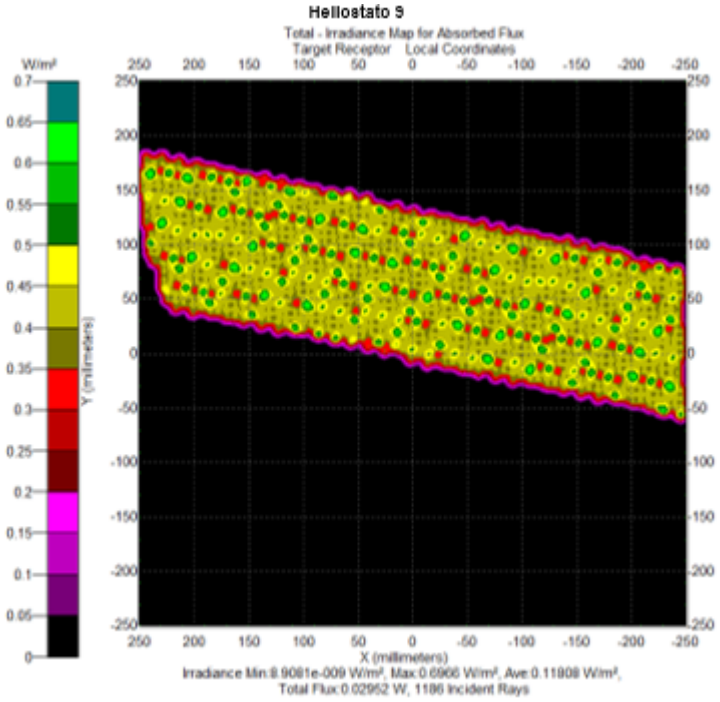


Figura 5.5: Mapa de irradiancia del heliostato 9.

tuvo como objetivo observar y comparar los resultados obtenidos con la nueva versión del algoritmo frente a los resultados anteriores. Se buscó determinar si la mejora propuesta tiene un impacto significativo en la reducción de la deriva del heliostato. Los resultados de la simulación permitieron evaluar la efectividad de las iteraciones adicionales y verificar la precisión del apuntamiento mejorado.

En las Figuras 5.6, 5.7 y 5.8 se observa que el algoritmo de iteración disminuye significativamente la deriva de los espejos y favorece el redireccionamiento al centro del objetivo.

Se observó una mejora significativa en la precisión de la redirección de los rayos solare tras la implementación del algoritmo mejorado, como se puede apreciar claramente en la Figuras 5.9. El error de redirección tiende a cero en la escala milimétrica, lo que demuestra que todos los puntos convergen en el objetivo sin importar su orientación y época del año. Esto indica una corrección efectiva de la deriva inicial con el primer algoritmo implementado.

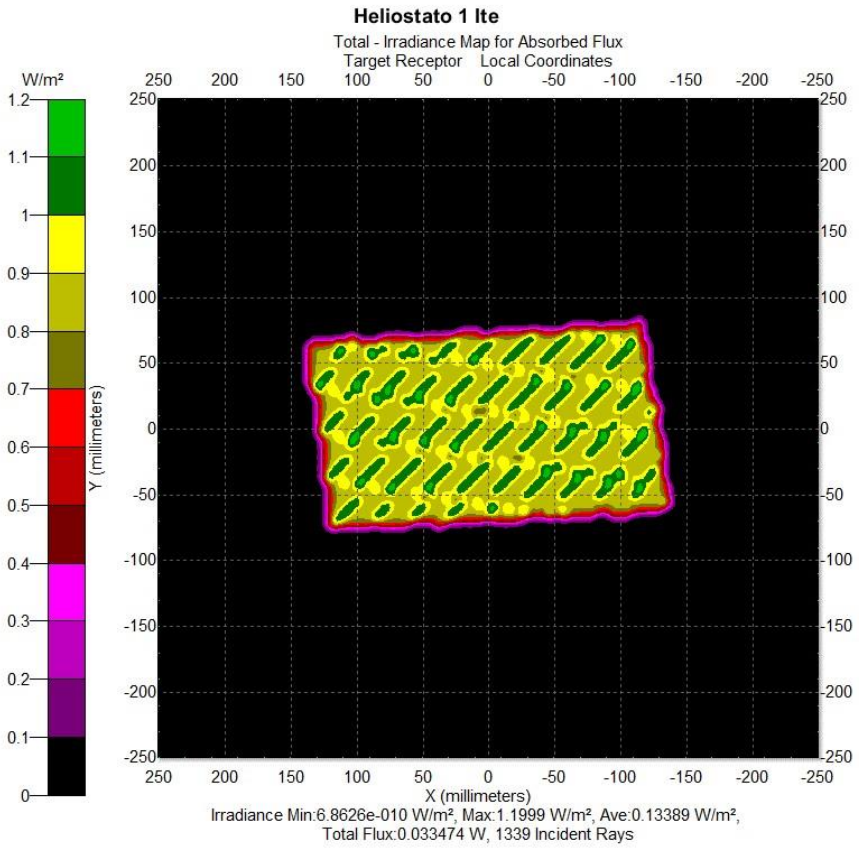


Figura 5.6: Mapa de irradiancia del heliostato 1 corregido con iteración.

Las Figuras 5.10 muestra que la deriva tiene resultados del orden de 10^{-11} mm, lo que se entiende como una deriva prácticamente nula causada por el desfase dinámico del centro ideal del espejo. Esto subraya la eficacia del nuevo algoritmo en reducir las desviaciones y mejorar la precisión global del sistema.

5.4 Simulación de desfase lineal de campo 3X3

Los resultados obtenidos a partir de las simulaciones realizadas en MATLAB y TracePro permiten evaluar el impacto de los errores de posicionamiento en la eficiencia de concentración del campo solar. En esta fase del estudio, se analizaron

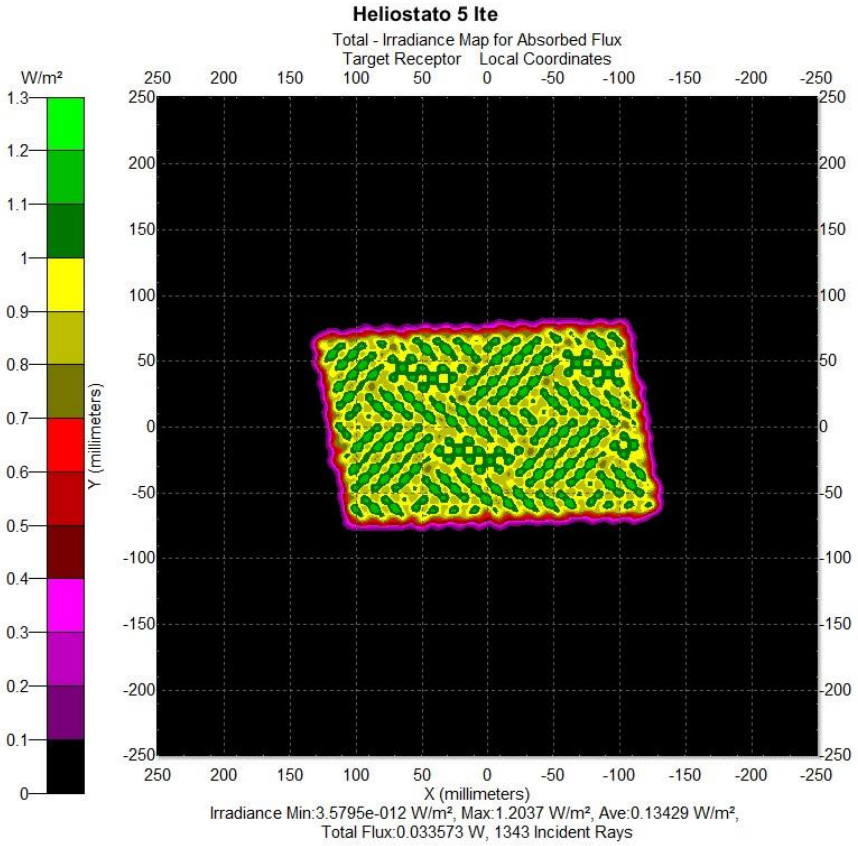


Figura 5.7: Mapa de irradiancia del heliostato 5 corregido con iteración.

desplazamientos en los ejes X y Y, variando las posiciones de los heliostatos con respecto a su ubicación ideal y evaluando cómo estos desplazamientos afectan la eficiencia de concentración (η) como se muestra en la tabla 5.4 .

5.4.1 Tendencias generales en los datos

Los valores de eficiencia reportados en la Tabla 5.4 muestran una tendencia clara: a medida que el desplazamiento en X y Y aumenta, la eficiencia del sistema disminuye progresivamente.

Para desplazamientos menores a 50 mm en X o Y, la eficiencia permanece

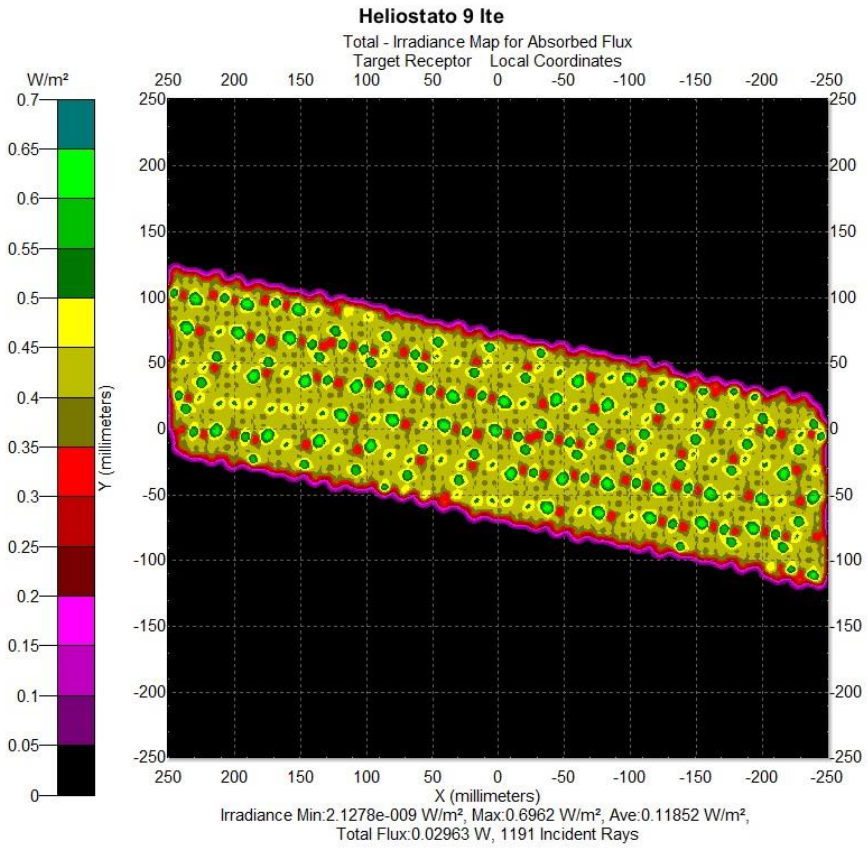


Figura 5.8: Mapa de irradiancia del heliostato 9 corregido con iteración.

elevada ($\eta > 0.9$), indicando que pequeños errores de instalación no afectan de manera crítica el rendimiento del campo.

A partir de 100 mm de desplazamiento en cualquier eje, la eficiencia comienza a decrecer notablemente, cayendo por debajo del 0.75 en la mayoría de los casos.

Cuando los heliostatos están desplazados en 500 mm o más en X y/o Y, la eficiencia sufre una disminución severa, llegando a valores inferiores a 0.4, lo que representa pérdidas significativas en la concentración solar.

Para desplazamientos extremos (1000 mm en X y/o Y), la eficiencia cae prácticamente a cero, indicando que en estos escenarios la radiación reflejada se desvía completamente del objetivo.

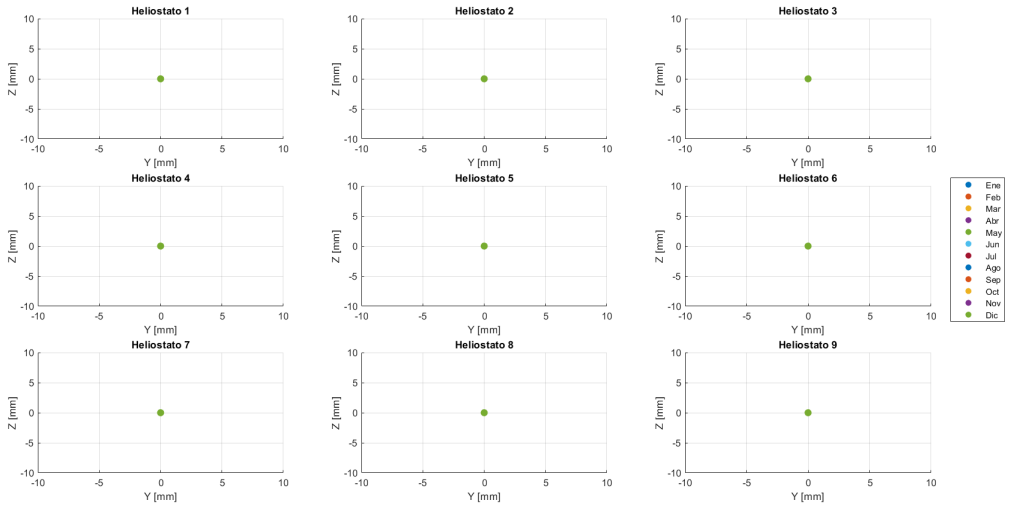


Figura 5.9: Apuntamiento del campo durante todo el año de cada heliostato con iteración.

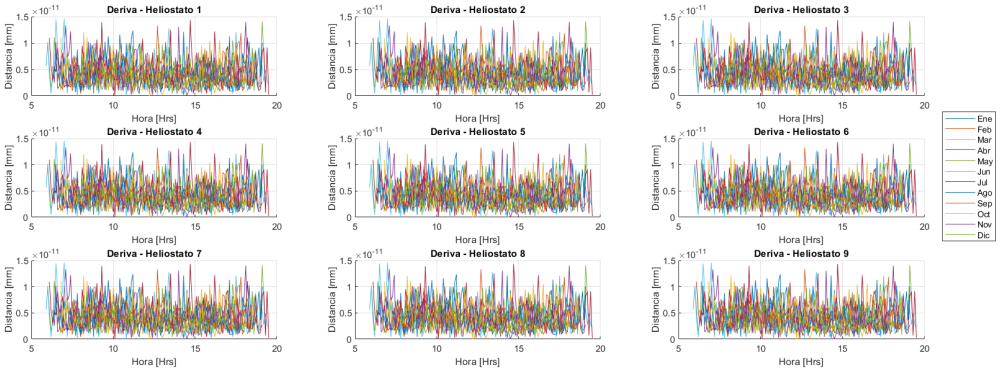


Figura 5.10: Deriva de campo durante todo el año con iteración.

Estos resultados reflejan la importancia de una instalación precisa del campo de heliostatos, ya que desplazamientos superiores a 100 mm pueden generar pérdidas de eficiencia considerables, afectando el rendimiento energético del sistema.

5.4.2 Análisis visual de los mapas de eficiencia

Las gráficas generadas proporcionan una representación visual del impacto del desfase en la eficiencia (Ver Figura 5.11). Se pueden identificar los siguientes patrones

Tabla 5.4: Eficiencia normalizada en función del desfase en X y Y

X,Y (mm)	**0**	**10**	**20**	**30**	**40**	**50**	**100**	**250**	**500**	**1000**
1000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0069	0.0423	0.0238
750	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0017	0.0600	0.0283	0.1217
500	0.0110	0.0125	0.0149	0.0189	0.0234	0.0290	0.0507	0.1216	0.1318	0.1980
400	0.0565	0.0595	0.0643	0.0687	0.0719	0.0758	0.1116	0.1851	0.1999	0.2234
350	0.1657	0.1675	0.1696	0.1733	0.1732	0.1790	0.2000	0.2525	0.2435	0.2328
300	0.3243	0.3259	0.3298	0.3348	0.3342	0.3351	0.3468	0.3409	0.3070	0.2497
250	0.5130	0.5125	0.5132	0.5122	0.5122	0.5135	0.5110	0.4480	0.3659	0.2819
200	0.7001	0.6957	0.6958	0.6939	0.6941	0.6921	0.6690	0.5574	0.4214	0.3106
100	0.9638	0.9539	0.9575	0.9540	0.9499	0.9439	0.8861	0.7201	0.4807	0.3127
50	0.9929	0.9900	0.9879	0.9813	0.9728	0.9612	0.9084	0.7484	0.4784	0.2998
0	1.0000	0.9997	0.9946	0.9885	0.9807	0.9710	0.9247	0.7596	0.4808	0.2955

clave:

- **Mapa de eficiencia en función del desfase en X**: La eficiencia se mantiene elevada en posiciones cercanas a la ubicación ideal ($X \approx 0mm$). Conforme el desplazamiento en X aumenta, se observa una transición progresiva de valores altos (verde) a valores bajos (rojo), con una caída drástica más allá de 250 mm.

- **Mapa de eficiencia en función del desfase en Y**: Presenta un comportamiento similar al caso del eje X, con una disminución gradual de la eficiencia conforme el desfase aumenta. Se observa que la caída de eficiencia es más pronunciada en desplazamientos mayores a 300 mm en Y.

- **Mapa tridimensional de eficiencia (X vs Y)**: Se aprecia una clara región de máxima eficiencia en el punto central (X = 0, Y = 0), la cual se reduce de manera simétrica conforme el desfase aumenta en cualquier dirección. La zona de baja eficiencia se expande en las regiones donde X y Y son superiores a 500 mm (Ver Figura 5.12).

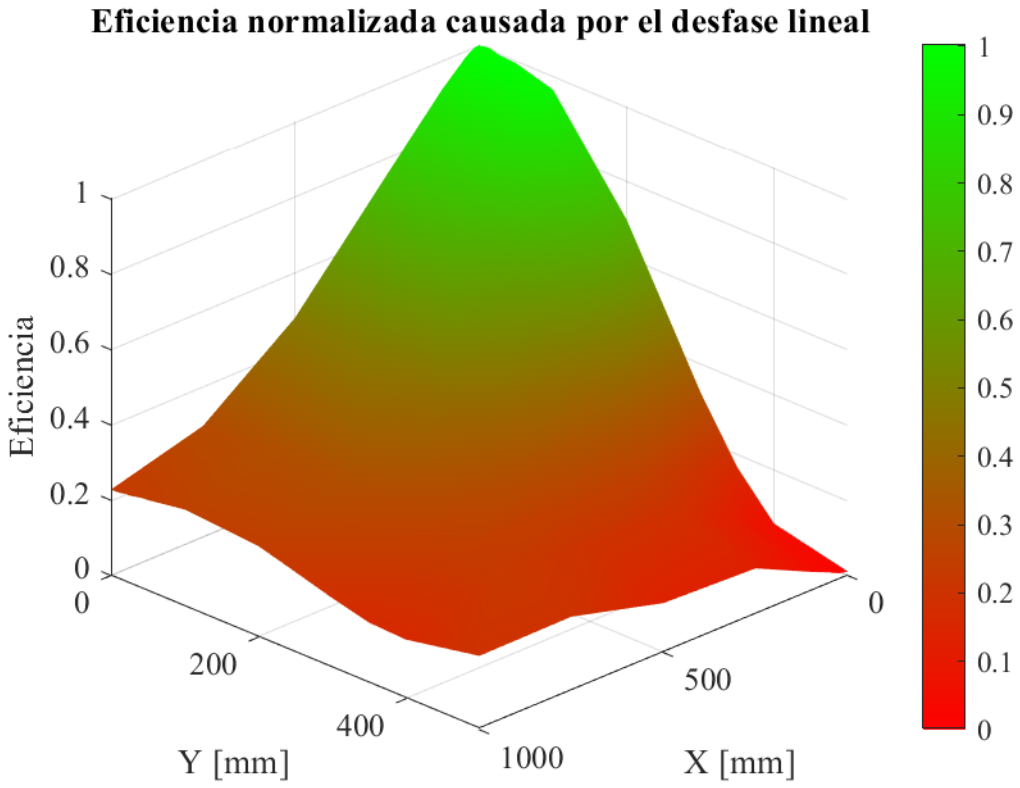


Figura 5.11: Resultados obtenidos en la simulación de desplazamiento lineal del campo 3X3.

5.5 Simulación de Rotación de campo 3X3

Los resultados obtenidos en la simulación realizada en MATLAB y TracePro demuestran una relación clara entre la eficiencia de concentración solar y la rotación del campo de heliostatos. La eficiencia, representada por el parámetro "η", alcanza su valor máximo cuando la rotación es de 0°, disminuyendo progresivamente a medida que el campo se inclina en cualquier dirección, ya sea positiva o negativa, tal como se observa en la Tabla 5.5.

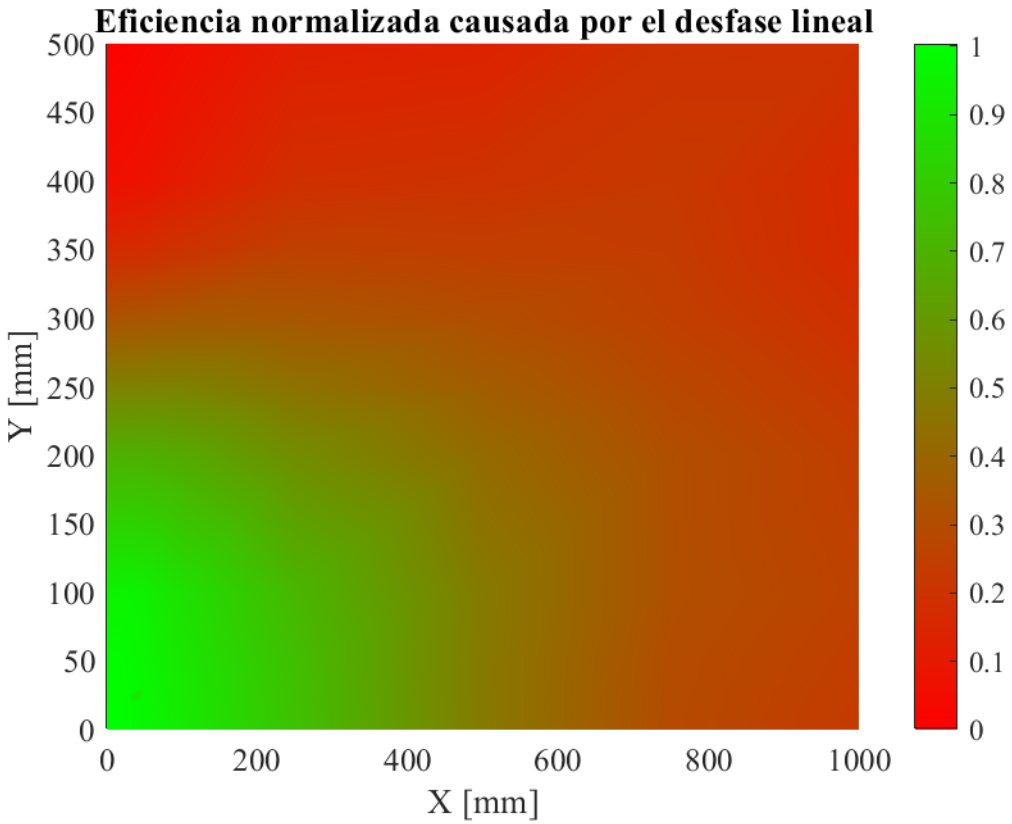


Figura 5.12: Resultados obtenidos en la simulación de desplazamiento lineal del campo 3X3 en vista aérea.

5.5.1 Impacto de la rotación en la eficiencia de concentración.

El comportamiento observado en la gráfica de la Figura 5.13 muestra una curva en forma de campana, donde el punto de máxima eficiencia, con un valor de $\eta = 1$, ocurre a 0° de rotación. Este resultado indica que, en la configuración ideal, la energía solar reflejada por los heliostatos es dirigida con la máxima precisión hacia el objetivo. No obstante, conforme el campo de heliostatos se desvía de esta orientación, la eficiencia disminuye debido a la pérdida de alineación óptica en la reflexión de la radiación solar.

Al analizar los datos numéricos, se observa que:

Tabla 5.5: Valores de azimut (Az), energía total (E Tot), eficiencia (η) y número de rayos (#Rayos).

Az	**E Tot [W]**	η	**#Rayos**
-2.25	15.128	0.3513	186624
-2.00	17.543	0.4073	186624
-1.75	20.72	0.4811	186624
-1.50	24.83	0.5765	186624
-1.25	29.48	0.6845	186624
-1.00	34.257	0.7954	186624
-0.75	38.382	0.8912	186624
-0.50	41.365	0.9605	186624
-0.25	42.827	0.9944	186624
0.00	43.067	1	186624
0.25	43.058	0.9998	186624
0.25	43.058	0.9998	186624
0.50	41.612	0.9662	186624
0.75	38.775	0.9003	186624
1.00	34.37	0.7981	186624
1.25	29.47	0.6843	186624
1.50	24.813	0.5761	186624
1.75	20.69	0.4804	186624
2.00	17.638	0.4095	186624
2.25	15.117	0.3510	186624

Para rotaciones menores a $\pm 0.5^\circ$, la eficiencia se mantiene alta ($\eta > 0.96$), lo que sugiere que pequeñas desviaciones no comprometen significativamente el rendimiento del sistema.

A $\pm 1^\circ$ de rotación, la eficiencia cae aproximadamente un 20 %, alcanzando

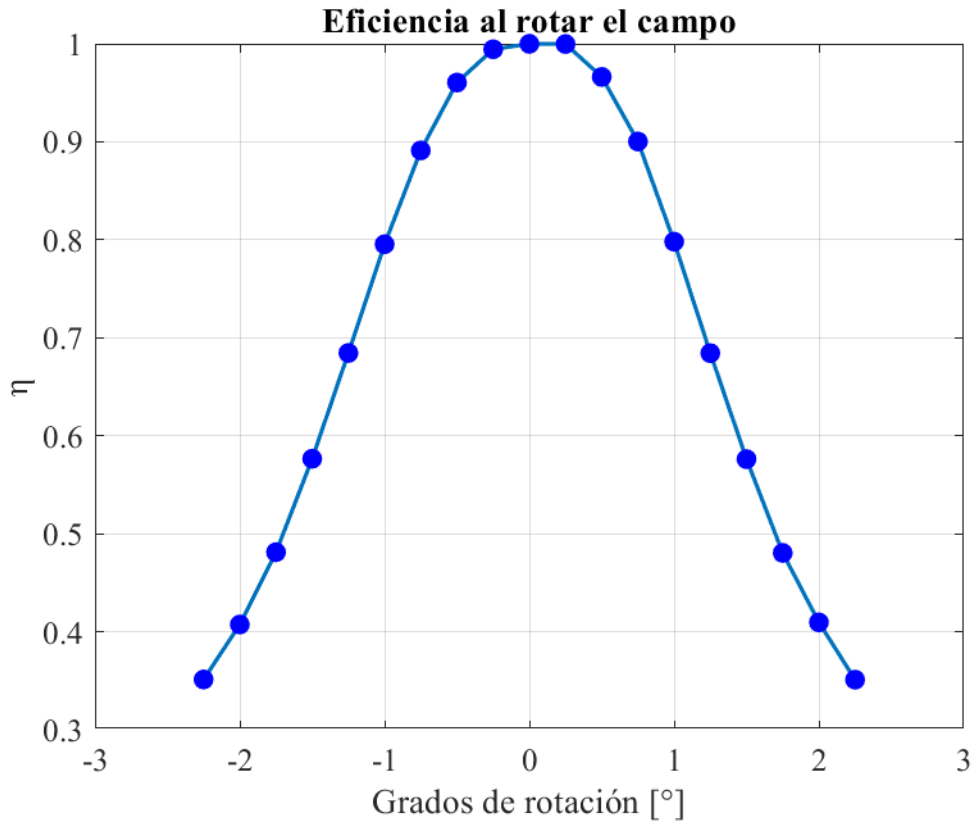


Figura 5.13: Resultados obtenidos en la simulación de rotación del campo 3X3.

valores cercanos a $\eta = 0.80$, lo que indica un impacto moderado en la concentración de la radiación solar.

Para desviaciones de $\pm 2°$ o mayores, la eficiencia cae por debajo del 50 %, y con una rotación de $\pm 2.25°$, la eficiencia es apenas de 0.35, evidenciando pérdidas significativas en la concentración de energía.

Este comportamiento resalta la importancia de un control preciso en la instalación y mantenimiento de campos solares de torre central. En plantas a gran escala, incluso un pequeño error en la rotación del campo puede traducirse en pérdidas energéticas significativas, afectando directamente la eficiencia y el rendimiento económico del sistema.

Capítulo 6

Conclusiones y trabajos futuros

En este capítulo, se presentan las conclusiones del trabajo, destacando las aportaciones realizadas al campo de los sistemas termosolares y proponiendo áreas de oportunidad para futuras investigaciones. Este trabajo se centró en el diseño e implementación de un algoritmo de control para sistemas de heliostatos autónomos, con el propósito de aumentar su desempeño técnico y económico en aplicaciones de concentración solar de torre central. Durante el estudio, se identificaron los principales desafíos asociados a la operación de estos sistemas, y se pretendió abordar la lógica de control proponiendo soluciones innovadoras que madecuados con la precisión en el seguimiento solar reducen los costos operativos y de implementación.

- Las pruebas realizadas validaron la capacidad del algoritmo para redirigir el haz solar con alta precisión hacia un punto fijo ante cambios en la posición solar. Además, se demostró que el microcontrolador ESP32 puede ejecutar cálculos equivalentes en precisión y confiabilidad a los realizados con MATLAB, consolidándolo como una opción viable para sistemas de bajo costo y reducir el desbordamiento alcanzando el punto objetivo del receptor.

- El análisis de los resultados de la simulación de desfase lineal confirmó que la eficiencia de los sistemas de concentración solar convencionales es altamente sensible a los errores de posicionamiento en X y Y. Para errores menores a 50

mm, las pérdidas de eficiencia son despreciables, lo que sugiere que el sistema tolera pequeñas desviaciones sin comprometer el rendimiento. Sin embargo, a partir de 100 mm de desfase, la eficiencia se ve comprometida de manera significativa, subrayando la importancia de una instalación precisa y alineada del campo solar. En casos donde los errores de posicionamiento alcanzan los 500 mm o más, la eficiencia sufre una caída drástica, lo que evidencia que en estos escenarios la radiación reflejada no impacta correctamente en el receptor, generando una reducción considerable en la concentración de energía.

Además, las gráficas tridimensionales muestran que el impacto del error es acumulativo en ambas direcciones (X e Y), lo que implica que errores combinados en ambos ejes aceleran aún más la disminución de la eficiencia. En términos de aplicación práctica, estos resultados resaltan la necesidad de procedimientos de calibración rigurosos durante la instalación y mantenimiento del campo de heliostatos. Asimismo, se sugiere la implementación de estrategias de corrección en tiempo real para minimizar los efectos del desfase y optimizar la eficiencia del sistema de concentración solar.

- El análisis de los datos obtenidos en la simulación de rotación del campo confirma que la eficiencia de concentración en un campo de heliostatos es altamente sensible a la rotación del sistema. Mientras que pequeñas desviaciones de hasta 0.5° no representan un problema crítico, errores superiores a 1° comienzan a reducir significativamente la eficiencia de captación de energía, y rotaciones de 2° o más pueden generar pérdidas de hasta el 65 %. Estos hallazgos subrayan la necesidad de implementar estrategias de control y mantenimiento que minimicen la rotación no deseada del campo de heliostatos, asegurando así un óptimo desempeño de las plantas solares de torre central. Además, el estudio proporciona una base cuantitativa para futuras investigaciones que busquen optimizar el diseño y la operación de estos sistemas en escenarios reales.

- La propuesta de implementación del algoritmo con iteración desarrollado para el control de heliostatos autónomos permitiría mejorar hasta en rangos de

$1x10^{-11}mm$ la precisión en el seguimiento solar. Este avance representa un paso significativo hacia la mejora en el afinamiento de los sistemas termosolares de torre central.

- La investigación bibliográfica realizada en la Sección 2.6 evidencía que errores como la desalineación de espejos, deformaciones estructurales y fallos en el cálculo de la posición solar afectan de manera significativa la eficiencia del sistema. Las estrategias propuestas en este trabajo contribuyen a mitigar dichos errores mediante un diseño robusto y algoritmos adaptativos.

- El desarrollo e implementación del gnomon para estimar la posición solar a través del análisis de sombras, demostró ser un enfoque económico en comparación con sistemas de seguimiento solar más complejos ya que al implementar este sistema se puede prescindir de estudios de geolocalización topográfica en la etapa de implementación del estos sistemas de torre central. El prototipo de gnomon, ofreció una solución innovadora para la adquisición de datos solares en tiempo real, integrando de manera efectiva hardware y software en un sistema robusto.

- Las simulaciones y experimentos realizados resaltan la necesidad de seguir innovando en áreas como sensores de alta precisión y algoritmos optimizados, con el objetivo de alcanzar mayores niveles de precisión operativa.

6.1 Trabajos futuros

6.1.1 Implementación autónoma del sistema de adquisición del gnomon en ESP32

Con el objetivo de reducir costos operativos y aumentar la accesibilidad tecnológica del sistema propuesto, se sugiere migrar el proceso de adquisición y análisis

de datos del gnomon, actualmente implementado en MATLAB, a un entorno embebido usando otro microcontrolador ESP32. Esto permitiría la obtención de los datos solares en tiempo real de manera autónoma y económica, sin necesidad de depender de software propietario o equipo adicional complejo. Además, abre la posibilidad de implementar soluciones IoT, facilitando la transmisión remota y gestión integral de los datos solares.

6.1.2 Integración del sistema en redes de helióstatos

Se propone explorar la escalabilidad del sistema desarrollado para su implementación en campos solares de gran escala. Esto implicaría investigar y evaluar técnicas avanzadas de comunicación y sincronización entre múltiples unidades de helióstatos para optimizar la eficiencia global del campo solar. Se recomienda analizar diferentes protocolos de comunicación inalámbrica y alámbrica que permitan una rápida respuesta y coordinación eficiente ante variaciones climáticas y de irradiancia solar, así como implementar algoritmos colaborativos que faciliten ajustes colectivos en tiempo real para maximizar la captación solar del sistema completo.

6.1.3 Análisis de durabilidad y mantenimiento predictivo

Se propone realizar un estudio integral centrado en la resistencia y comportamiento de los materiales y componentes empleados en el heliostato bajo condiciones reales de operación prolongada, incluyendo ciclos térmicos, exposición constante al sol, humedad y polvo. Complementariamente, se recomienda el desarrollo e implementación de técnicas predictivas de mantenimiento, utilizando algoritmos avanzados que analicen datos históricos y actuales del rendimiento del heliostato para anticipar fallos o desgaste en componentes críticos. Esto permitirá optimizar los

costos operativos y mejorar la confiabilidad a largo plazo del sistema.

Apéndice A

En este apéndice se presenta el código fuente de la función `angle_to_vector.m`.

```
 1 function V = angle_to_vector(az,al,fix)
 2 % angle_to_vector - Convierte una posición angular (acimut y altura)
 3 %                   a un vector 3D.
 4 % Sintaxis:
 5 %   V = angle_to_vector(az, al)
 6 %   V = angle_to_vector(az, al, fix)
 7 %
 8 % Entradas:
 9 %   az  - Ángulo de acimut en grados
10 %   al  - Ángulo de altura (elevación) solar en grados
11 %   fix - (opcional) Valor lógico que indica si se corrige
12 %         el eje Y (default = 0)
13 %
14 % Salida:
15 %   V   - Vector unitario [x, y, z] correspondiente a la dirección
16 %         definida por az y al
17 %
18 % Descripción:
19 %   Esta función toma los ángulos de acimut y altura solar y los
20 %   convierte en un vector tridimensional. Si se activa la corrección
21 %   con 'fix', se invierte el signo del eje Y, útil cuando el sistema
22 %   de referencia requiere un ajuste específico (por ejemplo,
23 %   conversión entre sistemas de coordenadas).
24
25 if nargin < 3
26      fix = 0;
27  end
28
29 V = [cosd(al)*cosd(az), cosd(al)*sind(az), sind(al)];
30 if fix
31      V = V .* [1, -1, 1];
32  end
33
34  end
```

Apéndice B

En este apéndice se presenta el código fuente de la función `vector_to_angle.m`.

```
function [az,al] = vector_to_angle(V,fix,flag)
% vector_to_angle - Convierte un vector tridimensional a coordenadas
% angulares (acimut y altura).
%
% Sintaxis:
%   [az, al] = vector_to_angle(V)
%   [az, al] = vector_to_angle(V, fix)
%   [az, al] = vector_to_angle(V, fix, flag)
%
% Entradas:
%   V     - Vector tridimensional [x, y, z]
%   fix   - (opcional) Valor lógico que indica si se ajusta el acimut
%           al cuadrante correcto (default = 0)
%   flag  - (opcional) Valor lógico que indica si se imprime el
%           cuadrante por consola (default = 0)
%
% Salidas:
%   az    - Ángulo de acimut en grados
%   al    - Ángulo de altura (elevación) en grados
%
% Descripción:
%   Esta función recibe un vector tridimensional y calcula sus
%   componentes angulares:
%   el acimut (az) y la altura (al). Si se habilita 'fix', se
%   realiza una corrección del ángulo de acimut de acuerdo al
%   cuadrante en el que se encuentra el vector en el plano XY.
%   Si 'flag' está activado, se muestra en consola el cuadrante
%   correspondiente.

% Asignar valores por defecto si no se especifican
if nargin < 3
    flag = 0;
end
if nargin < 2
    fix = 0;
end

% Normalizar el vector
V = V / norm(V);

% Calcular ángulo de elevación
al = asind(V(3));

% Calcular ángulo de acimut base
az = atand(V(2)/V(1));

```

```
47 % Corrección de acimut según el cuadrante (si se habilita 'fix')
48 if fix
49     az = abs(az); % Usar valor absoluto como base
50 
51     % Identificar cuadrante y ajustar el valor de azimut
52     if V(1) >= 0 && V(2) >= 0     % I cuadrante
53         if flag, disp('I cuadrante, azimut [-90 , 0]'); end
54         az = -az;
55     elseif V(1) >= 0 && V(2) <= 0 % II cuadrante
56         if flag, disp('II cuadrante, azimut [0 , 90]'); end
57         az = az;
58     elseif V(1) <= 0 && V(2) <= 0 % III cuadrante
59         if flag, disp('III cuadrante, azimut [90 , 180]'); end
60         az = (180 - az);
61     elseif V(1) <= 0 && V(2) >= 0 % IV cuadrante
62         if flag, disp('IV cuadrante, azimut [-180 , -90]'); end
63         az = -(180 - az);
64     end
65 end
66 
67 end
68 
```

Apéndice C

En este apéndice se presenta el código fuente de la función `Heliostato.ino`.

~\.platformio\Heliostato.cpp

```cpp
#include <ESP32Servo.h>

Servo ServoAz;  // Objeto servo para el movimiento azimutal (horizontal)
Servo ServoCe;  // Objeto servo para el movimiento cenital (vertical)

const int JoyAz = 34;       // Pin analógico para el joystick azimutal
const int pinServoAz = 18;  // Pin digital para el servo azimutal
const int JoyCe = 35;       // Pin analógico para el joystick cenital
const int pinServoCe = 19;  // Pin digital para el servo cenital

int valorJoyAz = 0;      // Valor leído del joystick azimutal (0-4095)
float anguloServoAz = 90; // Ángulo inicial del servo azimutal (grados)
int valorJoyCe = 0;      // Valor leído del joystick cenital (0-4095)
float anguloServoCe = 45; // Ángulo inicial del servo cenital (grados)

int save_r = 32; // Pin digital para el botón de guardar referencia (R)
int btn;         // Estado del botón pulsador
int opcion = 1;  // Estado actual del programa (1: Home, 2: Calibración, 3: Cálculo)

const int pinRojo = 33;   // Pin digital para el LED rojo
const int pinVerde = 25;  // Pin digital para el LED verde
const int pinAzul = 26;   // Pin digital para el LED azul
const int push_b = 27;    // Pin digital para el botón pulsador de cambio de modo

float az, al, vx, vy, vz, k; // Variables para cálculos vectoriales y angulares
const float Pi = 3.14159;    // Valor de Pi para conversiones de ángulos

float* R = nullptr; // Vector de referencia (R) para la calibración
float* N = nullptr; // Vector normal (N) calculado

// Declaración de funciones
float* vector_to_angle(float V[]); // Convierte un vector 3D a ángulos azimutal y cenital
float* angle_to_vector(float az, float al); // Convierte ángulos azimutal y cenital a un vector 3D
float Dot_P(float v1[], float v2[]); // Calcula el producto punto de dos vectores
float* Calibration_R(float N[], float S[]); // Calcula el vector de referencia (R)
float* Calculation_N(float R[], float S[]); // Calcula el vector normal (N)
float* Sum_Vec(float v1[], float v2[]); // Suma dos vectores
float* Esc_Vec(float v1[], float k); // Escala un vector por un factor k
float norm(float S[]); // Calcula la norma (magnitud) de un vector
float* Vec(float v1, float v2, float v3); // Crea un vector 3D a partir de tres componentes
void RGB(int rojo, int verde, int azul); // Controla el LED RGB
```

```
void freeVector(float* V); // Libera la memoria asignada a un vector

void setup() {
  Serial.begin(115200);
  pinMode(pinRojo, OUTPUT);
  pinMode(pinVerde, OUTPUT);
  pinMode(pinAzul, OUTPUT);
  pinMode(push_b, INPUT_PULLDOWN);
  pinMode(save_r, INPUT_PULLUP);
  ServoAz.attach(pinServoAz);
  ServoAz.write(anguloServoAz);
  ServoCe.attach(pinServoCe);
  ServoCe.write(anguloServoCe);
}

void loop() {
  btn = digitalRead(push_b); // Lee el estado del botón pulsador
  int v_state = digitalRead(save_r); // Lee el estado del botón de guardar referencia

  switch (opcion) {
    case 1: { // Modo Home: Posición inicial o de reposo
      RGB(255, 0, 255); // LED RGB: Morado parpadeando (indica modo Home)
      Serial.println("Home");
      if (btn == HIGH) { // Si se presiona el botón pulsador
        while (btn == HIGH) { // Espera a que se suelte el botón
          btn = digitalRead(push_b);
        }
        opcion = 2; // Cambia al modo de calibración
      }
      break;
    }

    case 2: { // Modo Calibración: Ajuste manual de la referencia (R)
      RGB(0, 0, 255); // LED RGB: Azul parpadeando (indica modo Calibración)
      Serial.print("Calibration Mode \t");
      Serial.print("Presione el botón una vez tenga direccionado R en el Heliostato \t");
      valorJoyAz = analogRead(JoyAz); // Lee el valor del joystick azimutal
      valorJoyCe = analogRead(JoyCe); // Lee el valor del joystick cenital
      float Azh = 90 - anguloServoAz; // Calcula el ángulo azimutal del heliostato
      float Ceh = anguloServoCe;      // Calcula el ángulo cenital del heliostato
      Serial.print(Azh);
      Serial.print(",");
      Serial.println(Ceh);
      float cambioAz = (valorJoyAz == 4095) ? 1 : (valorJoyAz == 0) ? -1 : 0; // Calcula el cambio en el ángulo azimutal
```

```
      float cambioCe = (valorJoyCe == 4095) ? 1 : (valorJoyCe == 0) ? -1 : 0; // Calcula el cambio en el ángulo cenital
      anguloServoAz += cambioAz; // Actualiza el ángulo azimutal
      anguloServoCe += cambioCe; // Actualiza el ángulo cenital
      anguloServoAz = constrain(anguloServoAz, 0, 180); // Limita el ángulo azimutal
      anguloServoCe = constrain(anguloServoCe, 0, 90);  // Limita el ángulo cenital
      ServoAz.write(anguloServoAz); // Mueve el servo azimutal
      ServoCe.write(anguloServoCe); // Mueve el servo cenital
      if (v_state == LOW) { // Si se presiona el botón de guardar referencia
        N = angle_to_vector(Azh, Ceh); // Convierte los ángulos del heliostato a un vector normal (N)
        if (N == nullptr) {
          Serial.println("Error: No se pudo asignar memoria para N.");
          break;
        }
        Serial.println("Introduzca los valores para S en formato Azs,Ces:");
        while (Serial.available() == 0) {} // Espera a recibir datos por el puerto serial
        if (Serial.available() > 0) {
          String input = Serial.readStringUntil('\n'); // Lee la entrada del puerto serial
          int commaIndex = input.indexOf(',');
          if (commaIndex != -1) {
            String AzsStr = input.substring(0, commaIndex);
            String CesStr = input.substring(commaIndex + 1);
            float Azs = AzsStr.toFloat();
            float Ces = CesStr.toFloat();
            float* S = Esc_Vec(angle_to_vector(Azs, Ces), -1); // Convierte los ángulos del sol a un vector S (inverso)
            if (S == nullptr) {
              Serial.println("Error: No se pudo asignar memoria para S.");
              freeVector(N);
              break;
            }
            Serial.print("S=[ ");
            Serial.print(S[0], 6);
            Serial.print(", ");
            Serial.print(S[1], 6);
            Serial.print(", ");
            Serial.print(S[2], 6);
            Serial.println("]");
            R = Calibration_R(N, S); // Calcula el vector de referencia (R)
            if (R == nullptr) {
              Serial.println("Error: No se pudo asignar memoria para R.");
              freeVector(N);
              freeVector(S);
              break;
            }
```

```
            Serial.print("R=[ ");
            Serial.print(R[0], 6);
            Serial.print(", ");
            Serial.print(R[1], 6);
            Serial.print(", ");
            Serial.print(R[2], 6);
            Serial.println("]");
            freeVector(S); // Libera la memoria de S
          } else {
            Serial.println("Error: Entrada inválida. Asegúrese de que esté en el formato correcto 'Azs,Ces'.");
          }
          freeVector(N); // Libera la memoria de N
        }
        delay(5000); // Espera 5 segundos
      }
      if (btn == 1) { // Si se presiona el botón pulsador
        while (btn == 1) {
          btn = digitalRead(push_b);
        }
        opcion = 3; // Cambia al modo de cálculo
      }
    }
    break;

    case 3: { // Modo Cálculo: Cálculo de la posición del heliostato
      RGB(0, 255, 0); // LED RGB: Verde parpadeando (indica modo Cálculo)
      Serial.println("Calculation Mode");
      Serial.println("Introduzca los valores para S en formato Azs,Ces:");
      while (Serial.available() == 0) {
        btn = digitalRead(push_b);
        if (btn == HIGH) {
          opcion = 1;
          break;
        }
      }
      if (Serial.available() > 0) {
        String input = Serial.readStringUntil('\n');
        int commaIndex = input.indexOf(',');
        if (commaIndex != -1) {
          String AzsStr = input.substring(0, commaIndex);
          String CesStr = input.substring(commaIndex + 1);
          float Azs = AzsStr.toFloat();
          float Ces = CesStr.toFloat();
          Serial.print("S=[ ");
          Serial.print(Azs, 6);
          Serial.print(", ");
          Serial.print(Ces, 6);
          Serial.println("]");
          float* S = Esc_Vec(angle_to_vector(Azs, Ces), -1); // Convierte los ángulos del sol a un vector S (inverso)
```

```
178           if (S == nullptr) {
179             Serial.println("Error: No se pudo asignar memoria para S.");
180             break;
181           }
182           Serial.print("S=[ ");
183           Serial.print(S[0], 6);
184           Serial.print(", ");
185           Serial.print(S[1], 6);
186           Serial.print(", ");
187           Serial.print(S[2], 6);
188           Serial.println("]");
189           N = Calculation_N(R, S); // Calcula el vector normal (N)
190           if (N == nullptr) {
191             Serial.println("Error: No se pudo asignar memoria para N.");
192             freeVector(S);
193             break;
194           }
195           Serial.print("N=[ ");
196           Serial.print(N[0], 6);
197           Serial.print(", ");
198           Serial.print(N[1], 6);
199           Serial.print(", ");
200           Serial.print(N[2], 6);
201           Serial.println("]");
202           float* Nh = vector_to_angle(N); // Convierte el vector normal (N)
a ángulos
203           if (Nh != nullptr) {
204             if (isnan(N[0])) {
205               Serial.println("xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx El
vector N es Nan xxxxxxxxxxxxxxxxxxxxxxxxxxxxx");
206               Serial.print("R=[ ");
207               Serial.print(R[0], 6);
208               Serial.print(", ");
209               Serial.print(R[1], 6);
210               Serial.print(", ");
211               Serial.print(R[2], 6);
212               Serial.println("]");
213               break;
214             }
215             anguloServoAz = Nh[0]; // Actualiza el ángulo azimutal del
servo
216             anguloServoCe = Nh[1]; // Actualiza el ángulo cenital del servo
217             Serial.print("N=[ ");
218             Serial.print(anguloServoAz, 6);
219             Serial.print(", ");
220             Serial.print(anguloServoCe, 6);
221             Serial.println("]");
222             ServoAz.write(90 - anguloServoAz); // Mueve el servo azimutal
223             ServoCe.write(anguloServoCe);      // Mueve el servo cenital
224             freeVector(S);                     // Libera la memoria de S
225             freeVector(N);                     // Libera la memoria de N
```

```
          } else {
            Serial.println("Error: No se pudo convertir N a ángulos.");
          }
        } else {
          Serial.println("Error: Entrada inválida. Asegúrese de que esté en el formato correcto 'Azs,Ces'.");
        }
      }
      if (btn == HIGH) {
        while (btn == HIGH) {
          btn = digitalRead(push_b);
        }
        opcion = 1;
      }
      break;
    }

    default: { // Modo Error: Estado desconocido
      Serial.println("Error");
      RGB(255, 0, 0); // LED RGB: Rojo parpadeando (indica modo Error)
      delay(1000);
      break;
    }
  }
  delay(100); // Espera 100 ms antes de repetir el bucle
}

void RGB(int rojo, int verde, int azul) {
  analogWrite(pinRojo, rojo);
  analogWrite(pinVerde, verde);
  analogWrite(pinAzul, azul);
  delay(50);
  analogWrite(pinRojo, 0);
  analogWrite(pinVerde, 0);
  analogWrite(pinAzul, 0);
}

// Convierte un vector 3D a ángulos azimutal y cenital
float* vector_to_angle(float V[]) {
  float* Angle = new float[2];
  if (Angle == nullptr) return nullptr;
  vx = V[0];
  vy = V[1];
  vz = V[2];
  Angle[0] = atan2(vy, vx) * (180 / Pi); // azimut (ángulo horizontal)
  Angle[1] = asin(vz / sqrt(vx * vx + vy * vy + vz * vz)) * (180 / Pi); // elevación (ángulo vertical)
  if (vy == 0 && vx == 0) {
    Angle[1] = 90; // Si el vector es vertical, la elevación es 90 grados
  }

```

```
  Angle[0] = fabs(Angle[0]);
  if (V[0] >= 0 && V[1] >= 0) {
    Angle[0] = -Angle[0];
  } else if (V[0] >= 0 && V[1] <= 0) {
    Angle[0] = Angle[0];
  } else if (V[0] <= 0 && V[1] <= 0) {
    Angle[0] = 180 - Angle[0];
  } else if (V[0] <= 0 && V[1] >= 0) {
    Angle[0] = -(180 - Angle[0]);
  }

  return Angle;
}

// Convierte ángulos azimutal y cenital a un vector 3D
float* angle_to_vector(float az, float al) {
  float* V = new float[3];
  if (V == nullptr) return nullptr;
  V[0] = cos(al * (Pi / 180)) * cos(az * (Pi / 180));
  V[1] = -cos(al * (Pi / 180)) * sin(az * (Pi / 180));
  V[2] = sin(al * (Pi / 180));
  return V;
}

// Calcula el producto punto de dos vectores
float Dot_P(float v1[], float v2[]) {
  float resultado = 0.0;
  for (int i = 0; i < 3; i++) {
    resultado += v1[i] * v2[i];
  }
  return resultado;
}

// Calcula el vector de referencia (R)
float* Calibration_R(float N[], float S[]) {
  float* R = new float[3];
  if (R == nullptr) return nullptr;
  float* N_unit = Esc_Vec(N, 1 / norm(N)); // Normaliza N
  float* S_unit = Esc_Vec(S, 1 / norm(S)); // Normaliza S
  R = Sum_Vec(Esc_Vec(N_unit, 2 * (Dot_P(N_unit, Esc_Vec(S_unit, -1)))),
S_unit); // Calcula R
  delete[] N_unit; // Libera la memoria de N_unit
  delete[] S_unit; // Libera la memoria de S_unit
  return R;
}

// Calcula el vector normal (N)
float* Calculation_N(float R[], float S[]) {
  float* N = new float[3];
  if (N == nullptr) return nullptr;
  float* R_unit = Esc_Vec(R, 1 / norm(R)); // Normaliza R
```

```
  float* S_unit = Esc_Vec(S, 1 / norm(S)); // Normaliza S
  N = Esc_Vec(Sum_Vec(R_unit, Esc_Vec(S_unit, -1)), 1 /
norm(Sum_Vec(R_unit, Esc_Vec(S_unit, -1)))); // Calcula N
  delete[] R_unit; // Libera la memoria de R_unit
  delete[] S_unit; // Libera la memoria de S_unit
  return N;
}

// Suma dos vectores
float* Sum_Vec(float v1[], float v2[]) {
  float* resultado = new float[3];
  if (resultado == nullptr) return nullptr;
  for (int i = 0; i < 3; i++) {
    resultado[i] = v1[i] + v2[i];
  }
  return resultado;
}

// Escala un vector por un factor k
float* Esc_Vec(float v1[], float k) {
  float* resultado = new float[3];
  if (resultado == nullptr) return nullptr;
  for (int i = 0; i < 3; i++) {
    resultado[i] = v1[i] * k;
  }
  return resultado;
}

// Calcula la norma (magnitud) de un vector
float norm(float S[]) {
  return sqrt(Dot_P(S, S));
}

// Crea un vector 3D a partir de tres componentes
float* Vec(float v1, float v2, float v3) {
  float* resultado = new float[3];
  if (resultado == nullptr) return nullptr;
  resultado[0] = v1;
  resultado[1] = v2;
  resultado[2] = v3;
  return resultado;
}

// Libera la memoria asignada a un vector
void freeVector(float* V) {
  if (V != nullptr) {
    delete[] V;
  }
}
```

Apéndice D

En este apéndice se presenta el código fuente de la función `Gnomon.m`.

```
%% INICIO

% Programa principal para ejecutar el proceso completo de
% adquisición,procesamiento y análisis de imágenes con
% detección de colores y cálculo de sombra solar.
% Versión con loop del 07-ago-24.

if exist('Resultados','var') == 0
    clear;
    clc;
    % Directorio para guardar los resultados
    foldersave = 'Pruebas_Job_SolarTracker_LOOP';
    mkdir(foldersave);  % Crear carpeta si no existe
    % Inicializar vectores de resultados
    Resultados_time = ['Timestamp          '];
    Resultados = [999 999];
end

%% Captura de imagen desde la cámara

% Crear objeto de cámara (se usa la última detectada)
cameras = webcamlist;
cam = webcam(cameras{length(cameras)});

pause(3);  % Esperar a que estabilice la imagen
imagen = snapshot(cam);  % Capturar imagen
imshow(imagen);  % Mostrar imagen capturada
clear cam;  % Liberar cámara

% Crear nombre de archivo con timestamp
timestamp = char(datetime('now', 'Format', 'yyyy-MM-dd HH.mm.ss'));
filename = fullfile([pwd, '/', foldersave, '/', timestamp]);

% Guardar imagen capturada
imwrite(imagen, [filename, '.jpg'], 'Quality', 100);

pause(3);
close;

%% Opción: seleccionar imagen manualmente (no activa por defecto)
if(0)
    clear;
    clc;
    pause(3);
    [filename, pathname] = uigetfile({'*.jpg;*.png;*.tif;*.bmp', ...
        'All Image Files'; '*.*','All Files'}, 'Seleccione una imagen');
    fullpath = fullfile(pathname, filename);
    imagen = imread(fullpath);  % Leer imagen seleccionada
end

%% Conversión de imagen a HSV
```

```
52
53 hsv_image2 = rgb2hsv(imagen);  % Convertir a espacio de color HSV
54
55 % Definir umbrales para cada color a detectar (HSV)
56 thresholds = {
57  'Rojo', [0.8966, 0.9966; 0.4435, 1; 0.4412, 1]  % Ajustado para #8a3f5e
58  'Verde', [0.308, 0.408; 0.657, 1; 0.19, 1];     % Ajustado para #124a19
59  'Azul', [0.588, 0.688; 0.572, 1; 0.32, 1];      % Ajustado para #232f6b
60  'Amarillo', [0.1, 0.2; 0.5, 1; 0.5, 1];         % Genérico
61 };
62
63 % Colores RGB correspondientes para visualización
64 color_values = {
65     [1, 0, 0];    % Rojo
66     [0, 1, 0];    % Verde
67     [0, 0, 1];    % Azul
68     [1, 1, 0];    % Amarillo
69 };
70
71 % Inicializar coordenadas
72 coordinates = zeros(4, 2);
73
74 % Mostrar imagen original con detección de colores
75 figure; imshow(imagen); hold on;
76
77 % Iterar sobre los colores definidos
78 for i = 1:size(thresholds, 1)
79     color_name = thresholds{i, 1};
80     threshold = thresholds{i, 2};
81     color_value = color_values{i};
82
83     % Crear máscara binaria del color actual
84     mask = (hsv_image2(:,:,1) >= threshold(1,1) & hsv_image2(:,:,1)  ...
85         <= threshold(1,2)) &  ...
86            (hsv_image2(:,:,2) >= threshold(2,1) & hsv_image2(:,:,2)  ...
87            <= threshold(2,2)) &  ...
88            (hsv_image2(:,:,3) >= threshold(3,1) & hsv_image2(:,:,3)  ...
89            <= threshold(3,2));
90
91     % Obtener coordenadas del centro del color detectado
92     [y, x] = find(mask);
93     if isempty(y)
94         warning(['No se encontró ningún punto ', color_name, ...
95             ' en la imagen.']);
96         continue;
97     end
98     x_center = mean(x);
99     y_center = mean(y);
100    coordinates(i, :) = [x_center, y_center];
101
102    % Mostrar en imagen
```

```
    plot(x_center, y_center, 'o', 'MarkerSize', 5, 'MarkerEdgeColor', ...
        color_value, 'MarkerFaceColor', color_value);
    text(x_center + 5, y_center, color_name, 'Color', color_value, ...
        'FontSize', 10);
end

%% Corrección de perspectiva

close;
puntos_origen = coordinates;
puntos_destino = [0, 0; 500, 0; 500, 500; 0, 500];   % Coordenadas destino

% Ajustar perspectiva usando transformación proyectiva
tform = fitgeotrans(puntos_origen, puntos_destino, 'projective');
imagen_transformada = imwarp(imagen, tform);

% Mostrar imagen corregida
figure;
imshow(imagen_transformada);
title('Imagen con Perspectiva Corregida');
imwrite(imagen_transformada, 'imagen_procesada_2.jpg');

## (continúa en siguiente bloque por límite de tokens)
%% Detección de esquinas en imagen corregida

close;
hsv_image2 = rgb2hsv(imagen_transformada);
corners = zeros(4, 2);

figure; imshow(imagen_transformada); hold on;

for i = 1:size(thresholds, 1)
    color_name = thresholds{i, 1};
    threshold = thresholds{i, 2};
    color_value = color_values{i};

    mask = (hsv_image2(:,:,1) >= threshold(1,1) & hsv_image2(:,:,1) ...
        <= threshold(1,2)) & ...
           (hsv_image2(:,:,2) >= threshold(2,1) & hsv_image2(:,:,2) ...
           <= threshold(2,2)) & ...
           (hsv_image2(:,:,3) >= threshold(3,1) & hsv_image2(:,:,3) ...
           <= threshold(3,2));

    [y, x] = find(mask);
    if isempty(y)
        warning(['No se encontró ningún punto ', color_name, ...
            ' en la imagen.']);
        continue;
    end

    x_center = mean(x);
```

```
154     y_center = mean(y);
155     corners(i, :) = [x_center, y_center];
156 
157     plot(x_center, y_center, 'o', 'MarkerSize', 5, 'MarkerEdgeColor', ...
158         color_value, 'MarkerFaceColor', color_value);
159     text(x_center + 5, y_center, color_name, 'Color', color_value, ...
160         'FontSize', 10);
161 end
162 
163 %% Recorte de imagen (crop)
164 
165 rect = [corners(1,1), corners(1,2), 500, 500];
166 cropped_img = imcrop(imagen_transformada, rect);
167 
168 close;
169 figure; imshow(cropped_img);
170 
171 %% Re-detección de esquinas en imagen recortada
172 
173 hsv_image2 = rgb2hsv(cropped_img);
174 corners = zeros(4, 2);
175 cornerspost = zeros(4, 2);
176 
177 for i = 1:size(thresholds, 1)
178     color_name = thresholds{i, 1};
179     threshold = thresholds{i, 2};
180 
181     mask = (hsv_image2(:,:,1) >= threshold(1,1) & hsv_image2(:,:,1)  ...
182         <= threshold(1,2)) & ...
183            (hsv_image2(:,:,2) >= threshold(2,1) & hsv_image2(:,:,2)  ...
184            <= threshold(2,2)) & ...
185            (hsv_image2(:,:,3) >= threshold(3,1) & hsv_image2(:,:,3)  ...
186            <= threshold(3,2));
187 
188     [y, x] = find(mask);
189     if isempty(y)
190         warning(['No se encontró ningún punto ', color_name, ...
191             ' en la imagen.']);
192         continue;
193     end
194 
195     x_center = mean(x);
196     y_center = mean(y);
197     corners(i, :) = [x_center, y_center];
198 end
199 
200 xo = sum(corners(:,1)) / 4;
201 yo = sum(corners(:,2)) / 4;
202 centro = [xo, yo];
203 cornerspost = corners;
204 
```

```
%% Recorte adicional para eliminar bordes (delete corners)

close;
rect = [25, 25, 450, 450];
cropped_img2 = imcrop(cropped_img, rect);
imshow(cropped_img2);
centro = [225, 225];

%% Creación de máscara de sombra

close;
img_gray = rgb2gray(cropped_img2);   % Escala de grises
img_gray = im2double(img_gray);      % Normalizar entre 0 y 1

shadow_min = 0.15;
shadow_max = 0.7;
shadow_mask = (img_gray >= shadow_min) & (img_gray <= shadow_max);

figure;
imshow(shadow_mask);
title('Máscara de Sombra con Tratamiento');

%% Cálculo de dirección de la sombra

close;
[y, x] = find(shadow_mask);
xy_shadow = [x, y] - centro;

r = xy_shadow(:,1).^2 + xy_shadow(:,2).^2;
safe_zone_mask = false(size(shadow_mask));

for ix = 1:length(xy_shadow)
    if xy_shadow(ix,2) > xy_shadow(ix,1) && xy_shadow(ix,2)
        > -xy_shadow(ix,1)
        r(ix) = 0;
        safe_zone_mask(y(ix), x(ix)) = true;
    end
end

imshow(shadow_mask);
hold on;
visboundaries(safe_zone_mask, 'Color', 'r');

index = find(r == max(r));
endshadow = xy_shadow(index, :);

plot(endshadow(1) + centro(1), endshadow(2) + centro(2), 'x', ...
    'MarkerSize', 10, 'MarkerEdgeColor', "m");
plot(centro(1), centro(2), '*', 'MarkerSize', 10, ...
    'MarkerEdgeColor', "c");

```

```
256 %% Cálculo de longitud real de la sombra
257
258 lado_real = 20;  % cm
259 lado_pixel = norm(cornerspost(4,:) - cornerspost(1,:));
260 for lado = 1:3
261     l = norm(cornerspost(lado,:) - cornerspost(lado+1,:));
262     lado_pixel = l + lado_pixel;
263 end
264 lado_pixel = lado_pixel / 4;
265
266 sombra_p = norm(endshadow);  % Longitud de sombra en píxeles
267 sombra = (lado_real * sombra_p / lado_pixel) - 0.35; % Longitud real en cm
268
269 %% Cálculo de ángulo azimutal y altura solar
270
271 Al_c = 4;  % Altura del poste central en cm
272 sun_vec = [-endshadow(2), -endshadow(1)];
273
274 az = atand(sun_vec(2)/sun_vec(1));
275 Ce = atand(Al_c / sombra);
276 az = abs(az);
277
278 % Determinar el cuadrante de azimut
279 if sun_vec(1)>=0 && sun_vec(2)>=0
280     az = -az;
281 elseif sun_vec(1)>=0 && sun_vec(2)<=0
282     az = az;
283 elseif sun_vec(1)<=0 && sun_vec(2)<=0
284     az = (180 - az);
285 elseif sun_vec(1)<=0 && sun_vec(2)>=0
286     az = -(180 - az);
287 end
288
289 disp(['Azimuth = ', num2str(az), '     Altura Solar = ', num2str(Ce), ...
290     '    ', char(datetime)])
291
292 % Guardar resultados en archivo .mat
293 save([filename, '.mat']);
294
295 %% Exportar resultados
296
297 Resultados_time = [Resultados_time; timestamp];
298 Resultados = [Resultados; [az, Ce]];
299 save([filename, '.mat']);
300
```

Apéndice E

En este apéndice se presenta el código fuente de `Analisis.deriva.m`.

```
%% VERIFICACIÓN INICIAL: AXIS DE GRÁFICAS Y ALGORITMO TRACKER DESEADO
%% Código maestro: INICIO

% Fecha de creación: 2025-02-10
% Propósito: Simulación para la tesis de JOB.
clear;
clc;

disp('Inicio de la simulación...')

%% -------------------- VARIABLES DE ENTRADA --------------------

ExcelFileNameRange = 'NOAA2024.xlsx';   % Archivo Excel con datos de
% radiación solar (PE-202304)
foldersave = 'Prueba_JOB250210';     % Carpeta para guardar los
% resultados de la simulación <<<<<<<<<<<<<<<<<<<<
folderPiezas = 'Piezas';    % Carpeta donde se encuentran los
% archivos .sat (TracePro) de los componentes

% Nombres de las superficies relevantes (definidas en TracePro)
str_espejos = 'Espejo : Default-Perfect Mirror'; % Superficie reflectante
% del espejo
str_espejopost = 'CaraPosterior : Default-Perfect Absorber'; % Superficie
% posterior del espejo (absorbente)
str_target = 'Receptor : Default-Perfect Absorber'; % Superficie
% del receptor (absorbente)

% Parámetros para la selección del periodo de simulación
hora_start = 9;
hora_end = 17;
dia_start = 1;
dia_end = 31;

% Nombres de los archivos .sat de los objetos en TracePro
TraceProFileNameObject = 'espejo_JOB';          % Modelo del heliostato
TraceProFileNameObject2 = 'Base_Hexagono_imt';  % Modelo de la base
% hexagonal del heliostato (no utilizado en la inserción actual)
TraceProFileNameTarget = 'Target_JOB';          % Modelo del receptor
TraceProFileNameTarget_block = 'EsferaSombrilla2'; % Modelo del objeto
% bloqueador (para análisis de bloqueo/sombreado)

% Ajustes geométricos del heliostato (para la orientación precisa)
Ajuste_altura = 90;         % Distancia vertical desde la rótula al
% centro del espejo (mm)
Ajuste_EjeIncY = 0;         % Ajuste en el eje Y del punto de
% inclinación (mm) - Default = 0; para tesis
Ajuste_AlturaEjeIncY = 247;    % Altura del eje de inclinación
% respecto a la base (mm) - 247 prototipo impreso
Ajuste_Brazo_espejo = 90;     % Distancia horizontal entre el eje
% de inclinación y el centro del espejo (mm)

```

```
52 % Parámetros de la fuente solar simulada
53 SunSize = 1;         % Tamaño angular del sol (se ajusta posteriormente
54 % según el campo de heliostatos)
55 SunGridRes = 50;     % Resolución de la grilla de la fuente solar
56 % (aumentar disminuye el tiempo de cálculo) - default = 100
57
58 % Información de las características del heliostato individual
59 ancho_espejo = 0.26;     % Ancho del espejo (metros)
60 alto_espejo = 0.18;      % Alto del espejo (metros)
61 Area_espejo = ancho_espejo * alto_espejo;  % Área reflectante real de
62 % un espejo (m²)
63 numero_targets = 1; % Número de receptores en la simulación
64 % (en este caso, único)
65 n_material = 1; % Eficiencia del material del espejo
66 % (1 para espejo perfecto - 100% reflectividad)
67 altura_target = 2000; % Altura del centro del receptor respecto al plan
68 % o de los heliostatos (mm)
69 nombre = [1, 1; 1, 2; 1, 3; 2, 1; 2, 2; 2, 3; 3, 1; 3, 2; 3, 3];  % Matriz
70 % para nombrar los heliostatos (fila, columna)
71
72 % Coordenadas del receptor en el sistema de coordenadas global (mm)
73 Coordenadas_del_target = [0, 0, altura_target];
74
75 % Configuración del "campo" de heliostatos: una cuadrícula cuadrada
76 L = 12; % Lado del campo de heliostatos (metros)
77 Nh = 3;  % Número de heliostatos por lado de la cuadrícula
78 dist_campo_target = 3000; %% Distancia aproximada entre el centro del
79 % campo y el target (mm)
80
81 % Ajuste del tamaño angular del sol basado en la extensión del campo
82 SunSize = L * 1.8 * 1000; % Se estima un tamaño para cubrir el campo
83 disp('Variables de entrada...DONE')
84
85 %% -------------------- LECTURA DE DATOS EXTERNOS --------------------
86
87 disp('Leyendo datos de radiación del archivo Excel...')
88 [data2] = xlsread(ExcelFileNameRange); % Lee todos los datos numéricos
89 % del archivo Excel
90 disp('...DONE')
91
92 %% -------------------- VARIABLES DE TRABAJO --------------------
93
94 % Cálculo de las coordenadas de los heliostatos en la cuadrícula
95 x = 1000 * (-L/2 : L/(Nh-1) : L/2); % Coordenada x de los  heliostatos (mm)
96 y = 1000 * (-L/2 : L/(Nh-1) : L/2); % Coordenada y de los  heliostatos (mm)
97 [X, Y] = meshgrid(x, y); % Crea matrices de coordenadas para la cuadrícula
98 CoordenadasX = reshape(X, 1, [])'; % Vector columna con las coordenadas x
99 % de todos los heliostatos
100 CoordenadasY = reshape(Y, 1, [])'; % Vector columna con las coordenadas y
101 % de todos los heliostatos
102 numero_elementos = length(CoordenadasY); % Número total de heliostatos
```

```
posiciones_xyz = [CoordenadasX, CoordenadasY, 250 * ones ...
    ...(numero_elementos, 1)]; % Matriz con las coordenadas (x, y, z)
    % de cada heliostato (z inicial = 250 mm)

% Desplaza toda la cuadrícula en la dirección -x para alejarla del target
posiciones_xyz = posiciones_xyz - (max(x) + dist_campo_target) * ...
...[ones(numero_elementos, 1), zeros(numero_elementos, 2)];

% Coordenadas x e y fijas donde se considera que está el "centro"
% del sol para algunos cálculos
sun_x_fix = (max(posiciones_xyz(:, 1)) + min(posiciones_xyz(:, 1))) / 2;
sun_y_fix = (max(posiciones_xyz(:, 2)) + min(posiciones_xyz(:, 2))) / 2;

% Coordenadas del target (se asignan a una variable para mayor claridad)
xyz_target = Coordenadas_del_target;

% Extracción de las columnas relevantes de los datos de radiación
n = data2(:, 1);        % Día del año
h = data2(:, 2);        % Hora del día (decimal)
rad = data2(:, 3);      % Radiación solar incidente (W/m²)
azimut_s = data2(:, 4); % Azimut solar (grados)
altura_s = data2(:, 5); % Altura solar (grados)
instantes = length(rad); % Número total de instantes
% de tiempo en los datos

% Inicialización de matrices para almacenar los ángulos de los heliostatos
IncY = zeros(instantes, numero_elementos);     % Inclinación en el eje Y
% (Al_n o Ce_n) para cada instante y heliostato
RotZ = zeros(instantes, numero_elementos);     % Rotación en el eje Z
% (Az_n) para cada instante y heliostato

% Inicialización de matrices para el cálculo del área efectiva
Areae = zeros(instantes, numero_elementos); % Área efectiva de cada
% espejo para cada instante (considerando el ángulo de incidencia)
TotalArea = zeros(instantes, 1);           % Área total de
% todos los espejos
EffectiveArea = zeros(instantes, 1);       % Área efectiva total
% de todos los espejos

% Definición de la carpeta de trabajo y creación de la carpeta
% para guardar los resultados
folder = pwd; % Obtiene la ruta de la carpeta actual de Matlab
folder = strrep(folder, '\', '/'); % Reemplaza barras invertidas
% por barras diagonales para compatibilidad
mkdir(foldersave); % Crea la carpeta para guardar las simulaciones
WorkspaceFileName = [pwd, '/', foldersave, '/', 'Workspace - ',...
...foldersave, '.mat']; % Nombre del archivo .mat para guardar el workspace

% Conversión de las horas a diferentes formatos
h2 = h;         % Variable para ajustar la hora de verano si es necesario
h3 = floor(h2) * 100 + floor(((h2) - floor(h2)) * 60 + 0.01);  % Hora
```

```
% en formato militar (HHMM)
Hora_excel = h2 / 24;  % Hora en formato de Excel (fracción de día)

% Inicialización de variables para el cálculo de eficiencias
n_cos = zeros(instantes, 1);    % Eficiencia del factor coseno
n_spi = zeros(instantes, 1);    % Eficiencia por desbordamiento (spillage)
n_block = zeros(instantes, 1); % Eficiencia por bloqueo
n_shade = zeros(instantes, 1); % Eficiencia por sombreado
n_abs = ones(instantes, 1);     % Eficiencia de absorción (1 para espejo
% perfecto)
n_total = zeros(instantes, 1); % Eficiencia total
TargetFlux = zeros(instantes, 1); % Flujo incidente total en el receptor
% (Watts)
sumT = zeros(instantes, 1);      % Suma del flujo incidente en el receptor
sumHF = zeros(instantes, 1);     % Suma del flujo incidente en la cara
% reflectante de los heliostatos
sumBF = zeros(instantes, 1);     % Suma del flujo incidente en la cara
% posterior de los heliostatos
Eff_aparente = zeros(instantes, 1); % Eficiencia aparente (flujo en
% el target / radiación máxima posible)
RadMax = zeros(instantes, 1);     % Radiación máxima posible incidente
% en el área total de los espejos

% Variable para almacenar la posición de intersección de los rayos
% con el plano del target y la deriva
MT = NaN(instantes, numero_elementos, 3); % Matriz para almacenar
% las coordenadas (x, y, z) donde el rayo reflejado del espejo j
% en el instante i intersecta el plano del target
deriva = NaN(instantes, numero_elementos); % Distancia entre el punto
% de intersección y el centro del target para cada espejo y cada instante

disp('Variables de trabajo...DONE')

%% --------------- CÁLCULO DE ROTACIONES E INCLINACIONES ---------------

for i = 1:1:instantes
    if rad(i) > 0 % Realiza el cálculo solo si hay
        % radiación solar incidente
        az = azimut_s(i);   % Azimut solar actual
        al = altura_s(i);   % Altura solar actual
        if altura_s(i) > 0 % Realiza el cálculo solo si el sol
            % está sobre el horizonte
            % Encontrar rotaciones e inclinaciones óptimas para
            % todos los espejos
            for j = 1:1:numero_elementos
                xyz_espejo = posiciones_xyz(j, :); % Coordenadas
                % del heliostato actual
                % Cálculo de los ángulos de rotación e inclinación
                % utilizando la función Tracker_heliostato
                % Esta función determina la orientación del espejo
                % para reflejar los rayos solares hacia el target.
```

```
205                 [az_n, al_n, n_cose] = Tracker_heliostato(xyz_espejo, ...
206                     xyz_target, az, al);
207                 RotZ(i, j) = (az_n);         % Almacena el ángulo de
208                 % rotación en el eje Z (azimutal)
209                 IncY(i, j) = (90 - al_n);     % Almacena el ángulo de
210                 % inclinación en el eje Y (cenital)
211
212                 Areae(i, j) = n_cose * Area_espejo;  % Calcula el área
213                 % efectiva del espejo j en el instante i considerando
214                 % el ángulo de incidencia (factor coseno)
215             end
216         end
217     end
218     disp(['Dia ', num2str(n(i), '%03d'), ' | hora ', num2str(h(i) ...
219         , '%.2f'), ' ...']) % Muestra el progreso de la simulación
220     % en la consola
221 end
222
223 % Aplica el desfase en x a las posiciones de los heliostatos
224 posiciones_xyz = posiciones_xyz + desfase_x * [ones(length ...
225     (CoordenadasX), 1), zeros(length(CoordenadasX), 2)];
226 save(WorkspaceFileName) % Guarda las variables calculadas
227 % hasta este punto en el archivo .mat
228 disp('Cálculo de rotación e inclinación...DONE')
229
230 %% ---------------- CÁLCULO DE MANCHA SIN ITERACIÓN ----------------
231
232 for i = 1:1:instantes
233     if rad(i) > 0
234         az = azimut_s(i);
235         al = altura_s(i);
236         if altura_s(i) > 0
237             % Calcular la posición de la mancha reflejada en el
238             % plano del target para cada espejo
239             for j = 1:1:numero_elementos
240                 xyz_espejo = posiciones_xyz(j, :);
241                 az_n = RotZ(i, j);
242                 al_n = 90 - IncY(i, j);
243
244                 % Cálculo del vector normal al espejo (N) y el
245                 % vector unitario de la dirección del sol (S)
246                 N = angle_to_vector(az_n, al_n, 1);
247                 S = -angle_to_vector(az, al, 1);  % El signo negativo
248                 % indica la dirección desde el sol hacia el espejo
249
250                 % Cálculo del vector de reflexión ideal (R)
251                 R = 2 .* (dot(N, (-S))) .* N + S;
252                 R = R / norm(R);  % Normalización del vector de reflexión
253
254                 % Estimación del punto de colisión en el plano del target
255                 Ajuste_xyz = Ajuste_Brazo_espejo * ...
```

```
256                 angle_to_vector(az_n, al_n, 1);  % Vector desde la
257                 % rótula al centro del espejo
258                 Pc = xyz_espejo + Ajuste_xyz;  % Coordenadas del
259                 % centro del espejo
260                 T0 = (Coordenadas_del_target(1) - Pc(1)) / R(1);
261                 % Parámetro para encontrar la intersección con
262                 % el plano x del target
263                 MT(i, j, :) = Pc + T0 * R;  % Coordenadas (x, y, z)
264                 % del punto de intersección estimado
265
266                 % Cálculo de la deriva (distancia) entre el punto de
267                 % intersección y el centro del target
268                 deriva(i, j) = sqrt((MT(i, j, 2) - ...
269                     Coordenadas_del_target(2))^2 + (MT(i, j, 3) ...
270                     - Coordenadas_del_target(3))^2);
271             end
272         end
273     end
274 end
275 disp('Cálculo de posiciones de intersección a Target...DONE')
276
277 %% ------------------------ GRAFICACIÓN ------------------------
278
279 % Definir la paleta de colores para los 12 meses
280 Mes = {'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago',...
281     'Sep', 'Oct', 'Nov', 'Dic'};
282 colors = lines(12);  % Genera 12 colores distintos para cada mes
283 close all; % Cierra todas las figuras abiertas
284
285 % Graficar la posición de la mancha en el target para cada heliostato,
286 % diferenciando por mes
287 for j = 1:numero_elementos
288     figure(j);
289     hold on;
290     grid on;
291     % Bucle para cada mes del año
292     for mes = 1:12
293         % Índices para los 13 instantes de cada mes
294         idx_start = (mes - 1) * 240 + 1;
295         idx_end = mes * 240;
296         % Extraer las coordenadas y y z de la mancha para el mes actual
297         yyy = MT(idx_start:idx_end, j, 2) - Coordenadas_del_target(2);
298         % Desplazamiento en Y respecto al centro del target
299         zzz = MT(idx_start:idx_end, j, 3) - Coordenadas_del_target(3);
300         % Desplazamiento en Z respecto al centro del target
301         % Graficar la dispersión de la mancha usando el color
302         % correspondiente al mes
303         scatter(yyy, zzz, 50, colors(mes, :), 'filled', 'DisplayName' ...
304             , Mes{mes});
305     end
306
```

```
307     % Añadir título y etiquetas a la gráfica
308     title(['Gráfica por Mes - Heliostato ', num2str(j)]);
309     xlabel('Desplazamiento en Y [mm]');
310     ylabel('Desplazamiento en Z [mm]');
311     axis([-10 10 -10 10]); % Limitar los ejes para mejor visualización
312     % (ajustar si es necesario)
313     %axis([-100 100 -20 130]);
314
315     % Mostrar la leyenda para identificar los meses
316     legend('show');
317     legend('Location', 'eastoutside'); % Ubicar la leyenda fuera del
318     % área de la gráfica
319     hold off;
320 end
321
322 %% Graficar subplots de la posición de la mancha por heliostato y mes
323 % Definir la paleta de colores para los 12 meses (se reutiliza la anterior)
324 % Mes = {'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep',
325 % 'Oct', 'Nov', 'Dic'};
326 % colors = lines(12);
327 close all;
328
329 % Calcular la disposición de los subplots en función del número
330 % de heliostatos
331 num_rows = ceil(sqrt(numero_elementos));
332 num_cols = ceil(numero_elementos / num_rows);
333 figure; % Crear una nueva figura para los subplots
334
335 % Iterar sobre cada heliostato para crear un subplot individual
336 for j = 1:numero_elementos
337     subplot(num_rows, num_cols, j);
338     hold on;
339     grid on;
340
341     % Bucle para graficar los datos de cada mes en el subplot actual
342     for mes = 1:12
343         % Índices de los datos correspondientes al mes actual
344         idx_start = (mes - 1) * 240 + 1;
345         idx_end = min(mes * 240, instantes);
346
347         % Extraer las coordenadas y y z de la mancha para el mes actual
348         yyy = MT(idx_start:idx_end, j, 2) - Coordenadas_del_target(2);
349         zzz = MT(idx_start:idx_end, j, 3) - Coordenadas_del_target(3);
350
351         % Graficar la dispersión de la mancha con el color del mes
352         scatter(yyy, zzz, 50, colors(mes, :), 'filled', 'DisplayName', ...
353             Mes{mes});
354     end
355
356     % Añadir título y etiquetas al subplot
357     title(['Heliostato ', num2str(j)]);
```

```
358     xlabel('Desplazamiento en Y [mm]');
359     ylabel('Desplazamiento en Z [mm]');
360     %axis([-10 10 -10 10]);
361     axis([-100 100 -20 130]); % Ajustar los límites de los ejes para
362     % todos los subplots
363 
364     % Mostrar la leyenda solo en el primer subplot para evitar
365     % repeticiones
366     if j == 1
367         legend('show');
368         legend('Location', 'eastoutside');
369     end
370     hold off;
371 end
372 % Ajustar el espaciado entre los subplots para una mejor visualización
373 tight_layout();
374 
375 %% GRAFICACIÓN DE LA DERIVA (DISTANCIA AL CENTRO DEL TARGET)
376 
377 for j = 1:numero_elementos
378     figure(100 + j); % Crear nuevas figuras con identificadores diferentes
379     hold on;
380     grid on;
381     % Bucle para graficar la deriva para cada mes
382     for mes = 1:12
383         % Índices de los datos del mes actual
384         idx_start = (mes - 1) * 240 + 1;
385         idx_end = min(mes * 240, instantes);
386 
387         % Extraer los valores de la deriva para el mes actual
388         yyy = deriva(idx_start:idx_end)';
389         xxx = 0:0.1:23.999; % Eje temporal aproximado en horas (asumiendo
390         % un dato cada 6 minutos)
391         % Graficar la deriva como una línea con el color del mes
392         plot(xxx, yyy, 'color', colors(mes, :), 'DisplayName', Mes{mes});
393     end
394 
395     % Añadir título y etiquetas a la gráfica de deriva
396     title(['Deriva - Heliostato ', num2str(j)]);
397     xlabel('Hora [Hrs]');
398     ylabel('Distancia al Centro del Target [mm]');
399     % Mostrar la leyenda para identificar los meses
400     legend('show');
401     legend('Location', 'eastoutside');
402     hold off;
403 end
404 disp('Graficas...DONE')
405 
406 %% GRAFICACIÓN DE LA DERIVA EN SUBPLOTS
407 
408 % Definir la paleta de colores para los 12 meses (se reutiliza)
```

```
409 % Mes = {'Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep',
410 % 'Oct', 'Nov', 'Dic'};
411 % colors = lines(12);
412 close all;
413
414 % Calcular la disposición de los subplots para la deriva
415 num_rows = ceil(sqrt(numero_elementos));
416 num_cols = ceil(numero_elementos / num_rows);
417 figure; % Nueva figura para los subplots de deriva
418
419 % Iterar sobre cada heliostato para crear un subplot de su deriva
420 for j = 1:numero_elementos
421     subplot(num_rows, num_cols, j);
422     hold on;
423     grid on;
424
425     % Bucle para graficar la deriva de cada mes en el subplot actual
426     for mes = 1:12
427         % Índices de los datos del mes actual
428         idx_start = (mes - 1) * 240 + 1;
429         idx_end = min(mes * 240, instantes);
430
431         % Extraer los valores de la deriva
432         yyy = deriva(idx_start:idx_end)';
433         xxx = 0:0.1:23.999; % Eje temporal aproximado en horas
434         % Graficar la deriva como una línea con el color del mes
435         plot(xxx, yyy, 'color', colors(mes, :), 'DisplayName', Mes{mes});
436     end
437
438     % Añadir título y etiquetas al subplot de deriva
439     title(['Deriva - Heliostato ', num2str(j)]);
440     xlabel('Hora [Hrs]');
441     ylabel('Distancia [mm]');
442
443     % Mostrar la leyenda solo en el primer subplot
444     if j == 1
445         legend('show');
446         legend('Location', 'eastoutside');
447     end
448     hold off;
449 end
450 disp('Deriva Subplot...DONE');
451
452 %% --------------------- CONEXIÓN CON TRACEPRO ---------------------
453
454 TP_COMMAND = ddeinit('TracePro', 'Scheme'); % Inicializa la conexión
455 % DDE con TracePro
456 if ~TP_COMMAND
457     disp('ERROR en Conexion con TracePro... Abrir TracePro' )
458 else
459     disp('Conexion con TracePro...DONE' )
```

```
460 end
461
462 %% ------------ GENERACIÓN DEL ARCHIVO BASE DE TRACEPRO ------------------
463
464 ddeexec(TP_COMMAND, '(file:close-all)'); % Cierra todas las ventanas
465 % abiertas en TracePro
466 ddeexec(TP_COMMAND, '(file:new)');       % Crea un nuevo archivo
467 % en TracePro
468 ddeexec(TP_COMMAND, '(window:horizontal-tile)'); % Organiza las ventanas
469 % horizontalmente
470 TP_setView(TP_COMMAND, 'xyz2', 1); % Establece la vista 3D isométrica
471
472 % Insertar los modelos de los heliostatos en la escena de TracePro
473 for j = 1:1:numero_elementos
474     TP_name = strcat('Fila_', num2str(nombre(j, 1)), '-', num2str ...
475         (nombre(j, 2))); % Genera un nombre único para cada heliostato
476     xyz = posiciones_xyz(j, :) + [0, 0, Ajuste_altura]; % Coordenadas
477     % del heliostato, elevando el modelo por la altura del ajuste
478     TP_InsertFile(TP_COMMAND, TraceProFileNameObject, TP_name, xyz, ...
479         [0, 0, 0], folderPiezas); % Inserta el archivo .sat del heliostato
480 end
481
482 % Insertar el modelo del receptor en la posición definida
483 TP_InsertFile(TP_COMMAND, TraceProFileNameTarget, 'Target', ...
484     Coordenadas_del_target, [0, 0, 0], folderPiezas); % OPCIONAL
485
486 ddeexec(TP_COMMAND, '(zoom-all))'); % Ajusta el zoom para ver todos los↙
objetos
487 flag_base = 0; % Variable para controlar la inserción de las bases
488 % hexagonales (actualmente desactivada)
489
490 % Insertar las bases hexagonales (condicionalmente)
491 if flag_base
492     for j = 1:1:numero_elementos
493         TP_name = strcat('Base_', num2str(nombre(j, 1)), '-', ...
494             num2str(nombre(j, 2)));
495         xyz = posiciones_xyz(j, :) - [0, 0, 0];
496         TP_InsertFile(TP_COMMAND, 'BaseEsfera', TP_name, xyz, ...
497             [0, 0, 0], folderPiezas);
498     end
499 end
500
501 % Guardar el archivo base de TracePro
502 filename = 'base.oml';
503 ddeexec(TP_COMMAND, ['(file:save-as "', folder, '/', foldersave, '/',↙
filename, '")']);
504 %TP_setView(TP_COMMAND,'iso3',1);
505 TP_setView(TP_COMMAND, 'xy2', 1); % Cambia a la vista XY
506 disp('Generar archivo base...DONE')
507
508 %% -------- EJECUCIÓN DE TRACEPRO PARA CADA INSTANTE DE TIEMPO ---------
```

```
509
510 simulation_time = zeros(instantes + 1, 1);  % Inicializa un vector
511 % para almacenar los tiempos de simulación
512 ShiftAxis = Ajuste_EjeIncY;  % Asigna el valor del ajuste del eje Y
513 fprintf('Hora de inicio %s\n', datestr(now, 'HH:MM:SS')); % Muestra
514 % la hora de inicio de la simulación
515 t1 = datestr(now, 'HH:MM:SS'); % Guarda la hora de inicio
516
517 % Bucle principal para simular cada instante de tiempo (se ha reducido
518 % el número de iteraciones para prueba)
519 for i = 1:10:min(1320, instantes)  % Iterar sobre los instantes de tiempo
520     % (se toma un instante cada 10 para reducir el tiempo de simulación)
521     simulation_time(i) = now;  % Guarda la hora de inicio de la simulación
522     % para este instante
523     if rad(i) > 0
524         az = azimut_s(i);
525         al = altura_s(i);
526         if altura_s(i) > 0
527
528             % Abrir el archivo base de TracePro
529             ddeexec(TP_COMMAND, '(file:close-all)');
530             filenamebase = ['base', '.oml'];
531             ddeexec(TP_COMMAND, ['(file:open "', folder, '/', ...
532                 foldersave, '/', filenamebase, '")']);
533
534             % Guardar el archivo de trabajo con un nombre específico
535             % para este instante
536             filename = ['Dia_', num2str(n(i), '%03d'), '_hora_',...
537                 num2str(h3(i), '%04d'), '.oml'];
538             filename_bmp = ['Dia_', num2str(n(i), '%03d'),...
539                 '_hora_', num2str(h3(i), '%04d'), '.bmp'];
540             ddeexec(TP_COMMAND, ['(file:save-as "', folder, ...
541                 '/', foldersave, '/', filename, '")']);
542
543             % Cerrar todas las ventanas (para asegurar que se
544             % abre el archivo correcto)
545             ddeexec(TP_COMMAND, '(file:close-all)');
546
547             % Abrir el archivo de trabajo guardado
548             ddeexec(TP_COMMAND, ['(file:open "', folder, '/', ...
549                 foldersave, '/', filename, '")']);
550             ddeexec(TP_COMMAND, ['(window:horizontal-tile)']);
551             TP_setView(TP_COMMAND, [[-1000, -300, 500], [0, 0, 400],  ...
552                 [0, 0, 1]]);  % Establece una vista específica
553
554             % Rotar los espejos según los ángulos calculados
555             c = posiciones_xyz;  % Coordenadas de los centros de
556             % los heliostatos
557             for j = 1:1:numero_elementos
558                 TP_name = strcat('Fila_', num2str(nombre(j, 1)), ...
559                     '-', num2str(nombre(j, 2)));
```

```
560                 entity = ['entity:get-by-name "', TP_name, '"'];
561                 % Obtiene la entidad del espejo por su nombre
562                 % Aplica la rotación en el eje Y (inclinación) y
563                 % luego en el eje Z (azimutal)
564                 ddeexec(TP_COMMAND, ['(edit:rotate (', entity, ') ', ...
565                     num2str(c(j, 1) + ShiftAxis), ' ', ...
566                     num2str(c(j, 2)), ' ', num2str(c(j, 3)), ' 0 1 0 ', ...
567                     num2str(IncY(i, j)), ' )']);
568                 ddeexec(TP_COMMAND, ['(edit:rotate (', entity, ') ', ...
569                     num2str(c(j, 1)), ' ', num2str(c(j, 2)), ' ', ...
570                     num2str(c(j, 3)), ' 0 0 1 ', num2str(-RotZ(i, j)), ...
571                     ' )']);
572             end
573
574             % Guardar el archivo con las rotaciones aplicadas
575             ddeexec(TP_COMMAND, ['(file:save-as "', folder, '/', ...
576                 foldersave, '/', filename, '")']);
577             TP_setView(TP_COMMAND, 'xyz2'); % Restablece la vista
578             % isométrica
579             ddeexec(TP_COMMAND, ['(file:save-as "', folder, '/', ...
580                 foldersave, '/', filename_bmp, '")']);
581             % Guarda una captura de pantalla
582
583             % Insertar el bloqueador del target (para análisis de
584             % bloqueo/sombreado)
585             TP_InsertFile(TP_COMMAND, TraceProFileNameTarget_block, ...
586                 'EsferaSombrilla', Coordenadas_del_target, ...
587                 [0, (90 - al), (-az)], folderPiezas); % OPCIONAL
588             ddeexec(TP_COMMAND, '(zoom-all))');
589
590             % Insertar la fuente solar (Grid Source)
591             xyzSUN = round(50000 * angle_to_vector(az, al, 1)); % Calcula
592             % la posición del centro del sol (lejano)
593             xyzSUN = xyzSUN + [sun_x_fix, sun_y_fix, 0]; % Ajusta la
594             % posición del sol relativa al campo
595             ANG = [180, (90 - al), (-az)]; % Ángulos de orientación
596             % del sol (Euler)
597             Lado = SunSize; % Lado de la fuente de rejilla (mm)
598             xyGRID = Lado / 2; % Semilado de la rejilla
599             xyRAYS = floor(Lado / SunGridRes); % Número de rayos
600             % por lado de la rejilla
601             raysm2 = (xyRAYS^2) / ((Lado^2) / 1e6); % Densidad
602             % de rayos por m²
603             PeakFlux = rad(i) / raysm2; % Flujo por rayo
604
605             % Configurar la fuente de rejilla en TracePro
606             ddeexec(TP_COMMAND, ['(raytrace:set-grid-origin ' ...
607                 '(position ', num2str(xyzSUN), '))']);
608             ddeexec(TP_COMMAND, ['(raytrace:set-grid-boundary' ...
609                 '-rectangular ', num2str(xyGRID), ' ', ...
610                 num2str(xyGRID), ')']);
```

```
611             ddeexec(TP_COMMAND, ['(raytrace:set-grid-pattern' ...
612                 '-rectangular ', num2str(xyRAYS), ' ', num2str ...
613                 (xyRAYS), ' ', num2str(PeakFlux), ')']);
614             ddeexec(TP_COMMAND, ['(raytrace:set-grid-orientation' ...
615                 '-euler-degrees (gvector ', num2str(ANG), '))']);
616             ddeexec(TP_COMMAND, '(raytrace:set-analysis-mode-on)');
617             ddeexec(TP_COMMAND, ['(raytrace:set-grid-source-flag ' ...
618                 '"Grid Source 1" #t)']);
619             ddeexec(TP_COMMAND, '(raytrace:grid)'); % Lanza el
620             % trazado de rayos desde la fuente de rejilla
621             pause(2)
622
623             % Configurar el análisis para guardar solo los
624             % rayos que inciden en el target
625             entity = ['entity:get-by-name "','Target','"'];
626             ddeexec(TP_COMMAND, ['(edit:clear-selection)']);
627             ddeexec(TP_COMMAND, ['(edit:add-selection (tools:' ...
628                 'face-in-body 3 (', entity, ')))']);
629             ddeexec(TP_COMMAND, ['(analysis:ray-sorting' ...
630                 ' 1 0 0 100 0.0 1.0)']);  % Mostrar solo los rayos
631             % que inciden en el target
632
633             % Guardar el mapa de irradiancia en el target
634             pause(1);
635             ddeexec(TP_COMMAND, '(window:vertical-tile)'); % Organiza
636             % las ventanas verticalmente
637             IrradianceMapfilename = [num2str(i, '%03d'), '_Dia_',...
638                 num2str(n(i), '%03d'), '_hora_', num2str(h3(i), ...
639                 '%04d'), ' - Map', '.bmp'];
640             cmdfr = ['(analysis:irradiance-save "', folder,...
641                 '/', foldersave, '/', IrradianceMapfilename, '")'];
642             ddeexec(TP_COMMAND, cmdfr);
643
644             % Guardar el archivo de TracePro con los
645             % resultados del trazado
646             ddeexec(TP_COMMAND, ['(file:save-as "', folder, ...
647                 '/', foldersave, '/', filename, '")']);
648
649             % Guardar capturas de pantalla de diferentes vistas (opcional)
650             TP_setView(TP_COMMAND, 'xzy')
651             ddeexec(TP_COMMAND, ['(file:save-as "', folder, ...
652                 '/', foldersave, '/', 'xz_', filename_bmp, '")']);
653             TP_setView(TP_COMMAND, 'xyz2');
654             ddeexec(TP_COMMAND, ['(file:save-as "', folder, ...
655                 '/', foldersave, '/', 'ray_', filename_bmp, '")']);
656             TP_setView(TP_COMMAND, 'yzx')
657             ddeexec(TP_COMMAND, ['(file:save-as "', folder, ...
658                 '/', foldersave, '/', 'yz_', filename_bmp, '")']);
659         end
660     end
661     % Mostrar el progreso de la simulación en la consola de Matlab
```

```
662     disp(['Dia ', num2str(n(i), '%03d'), ' - hora ', num2str(h(i), ...
663         '%04d'), ' listo | [', num2str(i), '/', num2str(length(rad)), ']'])
664     disp(['TracePro | [', num2str(i), '/', num2str(instantes), ...
665         '] | ', char(datetime)])
666
667     % Establecer vistas específicas en TracePro (opcional)
668     TP_setView(TP_COMMAND, 'xzy');
669     TP_setView(TP_COMMAND, 'iso3');
670
671 end
672
673 simulation_time(i + 1) = now;  % Guarda la hora de finalización
674 % de la última simulación
675 t2 = datestr(now, 'HH:MM:SS'); % Guarda la hora final
676 fprintf('Hora final %s\n', datestr(now, 'HH:MM:SS')); % Muestra
677 % la hora de finalización
678
679 %% -------- CÁLCULO DE EFICIENCIAS (ANÁLISIS POST-SIMULACIÓN) -------------
680
681 % Update: Mayo 2024
682 % Info: Lee los reportes de flujo generados por TracePro y calcula
683 % las eficiencias de los diversos factores de pérdida en el sistema
684 % heliostático.
685 % Exporta un archivo de texto (bitácora) con posibles errores.
686
687 mkdir([pwd, '/', foldersave, '/', 'log files']); % Crea la carpeta
688 % para los archivos de registro
689 fileID = fopen([pwd, '/', foldersave, '/', 'log files', ...
690     '\Error_Log_', foldersave, '.txt'], 'w'); % Abre un archivo
691 % para registrar errores
692 formatSpec = 'Simulación de %s \n\n';
693 fprintf(fileID, formatSpec, foldersave);  % Escribe el encabezado
694 % en el archivo de registro
695
696 for i = 1:1:instantes
697     if rad(i) > 0
698         az = azimut_s(i);
699         al = altura_s(i);
700         if altura_s(i) > 0
701             %% Lectura de datos del Reporte de Flujo de TracePro
702             FluxReportfilename = ['Dia_', num2str(n(i), '%03d'),
703                 '_hora_', num2str(h3(i), '%04d'),...
704                 ' - Flux Report', '.txt'];
705             names = importfileNAMES(FluxReportfilename, 1, inf,  ...
706                 foldersave); % Importa los nombres de las entidades
707             % del reporte
708             r = size(names, 1); % Número de entidades en el reporte
709
710             Espejos_W = importfile_W(FluxReportfilename, 1, inf,  ...
711                 foldersave); % Importa la potencia incidente
712             % en cada entidad (Watts)
```

```
713
714             %% Buscar datos de incidencia en la superficie
715             % reflectante del espejo
716             HeliostatFlux = zeros(numero_elementos, 1);
717             gi = 1;
718             for g = 1:1:r
719                 str_names = names(g, 1:1);
720                 comp = strcmp(str_names, str_espejos);  % Compara el
721                 % nombre de la entidad con el nombre del espejo
722                 if comp == 1
723                     HeliostatFlux(gi, 1) = Espejos_W(g, 1);  % Almacena
724                     % la potencia incidente en el espejo
725                     gi = gi + 1;
726                 end
727             end
728
729             %% Buscar datos de incidencia en la superficie posterior del
730             % espejo (pérdidas por transmisión/absorción imperfecta)
731             HeliostatBlockFlux = zeros(numero_elementos, 1);
732             gi = 1;
733             for g = 1:1:r
734                 str_names = names(g, 1:1);
735                 comp = strcmp(str_names, str_espejopost);  % Compara con
736                 % la superficie posterior del espejo
737                 if comp == 1
738                     HeliostatBlockFlux(gi, 1) = Espejos_W(g, 1);  %Potencia
739                     % incidente en la parte posterior (idealmente cero
740                     % para espejo perfecto)
741                     gi = gi + 1;
742                 end
743             end
744
745             % Para un target plano, utilizar (descomentar) esta parte
746             % para leer el flujo incidente
747             %% Buscar datos de incidencia en el target (receptor)
748             Target = zeros(numero_targets, 1);
749             gi = 1;
750             for g = 1:1:r
751                 str_names = names(g, 1:1);
752                 comp = strcmp(str_names, str_target);  % Compara
753                 % con el nombre del receptor
754                 if comp == 1
755                     Target(gi, 1) = Espejos_W(g, 1);  % Potencia
756                     % incidente en el receptor
757                     gi = gi + 1;
758                 end
759             end
760
761             %% Preparación de variables para el cálculo de eficiencias
762             RadMax(i) = numero_elementos * rad(i) * Area_espejo; % Potencia
763             % máxima incidente posible en el área total de los
```

```
764             % espejos (Watts)
765
766             sumT(i) = sum(Target);  % Suma de la potencia incidente
767             % en todos los receptores (en este caso, uno)
768             sumHF(i) = sum(HeliostatFlux);  % Suma de la potencia
769             % incidente en la cara reflectante de todos los heliostatos
770             sumBF(i) = sum(HeliostatBlockFlux);  % Suma de la potencia
771             % incidente en la cara posterior de todos los heliostatos
772             TotalArea(i) = numero_elementos * Area_espejo;   % Área total
773             % de todos los espejos (m²)
774             EffectiveArea(i) = sum(Areae(i, :));  % Área efectiva total
775             % de los espejos considerando el factor coseno (m²)
776
777             %% Cálculo del Factor Coseno (eficiencia geométrica)
778             n_cos(i) = EffectiveArea(i) / TotalArea(i);
779             if n_cos(i) > 1
780                 formatSpec = ['Error en factor coseno: \t Dia %d - ' ...
781                     'Hora %4.2f \t- ERROR eff = %f \n'];
782                 fprintf(fileID, formatSpec, n(i), h(i), n_cos(i));
783                 disp(['Error (n_cos: Dia ', num2str(n(i)), ' Hora ', ...
784                     num2str(h(i)), '     -  ¿? eff=', ...
785                     num2str(n_cos(i)), '    <<<<<'])
786                 n_cos(i) = 1;
787             end
788
789             %% Cálculo de las Pérdidas por Sombreado (Shade)
790             n_shade(i) = sumHF(i) / (rad(i) * EffectiveArea(i));
791             if n_shade(i) > 1
792                 formatSpec = ['Error en factor de sombras: \t Dia %d' ...
793                     ' - Hora %4.2f \t- ERROR eff = %f \n'];
794                 fprintf(fileID, formatSpec, n(i), h(i), n_shade(i));
795                 disp(['Error (n_shade: Dia ', num2str(n(i)), ' Hora' ...
796                     ' ', num2str(h(i)), '    -  ¿? eff=', ...
797                     num2str(n_shade(i))])
798                 n_shade(i) = 1;
799             end
800
801             %% Cálculo de las Pérdidas por Bloqueo (Block)
802             n_block(i) = 1 - (sumBF(i) / (sumHF(i) * n_material));
803             if n_block(i) > 1
804                 formatSpec = ['Error en factor bloqueo: \t Dia %d' ...
805                     ' - Hora %4.2f \t- ERROR eff = %f \n'];
806                 fprintf(fileID, formatSpec, n(i), h(i), n_block(i));
807                 disp(['Error (n_block: Dia ', num2str(n(i)), ' Hora ' ...
808                     '', num2str(h(i)), '     -  ¿? eff=', ...
809                     num2str(n_block(i))])
810                 n_block(i) = 1;
811             end
812
813             %% Cálculo de las Pérdidas por Desbordamiento (Spillage)
814             n_spi(i) = sumT(i) / (sumHF(i) * n_material - sumBF(i));
```

```
815             if n_spi(i) > 1
816                 formatSpec = ['Error en desbordamiento: \t Dia %d' ...
817                     ' - Hora %4.2f \t- ERROR eff = %f \n'];
818                 fprintf(fileID, formatSpec, n(i), h(i), n_spi(i));
819                 disp(['Error (n_spi: Dia ', num2str(n(i)), ' Hora' ...
820                     ' ', num2str(h(i)), '    -  ¿? eff=', ...
821                     num2str(n_spi(i))])
822                 n_spi(i) = 1;
823             end
824 
825             %% Cálculo de la Eficiencia Total
826             n_total(i) = n_cos(i) * n_spi(i) * n_block(i) ...
827             * n_shade(i) * n_abs(i);
828         end
829     end
830 end
831 
832 Eff_aparente = sumT ./ RadMax; % Cálculo alternativo dela eficiencia
833 % total aparente
834 % Variables importantes para el análisis:
835 % sumT: Radiación que impacta en el target (W) en el instante i
836 % RadMax: Máxima cantidad de radiación incidente posible en el
837 % área total de los espejos
838 
839 disp(['     Dia       Hora     Rad       n_cos     n_shade' ...
840     '   n_block    n_spi    n_total   Eff_aparente'])
841 ALL_DATA1 = [n, h, rad, n_cos, n_shade, n_block, n_spi, ...
842     n_total, Eff_aparente]
843 disp(['     Dia       Hora     Rad       n_cos     n_shade' ...
844     '   n_block    n_spi    n_total   Eff_aparente'])
845 ALL_DATA2 = [lado_rombo; numero_elementos; Area_espejo * ...
846     numero_elementos; (nr0 * lado_rombo / 1000)^2; sum(sumT)]; % Informa-
847 % ción adicional (algunas variables no están definidas en este fragmento)
848 save(WorkspaceFileName); % Guarda el workspace con los resultados
849 % de las eficiencias
850 disp(['   Número de espejos: ', num2str(numero_elementos), ' espejos'])
851 disp(['   Area disponible: ', num2str(Area_espejo * ...
852     numero_elementos), ' m²'])
853 disp(['   Energía* TOTAL: ', num2str(sum(sumT) / 1000), ...
854     ' kW*']) % Energía total incidente en el target durante la simulación
855 disp(['   Número de rayos: ', num2str(floor(xyRAYS) * floor(xyRAYS))])
856 formatSpec = ['\n\n Numero de espejos: %d \n ' ...
857     'Area disponible: %.2f m² \n Energia* TOTAL: %.2f kW\n'];
858 fprintf(fileID, formatSpec, numero_elementos, ...
859     Area_espejo * numero_elementos, sum(sumT) / 1000); % Escribe
860 % resumen en el archivo de registro
861 fclose(fileID); % Cierra el archivo de registro
862 
863 %% ------------------- GUARDAR WORKSPACE FINAL -------------------
864 save(WorkspaceFileName)
865 disp(['Workspace guardado en: ', foldersave, ' ...']);
```

866

Apéndice F

En este apéndice se presenta el código fuente de
`Simulacion.desfase.xy. and.rotacional.m`.

```
%% Código maestro: INICIO

% Fecha de creación: 2025-03-07
% Propósito: Simulación para la tesis de JOB con desfase lineal (en X e Y)
%            y rotacional automático.

% Variables para el bucle de simulación (se espera que 'Variables'
% esté definido previamente)
for bbb = 1:1:length(Variables)
    desfase_x = Variables(bbb, 1);  % Desfase en la coordenada x de los
    % heliostatos (milímetros) - Extraído de la matriz 'Variables'
    desfase_y = Variables(bbb, 2);  % Desfase en la coordenada y de los
    % heliostatos (milímetros) - Extraído de la matriz 'Variables'
    desfase_az_s = 0;  % Ángulos de rotación en azimut aplicados al sol
    % para simular desfase (grados) - Actualmente fijo en 0

    disp(['Inicio de la simulación para desfase X=', num2str ...
        (desfase_x), 'mm, Y=', num2str(desfase_y), 'mm, Az= ', ...
        num2str(desfase_az_s), '°...'])

    %% --------------------- VARIABLES DE ENTRADA ---------------------

    ExcelFileNameRange = 'NOAA2024.xlsx';   % Archivo Excel con datos
    % de radiación solar (PE-202304)
    foldersave = ['Prueba_JOB250210_desfase_', num2str(desfase_x),...
        'x_', num2str(desfase_y), 'y_0az'];     % Carpeta para guardar
    % los resultados de esta simulación <<<<<<<<<<<<<<<<<<<<<<
    folderPiezas = 'Piezas';    % Carpeta donde se encuentran los
    % archivos .sat (TracePro) de los componentes

    % Nombres de las superficies relevantes (definidas en TracePro)
    str_espejos = 'Espejo : Default-Perfect Mirror'; % Superficie
    % reflectante del espejo
    str_espejopost = 'CaraPosterior : Default-Perfect Absorber';
    % Superficie posterior del espejo (absorbente)
    str_target = 'Receptor : Default-Perfect Absorber'; % Superficie del
    % receptor (absorbente)

    % Parámetros para la selección del periodo de simulación (se
    % utilizan todos los datos del archivo Excel)
    hora_start = 9;
    hora_end = 17;
    dia_start = 1;
    dia_end = 31;

    % Nombres de los archivos .sat de los objetos en TracePro
    TraceProFileNameObject = 'espejo_JOB';        % Modelo del heliostato
    TraceProFileNameObject2 = 'Base_Hexagono_imt';  % Modelo de la base
    % hexagonal del heliostato (no utilizado en la inserción actual)
    TraceProFileNameTarget = 'Target_JOB';          % Modelo del receptor
    TraceProFileNameTarget_block = 'EsferaSombrilla2'; % Modelo del objeto
```

```
52      % bloqueador (para análisis de bloqueo/sombreado)
53
54      % Ajustes geométricos del heliostato (para la orientación precisa)
55      Ajuste_altura = 90;          % Distancia vertical desde la rótula al
56      % centro del espejo (mm)
57      Ajuste_EjeIncY = 0;          % Ajuste en el eje Y del punto de
58      % inclinación (mm) - Default = 0; para tesis
59      Ajuste_AlturaEjeIncY = 247;    % Altura del eje de inclinación
60      % respecto a la base (mm) - 247 prototipo impreso
61      Ajuste_Brazo_espejo = 90;     % Distancia horizontal entre el eje
62      % de inclinación y el espejo (mm)
63
64      % Parámetros de la fuente solar simulada
65      SunSize = 1;         % Tamaño angular del sol (se ajusta posteriormente
66      % según el campo de heliostatos)
67      SunGridRes = 50;     % Resolución de la grilla de la fuente solar
68      % (aumentar disminuye el tiempo de cálculo) - default = 100
69
70      % Información de las características del heliostato individual
71      ancho_espejo = 0.26;     % Ancho del espejo (metros)
72      alto_espejo = 0.18;      % Alto del espejo (metros)
73      Area_espejo = ancho_espejo * alto_espejo;  % Área reflectante real de
74      % un espejo (m²)
75      numero_targets = 1; % Número de receptores en la
76      % simulación (en este caso, único)
77      n_material = 1; % Eficiencia del material del
78      % espejo (1 para espejo perfecto - 100% reflectividad)
79      altura_target = 2000; % Altura del centro del receptor respecto al
80      % plano de los heliostatos (mm)
81      nombre = [1, 1; 1, 2; 1, 3; 2, 1; 2, 2; 2, 3; 3, 1; 3, 2; 3, 3];
82      % Matriz para nombrar los heliostatos (fila, columna)
83
84      % Coordenadas del receptor en el sistema de coordenadas global (mm)
85      Coordenadas_del_target = [0, 0, altura_target];
86
87      numero_elementos = 9; % Número total de heliostatos en la simulación
88      num_espejos = numero_elementos; % Alias para el número de elementos
89
90      % Configuración del "campo" de heliostatos: una cuadrícula cuadrada
91      L = 12; % Lado del campo de heliostatos (metros)
92      Nh = 3;  % Número de heliostatos por lado de la cuadrícula
93      dist_campo_target = 3000; %% Distancia aproximada entre el centro
94      % del campo y el target (mm)
95
96      % Ajuste del tamaño angular del sol basado en la extensión del campo
97      SunSize = L * 1.8 * 1000; % Se estima un tamaño para cubrir el campo
98      disp('Variables de entrada...DONE')
99
100     %% ------------------- LECTURA DE DATOS EXTERNOS -------------------
101
102     disp('Leyendo datos de radiación del archivo Excel...')
```

```
103     [data2] = xlsread(ExcelFileNameRange);  % Lee todos los datos numéricos
104     % del archivo Excel
105     disp('...DONE')
106
107     %% -------------------- VARIABLES DE TRABAJO --------------------
108
109     % Cálculo de las coordenadas de los heliostatos en la cuadrícula
110     x = 1000 * (-L/2 : L/(Nh-1) : L/2);  % Coordenadas x de los centros
111     % de los heliostatos (mm)
112     y = 1000 * (-L/2 : L/(Nh-1) : L/2);  % Coordenadas y de los centros
113     % de los heliostatos (mm)
114     [X, Y] = meshgrid(x, y);  % Crea matrices de coordenadas para
115     % la cuadrícula
116     CoordenadasX = reshape(X, 1, [])';  % Vector columna con las
117     % coordenadas x de todos los heliostatos
118     CoordenadasY = reshape(Y, 1, [])';  % Vector columna con las
119     % coordenadas y de todos los heliostatos
120     numero_elementos = length(CoordenadasY);  % Número total de heliostatos
121     posiciones_xyz = [CoordenadasX, CoordenadasY, 250 * ...
122         ones(numero_elementos, 1)];  % Matriz con las coordenadas (x, y, z)
123     % de cada heliostato (z inicial = 250 mm)
124
125     % Desplaza toda la cuadrícula en la dirección -x para
126     % alejarla del target
127     posiciones_xyz = posiciones_xyz - (max(x) + dist_campo_target) ...
128     * [ones(numero_elementos, 1), zeros(numero_elementos, 2)];
129
130     % Coordenadas x e y fijas donde se considera que está el "centro" del
131     % sol para algunos cálculos
132     sun_x_fix = (max(posiciones_xyz(:, 1)) + min(posiciones_xyz  ...
133         (:, 1))) / 2;
134     sun_y_fix = (max(posiciones_xyz(:, 2)) + min(posiciones_xyz  ...
135         (:, 2))) / 2;
136
137     % Coordenadas del target (se asignan a una variable para
138     % mayor claridad)
139     xyz_target = Coordenadas_del_target;
140
141     % Extracción de las columnas relevantes de los datos de radiación
142     n = data2(:, 1);        % Día del año
143     h = data2(:, 2);        % Hora del día (decimal)
144     rad = data2(:, 3);      % Radiación solar incidente (W/m²)
145     azimut_s = data2(:, 4);  % Azimut solar (grados)
146     altura_s = data2(:, 5);  % Altura solar (grados)
147     instantes = length(rad);  % Número total de instantes de tiempo en
148     % los datos
149
150     % Inicialización de matrices para almacenar los ángulos de
151     % los heliostatos
152     IncY = zeros(instantes, numero_elementos);     % Inclinación en el
153     % eje Y (Al_n o Ce_n) para cada instante y heliostato
```

```
154     RotZ = zeros(instantes, numero_elementos);     % Rotación en el eje Z
155     % (Az_n) para cada instante y heliostato
156
157     % Inicialización de matrices para el cálculo del área efectiva
158     Areae = zeros(instantes, numero_elementos);  % Área efectiva de cada
159     % espejo para cada instante (considerando el ángulo de incidencia)
160     TotalArea = zeros(instantes, 1);           % Área total de todos los
161     % espejos
162     EffectiveArea = zeros(instantes, 1);       % Área efectiva total de todos
163     % los espejos
164
165     % Definición de la carpeta de trabajo y creación de la carpeta para
166     % guardar los resultados
167     folder = pwd; % Obtiene la ruta de la carpeta actual de Matlab
168     folder = strrep(folder, '\', '/'); % Reemplaza barras invertidas por
169     % barras diagonales para compatibilidad
170     mkdir(foldersave); % Crea la carpeta para guardar las simulaciones de
171     % este desfase
172     WorkspaceFileName = [pwd, '/', foldersave, '/', 'Workspace - ',...
173         foldersave, '.mat']; % Nombre del archivo .mat para guardar el
174     % workspace de este desfase
175
176     % Conversión de las horas a diferentes formatos
177     h2 = h;      % Variable para ajustar la hora de verano si es necesario
178     h3 = floor(h2) * 100 + floor(((h2) - floor(h2)) * 60 + 0.01);  % Hora
179     % en formato militar (HHMM)
180     Hora_excel = h2 / 24;   % Hora en formato de Excel (fracción de día)
181
182     % Inicialización de variables para el cálculo de eficiencias
183     n_cos = zeros(instantes, 1);  % Eficiencia del factor coseno
184     n_spi = zeros(instantes, 1);  % Eficiencia por desbordamiento (spillage)
185     n_block = zeros(instantes, 1);% Eficiencia por bloqueo
186     n_shade = zeros(instantes, 1);% Eficiencia por sombreado
187     n_abs = ones(instantes, 1);   % Eficiencia de absorción (1 para
188     % espejo perfecto)
189     n_total = zeros(instantes, 1);  % Eficiencia total
190     TargetFlux = zeros(instantes, 1);  % Flujo incidente total en el
191     % receptor (Watts)
192     sumT = zeros(instantes, 1);      % Suma del flujo incidente en
193     % el receptor
194     sumHF = zeros(instantes, 1);     % Suma del flujo incidente en la
195     % cara reflectante de los heliostatos
196     sumBF = zeros(instantes, 1);     % Suma del flujo incidente en la
197     % cara posterior de los heliostatos
198     Eff_aparente = zeros(instantes, 1);  % Eficiencia aparente (flujo en
199     % el target / radiación máxima posible)
200     RadMax = zeros(instantes, 1);     % Radiación máxima posible incidente
201     % en el área total de los espejos
202
203     % Variable Para realizar gráficas de aiming y distancia de
204     % target (no se utiliza en el código actual)
```

```
205     MT = NaN(instantes, numero_elementos, 3);
206     deriva = NaN(instantes, numero_elementos);
207
208     disp('Variables de trabajo...DONE')
209
210     %% ------------ CÁLCULO DE ROTACIONES E INCLINACIONES ------------
211
212     for i = 1:10:instantes % Iterar sobre los instantes de tiempo
213         % (se toma un instante cada 10)
214         if rad(i) > 0 % Realiza el cálculo solo si hay
215             % radiación solar incidente
216             az = azimut_s(i);    % Azimut solar actual
217             al = altura_s(i);    % Altura solar actual
218             if altura_s(i) > 0 % Realiza el cálculo solo si el sol está
219                 % sobre el horizonte
220                 % Encontrar rotaciones e inclinaciones óptimas para todos
221                 % los espejos
222                 for j = 1:1:numero_elementos
223                     xyz_espejo = posiciones_xyz(j, :);  % Coordenadas del
224                     % heliostato actual
225                     % Cálculo de los ángulos de rotación e inclinación
226                     % utilizando la función Tracker_heliostato
227                     [az_n, al_n, n_cose] = Tracker_heliostato ...
228                     (xyz_espejo, xyz_target, az, al);
229                     RotZ(i, j) = (az_n);         % Almacena el ángulo de
230                     % rotación en el eje Z (azimutal)
231                     IncY(i, j) = (90 - al_n);     % Almacena el ángulo de
232                     % inclinación en el eje Y (cenital)
233
234                     Areae(i, j) = n_cose * Area_espejo;  % Calcula el área
235                     % efectiva del espejo j en el instante i
236                 end
237             end
238         end
239         disp(['Dia ', num2str(n(i), '%03d'), ' | hora ', ...
240             num2str(h(i), '%.2f'), ' ...']) % Muestra el progreso
241     end
242
243     % Aplica el desfase lineal a las posiciones de los heliostatos
244     posiciones_xyz = posiciones_xyz + [desfase_x, desfase_y, 0];
245     % Aplica el desfase rotacional al azimut solar
246     % (actualmente se aplica después del cálculo de rotaciones)
247     azimut_s = data2(:, 4) + [desfase_az_s];
248
249     save(WorkspaceFileName)  % Guarda las variables calculadas hasta
250     % este punto
251     disp('Calculo de rotacion e inclinacion...DONE')
252
253     %% -------------------- CONEXIÓN CON TRACEPRO --------------------
254
255     TP_COMMAND = ddeinit('TracePro', 'Scheme'); % Inicializa la conexión
```

```
256     % DDE con TracePro
257     if ~TP_COMMAND
258         disp('ERROR en Conexion con TracePro... Abrir TracePro')
259     else
260         disp('Conexion con TracePro...DONE')
261     end
262
263     %% ------------ GENERACIÓN DEL ARCHIVO BASE DE TRACEPRO ------------
264
265     ddeexec(TP_COMMAND, '(file:close-all)'); % Cierra todas las ventanas
266     % abiertas en TracePro
267     ddeexec(TP_COMMAND, '(file:new)');       % Crea un nuevo archivo en
268     % TracePro
269     ddeexec(TP_COMMAND, '(window:horizontal-tile)'); % Organiza las
270     % ventanas horizontalmente
271     TP_setView(TP_COMMAND, 'xyz2', 1); % Establece la vista 3D isométrica
272
273     % Insertar los modelos de los heliostatos en la escena de TracePro
274     for j = 1:1:numero_elementos
275         TP_name = strcat('Fila_', num2str(nombre(j, 1)), '-', ...
276             num2str(nombre(j, 2))); % Genera un nombre único
277         xyz = posiciones_xyz(j, :) + [0, 0, Ajuste_altura]; % Coordenadas
278         % con ajuste de altura
279         TP_InsertFile(TP_COMMAND, TraceProFileNameObject, ...
280             TP_name, xyz, [0, 0, 0], folderPiezas); % Inserta el
281         % archivo .sat
282     end
283
284     % Insertar el modelo del receptor
285     TP_InsertFile(TP_COMMAND, TraceProFileNameTarget, 'Target', ...
286         Coordenadas_del_target, [0, 0, 0], folderPiezas); % OPCIONAL
287     ddeexec(TP_COMMAND, '(zoom-all))'); % Ajusta el zoom
288
289     flag_base = 0; % Control para la inserción de bases
290     % (actualmente desactivado)
291     if flag_base
292         for j = 1:1:numero_elementos
293             TP_name = strcat('Base_', num2str(nombre(j, 1)), '-', ...
294                 num2str(nombre(j, 2)));
295             xyz = posiciones_xyz(j, :) - [0, 0, 0];
296             TP_InsertFile(TP_COMMAND, 'BaseEsfera', TP_name, xyz, ...
297                 [0, 0, 0], folderPiezas);
298         end
299     end
300
301     % Guardar el archivo base de TracePro
302     filename = 'base.oml';
303     ddeexec(TP_COMMAND, ['
```

Apéndice G

El presente apéndice incluye el artículo titulado “Diseño de sistema de seguimiento y redireccionamiento para concentración de energía termosolar”, que se presentó en el XXIX Congreso Internacional Anual de la Sociedad Mexicana de Ingeniería Mecánica (SOMIM). Dicho congreso se llevó a cabo del 20 al 22 de septiembre de 2023 en Ciudad Juárez, Chihuahua, México, ofreciendo un espacio para la difusión de avances en la ingeniería mecánica.

DOI: 10.59920/KFRG1530

Tema A4 Termo fluidos: Sustentabilidad energética y ahorro de energía

"Diseño de sistema de seguimiento y redireccionamiento para concentración de energía termosolar"

Job Ordaz-Castillo[a], Héctor D. Garcia-Lara[a,*], Mariana Bautista-Gutierrez[a], Carlos D. Carrillo-Torres[a], Oscar A. Morales-Almaguer[a], Santos Méndez-Díaz[a]

[a] *Universidad Autónoma de Nuevo León, Facultada de Ingeniería Mecánica y Eléctrica. Av. Universidad, S/N. Ciudad Universitaria, 66455. San Nicolas de los Garza, Nuevo León.*
**Autor de contacto. hector.garcialra@uanl.edu.mx*

R E S U M E N

Se presenta el diseño, metodología y simulación de un dispositivo de seguimiento solar destinado a sistemas de heliostatos. El diseño y desarrollo de este dispositivo, emplea un mecanismo de movimiento altazimutal. Se fundamenta en el uso de sensores fotosensibles, dispositivos de posicionamiento, actuadores y una técnica propuesta para determinar con precisión la posición del sol. Además, se establece la comunicación e integración con un dispositivo de redireccionamiento que permite la concentración de energía termo solar, y la configuración de posicionamiento objetivo ajustable.
Todas estas funcionalidades se simulan a partir de medidas reales de un prototipo físico usando el software TracePro. El prototipo se hizo a base de la utilización de sistemas mecatrónicos, CAD, programación y servomecanismos. El seguimiento solar permite que los espejos se muevan continuamente para mantener el enfoque en el punto focal, mientras que el redireccionamiento permite ajustar la dirección de los espejos y optimizar la captación de energía, concentrando los rayos solares en un punto en específico para su aprovechamiento.

Palabras clave: Seguimiento solar, Termosolar, Diseño mecánico, Heliostato

A B S T R A C T

The design, methodology and simulation of a solar tracking device for heliostat systems are presented. The design and development of this device employs an altazimuthal movement mechanism. It is based on the use of photosensitive sensors, positioning devices, actuators, and a proposed technique to accurately determine the position of the sun.
In addition, communication and integration are established with a redirection device that allows the concentration of solar thermal energy, and the configuration of adjustable target positioning.
All these functionalities are simulated from real measurements of a physical prototype using the TracePro software. The prototype was made based on the use of mechatronic systems, CAD, programming, and servomechanisms. Solar tracking allows the mirrors to move continuously to maintain focus on the focal point, while redirection allows adjusting the direction of the mirrors and optimizing energy capture, concentrating the sun's rays on a specific point for use.

Keywords: Solar tracking, Solar thermal, Mechanical design, Heliostat

ISSN 2954-4734 T 158

INTRODUCCIÓN

Uno de los problemas que enfrenta la sociedad actual es la creciente demanda energética ligada al aumento significativo de la población y al consumo de energía inherente.

La concentración de energía solar se ha convertido en una alternativa cada vez más popular y atractiva para la generación de energía limpia y renovable. Entre los dispositivos utilizados en sistemas de concentración solar se encuentran los heliostatos, que permiten reflejar y concentrar la luz solar en un punto focal [1].

El documento presenta una investigación sobre el diseño, construcción y evaluación de un heliostato para concentración de energía. Además, se plantea utilizar los principios de un seguidor solar esto en conjunto de sensores y sistemas mecatrónicos para generar un dispositivo que sea capaz de comunicar la posición solar a sistemas externos utilizando protocolos de comunicación digitales.

El uso de heliostatos para la captación de energía solar permite reducir costos energéticos en procesos industriales; ahora bien, un sistema de seguimiento permitirá desarrollar un prototipo capaz de interactuar con diversos componentes que conforman un campo.

El desarrollo de este proyecto permitirá retroalimentar sistemas de heliostatos con la posición solar especifica en cada momento del día, que en conjunto generarán un sistema de lazo cerrado para disminuir el error al momento de redireccionar los rayos solares a un objetivo específico.

1. MARCO TEORICO

1.1. Heliostatos

Un heliostato se define como un “aparato que, mediante un servomecanismo, hace que un espejo siga el movimiento diurno del sol, recogiendo así la máxima energía para su utilización calorífica” [2].

Los heliostatos están compuestos por al menos una superficie reflectante, la cual se mueve durante el día mediante un sistema de control de inclinación que le permite reflejar en todo momento la radiación solar directa hacia un objetivo estático determinado.

El seguimiento solar existente en los heliostatos y en sus espejos se da gracias a 2 mecanismos de accionamiento que permiten el movimiento en el azimut y la elevación. Previamente es necesario saber que el par de ejes azimut-elevación son aquellos que definen la posición de un astro en el cielo en un momento dado, y desde una localización determinada [6].

1.1.1. Tipos de heliostatos

Los sistemas de torre solar basan su funcionamiento en los heliostatos, los cuales reciben su nombre por la combinación de "helio" (sol) y "stato" (estacionario) debido a que la imagen del sol reflejada se mantiene fija durante todo el día. Estos heliostatos están compuestos por espejos prácticamente planos, que están equipados con un mecanismo de seguimiento para asegurar que reflexión del sol se mantenga constante. Los heliostatos concentran y recolectan la energía solar en un receptor montado en una torre, el cual puede ser utilizado para generar energía eléctrica a partir de la térmica [6].

Para garantizar que la imagen del sol se mantenga enfocada en el receptor solar, los heliostatos deben seguir el movimiento del sol durante todo el día. De esta manera, se asegura que la imagen reflejada del sol permanezca sobre el receptor en todo momento. La Figura 1 muestra los principales componentes de un heliostato según Valdés, Durán, & Lizana [6] los cuales se describen brevemente a continuación.

En primer lugar, se encuentran los espejos, que son los encargados de reflejar la luz solar hacia el receptor. Además, es importante contar con una estructura de soporte para estos espejos, así como un pilar y una base sólida que garantice la estabilidad del heliostato. Otro elemento crucial es el sistema de control de seguimiento, que permite que el heliostato siga el movimiento del sol con precisión. Por último, se necesitan accionamientos que permitan el movimiento del heliostato y su ajuste a lo largo del día. De esta forma, todos estos componentes trabajan en conjunto para garantizar que la energía solar sea recolectada de manera eficiente y enfocada en el receptor para su uso en sistemas de energía solar.

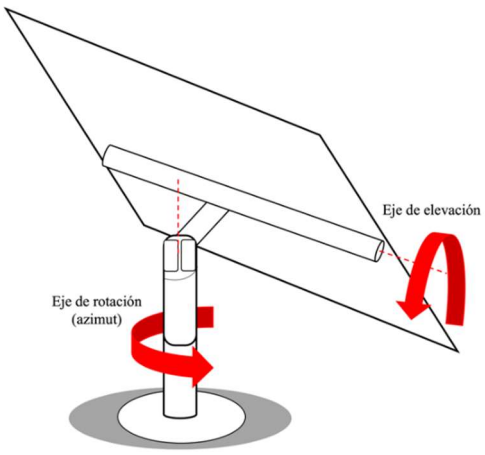


Figura 1: Diagrama de heliostato con ejes de rotación y elevación.

ISSN 2954-4734

1.2. Seguidor solar como dispositivo de optimización

En los sistemas fotovoltaicos existe la posibilidad de implementar un dispositivo adicional con el fin de aumentar la captación de radiación solar y por ende la energía suministrada por la instalación.

Un seguidor solar es un dispositivo conformado básicamente por una parte fija y una móvil, cuya finalidad es el aumento de la captación de radiación solar, para lo cual cuenta con una superficie de captación que debe permanecer perpendicular a los rayos del sol durante el día y dentro de su rango de movimiento. Los seguidores solares pueden ser clasificados según el tipo de movimiento que realicen y según el algoritmo de seguimiento [4].

2. METODOLOGÍA

Para el desarrollo de este proyecto, el objetivo principal fue establecer de manera física y simulada, el alcance que un heliostato puede conseguir en base a diferentes cualidades y parámetros, la eficiencia que se genera cuando se direcciona de forma correcta en un punto, de tal modo que, aunque la posición cambie, este seguirá captando de forma eficiente la luz solar.

El heliostato va en conjunto con el seguidor solar, el cual nos proporciona la capacidad de visualizar o captar de forma más eficiente la radiación solar.

2.1. Diseño mecánico del seguidor solar y heliostato

Para el diseño del heliostato y del seguidor solar se utilizaron dos ejes de movimiento; estos son necesarios para redireccionar los rayos del sol a un punto deseado, y en el caso del seguidor solar, posicionarse en la dirección actual del sol, transmitir sus coordenadas al heliostato y con el uso de la celda solar obtener la mayor cantidad de energía en ese punto.

La intención del diseño es crear un mecanismo que pueda ser utilizado de dos maneras distintas con la menor cantidad de cambios necesarios. En particular, se prevé su uso como heliostato o como seguidor solar, simplemente intercambiando la pieza 8. De esta manera, se obtienen dos variantes del mecanismo:

1) Con un soporte de celda solar y sensor de posición solar (Seguidor solar, Figura 2, pieza 8-9).
2) Con un soporte de espejo (Heliostato, Figura 3, pieza 8)

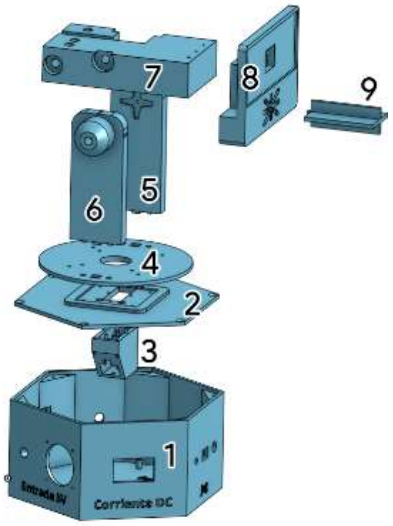


Figura 2: Vista general explosionada de la estructura del seguidor solar.

El diseño permite reutilizar las piezas del 1 al 7 para el sistema seguidor solar y el sistema del heliostato, obteniendo un estándar en cada una de las piezas individuales, permitiendo intercambiar o reemplazar piezas de ser necesario.

Para el movimiento de los dos ejes de ambos sistemas, se utilizan servomotores con 180° de movimiento (Ejes Z, Y). Para facilitar el movimiento en el eje Y se utiliza un tornillo roscado de metal que en conjunto de un rodamiento 6000z reduce la fricción al mínimo para generar movimientos precisos. En el eje Z se utiliza un rodamiento de base giratoria, mismo que se encarga de sostener el peso de las piezas 4 al 9 y reducir la fricción de dicho eje.

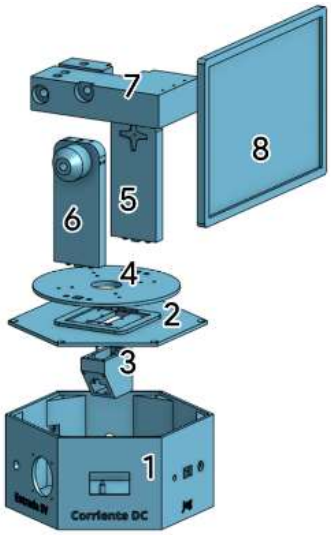


Figura 3: Vista general explosionada de la estructura del heliostato.

ISSN 2954-4734

2.2. Diseño electrónico para heliostato y seguidor solar

Para el diseño electrónico se contempla en una misma placa PCB para ambos sistemas. En la Figura 4 se muestra el circuito diseñado para el control y potencia de los sistemas fabricados.

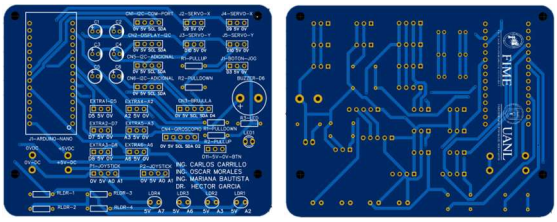


Figura 4: PCB diseñada para ambos sistemas.

En la PCB se contempla un bus de voltaje adicional a la fuente de entrada, para su funcionamiento se contempla una entrada de voltaje de 5VDC 3A.

La etapa de control es ejecutada mediante el uso de un Arduino nano programado previamente para controlar el sistema. La obtención y procesamiento de los sensores, transmisión y lectura de datos se realizan mediante protocolo I2C.

2.3. Comunicación

El sistema del heliostato utiliza el protocolo de comunicación I2C que viene integrado de forma nativa en las placas de desarrollo Arduino, esto permite comunicar mediante un bus en común los datos que nos interesan trasmitir entre sistema de seguimiento solar y sistemas de heliostatos.

Para su funcionamiento, se le asigna una dirección única a cada componente que se encuentre conectado al bus de comunicación I2C, de esta manera los datos que se envíen solamente llegarán y serán leídos por el componente que tenga la misma dirección con la que fueron enviados los datos.

2.4. Simulación del sistema

Para la simulación del sistema, se utilizaron dos programas, con interacción entre sí; el primero de ellos es Matlab para la creación del entorno y definición de variables de trabajo.

Para la parametrización de los resultados obtenidos, se utilizó TracePro que es un software de trazado de rayos, realizando la simulación de un escenario ideal para el sistema. Para la implementación y entendimiento del código utilizado dentro de Matlab, se realizó la Figura 5, donde se puede observar paso a paso la codificación de forma gráfica.

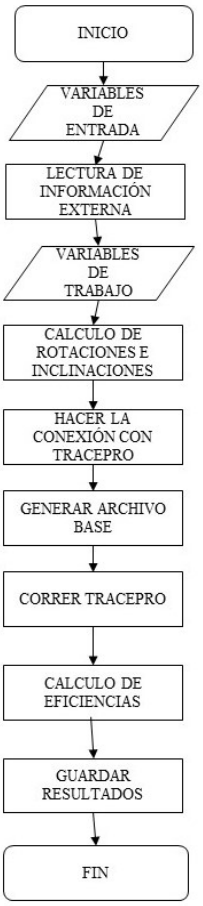


Figura 5: Diagrama de flujo del código implementado.

En un sistema ideal con variables de matrices en 3 dimensiones, las ecuaciones de transformación lineal para apuntar un heliostato hacia un objetivo desde la posición del sol pueden involucrar las siguientes variables:

1. Posición del sol: La representación fue mediante un vector unitario, ya que representamos la dirección de los rayos solares. Por ejemplo, podemos representar dicha dirección como un vector columna:

$$S = [X_S;\ Y_S;\ Z_S] \qquad (1)$$

donde X_S, Y_S, Z_S son las componentes x, y, z del vector unitario que representan los rayos del sol en el sistema de coordenadas global.

2. Direccionamiento del heliostato: También puede representarse mediante un vector unitario en coordenadas cartesianas. Por ejemplo, se puede representar la dirección del heliostato como un vector columna:

ISSN 2954-4734

$$H = [X_h;\ Y_h;\ Z_h] \quad (2)$$

donde X_h, Y_h, Z_h son las componentes x, y, z del del vector unitario que indica a donde debe apuntar el heliostato en el sistema de coordenadas global.

3. La matriz de transformación T: representa la relación lineal entre las direcciones del sol y el heliostato que convierte el vector del sol en el vector del heliostato se puede construir al tomar el producto exterior de los vectores:

$$T = S * H' \quad (3)$$

El producto exterior de dos vectores produce una matriz, y en este caso, la matriz resultante T será de tamaño 3x3. Es esencial tener en consideración que la matriz de transformación mencionada anteriormente se aplica bajo la suposición de que los vectores unitarios del sol y del heliostato están definidos en el mismo sistema de coordenadas y no existen otras transformaciones, como rotaciones o escalas, entre ellos.

2.5. Características de la simulación

En la simulación se pueden obtener la radiación incidente sobre una superficie objetivo, estos considerados como punto de partida para casos ideales del prototipo de heliostato utilizado apuntando a un objetivo representado por una superficie de tamaño de 1m x 1m en un campo a una distancia de 6 metros.

Dentro de las consideraciones que se tienen para la simulación se encuentran:

- Un único heliostato en funcionamiento, con posicionamiento a tiempo real.
- Una superficie reflectiva perfecta como espejo del heliostato.
- Un receptor con propiedades de absorción perfecto.
- Se calcularon los valores de ángulos de azimut y zenit del sol para el día 8 de mayo en intervalos de tiempo de 5 minutos usando la calculadora de posición solar del NOAA que se basa en las ecuaciones de algoritmos astronómicos de Jean Meeus. [9]
- Radiación solar unitaria (1 W/m²) como punto de referencia.

La base para la simulación se tiene con el funcionamiento del heliostato como robot de 2 grados de libertad que actúa en 3 dimensiones. Para simular correctamente el sistema, es necesario tomar en cuenta las medidas reales de las articulaciones del prototipo usado, como se muestra en la Figura 6.

La Tabla 1 contiene las distancias que se tomaron como base, la del suelo al eje y del eje al espejo.

Tabla 1 - Medidas del prototipo físico para la simulación

Articulación	Medidas
Suelo – eje	250 mm
Eje - espejo	150 mm

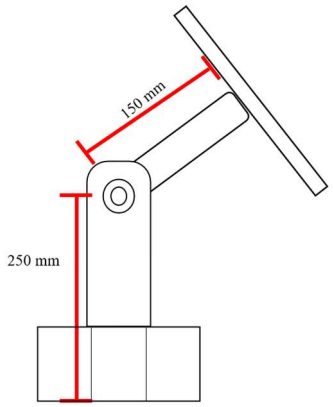


Figura 6: Diagrama del prototipo con medidas de articulaciones de robot.

De igual manera, se cargaron los modelos CAD de las geometrías utilizadas en el experimento, como el espejo del heliostato (260 mm x 180mm), el receptor de rayos, que es el objetivo y una estructura de sombra diseñada para bloquear los rayos directos del sol hacia el receptor con el fin de medir únicamente la energía captada a partir del heliostato. Por último, en la Figura 7 se muestra el diagrama del modelo del heliostato realizado en CAD exportado en TracePro, con esto se definieron las coordenadas espaciales de las geometrías mencionadas.

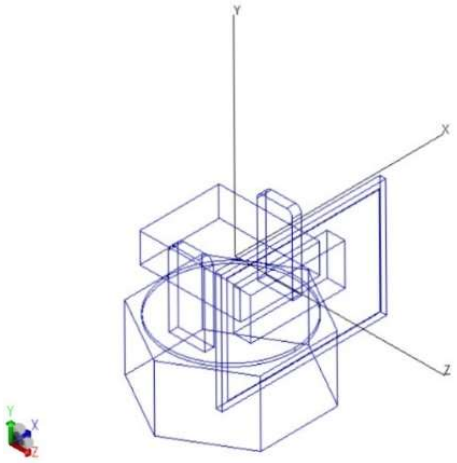


Figura 7: Modelo del heliostato en TracePro.

ISSN 2954-4734

Para realizar la simulación se tomaron en cuenta datos tomados de una prueba física del heliostato y del seguidor.

2.6. Pruebas físicas

Se realizó la experimentación para validar el funcionamiento del conjunto de heliostato y seguidor solar. El objetivo de las pruebas es comparar la precisión que tiene el mecanismo implementado físicamente contra los resultados de simular el mismo sistema bajo las mismas condiciones.

Dadas las medidas del heliostato, se creó un marco representando el objetivo para percibir la forma y ubicación del reflejo de la luz solar dada por el heliostato. Las medidas determinadas para la representación del objetivo son 300x300 mm.

La prueba física se realizó a la 1:40 pm del día 24 de mayo de 2023 en San Nicolás de los Garza, Nuevo León, México. En la tabla 2, se muestran las medidas que se utilizaron para las pruebas físicas.

Tomando el objetivo como origen del sistema de coordenadas, se obtiene la posición del heliostato en sus tres componentes (ver tabla 2).

Tabla 2 Medidas de posicionamiento de heliostato respecto al objetivo

Medida	Valor de la medición
Posición objetivo eje X	1200 mm
Posición objetivo eje Y	0.00 mm
Posición objetivo eje Z	1800 mm

Además, se considera que los cálculos realizados para determinar la dirección del objetivo respecto al heliostato se hacen en la comunicación de datos tomando la posición del heliostato respecto del seguidor como parte del procesamiento. En la Figura 8 se muestra de forma gráfica la posición que tomo el heliostato para realizar las pruebas.

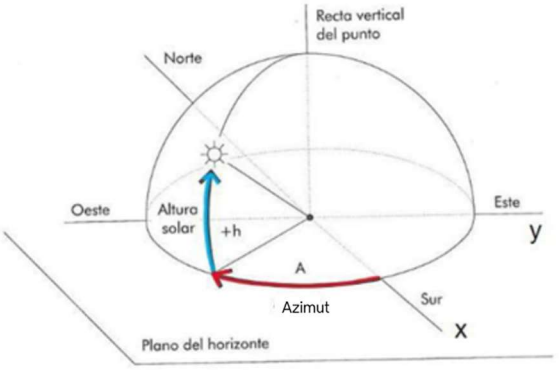


Figura 8: Diagrama de orientación del heliostato.

2.7. Simulación en TracePro

Para el cálculo de las variables de trabajo en la simulación, se usaron los datos de la tabla 2, así como que el objetivo se ubicaba en el origen.

Se llevó a cabo un post procesamiento de los datos adquiridos para su evaluación en Trace Pro. En la Figura 9, se observa el rayo resultante del heliostato hacia el objetivo deseado en TracePro.

A medida que transcurre el tiempo, el heliostato realiza cálculos para determinar de manera óptima su orientación y ángulo de inclinación, permitiéndole seguir y redirigir continuamente los rayos solares que inciden sobre su superficie. Hay que mencionar que para la prueba que se tomó, el seguidor solar nos indicó la posición en la cual se ubicaría el heliostato, y en base a esto el heliostato puede comenzar a reacomodar o redirigirse a la ubicación óptima.

La Figura 9 también, muestra el rayo y la forma en la que pega en el objetivo deseado tomando las medidas reales de la prueba física.

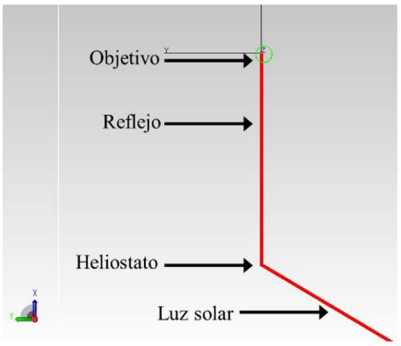


Figura 9: Resultado de la simulación del rayo a la 1:40 pm.

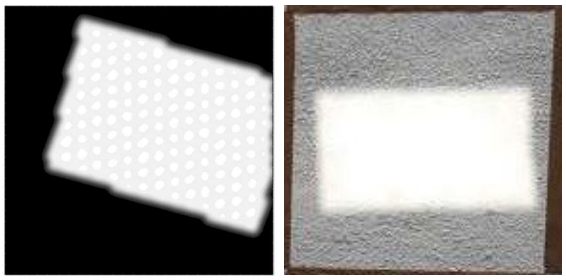
3. RESULTADOS

3.1. Comparación de prueba física y simulación

Para demostrar las similitudes y diferencias entre la imagen física y de la simulación, se obtuvo un mapa de irradiancia de la hora en la que se realizó la prueba física. El mapa de irradiancia tiene el objetivo de mostrar de manera grafica la cantidad de energía que logra reflejar el heliostato al receptor deseado. El mapa de la Figura 10 a), muestra la potencia solar que se obtuvo a una hora determinada.

Recordando que para el presente trabajo cuenta con un área superficial del espejo de 0.0468 m², y una radiación unitaria de 1 W/m², por lo que la radiación incidente máxima reflejada seria de 0.0468 W.

ISSN 2954-4734



a) b)

Figura 10: Comparación entre el mapa de irradiancia de la simulación y la prueba física del heliostato.

Como se muestra en la Figura 10, las imágenes obtenidas difieren entre sí. La simulación del rayo muestra el trazo o la dirección que este debería de tener según los datos de la tabla 2, el mapa representa cómo podría de verse de forma ideal la incidencia del rayo en el objetivo y qué geometría debe de tener. La prueba física obtuvo una forma distinta, indicando que hay factores externos de la orientación del heliostato que hacen que no se redireccione de manera correcta al objetivo. El mecanismo propuesto en el prototipo es funcional y cumple con la expectativa de redireccionar los rayos solares al objetivo deseado con precisión.

3.2. Prototipo físico

Durante todo el proceso de elaboración del proyecto, el objetivo principal fue realizar la simulación en Matlab y TracePro en base al prototipo que se ha estado trabajando durante la investigación.

Figura 11: Prototipo de seguidor solar.

En las figuras 11 y 12 se puede ver el prototipo físico del seguidor solar y del heliostato que se estuvieron utilizando para la simulación y prueba física. Como se menciona, el diseño del prototipo aplica para el heliostato y el seguidor solar. Es importante destacar que ambos prototipos están hechos para funcionar individualmente, y por medio de comunicación I2C estén conectados a funcionar a la par.

Figura 12: Prototipo de heliostato.

4. CONCLUSIÓN

Gracias a los resultados obtenidos, el diseño y desarrollo de sistemas de seguimiento y redireccionamiento desarrollados en este trabajo han demostrado ser un avance significativo en la búsqueda de soluciones más eficientes y amigables con el medio ambiente. Debido al uso de sensores y algoritmos de control, estos mecanismos permiten orientar con precisión los concentradores solares hacia la posición óptima del sol, lo que asegura una captura eficiente de la radiación solar durante todo el día. La realización de simulaciones y experimentos ha proporcionado información valiosa sobre la radiación solar incidente y la optimización del sistema. Este enfoque en la energía renovable resalta su importancia y abre nuevas posibilidades para el desarrollo sostenible y la exploración continua de energías no convencionales. El trabajo en este campo ha contribuido al conocimiento y comprensión de la potencialidad de la energía termosolar, y su implementación física promete impulsar aún más el avance hacia un futuro más limpio y sostenible.

ISSN 2954-4734

5. AGRADECIMIENTOS

Los autores agradecen al Consejo Nacional de Ciencia y Tecnología (CONACYT).

6. REFERENCIAS

[1] Cock Martínez. (2018). Diseño mecánico de un heliostato modular (Tesis de Maestría en Optomecatrónica). Centro de Investigaciones en Óptica, A.C. León, Guanajuato. 121 pp.

[2] Asale, R., & Rae. (2016). Heliostato: Diccionario de la Lengua Española, 'Diccionario de la lengua española' - Edición del Tricentenario. Recuperado de https://dle.rae.es/heliostat (Citado el 16 de mayo de 2023).

[3] National Renewable Energy Laboratory. (2013). Power Tower Projects. Recuperado de http://www.nrel.gov/csp/solarpaces/power_tower.cfm (Citado el 16 de mayo de 2023).

[4] Mejía, A. E., Londoño, M. H., & Osorio, J. C. (2010). Diseño e implementación de un seguidor solar para la optimización de un sistema fotovoltaico. Scientia et technica, 1(44), 245-250. (Citado el 25 de marzo de 2023).

[5] Valdés, M., Durán, M., & Lizana, J. (2012). Sistemas de torre solar Estado y Perspectiva. Programa de Energías Renovables y Eficiencia Energética en Chile: 4E Chile.

[6] Valls Pepió, I., & Blay Pozo, D. (2018). Impresora 3D de resina por estereolitografía (Bachelor's thesis, Universitat Politècnica de Catalunya).

[7] Mejía, A. E., Londoño, M. H., & Osorio, J. C. (2010). Diseño e implementación de un seguidor solar para la optimización de un sistema fotovoltaico. *Scientia et technica*, *1*(44), 245-250.

[8] Granda-Gutiérrez, E. E., Orta, O. A., Díaz-Guillén, J. C., Jiménez, M. A., Osorio, M., & González, M. A. (2013, October). Modelado y simulación de celdas y paneles solares (conference paper).

[9] Global Monitoring Laboratory, National Oceanic & Atmospheric Administration (NOAA), Recuperado de https://gml.noaa.gov/grad/solcalc/index.html (Citado el 16 de mayo de 2023).

ISSN 2954-4734

Apéndice H

El presente apéndice incluye el artículo "[An open-loop control algorithm for improved tracking in a heliostat]"que fue aceptado y publicado en la revista Renewable Energy, Biomass & Sustainability (REB&S), Vol. 6, No. 1. Este trabajo es una extensión de la ponencia que se presentó en el Congreso Internacional de Desarrollo Sustentable y Energías Renovables (CIDSER), un importante congreso organizado por la Asociación Latinoamericana de Desarrollo Sustentable y Energías Renovables A.C. (ALDESER).

An open-loop control algorithm for improved tracking in a heliostat

Job Ordaz-Castillo, Héctor Daniel García-Lara *, Nilda Gabriela Trejo-Luna and Santos Méndez-Díaz

Laboratorios de Investigación e Innovación en Tecnología Energética (LIITE), Facultad de Ingeniería Mecánica y Eléctrica (FIME), Universidad Autónoma de Nuevo León (UANL), San Nicolás de los Garza, Nuevo León, México.
* Corresponding author: hector.garcialra@uanl.edu.mx

Received: December 5, 2023 Accepted: April 8, 2024 Published: April 11, 2024
DOI: https://doi.org/10.56845/rebs.v6i1.90

Abstract: The growing energy demand and its relation to climate change have driven the search for sustainable alternatives, such as concentrated solar energy. In this context, heliostats play a crucial role by reflecting and concentrating solar light onto a receiver. However, traditional control approaches based on geographical data have limitations. This study introduces an autonomous control system for heliostats that eliminates the need for preloaded geographical data. The approach is based on communication between the heliostat and the solar tracker, with two configuration modes: map calibration and automatic. Centralized and autonomous heliostats are distinguished, with the latter being the focus of the study. Autonomous heliostats have their own control system and can make decisions regarding positioning and safety. The methodology involves a mathematical algorithm that calculates the optimal rotation and tilt of the heliostat to redirect light toward a target. Simulation and physical prototype testing validate a remarkable consistency between simulated and experimental data. A key result is the surprising similarity of 97.9% between the obtained data, validating the algorithm's effectiveness. This study provides a robust approach for designing autonomous heliostat control systems, integrating simulation and experimentation. These results support the algorithm's precision and ability to direct solar radiation effectively. Expanding towards autonomous control and complete heliostat system evaluation facilitates the path toward more efficient and sustainable concentrated solar energy.

Keywords: heliostat; solar tracker; simulation; algorithm design.

Introduction

One of society's contemporary challenges is the rising energy need linked to the growing population, high greenhouse emissions gases, and consequently global warming. Central solar power energy has become a more popular and attractive option for renewable energy generation moreover, it respects the environment. Among devices used in concentrating systems, heliostats can reflect and redirect solar light toward a target. (Cock Martínez, 2018).

In a heliostat field, a compound for controlled mirrors that reflects and concentrates solar light in a central collector requires an efficient control system. During the operation of the field, each heliostat is individually aligned in such a way that the normal to the surface bisects the angle between the sun's position and the target point coordinates on the receiver. However, achieving precise alignment of up to 1 milliradian for all heliostats relative to the target points on the receiver without a calibration system is considered unrealistic due to various sources of tracking error. This challenge highlights the importance of a calibration system to improve the accuracy of redirection to achieve desired flow distributions, as well as to reduce or eliminate spillage, as presented by Sattler and others, who offer an overview of the sources of tracking error and the basic requirements of an ideal calibration system (Sattler *et al.,* 2020).

Tracking errors can occur for several reasons. In an exhaustive study conducted by Jones and Stone, the sources of heliostat tracking error at the Solar Two plant were examined and detailed, including bending due to gravity, pivot point shift, atmospheric refraction, among others. This in-depth analysis helps to understand the specific challenges that need to be addressed to improve accuracy in the heliostat field (Jones & Stone, 1999).

Additionally, Díaz-Félix and others evaluated the global distributions of tracking error in the heliostat field, identifying specific errors such as angular displacement in the reference position of the tracking mechanisms, imperfect leveling of the heliostat pedestal, and lack of perpendicularity between the tracking axes (Díaz-Félix *et al.,* 2014). These findings, along with observations by Gross and Balz about disagreements between the coordinate systems used by solar, civil, and topographic engineers, illustrate the complexity of tracking errors and underline the need to comprehensively address these errors (Gross & Balz, 2020).

Furthermore, a detailed study conducted by Freeman and others, focused on the errors affecting the design of calibration and control systems, assessed the impacts of individual errors. Errors with a strong impact include those in sensors and actuators, as well as installation errors, highlighting the importance of carefully considering these factors in the design of calibration systems to mitigate their impact and optimize the performance of the heliostat system (Freeman *et al.,* 2014).

One of the most used strategies in open-loop heliostat redirection systems is using algorithms that calculate and can predict the solar trajectory during the day. Carvajal presents a heliostat design with solar tracking to redirect incident radiation to a parabolic concentrator, which is based on real-time zenithal and azimuthal calculations with short time steps. The core algorithm contains a considerable number of initial parameters for its execution (Carvajal, 2018).

This work is focused on developing a system control design for a heliostat field interconnected to a solar tracker which dismisses the need for preloading tables data and geographic location process. In this first stage, this design is for just one heliostat. To raise this, an innovative approach is proposed based on the communication between solar tracker and heliostat. In this system, it's considered two configuration modes:

1. Map calibration: The Heliostat is positioned at the desired target and collects a defined number of instances where it reports to the solar tracker time, azimuth, and zenith data corresponding to the location. With this information, the solar tracker constructs a map of the work area.
2. Automatic: The solar tracker collects the sun's current position, both in terms of azimuth and altitude, and simultaneously transfers it to the heliostat. This information allows the heliostat to adjust its position optimally to maximize the capture of solar energy at each required moment. The implementation and communication process must be validated through simulation and experimentation, and it is expected that the results obtained will support the feasibility of this approach and open new possibilities in the field of concentrated solar energy.

Materials and Methods

Methodology

According to their management and operating system, heliostats are divided into centralized and autonomous heliostats. Centralized heliostats are connected to a central control system that sends signals to each actuator of the heliostats belonging to the same solar thermal energy concentration array. This control can be based on tracking the sun's movement, which requires a shared solar tracker for all heliostats in the group. However, it is also possible to employ an approach that relies on astronomical calculations to predict the sun's position at separate times of the day, thus eliminating the need for the solar tracker.

On the other hand, autonomous heliostats have their own independent control system, which operates independently of other heliostats on the same platform as well as the central collection system. Like centralized heliostats, autonomous ones can either use a solar tracking system or rely on astronomical calculations to determine the relative position of the Sun and Earth. In the first case, each autonomous heliostat requires its own solar tracker, along with a synchronization mechanism (either mechanical or electronic) to control the actuators. The concept of autonomy applied to heliostats involves incorporating a local control unit that modifies various aspects of their operation, calculations, safety, and power supply, among others. This ensures that the heliostats consistently reflect direct solar radiation toward a fixed target, achieving solar tracking without depending on external devices. (Nakamura, 2004).

An autonomous heliostat can perform the following functions on its own:

- Perform positioning, axis guidance functions, and calculations and determine solar positioning and focus control, eliminating the need for central control assistance (autonomous calculations).
- Ensure making decisions about its own safety, protection, and integrity, thanks to the knowledge it acquires of external weather conditions or internal malfunctions (safety autonomy).
- Accomplish operational cycles thanks to the equipped local control. Cycles can be identified remotely or declared in advance (García Navajas & Egea Gea, 2000).

For these reasons, it demands extreme precision during solar tracking, otherwise, it will arise following deviations. They are referred to as a disparity presented between theoretical and real orientation and heliostat's optic axis through a year. This kind of deviation comes from a wrong alignment of activators or a failure of the calculus of solar position. Deviations can be tackled individually to quantify and fix them, achieving this by obtaining discrepancies for different solar incidence angles. This, in turn, is accomplished by directing the focus toward a camera that calculates and evaluates the deviation of the given focal point. (Pfahl *et al*., 2017).

As a pilot stage of this project, a strategic mathematical data processing design is proposed to control an autonomous heliostat communicating with a solar tracker. Once the design is achieved, simulations are planned (to analyze and adjust variables and pseudocode processes to optimize their performance before physical implementation, which can lead to improvements in efficiency and productivity), validating the model that will be implemented in the heliostat and solar tracker prototypes, respectively. Therefore, the physical experimentation will be carried out manually in the prototypes as the first stage.

Algorithm design

In this first stage of the project, a control algorithm of a mathematical and trigonometric nature was established for processing, which will have the function of performing optimal calculations for the rotation and tilt of the heliostat to redirect incident rays from its surface toward the target, regardless of its location and geographical orientation, while also eliminating the time variable, as shown in Figure 1.

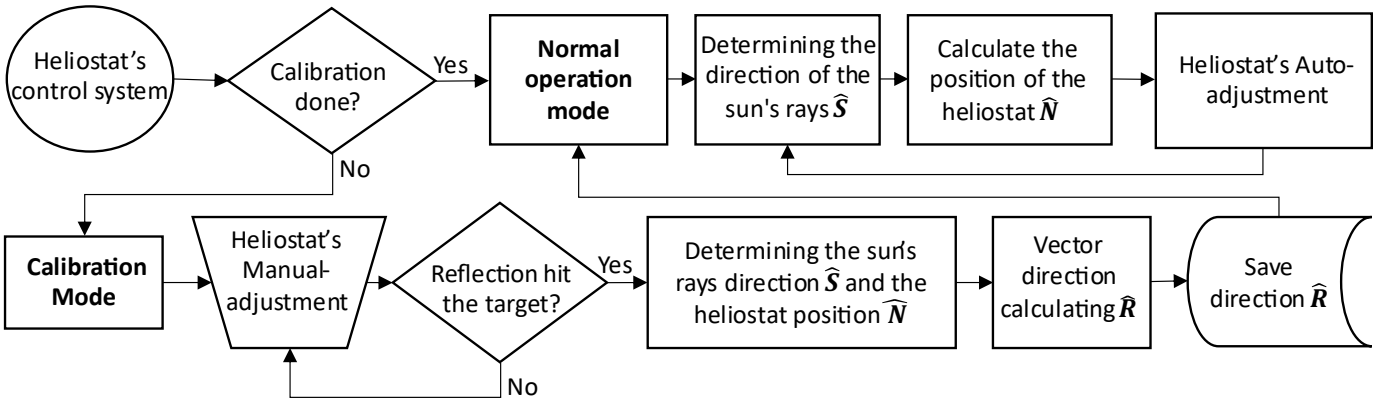


Figure 1. Flowchart.

The solar tracker is responsible for tracking the movement of the sun and reporting the azimuth (γ_s) and solar altitude α_s, which can be represented as a unit vector $\widehat{S}$ (solar ray) in our reference system. The heliostat always knows its rotation and tilt positioning, which can be represented as a unit orientation vector $\widehat{N}$ that is perpendicular to the reflective surface.

Calibration mode can permit that at a specific moment, the user manually provides the heliostat with the correct orientation using a joystick to accurately redirect the sunlight to a predetermined target. For this moment, it is possible to calculate the direction of a third unit direction vector $\widehat{R}$ using Equation 1, as seen in Figure 2. This direction $\widehat{R}$ will be a constant associated with the heliostat since the target is a fixed point that does not change throughout the day.

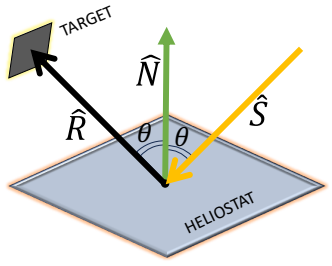


Figure 2. Established vector system.

$$\hat{\boldsymbol{R}} = 2\left(\hat{\boldsymbol{N}} \cdot \left(-\hat{\boldsymbol{S}}\right)\right)\hat{\boldsymbol{N}} + \hat{\boldsymbol{S}} \quad \textbf{(1)}$$

In normal operation mode, it is possible to calculate the position vector $\hat{\boldsymbol{N}}$ using Equation 2, thanks to having the direction $\hat{\boldsymbol{S}}$, which is a variable reported by the solar tracker, while $\hat{\boldsymbol{R}}$ was previously calculated.

$$\hat{\boldsymbol{N}} = \frac{\hat{\boldsymbol{R}} + (-\hat{\boldsymbol{S}})}{\left|\hat{\boldsymbol{R}} + (-\hat{\boldsymbol{S}})\right|} \quad \textbf{(2)}$$

To validate the a priori calculation algorithm through physical experimentation, TracePro software was used. TracePro is a tool used for optical-mechanical simulation through ray tracing. This application is responsible for tracking the intensity of each ray as it propagates through different paths in the solid models imported into its working environment (see Figure 3). Additionally, it incorporates a function known as Macro, which allows the software to receive instructions through lines of code, providing the capability to be controlled by another application using the Dynamic Data Exchange communication protocol. (García-Lara *et al*., 2021).

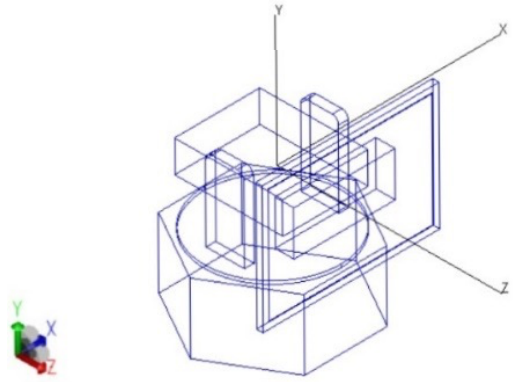


Figure 3. 3D model of heliostat's prototype in TracePro.

Experimentation and simulation

Experimental tests were carried out with the heliostat and tracker prototypes (see Figure 4) for five days in July 2023 in the municipality of Escobedo, Nuevo León. For each test, different times of the day and different reading frequencies (every 20, 30, or 60 minutes) were selected. It is important to note that in this initial stage of the study, the heliostat was operated electronically (non-autonomously) using a joystick for control. It should be emphasized that for this study, a functional mechanized prototype of the heliostat and solar tracker is available, and therefore, the mirror's geometry, rotation axes, link dimensions, and electronic control have already been determined in a previous study (Ordaz *et al*., 2023) and are not the focus of this study.

(a) **(b)**

Figure 4. Prototypes for physical experimentation: (a) heliostat and (b) Solar tracker.

During the experimentation, the solar azimuth angles (γ_s) and solar elevation angles (α_s) were documented, as well as the rotation and inclination of the heliostat (angles γ_h, α_h) for each moment. With this information, a simulation was conducted using Matlab and TracePro by inputting the real variables required by the vector calculation algorithm to verify the operation and feasibility.

Results and Discussion

The physical experimentation spanned several days in July 2023; however, data (Time, rotation angles, and inclination) from the heliostat operation are reported for 4 hours starting at 8:45 AM on July 13, 2023, with 30-minute intervals evaluated. Taking this into account, the obtained data from the Matlab-TracePro simulation were recorded, analyzing the same input variables (γ_s, α_s) from the solar tracker with the same sampling rate as the event, as shown in Table 1.

Table 1. Heliostat output angles in the experimentation and simulation on July 13, 2023.

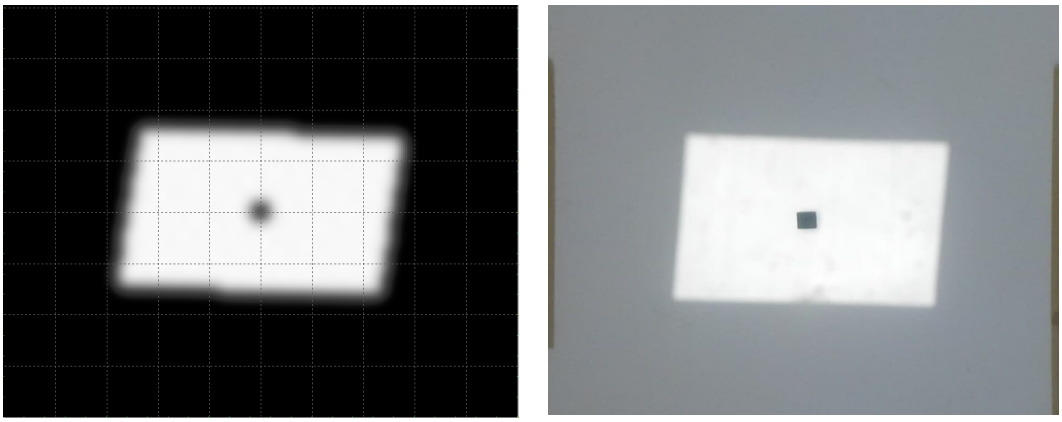
	Experimentation		Simulation	
Hour	Rotation [°]	Inclination [°]	Rotation [°]	Inclination [°]
08:45	54	34.5	54.77	32.68
09:15	58	37.5	58.24	35.63
09:45	63	39.5	62.15	38.61
10:15	68	41.5	65.98	41.02
10:45	73	42.5	70.66	43.14
11:15	77	44.5	75.45	45.00
11:45	84	45.5	80.60	46.71
12:15	86	46.5	86.76	47.59
12:45	91	46.5	91.38	48.10

In the same way a table was created that shows the percentage of similarity between the output angles for both processes of the methodology at the same moments, as shown in Table 2.

Table 2. Percentage of similarities of angles between simulation and experimentation.

Hour	Rotation	Inclination
08:45	98.6%	94.7%
09:15	99.6%	95.0%
09:45	98.7%	97.7%
10:15	97.0%	98.8%
10:45	96.8%	98.5%
11:15	98.0%	98.9%
11:45	96.0%	97.4%
12:15	99.1%	97.7%
12:45	99.6%	96.7%

Photographs of the incident light on the predetermined target were taken during the experimentation. The comparison of the images resulting from the simulation and those captured during the experimentation plays a crucial role in this study (Figure 5). By comparing the images representing the redirection of light onto the target with the irradiance maps calculated through simulations, we aim to evaluate the consistency and validity of the simulation model used. This direct comparison will help determine whether the degree of similarity observed in Table 2 has a coherent correlation with experimental reality. In this way, we seek to support the reliability of the simulation approach and provide greater confidence in the results obtained, thereby strengthening the conclusions and contributions of this multidisciplinary study. It is important to emphasize that the black center on the target serves as a reference point for image capture and does not contribute positively or negatively to the results of the tests obtained.



(a) (b)

Figure 5. Images obtained in both methodological processes for the same moment: (a) simulation in TracePro, and (b) physical experimental test.

Conclusions

This research materializes in the solid and credible validation of our approach through a consistent combination of simulation and physical experimentation. Valuable results have been obtained that allow us to understand the operation of the developed redirection algorithm and confirm its effectiveness. The surprising congruence between the simulation data and those obtained in the experimental test, with a significant 97.9% similarity as shown in the results section, validates the reliability of our approach and its ability to accurately capture expected behaviors. From this, it is inferred that the development of this algorithm contributes to the improvement of conventional systems such as those of Carvajal since many initial parameters are not needed in the map calibration mode (only the capture of the target vector R) and even less so in automatic mode. This leads to the elimination of redirection errors caused by the low resolution of the initial parameters and disparity in the reference systems.

Looking ahead, numerous opportunities for expanding and refining this study come into view. Firstly, there is an intention to enhance the resolution of the prototypes, specifically the solar tracker and the heliostat, to obtain more precise angle data at the tracker's output and allow for smoother movements in the heliostat. As a consequent step, the implementation of autonomous control of the heliostat is shaping up as a crucial second stage for future experimental tests. This expansion will enable the evaluation of the system's efficiency and adaptability in more challenging and dynamic situations, thus contributing to its robustness and applicability in diverse conditions.

Furthermore, it is imperative to shift the focus of simulation in future research. Instead of being limited to the validation of the redirection algorithm, the simulation could be redirected towards the precise quantification of the energy redirected by the field of heliostats. This would provide a more quantitative assessment of the system's performance and allow for even more rigorous optimization of its energy efficiency.

To conclude, there is a desire to transition from experimentation with a single heliostat to a full field of heliostats instead, as was done in this initial phase. This promises to provide a more comprehensive and applicable understanding of the interaction and coordination of multiple heliostats in a real-world setting. Consequently, this study is not only seen as a significant achievement but also as a solid foundation for more complex research and even deeper discoveries in the field of solar concentration technology and renewable energy.

Acknowledgments and Funding: This work was supported by the Science, Technology, and Innovation Support Program of the autonomous university of Nuevo Leon (Programa de Apoyo a la Ciencia, Tecnología e Innovación - ProACTI 2023) grant number 126-IDT-2023.

Author contributions: J.O.-C.: writing, analysis and interpretation of data; H.D.G.-L.: conceptualization, supervision, funding acquisition and editing; N.G.T.-L.: data collection; S.M.-D.: editing.

References

Carvajal Carrasco, E. (2018). Diseño y construcción de un helióstato con seguimiento solar en dos ejes para re-direccionar radiación incidente hacia un disco concentrador parabólico, Tesis de Licenciatura. Valparaiso, Chile: Laboratorio de Energías Renovables. UTFSM. http://hdl.handle.net/11673/41565

Cock Martínez, F. (2018). Diseño mecánico de un heliostato modular, Tesis de Maestría. León, Guanajuato, México: Centro de Investigaciones en Óptica A.C. https://cio.repositorioinstitucional.mx/jspui/bitstream/1002/641/1/17354.pdf

Delgado Carreño, O. R., (2019). Metodología para la evaluación del desempeño anual de sistemas de concentración de energía solar, Tesis de Maestría. Monterrey, Nuevo León, Universidad Autónoma de Nuevo León. http://eprints.uanl.mx/id/eprint/17871

Díaz-Félix, L., Escobar-Toledo, M., Waissman, J., Pitalúa-Díaz, N., & Arancibia-Bulnes, C. (2014). Evaluation of Heliostat Field Global Tracking Error Distribution by Monte Carlo Simulations. *Energy Procedia*, 49, 1308-1317. https://doi.org/10.1016/j.egypro.2014.03.140

Freeman, J., E.U., K., & S.R., R. (2014). Study od the errors influencing heliostats for calibration and control system desing. International Conference on Recent Advances and Innovations In Engineering, 1-8. https://doi.org/10.1109/ICRAIE.2014.6909113

García-Lara, H.D., Velázquez-Domínguez, A.A., Arriola-Gil, Y.Y., García-Yera, M. (2021). Evaluación de pérdidas en un campo de heliostatos mediante software de trazado de rayos. *Renewable Energy, Biomass & Sustainability (REB&S)*, *3*(2), 1-9. https://doi.org/10.56845/rebs.v3i2.46

García Navajas, Ginés, And A Egea Gea (2011). El Heliostato Autónomo. Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas CIEMAT. http://dx.doi.org/10.13140/2.1.3425.1525

Gross, F., & Balz, M. (2020). Potentially Confusing Coordinate Systems for Solar Tower Plants. AIP Conference Proceedings. https://doi.org/10.1063/5.0028942

Jones, S., & Stone, K. (1999). Analysis of solar two heliostat tracking error sources. ASME Internatinal *Solar Energy Conference*. Retrieved from: https://www.osti.gov/biblio/3312

Nakamura Katsushige (2004). Autonomous Heliostat, European Patent Office, No. EP1475582A2. https://patentimages.storage.googleapis.com/2f/43/e7/fcd3686c164516/EP1475582A2.pdf

Ordaz-Castillo J., García-Lara, H.D., Bautista-Gutiérrez, M., Carrillo-Torres, C.D., Morales-Almaguer, O.A., Méndez-Díaz, S. Diseño de sistema de seguimiento y redireccionamiento para concentración de energía termosolar. Congreso Internacional Anual de la SOMIM 2023, Ciudad Juárez, Chihuahua, México. https://somim.org.mx/memorias/memorias2023/inicio.html

Pfahl, Andreas, Joe Coventry, Marc Röger, *et al.,*. (2017). Progress in Heliostat Development. *Solar Energy* 152(Supplement C). Progress in Solar Energy Special Issue: Concentrating Solar Power (CSP): 3–37. https://doi.org/10.1016/j.solener.2017.03.029

Sattler, J. C., Röger, M., Schwarzbözl, P., Buck, R., Macke , A., Raeder, C., & Göttsche, J. (2020). Review of heliostat calibration and tracking control methods. *Solar Energy* (207), 110-132. https://doi.org/10.1016/j.solener.2020.06.030

Bibliografía

[1] David Meneses-Rodrıguez, Paul P. Horley, Jesús González-Hernández, Yuri V. Vorobiev, and Peter N. Gorley. Photovoltaic solar cells performance at elevated temperatures. *Solar Energy*, 78(2):243–250, 2005. ISES Solar World Congress 2003.

[2] Yuting Jia, Guruprasad Alva, and Guiyin Fang. Development and applications of photovoltaic–thermal systems: A review. *Renewable and Sustainable Energy Reviews*, 102:249–265, 2019.

[3] Ali Salari, Ali Parcheforosh, Ali Hakkaki-Fard, and Ali Amadeh. A numerical study on a photovoltaic thermal system integrated with a thermoelectric generator module. *Renewable Energy*, 153:1261–1271, 2020.

[4] Jonathan John Gwiazda and Francis Anthony DiBella. Power generation by solar/pneumatic cogeneration in a large, natural or man-made, open pit, 7 2005. Application No.: 10/752,506 filed on January 8, 2004.

[5] Real Academia Española. Diccionario de la lengua española, 2023. Consulta: 25 marzo 2023.

[6] Indranil Paul and Shireesh B. Kedare. Determination of optically feasible heliostat field region for solar power tower system employing innovative receiver aperture. *Solar Energy*, 271:112404, 2024.

[7] Javier Collado Hernández, Joaquín Gracia Grijota, and Jordi Fort I Comaposada. Sistema, procedimiento y programa informático de calibración del posi-

cionamiento de los espejos en heliostatos, September 2014. Titular: Ingemetal Energias S.A.

[8] D. Todd Griffith, Clifford K. Ho, Patrick S. Hunter, et al. Modal analysis of a heliostat for concentrating solar power. In *Topics in Modal Analysis I, Volume 5*, Conference Proceedings of the Society for Experimental Mechanics Series, pages 415–423, New York, NY, 2012. Springer.

[9] Manuel Torres Roldán. *Diseño de un helióstato polar innovador y simplificado para la integración en edificios y entornos urbanos.* Tesis doctoral, Universidad de Córdoba, Córdoba, 2016.

[10] Omar Behar, Abdallah Khellaf, and Kamal Mohammedi. A review of studies on central receiver solar thermal power plants. *Renewable and Sustainable Energy Reviews*, 23(Supplement C):12–39, 2013.

[11] Organización de las Naciones Unidas. Objetivos de desarrollo sostenible, 2023. [En línea]. Available: `https://www.un.org/sustainabledevelopment/es/energy/`.

[12] International Energy Agency. Technology roadmap solar thermal electricity. Technical report, International Energy Agency, 2014. [En línea]. Available: `http://www.solarpaces.org/wp-content/uploads/IEA_TechnologyRoadmapSolarThermalElectricity_2014edition.pdf`.

[13] International Energy Agency. Solar heat worldwide. Technical report, International Energy Agency, 2022. [En línea].

[14] J. C. Sattler, M. Röger, P. Schwarzbözl, R. Buck, A. Macke, C. Raeder, and J. Göttsche. Review of heliostat calibration and tracking control methods. *Solar Energy*, (207):110–132, 2020.

[15] S. Jones and K. Stone. Analysis of solar two heliostat tracking error sources. In *ASME International Solar Energy Conference*, 1999.

[16] L. Díaz-Félix, M. Escobar-Toledo, J. Waissman, N. Pitalúa-Díaz, and C. Arancibia-Bulnes. Evaluation of heliostat field global tracking error distribution by monte carlo simulations. *Energy Procedia*, 49:1308–1317, 2014.

[17] F. Gross and M. Balz. Potentially confusing coordinate systems for solar tower plants. In *AIP Conference Proceedings*, 2020.

[18] J. Freeman, K. E. U., and R. S. R. Study of the errors influencing heliostats for calibration and control system design. In *International Conference on Recent Advances and Innovations In Engineering*, pages 1–8, 2014.

[19] H. Zhang, Y. Li, and Z. Zhao. Heliostat tracking control systems in solar tower plants: A review. *Renewable and Sustainable Energy Reviews*, 76:1125–1141, 2017.

[20] T. Wang, J. Cao, and L. Liu. Performance analysis of open-loop and closed-loop control strategies for solar tracking systems. *Solar Energy*, 148:260–270, 2018.

[21] M. Smith and R. Lee. Comparative study of control systems for heliostat fields. *Applied Energy*, 185:243–252, 2016.

[22] J. J. Benitez and P. Fernandez. Design of cost-effective heliostat field control for solar power plants. *International Journal of Energy Research*, 45(9):1200–1212, 2020.

[23] F. Gonzalez. Challenges in heliostat tracking: Sensitivity to weather conditions. *Journal of Solar Engineering*, 141(6):061008–061017, 2019.

[24] D. Lopez. Calibrating heliostat fields: Methods and applications. *Renewable Energy*, 131:236–243, 2020.

[25] E. Carvajal Carrasco. Diseño y construcción de un helióstato con seguimiento solar en dos ejes para re-direccionar radiación incidente hacia un disco concentrador parabólico. Tesis de licenciatura, Laboratorio de Energías Renovables, Universidad Técnica Federico Santa María, Valparaiso, Chile, 2018.

[26] Katsushige Nakamura. Autonomous heliostat. European Patent Application, 11 2004. Accessed: 09/05/2024.

[27] G. García Navajas and A. Egea Gea. El heliostato autónomo. Informe Técnico Depósito Legal: M-14226-1995, ISSN: 1135-9420, NIPO: 238-00-002-0, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Ciudad Universitaria, 28040 Madrid, España, 10 2000.

[28] Andreas Pfahl, Joe Coventry, Marc Röger, Fabian Wolfertstetter, Juan Felipe Vásquez-Arango, Fabian Gross, Maziar Arjomandi, Peter Schwarzbözl, Mark Geiger, and Phillip Liedke. Progress in heliostat development. *Solar Energy*, 152:3–37, 2017.

[29] John A. Duffie and William A. Beckman. *Solar Engineering of Thermal Processes*. Wiley, Hoboken, NJ, USA, 4th edition, 2013.

[30] Iván Hernández Martínez. Diseño de un microhorno solar y campo de heliostatos para diversas aplicaciones. Tesis de grado de ingeniería mecánica, Universidad Nacional Autónoma de México, 2013. `http://132.248.52.100:8080/xmlui/handle/132.248.52.100/5955`.

[31] Thomas R. Mancini. Catalog of solar heliostats. Technical report, IEA-Solar Power and Chemical Energy Systems, Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), Solare Energietechnik (DLR, EN-SE), Köln, Germany, June 2000. Task III: Solar Technology and Applications.

[32] S. Jones and K. Stone. Analysis of strategies to improve heliostat tracking at solar two. In *ASME International Solar Energy Conference*, Maui, HI, 1999.

[33] Schalk J. Lombard and Willie J. Smit. Practical challenges to calibrate a heliostat with a multi-copter. In *AIP Conference Proceedings*, volume 2445, page 050005, 2022.

[34] Reynaldo Ledesma-Jaime, Marcos Rodríguez-Sánchez, Miguel Ángel Ferrer-Almaráz, and Humberto Ramos-López. Análisis dinámico estructural de un

helióstato concentrador de energía solar. *Revista de Energía Química y Física*, 3(8):1–11, 2016.

[35] Araya Sepúlveda and Gonzalo Matías. Análisis, comparación y evaluación económica de tecnologías termosolares, 2013. Accessed: 2023-11-03.

[36] Christoph Richter, Sven Teske, and Rebecca Short. *Energía Solar Térmica de Concentración. Perspectiva Mundial 2009*. Greenpeace Internacional, SolarPACES y ESTELA, España, 2009. Accessed: 2023-11-03.

[37] Duban Herley Morales Sánchez. Propuesta de una metodología para el cálculo del costo nivelado de energía (lcoe) en proyectos de energía solar, 2020.

[38] ENEA Grupo. Evaluación del recurso solar, 2017.

[39] Ministerio para la Transición Ecológica y el Reto Demográfico. Guía para la elaboración de estudios de impacto ambiental de proyectos de plantas solares fotovoltaicas y sus infraestructuras de evacuación, 2020.

[40] MathWorks. *MATLAB and Simulink Documentation*, 2023. Versión R2023a.

[41] Lambda Research Corporation. *TracePro 2023 Released*, 2023. Versión más reciente de TracePro.

[42] Fernando Arístides Saldaña Milla. Aplicación del algoritmo pid disminuye error de posicionamiento en helióstatos. *High Tech - Engineering Journal*, 2025.

[43] J. Ballestrín, A. Crespo, E. Guillot, and J. M. Martínez. Impact of heliostat tracking error on the optical performance of solar tower plants. *Energy Procedia*, 49:50–59, 2014.