Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID
ESCUELA DE INGENIERÍAS INDUSTRIALES

Grado en Ingeniería en Electrónica Industrial y Automática

MÓDULOS FOTOVOLTAICOS: ANÁLISIS Y ESTADO DE LA TÉCNICA

Autor:
Maroto García, Miguel Ángel

Tutor:
Buey Cuesta, José Julio
Tecnología Electrónica

Valladolid, Octubre de 2017

Resumen

En este trabajo se realizará un estudio del estado del arte y análisis de mercado sobre la tecnología solar fotovoltaica. Una vez realizada ésta parte del trabajo, se procederá al desarrollo de un simulador capaz de realizar los cálculos necesarios para replicar el comportamiento bajo condiciones reales de los módulos más populares del mercado en su punto de máxima potencia, sin incluir elementos posteriores al módulo.

Palabras clave

fotovoltaico, estado, mercado, simulador, precisol

Abstract

In this work a study of the state of the art and market analysis will be performed. After that work is done, a simulator will be developed and implemented, capable of performing the neccesary calculations to replicate the behavior of the photovoltaic market's most popular modules under real conditions at their maximum power point, ommiting posterior elements to the module.

Keywords

photovoltaic, state, market, simulator, precisol

Índice

Capítulo 1

Introducción

La tecnología fotovoltaica es una de las energías renovables que más atención e inversión está recibiendo en éste momento; la inversión en este tipo de sistemas (tanto investigación como instalación, con mucha diferencia en lo primero) es muy superior que el resto de energías renovables, creciendo año tras año, gracias a toda la atención que está recibiendo en I+D, con células de eficiencias en laboratorio de hasta un 46 %, y gracias a la rapidez de creación de infraestructuras fotovoltaicas y su baja relación de coste por potencia producida y peso por potencia producida.

La tecnología fotovoltaica se basa en producir electricidad directamente de la radiación recibida por el sol. El efecto fotovoltaico fue descubierto por Bequerel en 1838, por el que los fotones inciden sobre un material fotovoltaico, el cual genera corriente debido a la absorción de la energía de los fotones que impactan sobre electrones de valencia y los desplazan.

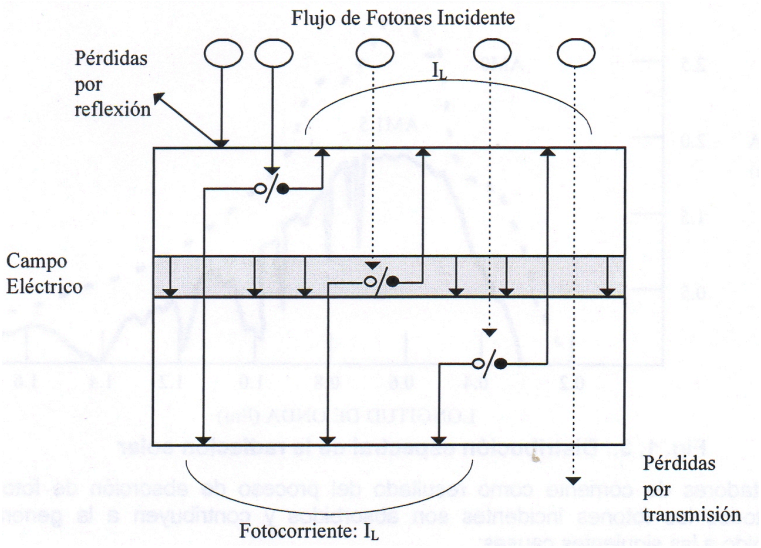


Figura 1.1 : esquema del funcionamiento interno de una célula solar, cortesía de Asociación LACECAL

En las células existen dos componentes principales de corriente: la fotogenerada, que corresponde a la generación y movimiento de los portadores debido a la transferencia de energía de los fotones a los electrones; y la corriente de oscuridad, la cual es una pérdida, debida a la recombinación de portadores, producida a su vez por la tensión que hay que aplicar entre extremos a la célula para crear un campo eléctrico en la misma que favorezca la conducción.

El mayor inconveniente al que se enfrentan las células solares actualmente (y a lo largo de toda su historia) es su ineficiencia energética, y a la gran necesidad de superficie para la generación de energía. Para paneles comerciales, la eficiencia está situada entre el 12 y el 16 % en células de silicio policristalino (las más habituales), con máximas del 20 % en paneles con células de silicio monocristalino, mucho más caro de producir. Éste inconveniente las hace inviables para ciertas aplicaciones de pequeña escala. Además, no toda la radiación es absorbida por las células solares, y éste tipo de pérdidas suelen ser las más importantes para una célula solar. Están influidas por el tipo de material (no todos los materiales absorben el mismo espectro de radiación), la reflectividad (corregida por capas de antirreflejante, pero no completamente) y la interconexión de las células, lo cual reduce en una pequeña cantidad la superficie útil de la célula. A esto se añaden más perdidas por causas ambientales (suciedad, ángulo de incidencia, temperatura, etc).

Objetivos

En este trabajo se han establecido los dos siguientes objetivos principales:

- Realizar un estudio de mercado y del estado del arte en tecnología fotovoltaica, observando el gasto, legislaturas relacionadas e inversión en infraestructura, a nivel internacional y de España, además de citar las tecnologías más importantes en la actualidad y nuevos descubrimientos en células y materiales fotovoltaicos.
- Desarrollo de un simulador de células y módulos fotovoltaicos, capaz de simular con precisión condiciones meteorológicas a las que el módulo será sometido (irradiancia y temperatura). Se desarrollarán dos modelos, los cuales diferirán en la estructura del circuito simplificado que representará la célula y el módulo.

Capítulo 2

Estado del arte

2.1. Internacional

A continuación se hará un resumen de la situación general de la tecnología solar fotovoltaica a nivel internacional. La inmensa mayoría de los datos corresponden a estudios realizados por REN21 [32], asociación la cual realiza reportes anuales muy completos sobre energías renovables (estudios de mercado, estado del arte, situación política, etc). Para los datos que no correspondan a dicho estudio se usará una cita que identifique el recurso utilizado.

Situación política y legal

Casi todos los países del mundo tienen normas relacionadas a las energías renovables, muchas de ellas regulando la tecnología solar fotovoltaica, y sigue siendo uno de los focos de atención más importante de legisladores. Éstas normas son muy variadas en naturaleza, ya sean acuerdos, objetivos, tasas o subvenciones, siendo la imposición de objetivos el método más usado para el desarrollo de éste campo. En generación de energía, otra manera muy utilizada de propulsar el desarrollo y uso de paneles solares fotovoltaicos son las FITs (Feed-In Tariffs, básicamente inversiones por parte del gobierno en proyectos), con algunas dedicadas específicamente a esta tecnología, sobre todo en países con mercados de energía renovables desarrollados.
No todas las políticas para la tecnología solar fotovoltaica son favorables a su desarrollo e inversión, como por ejemplo, las tasas e impuestos que se imponen sobre la instalación de paneles, inversores y generadores, sobre todo a nivel residencial. Algunos ejemplos de leyes similares son Hawaii, España, parte de E.E.U.U. (Arizona, Kansas, etc), con tasas regulatorias sobre el autoconsumo.

Inversión en tecnología solar fotovoltaica

Hacia 2011, la inversión de capital en tecnología solar fotovoltaica era liderada por países desarrollados, con Alemania a la cabeza, con unos 250 billones de dolares americanos (USD), pero ha ido decreciendo con los años, y hacia 2015, China pasó a la cabeza de inversión en tecnología fotovoltaica, tanto a gran como a pequeña escala (residencial o de aplicación específica).
A nivel mundial, en 2015 se invirtieron 81 billones de USD en países desarrollados y 82 en países en vías de desarrollo, lo cual supone un aumento del 12 % respecto a 2014. 67.4 billones de USD (a nivel global) corresponden a proyectos de pequeña escala. Es la tecnología renovable con más inversión a nivel mundial, y con el mayor crecimiento. El hecho de que la inversión en países en vías de desarrollo sea cumulativamente mayor que en países desarrollados demuestra la gran viabilidad económica que tienen dichas tecnologías a la hora de añadir potencia a la red de forma rápida, eficiente, bajo coste, con la obvia ventaja de la renovabilidad del recurso usado.
Respecto a la inversión en I+D en tecnología solar, en 2015 se situaba en 4.5 billones de USD, lo que supera la suma de la inversión en I+D del resto de tecnologías de energía renovable.

Instalaciones

Muy parecido al panorama visto en la inversión, el liderazgo en instalaciones de plantas solares está en Asia, con una cuota de adiciones en 2015 del 60 % mundial, con China a la cabeza, añadiendo una estimación de 15.2 GW en 2015, teniendo una capacidad acumulativa de 44 GW en energía solar, muy por delante de los otros dos líderes, Japón y Estados Unidos. En otros continentes, sobre todo en Europa, siendo el continente con más capacidad, instaló muy poca en 2015, reduciéndose aún más en años venideros, por lo que ha sido adelantada por Asia.

2.2. España

A continuación se hará un resumen de la situación general de la tecnología solar fotovoltaica a nivel nacional, en España. La inmensa mayoría de los datos corresponden a estudios realizados por REN21[32]. Para los datos que no correspondan a dicho estudio se usará una cita que identifique el recurso utilizado.

Situación política y legal

España, durante varios años, fue uno de los países con más inversión dedicada a la tecnología solar fotovoltaica, siendo los proyectos a pequeña escala, sobre todo residenciales, una gran parte de dicha inversión, y uno de los países con mayor esfuerzo en I+D sobre tecnología fotovoltaica, pero recientes cambios de la política sobre el autoconsumo ("impuesto al sol") en 2015 han frenado notablemente su desarrollo y viabilidad económica.
Muy recientemente, a principios de 2017, se han realizado esfuerzos para desarrollar una nueva legislación con respecto al autoconsumo, en parte debido a las presiones y multas por parte de Europa al no cumplir con objetivos marcados para el desarrollo de energías renovables.

Toda esta información se ha recavado de la revista Era Solar[36] y los informes anuales de la UNEF[9]

Inversión en tecnología solar fotovoltaica

El tejido empresarial ha evolucionado de 2010 a 2015, con un movimiento del sector desde la fabricación y distribución de módulos hacia la promoción y gestión de proyectos, los llamados Contratistas EPC, debido a la falta de creación de instalaciones en territorio nacional.
Respecto al I+D, ahora está más enfocado a investigadores: centros de investigación y universidades, debido al mencionado cambio del tejido industrial y su deterioro.

Toda esta información se ha recavado de los informes anuales de la UNEF[9]

Instalaciones

En España sólo se instaron 49MW en 2015, llegando a un total de 4667MW netos en la península ibérica. Desde 2009 hasta 2015 el número de instalaciones realizadas fue muy pequeño, con una potencia instalada en estos años de tan sólo unos 1300MW. Esto se debe a la hecatombe que sufrió el sector hacia 2008, con la disminución drástica de los precios de los paneles solares fotovoltaicos y el vacío legal que hubo en las primas aportadas en los 10 meses de diferencia entre el primer plan de energía solar fotovoltaica y el segundo; en ése pequeño período, debido a la rentabilidad que daban las primas, se instaló unas dos veces la capacidad

instalada en los años próximos, de alrededor de 2800MW en tan solo 10 meses.

Toda esta información se ha recavado de la revista Era Solar[36] y los informes anuales de la UNEF[9]

2.3. Tecnologías relevantes

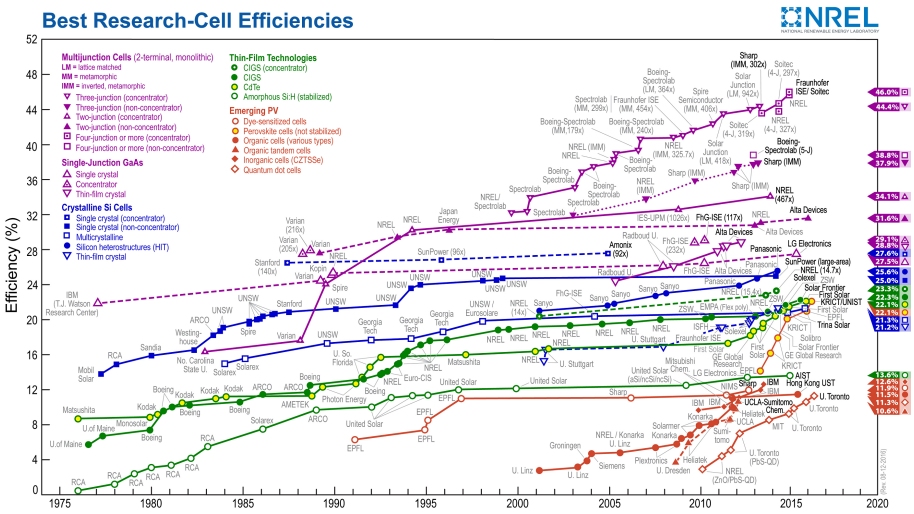


Figura 2.1 : actual progresión del desarrollo de células solar y sus eficiencias, cortesía de NREL

En la actualidad existen multitud de tecnologías fotovoltaicas, algunas llegando a eficiencias de laboratorio superiores al 45 % [27]. Actualmente sigue predominando la tecnología de silicio policristalino, con cuotas de mercado de un 46 % [39] para prácticamente todas las aplicaciones, ya sean grandes superficies, . Teniendo esto en cuenta, hay tecnologías emergentes que, debido a esfuerzos en I+D, están haciendo competencia en coste, eficiencia y mantenibilidad a las tecnologías tradicionales de silicio monocristalino y policristalino, como las tecnologías de capas finas de silicio amorfo, células de Perovskita, orgánicas, CIGS y Cd-Te. En este trabajo se van a analizar las tecnologías más relevantes, ya sea debido a sus altas cuotas de mercado, desarrollo reciente, prometedoras eficiencias, bajos costes de producción, obtención de materias primas y mantenimiento.

Silicio Monocristalino

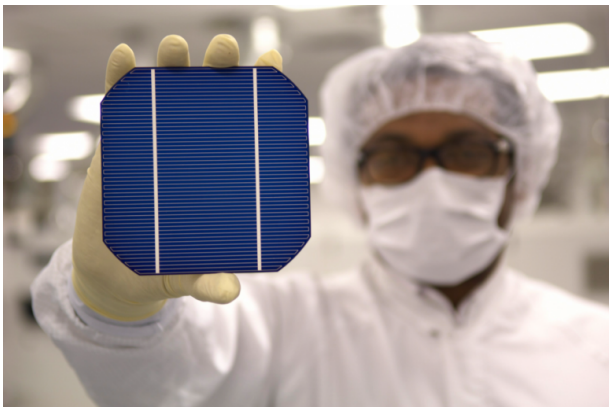


Figura 2.2 : célula solar fotovoltaica monocristalina, cortesía de Spectrolab

Una de las células más populares del mercado, con el segundo puesto en cuota de mercado, sólo por debajo de la tecnología policristalina, con eficiencias medias cercanas al 20 %, y eficiencias de laboratorio en células estado del arte llegando hasta el 25.3 % [13], aunque esta tecnología está intentando sustituirse, por su alto coste de fabricación; esto se debe principalmente a la dificultad del proceso de creación de las láminas de silicio monocristalino.

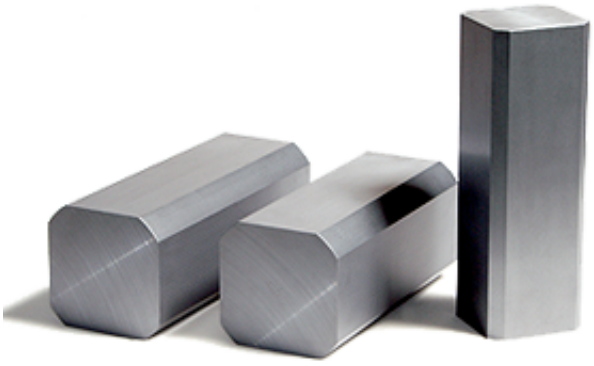


Figura 2.3 : lingotes de silicio monocristalino, cortesía de Solar Innova

El proceso de fabricación de éstas células se asemeja mucho al de un circuito impreso, salvando las distancias. El proceso comienza por el procesado del silicio en su estado bruto hasta convertirlo en un lingote de silicio prácticamente monocristalino mediante el método Czochralski (99.9999999 % de pureza, “9N”). Posterior a esto, se corta al tamaño de las células, se realiza una serie de dopados

para obtener una unión N-P (exactamente igual que la de un diodo) y se añaden las capas de metalizado para conectar las dos partes del semiconductor. Finalmente, se termina la célula tratando químicamente la superficie para aumentar su vida útil y, sobre todo, disminuir la reflectividad de la superficie de la célula.

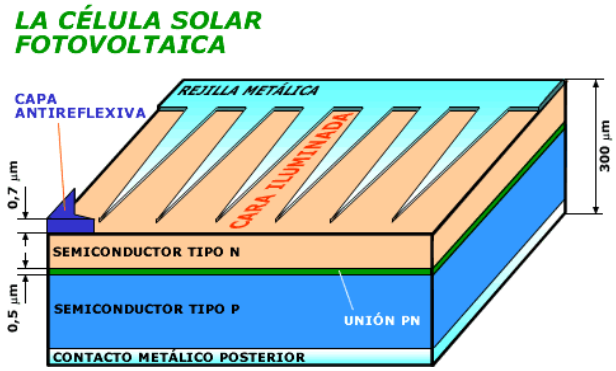


Figura 2.4 : esquema célula solar monocristalina, cortesía de la Universidad de Jaén

Se trata de una célula de capa ”gruesa”: no se fabrican por deposición de un elemento sobre otro, y son inflexibles. Tienen un aspecto y un color homogéneos, y una absorción teóricamente igual en todas las células cortadas en el mismo plano cristalino.

Silicio Policristalino

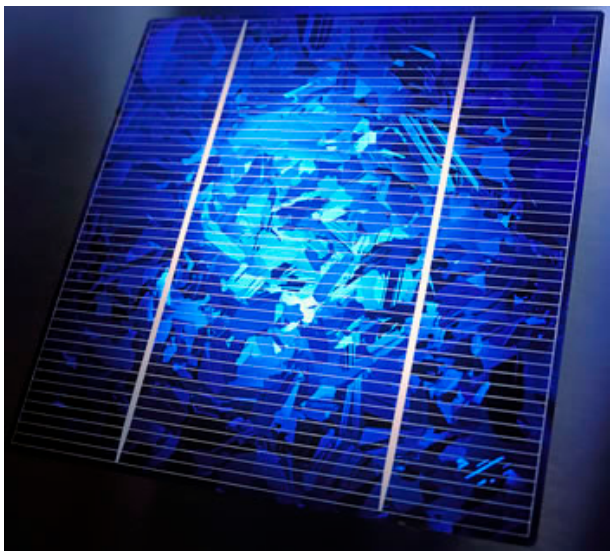


Figura 2.5 : célula solar fotovoltaica de silicio policristalino, cortesía de MOUKD

Tecnología más producida a nivel mundial, con eficiencias rondando el 16 %, y eficiencias de laboratorio en células estado del arte llegando hasta el 21.9 % [13]. Son mucho más baratas de fabricar que las anteriores (silicio monocristalino), por lo que su ratio coste/eficiencia las supera enormemente, convirtiéndolas en el tipo de célula más popular en el mercado.
Su fabricación es muy parecida a la del monocristalino, pero su mayor diferencia reside en la materia prima, o más bien, en el procesamiento metalúrgico de la materia prima. Al igual que el monosilicio, el polisilicio proviene de silicio elemental, pero los lingotes son de grado metalúrgico, siendo el proceso de obtención más habitual el proceso Siemens [44].

(a) : lingote de silicio grado metalurgico, cortesía de Warut Roonguthai

(b) : lingote de silicio grado metalurgico, cortesía de Polycrystalline Silicon Technology Corporation (P.S.T.)

Figura 2.6

Se fabrican por deposición de una capa de silicio monocristalino sobre una oblea de silicio [25].

Como se ha mencionado anteriormente, se trata de una célula de capa gruesa. Tienen un aspecto característico, emitiendo diferentes brillos, debido a los diferentes planos y granos en el lingote policristalino.

Silicio Amorfo

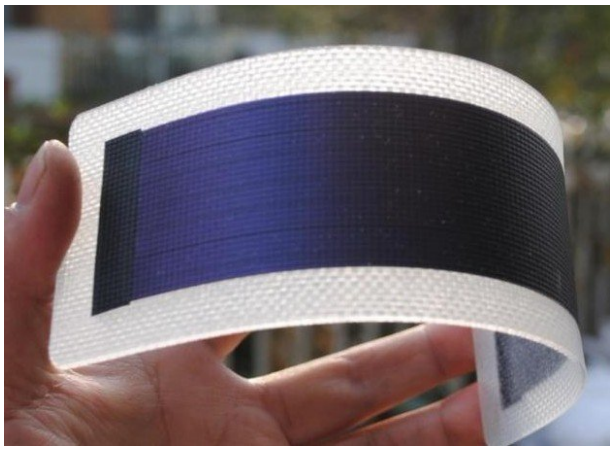


Figura 2.7 : célula solar fotovoltaica de silicio amorfo, cortesía de DHgate

Éste tipo de células son otra variante de las provenientes del silicio como materia prima principal. Como su propio nombre sugiere, están compuestas por silicio sin estructura cristalina definida, con un tamaño de grano prácticamente indistinguible, por lo que, a niveles macroscópicos, ninguna parte de la célula tiene una estructura cristalina. Inherentemente, debido a su estructura, son las células de silicio que, por si solas, tienen la menor eficiencia, rondando el 7 % de media, pero con eficiencias de laboratorio en células estado del arte de hasta un 21.2 % [38].

Figura 2.8 : horno de deposición de a-Si:H, cortesía de Keija Furnaces (fotografía de `https://www.alibaba.com/`)

Su proceso de fabricación es radicalmente diferente al resto de células solares basadas en silicio, ya que ésta tecnología es de capa fina, lo que quiere decir que no se obtiene de lingotes cortados en obleas, sino de deposición de finas capas

de silicio sobre un sustrato. Hay varios métodos para fabricarlas y en la actualidad, debido a su potencial, bajo coste y competitividad con las células monocristalinas, se están creando nuevos procesos, optimizando la fabricación de éstas células.

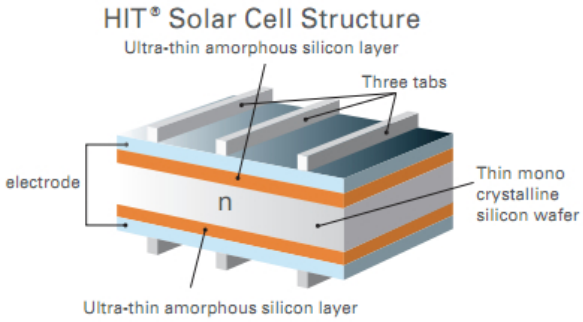


Figura 2.9 : célula solar fotovoltaica HIT, cortesía de Sanyo-Panasonic

En la actualidad el silicio amorfo se está juntando con otro tipos de células y creando hibridaciones, campo donde se encuentra el aparentemente verdadero potencial de ésta tecnología. Células como la HIT de Sanyo-Panasonic [34] son un excelente ejemplo: dicha célula consiste en la deposición de una capa de a-Si:H (abreviación de Silicio Amorfo Hidrogenado) sobre una célula monocristalina, lo que produce que la célula tenga una capacidad de absorción de espectros mayor y una reflectividad menor, llegando a eficiencias del 20 % en funcionamiento nominal. Además de ésta célula, existen multitud de células multicapa basadas en el a-Si:H, las cuales son las que más eficiencia están logrando. Actualmente, la célula más eficiente es una célula de cuatro capas, con una eficiencia del 46 % en laboratorio [13].

Éstas células, por lo general, tienen un aspecto de color oscuro, mate antirreflectivo, y la mayoría de ellas son flexibles, debido a su naturaleza de capa fina.

Células CIGS

Figura 2.10 : célula solar CIGS, cortesía de Solarion AG, fotografía de Dantor (Wikipedia)

Células compuestas por cobre (C), indio (I), galio (G) y selenio (S). Poseen eficiencias cercanas 14 %, un buen comportamiento respecto a la temperatura y con una gran capacidad de absorción de espectro solar, lo que conduce a unas eficiencias considerables, de alrededor del 15 % de media, con eficiencias de laboratorio de hasta un 22.3 % [4].

Como la tecnología a-Si:H, CIGS es una tecnología de capa fina, es decir, se forma por deposición de una o varias capas de material sobre un sustrato, y son flexibles.

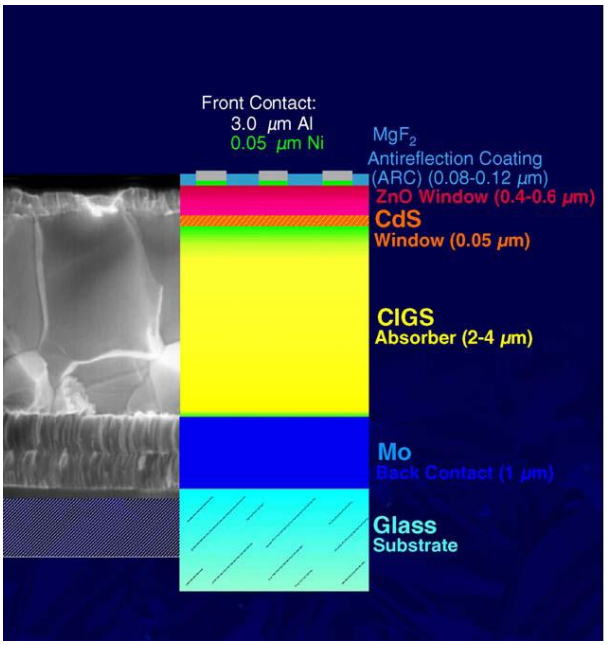


Figura 2.11 : estructura de una célula CIGS, cortesía de NREL

Éste tipo de tecnología no tiene un método de fabricación predominante, ya que existen una gran variedad de procesos, al estar todavía en desarrollo, aún con cuotas de mercado muy bajas, siguen siendo una nueva tecnología en vías de desarrollo. El método depende de la deposición de la capa CIGS, el sustrato (ya sea metálico, plástico o vidrio [1]), el orden de deposición de capas, etc.

Células Cd-Te

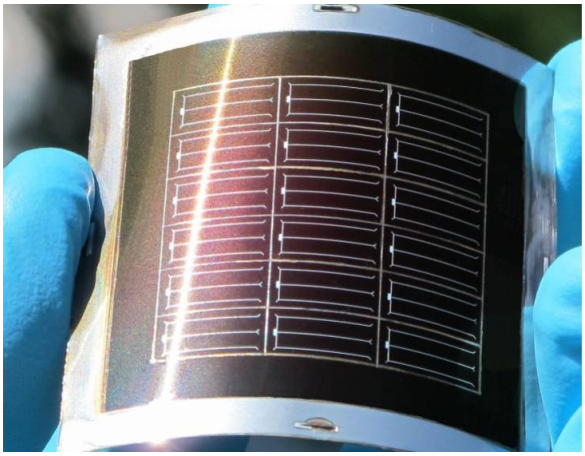


Figura 2.12 : célula solar fotovoltaica Cd-Te, cortesía de Empa

Compuestas por cadmio (Cd) y teluro (Te), es, junto a las células a-Si:H y CIGS, la tecnología de capa fina más relevante en el mercado[14], con eficiencias medias de un 17 % y eficiencias de laboratorio en células estado del arte del 22.1 % [37]. Son células flexibles, más baratas y rápidas de producir que sus principales competidoras en capa fina y, al tener un pequeño coeficiente de temperatura su eficiencia en entornos de trabajo reales se ve aumentada [35].

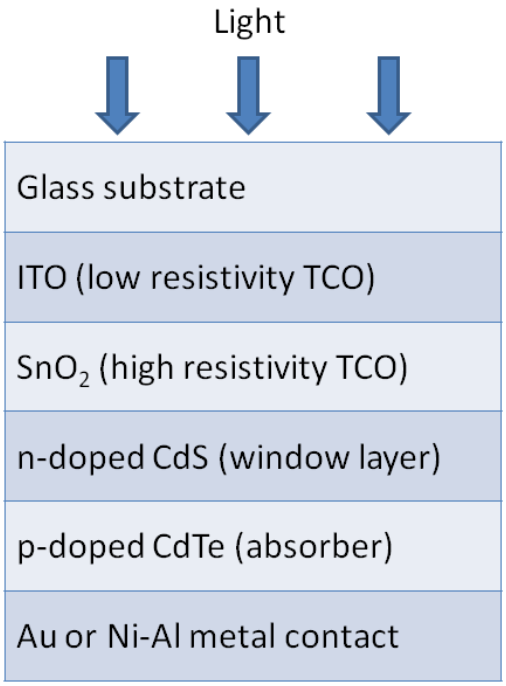


Figura 2.13 : esquema célula Cd-Te, cortesía de Chopra et al., *Progress in Photovoltaics*, fotografía de Mse395chris (Wikipedia)

Su proceso de fabricación es muy parecido al de todas las tecnologías de capa fina, lográndose por deposición de las capas necesarias (Cd-Te y demás compuestos químicos) sobre un sustrato, típicamente plástico o vidrio [8]. Eventos recientes y hallazgos de yacimientos de teluro, como el encontrado en el fondo del mar en aguas cercanas a Canarias [5], podrían cambiar la posición en el mercado de esta tecnología, debido a la viabilidad de coste que podría suponer la abundancia de dicho elemento.

Células orgánicas

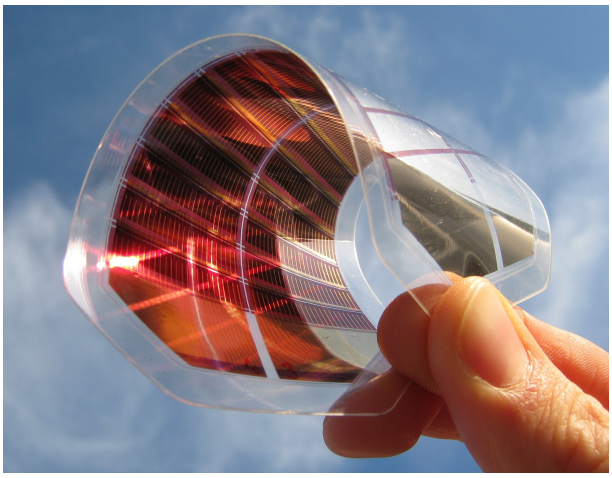


Figura 2.14 : célula solar fotovoltaica orgánica, cortesía de BusinessKorea

También denominadas células plásticas, ya que están construidas a partir de polímeros capaces de generar flujo de electrones por efecto fotovoltaico [42]. Es una tecnología que aún se encuentra bajo desarrollo, debido a su mayor problema, deducible por los materiales usados para construirlos (polímeros): grave degradación debido a efectos fotoquímicos [17].

Actualmente, éstas células tienen una eficiencia media de alrededor del 6 %, con eficiencias de laboratorio en células estado del arte del 11.3 % [41].

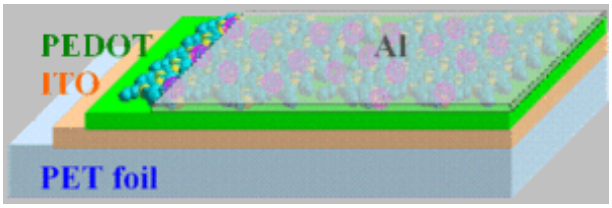


Figura 2.15 : esquema general de una célula orgánica, cortesía de Victor I. Krinichnyi

Debido a que se encuentran en continuo desarrollo y optimización, hacer un resumen de los procesos de fabricación escapa el alcance de este trabajo.

Actualmente existen varios tipos de células de éste tipo, muy variados en sus propiedades: hay células desde una capa hasta tres capas de polímeros fotovoltaicos, flexibles, rígidas, opacas y hasta transparentes (aplicación en ventanas, etc).

Células de Perovskita

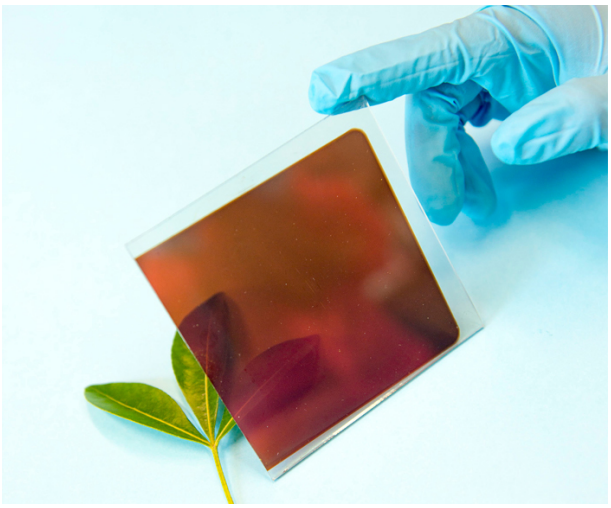


Figura 2.16 : célula solar de perovskita en vidrio, cortesía ce physicsworld.com

Son el tipo de células más recientemente descubiertas, y, por tanto, bajo desarrollo, y al parecer con un largo recorrido, debido a que, a pasos agigantados, se están desarrollando nuevas células con éste mineral [27] (perovskita, $CaTiO_3$ [12]). Con menos de 4 años de recorrido, ésta tecnología has sido capaz de casi duplicar sus eficiencias de laboratorio en diferentes dispositivos, con una eficiencia original en un dispositivo creado por la EPFL [6] de alrededor del 12 % hasta un 22.1 % [27] en un dispositivo creado por KRICT[19]/UNIST[40]

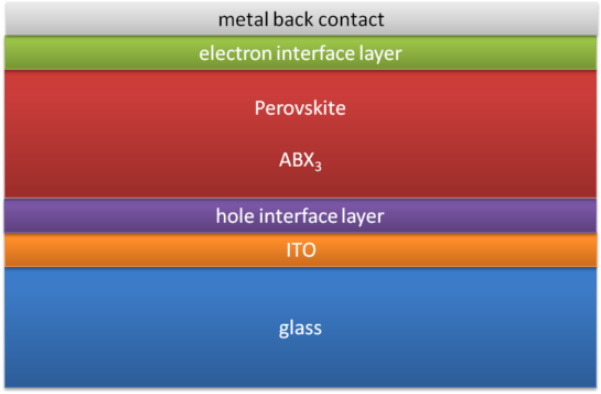


Figura 2.17 : esquema general de una célula de perovskita, cotesía de ossila.com

2.4. Empresas y módulos dominantes en el mercado

Según un estudio de mercado realizado por PV-Tech[29], los líderes mundiales en fabricación de paneles solares son las siguientes empresas:

Puesto	Empresa
1	JinkoSolar
2	Trina Solar
3	Canadian Solar
4	Hanwha Q-CELLS
5	JA Solar
6	GCL
7	First Solar
8	Yingli Green
9	Talesun
10	Risen

Cuadro 2.1 Clasificación de los mayores productores de módulos fotovoltaicos a nivel internacional

En 2015, el mercado ha estado liderado completamente por las células fotovoltaicas con tecnología de capa gruesa basada en silicio, con un 93 % del total producido[13], y ahora, la tecnología líder es la célula policristalina, con un total de cuota de producción del 68 % mundial. Detrás de la tecnología policristalina[13], se sitúan las células monocristalinas y las de silicio amorfo (a-Si:H), la última con una producción total en 2015 del 8 % mundial[13]. Las demás tecnologías suelen estar reservadas para pequeñas instalaciones y aplicaciones muy concretas.

Para la posterior evaluación del simulador desarrollado, se elegirá el módulo más popular, más fabricado o más vendido de cada uno de los fabricantes mencionados en la clasificación anterior:

Empresa	Módulo
JinkoSolar	JKM315P-72
Trina Solar	TSM-DD14A-365
Canadian Solar	Dymond CS6X-340M-FG
Hanwha Q-CELLS	Q.PLUS L-G4.1 340
JA Solar	JAM6 60/270
GCL	N/A
First Solar	Series 4^{TM} FS-4122A-3
Yingli Green	YGE 60 CELL Series 2 275
Talesun	HIPRO TP672M 350
Risen	Double Glass Module RS250P660

Cuadro 2.2 Paneles solares elegidos por cada empresa

2.5. Resumen del análisis del estado del arte y de mercado

La tecnología solar fotovoltaica está creciendo a ritmos mucho mayores que el resto de energías renovables, por todas las ventajas que supone respecto a todas sus competidoras, y sus decrecientes precios, incentivos y nuevas tecnologías de célula, que prometen mayor versatilidad (células de capa delgada, células integrables en construcciones, etc), mayor durabilidad y, sobre todo, mayor eficiencia (células de capa delgada multicapa).

En la actualidad, los módulos policristalinos dominan el mercado de grandes instalaciones, debido a su gran ratio precio/eficiencia, ya que la superficie no es un factor limitante. En aplicaciones donde la superficie es clave, como son las instalaciones de autoconsumo, domésticas, aeroespaciales, etc, sigue dominando la tecnología monocristalina, aunque en el futuro pueden ser sustituidas por competidoras con mayor eficiencia, como las células multicapa. La tecnología de silicio amorfo tiene mucho menor impacto en el mercado, aunque se va haciendo hueco por crecientes eficiencias, versatilidad debido a su flexibilidad y facilidad y bajo coste de fabricación.

El resto de tecnologías todavía siguen en vías de desarrollo, ya que ninguna ha alcanzado una rentabilidad que supere a las células basadas en silicio, que suponen un 93 % del mercado.

Capítulo 3

Simulación de células y módulos solares

Una vez conocida la situación actual del mercado y el estado del arte de la tecnología solar fotovoltaica, se procede a diseñar e implementar un simulador con capacidad para calcular la producción de energía de una instalación, dado el tipo de panel. Éste debe ser capaz de generar unos resultados a partir de datos reales, los cuales se obtendrán mediante algún proveedor de datos meteorológicos detallados que contengan cifras sobre la irradiancia en las localizaciones en las que se vaya a realizar una instalación, las cuales serán leídas por el programa, y dependiendo del panel elegido y la configuración de la instalación (número de paneles, conexión, etc) calculará todos los datos necesarios para analizar la propuesta.

3.1. Hito Nº1: análisis de propuestas de simuladores existentes

Antes de comenzar a realizar un simulador, conviene conocer la actual oferta de programas que cumplan con el mismo objetivo, para evaluar la posibilidad de usar diferentes alternativas ya existentes en vez de crear un simulador desde cero.

- OrientSol:

Figura 3.1 : ventana principal del Software OrientSol, cortesía de la Escuela Politécnica Superior de Jaén

Programa gratuito desarrollado en la Escuela Politécnica Superior de Jaén [7] por la alumna Miriam Jiménez Torres y los profesores Leocadio Hontoria García y Catalina Rus casas. Tiene un set predeterminado de ciudades sobre las cuales se pueden hacer diferentes estudios de situación (6 estudios de localización diferentes), con datos diarios y mensuales. En nuestro caso, éste acercamiento no es válido, ya que no se centra en un modelo de simulación de célula solar, ni módulo solar.

- PVSyst:

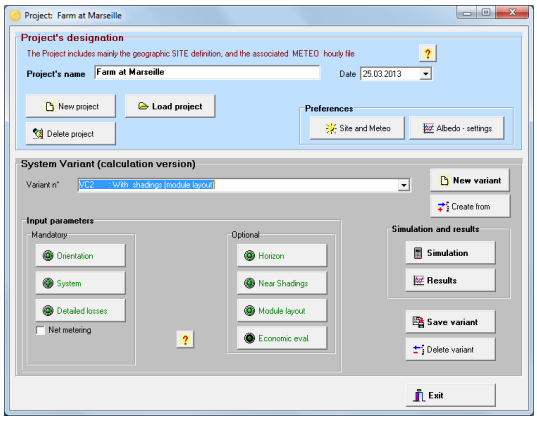


Figura 3.2 : ventana principal del Software PVSyst, cortesía de PVSyst [31]

Programa de simulación de instalaciones comercial [31], a un coste de licencia de alrededor de los 1300$, posee todas las funcionalidades necesarias para hacer los cálculos y estimaciones de una instalación hasta el diseño del proyecto, con un gran abanico de uso de datos meteorológicos de diversa procedencia. Es muy completo, lleva de principio a fin la estimación y diseño de

un proyecto de energía solar fotovoltaica, destinado a ingenieros, arquitectos e investigadores.

- PV_LIB: Set de funciones denominado “Photovoltaic Library”, proveniente del proyecto “Photovoltaic Performance Modeling Collaboration” (PVPMC), con librerías de funciones documentadas para la simulación de sistemas y módulos solares, para MATLAB o Python.

- CoBoGUI:

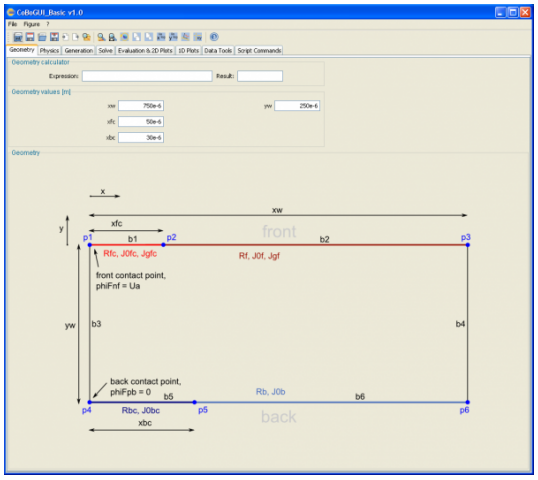


Figura 3.3 : ventana principal del Software CoBoGUI, cortesía de ISFH [15]

Programa denominado “Conductive-Boundary-model Graphical User Interface”, desarrollado por el ISFH [15], usado para una simulación muy completa a nivel físico de una célula solar determinada. Su mayor utilidad es en el campo de la investigación, no siendo útil para aplicaciones de diseño de instalación, cálculo de potencia, etc.

- SpiceGUI:

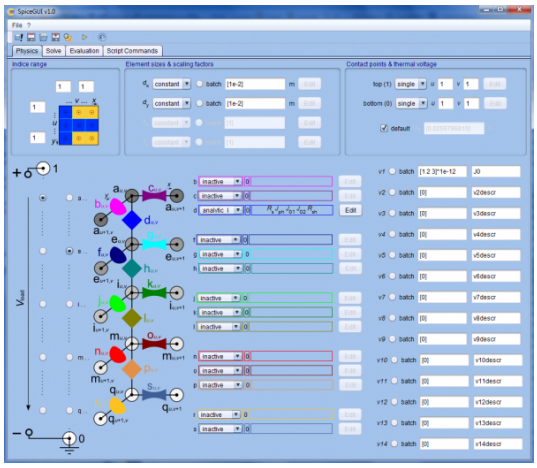


Figura 3.4 : ventana principal del Software SpiceGUI, cortesía de ISFH [15]

Programa de código abierto desarrollado por el ISFH [15], usado para una simulación de circuito equivalente de una célula determinada y desarrollo de dicho modelo.

3.2. Hito Nº2: acotación del trabajo

Modelo matemático

El objetivo primordial y la base del simulador es implementar un programa capaz de calcular la curva de potencia de una célula fotovoltaica dada, por lo que, para empezar, se necesitará un modelo matemático que represente el funcionamiento de una célula solar, que debe cumplir un compromiso entre rapidez y precisión.

Con éste fin se han buscado diferentes maneras de modelar un sistema fotovoltaico. Debido a que hay muchos tipos de tecnologías disponibles en el mercado, existen muchos modelos que representan a las mismas:

- Células basadas en silicio: debido a la semejanza en su funcionamiento, todas ellas se pueden modelar de la misma manera, con la diferencia radicando en los valores de los parámetros del modelo.
  Los modelos de éstas células suelen estar basados en circuitos equivalentes; teniendo más o menos complejidad, se consisten en el mismo principio: un diodo de unión P-N, ya que la construcción de éstas células son básicamente

un diodo de gran superficie. Según las componentes de corriente consideradas, se pueden diferenciar dos modelos:

- Modelo de dos diodos:

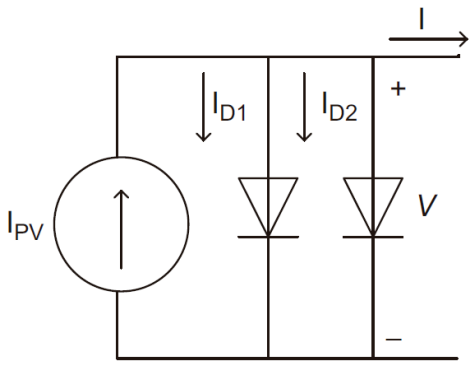


Figura 3.5 : esquema del circuito equivalente con dos diodos, cortesía de Elsevier

Provienen del desarrollo más completo y menos simplificado del enfoque de tratar a la célula solar fotovoltaica como un diodo, salvando las diferencias, que son la generación de corriente mediante el efecto fotovoltaico. Es, en principio, el modelo más preciso, pero puede que esto no se cumpla si se realizan simplificaciones erróneas en el factor de idealidad de los diodos que influyan en las corrientes que representan (a_1 y a_2) [20].

- Modelo de un diodo:

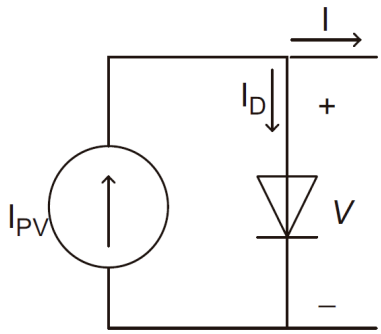


Figura 3.6 : esquema del circuito equivalente con un diodo, cortesía de Elsevier

Proviene de la simplificación del modelo de dos diodos, despreciando la componente de recombinación, ya que a temperatura ambiente, éste

término influye mucho menos que la corriente de difusión [20]. Como la aproximación da pie a pensar, el modelo pierde precisión al trabajar en temperaturas diferentes a las de test en condiciones estándar (STC), por lo que puede ser inviable para análisis en localizaciones en las que la temperatura cambia enormemente, como pueden ser las superficies desérticas.

Éstos dos modelos pueden ser mejorados añadiendo un parámetro de resistencia Shunt (Rs) y resistencia en paralelo (Rs):

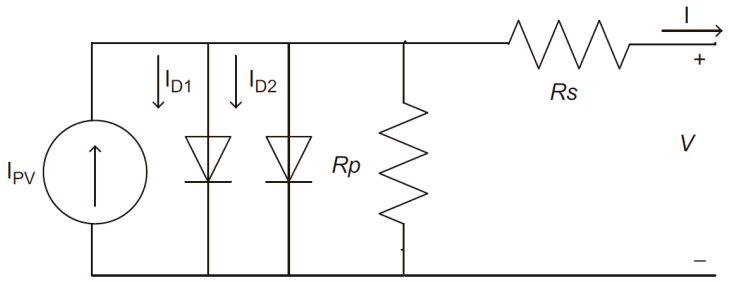


Figura 3.7 : esquema del circuito equivalente con dos diodos completo con resistencias Rs y Rp, cortesía de Elsevier

- Células CIGS: según un trabajo científico sobre el uso del modelo de dos diodos aplicado a células CIGS [3], dicho modelo se puede aplicar para éstas células “en un amplio rango de irradiancias y condiciones de temperatura de la célula” de forma válida.
- Células orgánicas: según un artículo científico sobre la simulación de células solar orgánicas [2], el modelo de un sólo diodo es inadecuado para éste tipo de células, mientras que el de dos diodos lo representa de manera más o menos precisa, variando el enfoque de dicho modelo para diferentes irradiancias.

Para los otros dos tipos de tecnologías mencionadas en el estudio del estado del arte (Cd-Te y Perovskita), los modelos exceden el compromiso de complejidad, con modelos físicos muy difíciles de implementar y de usar.

Para éste trabajo, se ha optado por utilizar el modelo de un diodo y un modelo de dos diodos simplificado, para una posterior comparación en precisión y tiempos de computación.

Implementación

Para esta aplicación se ha optado por tener la posibilidad de utilizar un lenguaje con capacidad para diseño por bloques funcionales, para una mayor sencillez y comprensibilidad del simulador desarrollado. Para éste fin, se ha elegido la herramienta SIMULINK de MathWorks , que será complementada con el lenguaje MATLAB para procesar los datos generados por el simulador creado en SIMULINK[1] [21]. Se han contemplado otros lenguajes, pero han sido descartados por la versatilidad y potencia que ofrecen las herramientas matemáticas de MathWorks, además del conocimiento previo adquirido debido al extenso trabajo realizado con estas herramientas a lo largo de la carrera.

3.3. Hito Nº3: desarrollo del modelo matemático del simulador

Una vez acotado el trabajo y decididas las funcionalidades a implementar, pasamos al desarrollo de los modelos matemáticos elegidos para las simulaciones.

La célula solar funciona como un diodo, el cual, estando conectado a una carga, al recibir iluminación sobre la capa dopada N genera una diferencia de potencial y una corriente fotogenerada. La corriente de una célula solar fotovoltaica de tipo P-N tiene dos componentes básicas: corriente fotogenerada I_{pv}, que es la interesante a la hora de generar corriente y se produce por la generación de portadores por irradiación, y la corriente de oscuridad I_D, que es opuesta a I_{pv} y se produce por recombinación de portadores por el voltaje externo, necesario para que la célula entregue corriente a la carga.

$$I = I_{pv} - I_D(V) \tag{3.1}$$

Ésta es la ecuación fundamental del diodo, con la corriente fotogenerada como positiva, debido a que es el principal interés en una célula solar [20].

La corriente fotogenerada depende de la irradiancia incidente en la célula, y se

[1] Además, a la hora de implementar el modelo matemático elegido, se ha observado que SIMULINK ha sido mucho más rápido a la hora de resolver la ecuación del modelo que la resolución simbólica de MATLAB, consiguiendo realizar alrededor de 1000000 puntos en lo que MATLAB realiza 100

calcula a partir de la intensidad de cortocircuito de la célula, su constante de temperatura respecto a la intensidad de cortocircuito y la irradiancia en condiciones de test estándar [18]:

$$I_{pv} = (I_{pv,STC} + K_I \Delta T) \frac{G}{G_{STC}} \tag{3.2}$$

Donde:

- I_{pv}: corriente fotogenerada, en amperios (A)
- $I_{pv,STC}$: corriente fotogenerada por la célula en condiciones de test estándar (STC), en amperios (A)

$$I_{pv,STC} = I_{sc}$$

- K_I: coeficiente de temperatura de la corriente de cortocircuito (I_{sc}), en miliamperios por unidad de temperatura ($mA/^{\circ}C$)
- ΔT: diferencia de temperatura de la célula respecto a la temperatura STC

$$\Delta T = T - T_{STC}$$

  - T: temperatura de unión de la célula, en grados Kelvin (K)
  - T_{STC}: temperatura de unión de la célula en STC, en grados Kelvin (K)

- G: irradiancia incidente sobre la superficie de la célula, en vatios por unidad de superficie (W/m^2)
- G_{STC}: irradiancia en STC, en vatios por unidad de superficie (W/m^2)

Ésta corriente es común a todos los modelos (uno o dos diodos), y posee un término que evalúa el efecto de la temperatura de la célula en la corriente fotogenerada.

Una vez definida la corriente fotogenerada, nos queda por resolver la corriente de oscuridad, que está formada por las siguientes componentes [20]:

- I_{o1}: representa la componente de difusión de los minoritarios.
- I_{o2}: representa la componente de generación/recombinación.

Éstas corrientes corresponden a los dos diodos del circuito equivalente de la célula solar. En su versión más completa y precisa, se representan las dos corrientes, pero se puede simplificar despreciando la corriente de generación/recombinación.

A partir de aquí, se desarrollarán los modelos de forma separada:

- Modelo de un diodo: se parte del siguiente esquema básico:

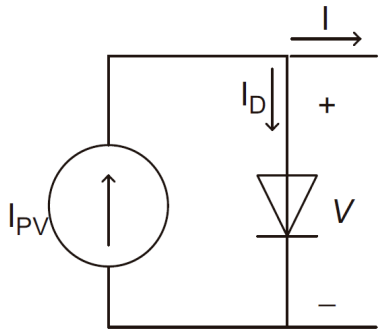


Figura 3.8 : esquema del circuito equivalente con un diodo, cortesía de Elsevier

Al cual, para mayor precisión, se le añade el parámetro de la resistencia Shunt (R_s):

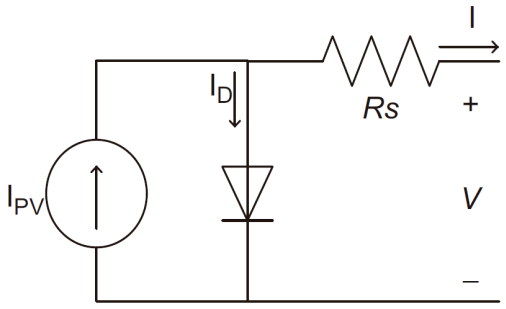


Figura 3.9 : esquema del circuito equivalente con un diodo con resistencia Shunt, cortesía de Elsevier

Cuya ecuación es la siguiente:

$$I = I_{pv} - I_o \left[exp\left(\frac{V + IR_s}{aV_T} \right) - 1 \right] \tag{3.3}$$

Donde:

- I: corriente a través de la célula fotovoltaica, en amperios (A)
- V: voltaje administrado a la célula, en voltios (V)
- I_{pv}: corriente fotogenerada, descrita en la ecuación 3.2, en amperios (A)
- I_o: corriente de difusión, descrita en la ecuación 3.4, en amperios (A)
- R_s: resistencia Shunt del circuito equivalente, en ohmios (Ω)

- a: factor de idealidad del diodo, que según la teoría de difusión de Shockley, debe ser la unidad [33]
- V_T: voltaje termal del diodo, descrito en la ecuación 3.5 en voltios (V)

En el caso del modelo de un diodo, la corriente de difusión es descrita por la siguiente ecuación mejorada, que considera la temperatura [18]:

$$I_o = \frac{(I_{sc} + K_I \Delta T)}{exp\left[(V_{oc} + K_V \Delta T)\, aV_T\right] - 1} \tag{3.4}$$

Donde:

- I_o: corriente de difusión, en amperios (A)
- I_{sc}: corriente de cortocircuito, en amperios (A)
- V_{oc}: tensión de circuito abierto, en voltios (V)
- K_I: coeficiente de temperatura de la corriente de cortocircuito (I_{sc}), en amperios por unidad de temperatura ($A/°C$)
- K_V: coeficiente de temperatura del voltaje en circuito abierto (V_{oc}), en voltios por unidad de temperatura ($V/°C$)
- V_T: voltaje termal del diodo, descrito en la ecuación 3.5, en voltios (V)
- a: factor de idealidad del diodo

También hay que definir el voltaje termal del diodo, el cual responde a la siguiente ecuación:

$$V_T = \frac{N_s k T}{q} \tag{3.5}$$

Donde:

- V_T: voltaje termal del diodo, en voltios (V)
- N_s: número de células en el módulo
- T: temperatura absoluta de la unión P-N de la célula, en grados Kelvin (K)
- k: constante de Boltzman, en Julios por grado Kelvin ($= 1{,}38064852x10^{-23} J/K$)
- q: carga del electrón, en Culombios ($= 1{,}60217662x10^{-19} C$)

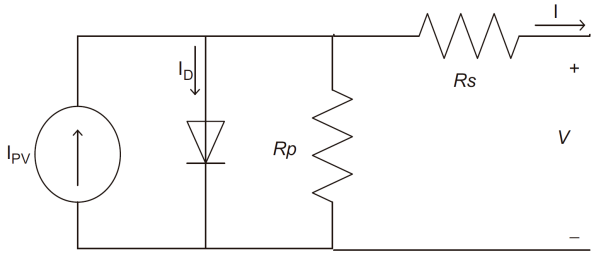


Figura 3.10 : esquema del circuito equivalente con un diodo con resistencia Shunt y resistencia en paralelo, cortesía de Elsevier

Como paso final para el desarrollo del modelo completo de un diodo, se añade un parámetro más: la resistencia en paralelo del circuito equivalente (R_p), que se añade a la ecuación 3.3:

$$I = I_{pv} - I_o \left[exp\left(\frac{V + IR_s}{aV_T}\right) - 1 \right] - \left(\frac{V + IR_s}{R_p}\right) \tag{3.6}$$

Donde:

- I: corriente a través de la célula fotovoltaica, en amperios (A)
- V: voltaje administrado a la célula, en voltios (V)
- I_{pv}: corriente fotogenerada, descrita en la ecuación 3.2, en amperios (A)
- I_o: corriente de difusión, descrita en la ecuación 3.4, en amperios (A)
- R_s: resistencia Shunt del circuito equivalente, en ohmios (Ω)
- R_p: resistencia en paralelo del circuito equivalente, en ohmios (Ω)
- a: factor de idealidad del diodo
- V_T: voltaje termal del diodo, descrito en la ecuación 3.5 en voltios (V)

Éste modelo posee varios parámetros, pero debido a su sencillez, sólo necesita una computación previa de las resistencias

- Modelo de dos diodos:

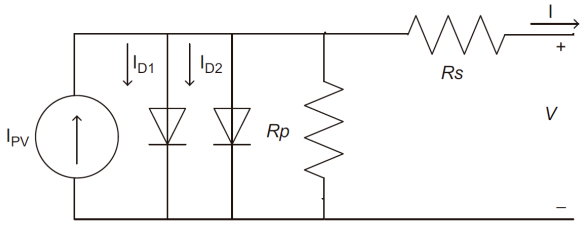


Figura 3.11 : esquema del circuito equivalente con dos diodos con resistencia Shunt y resistencia en paralelo, cortesía de Elsevier

Éste modelo es el desarrollo completo, incluyendo la corriente de generación/recombinación, correspondiente al otro diodo. Queda descrito por la siguiente ecuación [18]:

$$I = I_{pv} - I_{o1}\left[exp\left(\frac{V + IR_s}{a_1 V_T}\right) - 1\right] - I_{o2}\left[exp\left(\frac{V + IR_s}{a_2 V_T}\right) - 1\right] - \left(\frac{V + IR_s}{R_p}\right) \tag{3.7}$$

Donde:

- I: corriente a través de la célula fotovoltaica, en amperios (A)
- V: voltaje administrado a la célula, en voltios (V)
- I_{pv}: corriente fotogenerada, descrita en la ecuación 3.2, en amperios (A)
- I_{o1}: corriente de difusión, en amperios (A)
- I_{o2}: corriente de generación/recombinación, en amperios (A)
- R_s: resistencia Shunt del circuito equivalente, en ohmios (Ω)
- R_p: resistencia en paralelo del circuito equivalente, en ohmios (Ω)
- a_1: factor de idealidad del diodo de difusión
- a_2: factor de idealidad del diodo de generación/recombinación
- V_T: voltaje termal del diodo, en voltios (V)

Ésta ecuación requiere de resolución iterativa para varios parámetros, lo que la convierte en una alternativa extremadamente lenta, aunque más precisa. Para evitar tantos cálculos, se puede simplificar de varias maneras, pero en éste caso, seguiremos la simplificación realizada en un trabajo de investigación [18], el cual describe un modelo muy rápido a la hora de computarlo, y con un gran rango de condiciones de temperatura, manteniendo una gran precisión.
Dicha aproximación se basa en igualar las corrientes de difusión y generación recombinación y asignando unos parámetros fijos a los factores de idealidad (a_1 se iguala a la unidad y a_2 se iguala a 1.2):

$$I_o = I_{o2} = I_{o1} = \frac{(I_{sc} + K_I \Delta T)}{exp\left[(V_{oc} + K_V \Delta T)\left\{(a_1 + a_2)/p\right\} V_T\right] - 1} \tag{3.8}$$

Donde:

- I_o: corriente de difusión, en amperios (A)
- I_{sc}: corriente de cortocircuito, en amperios (A)

- V_{oc}: tensión de circuito abierto, en voltios (V)
- K_I: coeficiente de temperatura de la corriente de cortocircuito (I_{sc}), en amperios por unidad de temperatura ($A/°C$)
- K_V: coeficiente de temperatura del voltaje en circuito abierto (V_{oc}), en voltios por unidad de temperatura ($V/°C$)
- V_T: voltaje termal del diodo, descrito en la ecuación 3.5, en voltios (V)
- a_1: factor de idealidad del diodo correspondiente a la corriente de difusión
- a_2: factor de idealidad del diodo correspondiente a la corriente de generación/recombinación
- p: factor usado para compensar los factores de idealidad

$$p = a_1 + a_2$$

Lo que simplifica la ecuación de dos diodos a la siguiente expresión [18]:

$$I = I_{pv} - I_o \left[exp\left(\frac{V + IR_s}{V_T}\right) + exp\left(\frac{V + IR_s}{(p-1)\,V_T}\right) + 2 \right] - \left(\frac{V + IR_s}{R_p}\right) \tag{3.9}$$

Donde:

- I: corriente a través de la célula fotovoltaica, en amperios (A)
- V: voltaje administrado a la célula, en voltios (V)
- I_{pv}: corriente fotogenerada, descrita en la ecuación 3.2, en amperios (A)
- I_{o1}: corriente de difusión, en amperios (A)
- I_{o2}: corriente de generación/recombinación, en amperios (A)
- R_s: resistencia Shunt del circuito equivalente, en ohmios (Ω)
- R_p: resistencia en paralelo del circuito equivalente, en ohmios (Ω)
- V_T: voltaje termal del diodo, en voltios (V)
- p: factor usado para compensar los factores de idealidad

La cual tiene menos parámetros, y los únicos que hay que realizar de forma iterativa son las resistencias R_s y R_p. Una vez se obtienen sus valores, no necesitan ser recalculados para simular la misma célula.

En resumen:

- Modelo de un diodo: utilizará la ecuación principal 3.6, y adicionalmente necesitará las ecuaciones para el cálculo de I_o (3.4) y de V_T (3.5).
- Modelo de dos diodos: se utilizará el modelo propuesto por un artículo científico [18], utilizando la ecuación principal 3.9, y adicionalmente necesitará las ecuaciones para el cálculo de I_o (3.8) y de V_T (3.5).

Ahora, tendiendo los dos modelos definidos, pasaremos a la obtención de los parámetros por cálculos iterativos, que en este caso son las resistencias R_s y R_p.
Su cálculo se basa en el seguimiento del punto de máxima potencia: se despeja la ecuación de la célula para el punto de máxima potencia experimental ($P_{mp,e}$) y se compara con el obtenido mediante el cálculo con el modelo y los valores de resistencia elegidos ($P_{mp,c}$). A continuación se describirán las ecuaciones necesarias para dicho cálculo, las condiciones iniciales de los valores y el método a seguir para el cálculo iterativo, que será descrito en un diagrama de flujo:

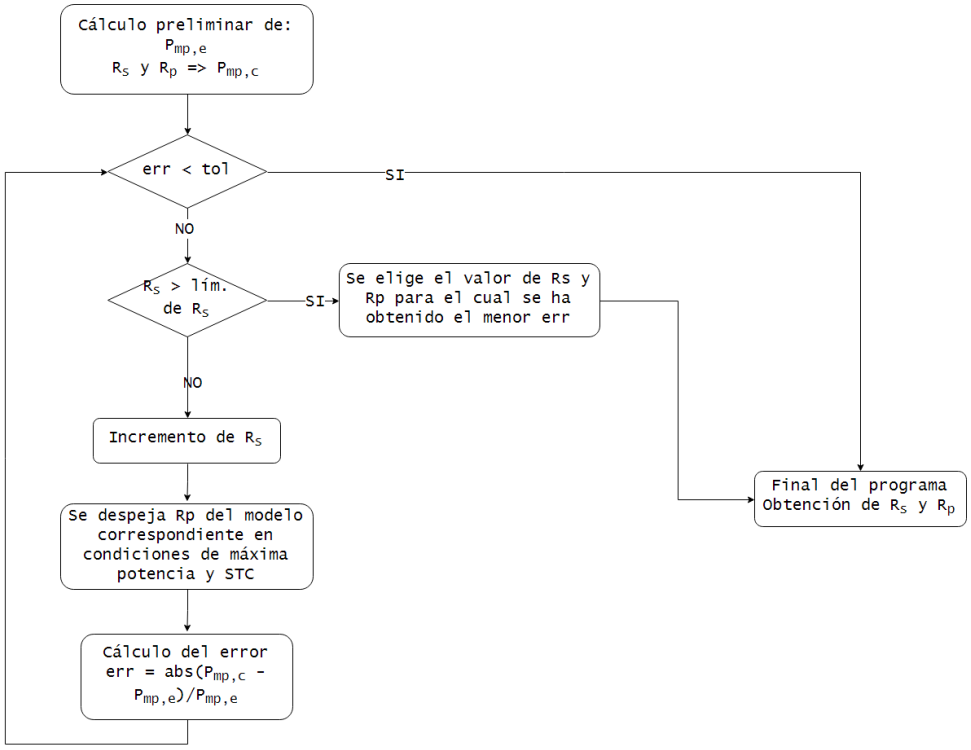


Figura 3.12 : diagrama de flujo de la obtención de R_s y R_p para los dos modelos, fuente propia

- Modelo de un diodo: Para empezar, se asignan los siguientes valores iniciales

a las resistencias [18]:

$$\begin{cases} R_s = 0 \\ R_p = \frac{V_{mp}}{I_{sc}-I_{mp}} - \frac{V_{oc}-V_{mp}}{I_{mp}} \end{cases}$$

Una vez tenemos éstos valores iniciales, se procede al cálculo del punto de máxima potencia ($P_{mp,c}$), usando éstos parámetros y la ecuación del modelo de un diodo, 3.6.

Después del primer cálculo, se compara el error entre el puntos de máxima potencia experimental y el calculado con la tolerancia deseada. Si está dentro de la tolerancia, aceptamos los valores de R_s y R_p iniciales. En caso negativo, se recalcula el valor de las resistencias de la siguiente manera: se incrementa el valor de R_s y el valor de R_p se despeja de la ecuación 3.6 con los valores de máxima potencia [18]:

$$R_p = \frac{V_{mp} + I_{mp}R_s}{I_{pv,ref} - I_o\left[exp\left(fracV_{mp} + I_{mp}R_sV_T\right) - 1\right] - I_{mp}} \tag{3.10}$$

Después de obtener los valores de R_s y R_p, se vuelve a comprobar el error, y se repite este paso hasta que el error entre dentro de la tolerancia, o se elegirán los valores para los que el error sea mínimo.

- Modelo de dos diodos: Para empezar, se asignan los siguientes valores iniciales a las resistencias [18]:

$$\begin{cases} R_s = 0 \\ R_p = \frac{V_{mp}}{I_{sc}-I_{mp}} - \frac{V_{oc}-V_{mp}}{I_{mp}} \end{cases}$$

Una vez tenemos éstos valores iniciales, se procede al cálculo del punto de máxima potencia ($P_{mp,c}$), usando éstos parámetros y la ecuación del modelo de un diodo, 3.7.

Después del primer cálculo, se compara el error entre el puntos de máxima potencia experimental y el calculado con la tolerancia deseada. Si está dentro de la tolerancia, aceptamos los valores de R_s y R_p iniciales. En caso negativo, se recalcula el valor de las resistencias de la siguiente manera: se incrementa el valor de R_s y el valor de R_p se despeja de la ecuación 3.7 con los valores de máxima potencia [18]:

$$R_p = \frac{V_{mp} + I_{mp}R_s}{I_{pv,ref} - I_o\left[exp\left(fracV_{mp} + I_{mp}R_sV_T\right) + exp\left(\frac{V_{mp}+I_{mp}R_s}{(p-1)V_T}\right) + 2\right] - I_{mp}} \tag{3.11}$$

Después de obtener los valores de R_s y R_p, se vuelve a comprobar el error, y se repite este paso hasta que el error entre dentro de la tolerancia, o se elegirán los valores para los que el error sea mínimo.

Una vez determinado por completo el modelo ideal pasaremos a definir y estimar las pérdidas de célula solar y también del módulo en general. Debido a que éste trabajo se centra en el desarrollo de un simulador de célula y módulo solar y no se incluyen elementos posteriores (convertidor DC/AC, seguidor de MPP, inversor, etc), sólo se estudiarán las pérdidas que no excedan éste alcance. Dichas pérdidas son las siguientes:

- Pérdidas por pérdida de potencia nominal: se definen como incumplimientos de potencia nominal de fábrica del módulo, y se aplican sobre la potencia obtenida por simulación. Se pueden deber a diferentes factores:

  - Pérdidas de fábrica: pérdidas de potencia del panel desde fábrica, estén dentro o fuera de la tolerancia establecida. Según datos del NREL [27] obtenidos en una noticia de Energy Informative [11], la potencia perdida de fábrica para los principales fabricantes de módulos solares se encuentra entre un 2 y un 10 %, con una media de un 3 % en general.

    $$P_{factory} = P_i \cdot (1 - F.L.)$$

    Donde *F.L.* son las pérdidas de fábrica (Factory Loss).

  - Pérdidas lineales por vida útil: se deben al deterioro del módulo a lo largo de su vida útil. La mayoría de fabricantes, sobre todo los principales, dan garantía de pérdidas de potencia lineales de 25 años, comenzando con su potencia nominal y bajando hasta una media de un 80 % de su potencia nominal al final de dicho período de garantía. Atendiendo a su tiempo de uso, se podría estimar el porcentaje de pérdidas de potencia por vida útil, hasta los 25 años; posterior a esa garantía, dichas pérdidas seguirían otra tendencia que no estudiaremos en este trabajo, ya que depende de demasiadas variables (ruptura de aislamientos, pulimento de las superficies acristaladas, oxidación de contactos, etc).

    $$P = P_{factory} \cdot (1 - 0{,}8 \cdot y)$$

    Donde *y* son los años de funcionamiento del módulo.

- Pérdidas por conexionado de paneles diferentes: al conectar paneles diferentes se limita la potencia total producida a la del panel con la menor producción

(limita la corriente si se conecta en serie y la tensión en paralelo). Aun entrando en el ámbito del trabajo, éste tipo de pérdidas no se tendrán en cuenta, debido a que en el simulador se suponen instalaciones homogéneas de un sólo tipo de módulo.

- Pérdidas de irradiancia por ángulo: debidas al ángulo de colocación de los paneles. En éste trabajo no se implementarán dichas pérdidas de irradiancia, ya que el programa usado para la obtención de los datos meteorológicos, Meteonorm [22], ya es capaz de realizar dichos cálculos, así como la orientación y adicionales.

- Pérdidas de caída de voltaje: las conexiones entre paneles cercanos son lo suficientemente cortas como para despreciar dichas pérdidas. Éstas pueden volverse considerables cuando se conectan a los diferentes elementos de una instalación.

- Pérdidas por temperatura: incluidas en el modelo matemático, ya que la temperatura del módulo influye en dichos cálculos.

- Pérdidas por suciedad: perdidas debidas al acumulamiento de polvo y diferentes materiales que reducen la irradiancia recibida por el panel, y además crean puntos donde se acentúa la temperatura, creando pérdidas irregulares. Éstas pérdidas afectan mayoritariamente la irradiancia recibida por el módulo, por lo que el cómputo de pérdidas se realizará sobre éste dato. Según varios estudios [23], las pérdidas de irradiancia se encuentran en una media del 4.7 % para Estados Unidos y entre un 33 y un 65 % en Arabia Saudi y Kuwait, respectivamente, entre otros. Éste dato es complejo de estimar, por lo que se recomienda la introducción manual de éste.

$$G_{Gh,soiling} = G_{Gh} \cdot (1 - S.F.)$$

Donde *S.F.* es el factor de suciedad (Soiling Factor).

3.4. Hito Nº4: implementación del simulador

Como se ha discutido en secciones anteriores, se ha decidido usar la herramienta SIMULINK para el diseño de los bloques y el lenguaje MATLAB para el procesamiento de los resultados generados por el simulador.

A continuación, se hará una descripción detallada de la implementación de los modelos matemáticos desarrollados en la sección anterior:

- Modelo de un diodo: como en las ecuaciones matemáticas, dividiremos los subgrupos los elementos que necesiten para su cálculo otra ecuación, a excepción de las resistencias, que, como se ha mencionado en la sección anterior, deben ser calculadas de forma iterativa.
  Primero se implementan los tres subgrupos correspondientes a los valores de V_t (3.5), I_o (3.4) e I_{pv} (3.2):

  - V_t: se implementa tal y como se indica en la ecuación 3.5. Común para los dos modelos.

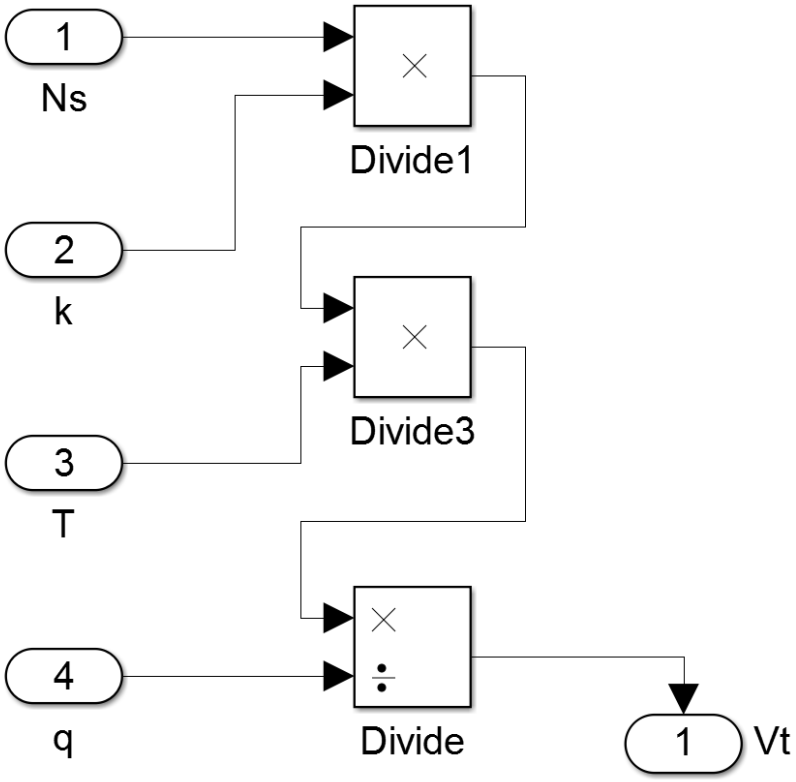


Figura 3.13 : diagrama de bloques de la ecuación de V_t para el modelo de 1 diodo

  - I_o: se implementa tal y como se indica en la ecuación 3.4. Ya que el elemento a siempre es la unidad, se omite la multiplicación por dicha constante.

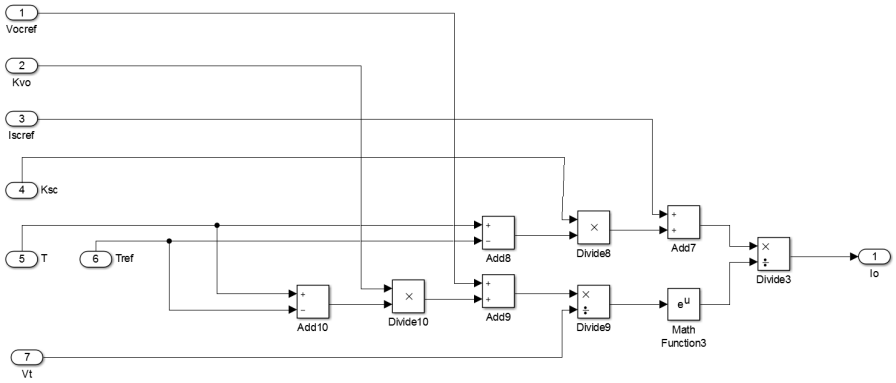


Figura 3.14 : diagrama de bloques de la ecuación de I_o para el modelo de 1 diodo

- I_{pv}: se implementa tal y como se indica en la ecuación 3.2. Común para los dos modelos.

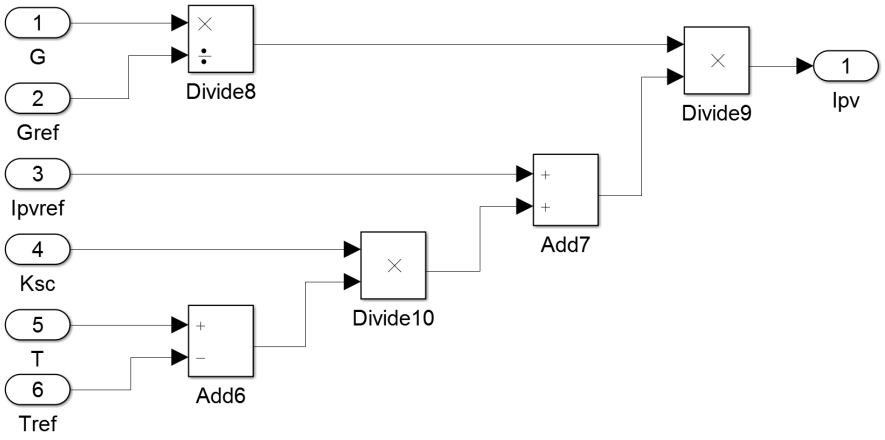


Figura 3.15 : diagrama de bloques de la ecuación de I_pv para el modelo de 1 diodo

Una vez tenemos los tres subgrupos para los términos más complejos, se procede a implementar el resto de la ecuación general del modelo de 1 diodo (3.6).

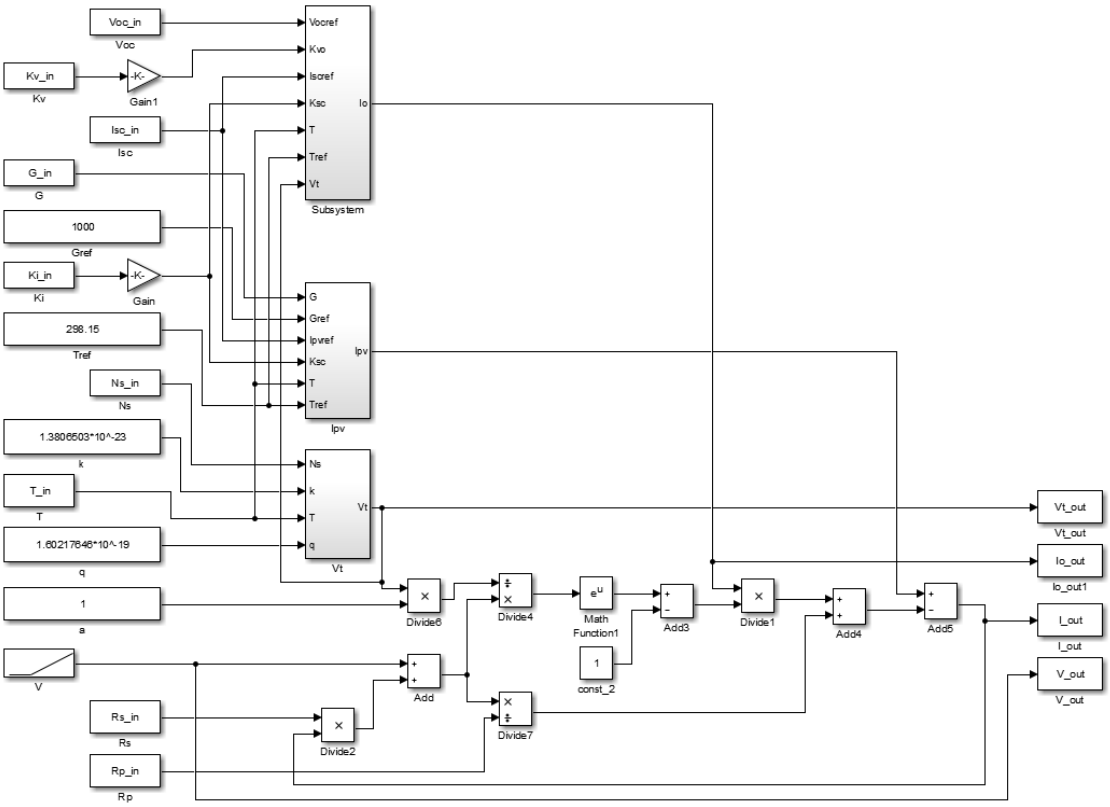


Figura 3.16 : diagrama de bloques de la ecuación general del modelo de 1 diodo

Éste modelo se ha implementado de tal manera que sea ejecutable de forma sencilla desde un script .m de MATLAB, por lo que todas las variables de entrada que no son constantes (irradiancia, temperatura del módulo, etc) se introducen como variables el espacio de trabajo utilizado por el script, y se ha seguido un procedimiento similar para las variables de salida del modelo.

- Modelo de dos diodos: Tal y como se ha realizado en el apartado del modelo de un diodo, se implementan primero los subgrupos de términos que necesitan de una ecuación y cálculo aparte y después se añade el resto de los términos y cálculos de la ecuación correspondientes.
  Primero se implementan los tres subgrupos correspondientes a los valores de V_t (3.5), I_o (3.8) e I_pv (3.2):

  - V_t: se implementa tal y como se indica en la ecuación 3.5. Común para los dos modelos.

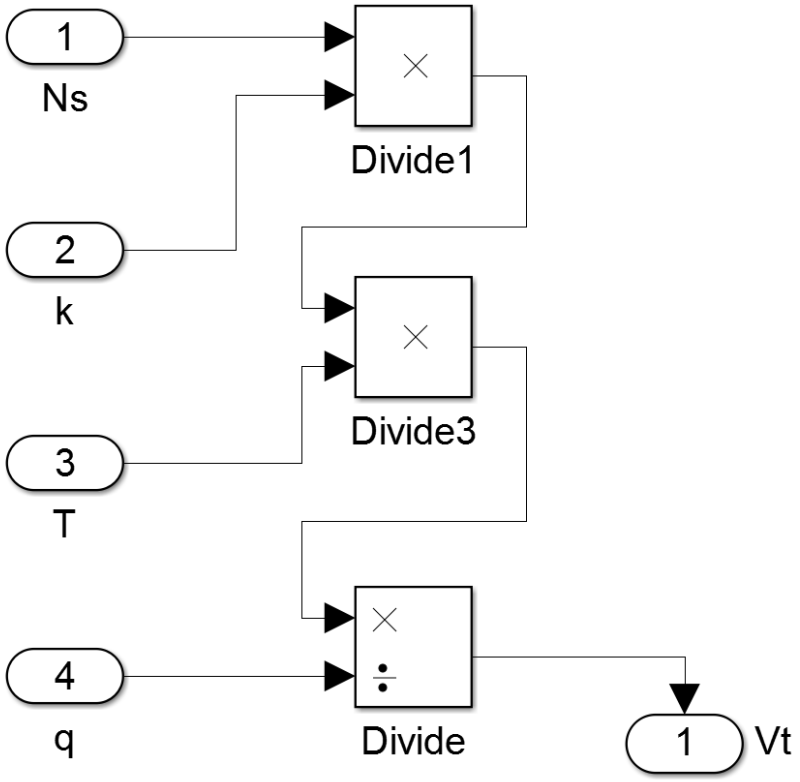


Figura 3.17 : diagrama de bloques de la ecuación de V_t para el modelo de 2 diodos

- I_o: se implementa tal y como se indica en la ecuación 3.8. Debido a que el término $\frac{a_1+a_2}{p}$ siempre es equivalente a la unidad, se ha omitido la multiplicación por dicho bloque constante.

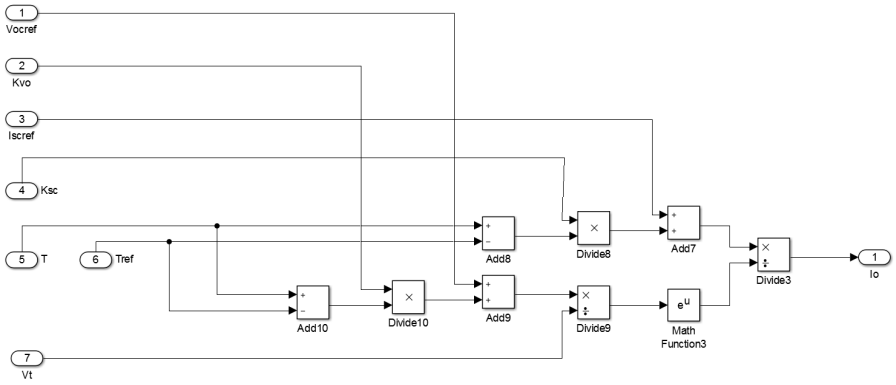


Figura 3.18 : diagrama de bloques de la ecuación de I_o para el modelo de 2 diodos

- I_{pv}: se implementa tal y como se indica en la ecuación 3.2. Común para

los dos modelos.

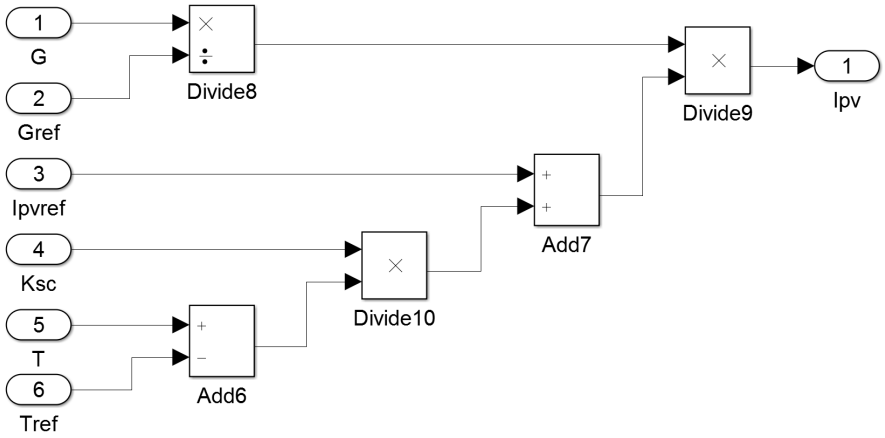


Figura 3.19 : diagrama de bloques de la ecuación de I_pv para el modelo de 2 diodos

Una vez tenemos los tres subgrupos para los términos más complejos, se procede a implementar el resto de la ecuación general del modelo de 2 diodos (3.7).

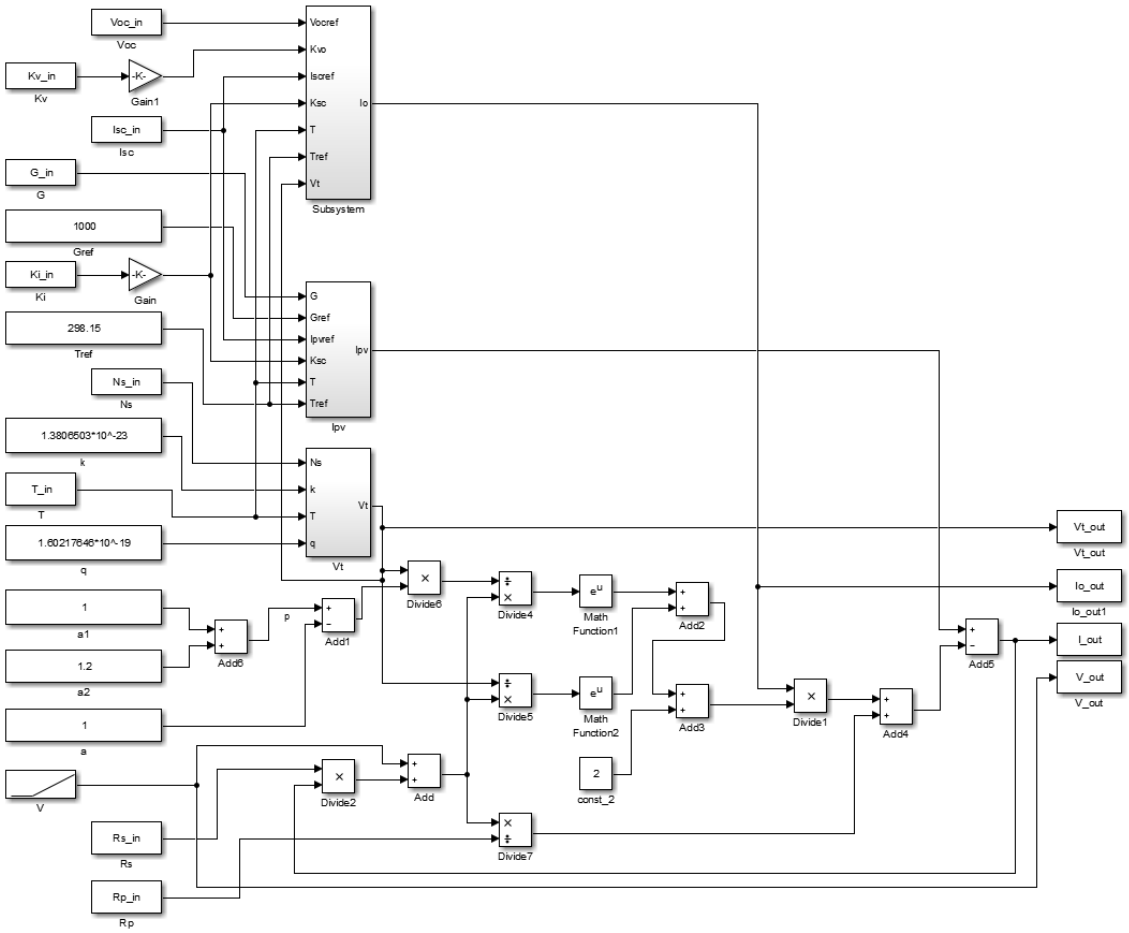


Figura 3.20 : diagrama de bloques de la ecuación general del modelo de 2 diodos

Éste modelo se ha implementado de tal manera que sea ejecutable de forma sencilla desde un script .m de MATLAB, por lo que todas las variables de entrada que no son constantes (irradiancia, temperatura del módulo, etc) se introducen como variables el espacio de trabajo utilizado por el script, y se ha seguido un procedimiento similar para las variables de salida del modelo.

Una vez desarrollados los modelos en SIMULINK, como se ha mencionado anteriormente, éstos se han de usar en programas .m en código MATLAB. Para este fin, se ha desarrollado una función con capacidad de selección de modo, para poder ejecutar el tipo de modelo que se precise, "*fast_diode*" [2].

[2]la palabra "fast" alude a la diferencia de velocidad de cálculo respecto a una implementación en código MATLAB del modelo

Datos de entrada:

1. Voc: tensión de circuito abierto
2. Isc: corriente de cortocircuito
3. Ns: número de células por módulo
4. Kv: coeficiente de temperatura respecto a V_{oc}
5. Ki: coeficiente de temperatura respecto a I_{sc}
6. Rs: vector con las resistencias en serie para los dos modelos
7. Rp: vector con las resistencias en paralelo para los dos modelos
8. G: irradiancia
9. T: temperatura del modulo
10. n_points: Nº de puntos deseados para la curva I-V (precisión). Por defecto, el modelo de SIMULINK utiliza 100000
11. mode: selector de modo de un diodo o dos diodos. Se trata de un dato booleano, si equivale a “false”, se usa el modelo de 1 diodo; si equivale a “true”, se usa el modelo de 2 diodos.

Datos de salida:

- I: vector con “*n_points*” valores con los valores de corriente de la curva I-V
- V: vector con “*n_points*” valores con los valores de tensión de la curva I-V

La llamada a la función se realiza escribiendo el nombre de la función, adjuntando todos los datos necesarios de entrada en el orden anteriormente establecido, precedido de los vectores de valores devueltos entre corchetes.

$$[I, V] = fast\_diode\,(Voc, Isc, Ns, Kv, Ki, Rs, Rp, G, T, n\_points, mode)$$

Como se mencionó en la sección del desarrollo matemático del modelo, se necesita calcular de forma iterativa las resistencias en serie y paralelo de los dos modelos para poder utilizarlos sin tener que incurrir en cálculos iterativos, para lo cual se han desarrollado dos funciones diferentes en dos ficheros .m separados, usadas para calcular las resistencias en serie de los modelos.

- "*fast_1_diode_resget*": script para el cálculo de las resistencias en serie y paralelo de un módulo determinado para el modelo de 1 diodo. La llamada se realiza de la siguiente manera:

$$fast\_1\_diode\_resget\,(Voc, Isc, Vmp, Imp, Ns, Kv, Ki)$$

- "*fast_2_diode_resget*": script para el cálculo de las resistencias en serie y paralelo de un módulo determinado para el modelo de 2 diodos. La llamada se realiza de la siguiente manera:

$$fast\_2\_diode\_resget\,(Voc, Isc, Vmp, Imp, Ns, Kv, Ki)$$

Las dos funciones toman los mismos datos de entrada, pero dependiendo de la función usada, se obtienen las resistencias para un modelo u otro.
Los dos scripts descritos cuentan con unos cuadros de diálogo que te permiten definir una estimación inicial de R_s y que avisan del comienzo y final del cálculo, siendo el cuadro final una ventana emergente con los valores de la resistencias obtenidas. Contiene además un sistema de detección de sobreestimación de la resistencia serie, lo que puede permitir un tiempo cálculo más corto[3].

En la implementación de éstos cálculos iterativos, existen dos diferencias principales con el diagrama de flujo descrito en anteriores secciones (figura 3.12):

- Valor inicial de R_p: en lugar de igualar el valor inicial de R_p al valor propuesto, se ha igualado a infinito, para eliminar por completo su efecto sobre el modelo. Posteriormente, se calcula tal y como se describe en las ecuaciones 3.10 para 1 diodo y 3.11 para 2 diodos.

- Condición de parada del cálculo iterativo: debido a que si se llegan a valores de R_s suficientemente altos, la resistencia en paralelo se vuelve negativa y que el error por el calculo iterativo de R_s no es continuamente decreciente, podría darse que no se llegara a unos valores de resistencias adecuados, o que no terminara nunca el cálculo iterativo, por lo que se ha optado por poner como condición de parada que la potencia máxima calculada sea menor que la potencia máxima experimental.

$$P_{mpc} < P_{mpe}$$

[3] Aunque ésta funcionalidad puede ser muy útil para evitar tiempos de cálculo de varios minutos (incluso horas, como se ha dado con un panel Cd-Te de FirstSolar) no es perfecto, y se puede sobreestimar el valor de R_s sin que el programa levante un mensaje de error. Esto se puede detectar observando la estimación final de R_s; si con un solo incremento de dicho valor se ha llegado a un valor conclusivo, es muy probable que se haya sobreestimado R_s.

De esta forma, y poniendo un paso de iteración lo suficientemente pequeño (0.001) nos aseguramos de que el cálculo de dichas resistencias es correcto.

Capítulo 4

Análisis de los modelos implementados

Una vez desarrollados los modelos se procederá a un análisis de los mismos, realizando al final una comparación entre ellos para determinar su precisión y fiabilidad en diferentes condiciones de trabajo.

4.1. Elección de módulo

Para poder realizar una comparación efectiva y poder observar la precisión de los dos modelos entre sí y con los valores reales, necesitaremos elegir un módulo antes. Idealmente, debe cumplir las siguientes características:

- Modelo popular en el mercado, a ser posible fabricado por las compañías mencionadas en el análisis de mercado 2.2
- Disposición de todos los datos necesarios para la ejecución de los modelos
- Disposición de datos experimentales de la curva I-V en diferentes condiciones de funcionamiento

Debido a que en éste caso sólo se desea hacer una comparativa de los modelos y una medida de su precisión, los dos últimos puntos serán predominantes frente al uso de un panel popular, debido a la necesidad de amplia información de laboratorio sobre el módulo que se simulará, y la precisión y fiabilidad de los valores reales.

Debido a que no se ha encontrado ningún recurso que contenga valores experimentales de la curva I-V de algún módulo fotovoltaico, se ha optado por elegir el módulo de CanadianSolar CS6X-340M-FG para realizar las simulaciones. En última

instancia, se compararán los puntos de importancia obtenidos por el simulador en condiciones STC y NOCT y se compararán con los datos reales. Valores a comparar:

- Tensión de circuito abierto, V_{oc}
- Corriente de cortocircuito, I_{sc}
- Potencia máxima, P_{mp}
- Tensión de potencia máxima, V_{mp}
- Corriente de potencia máxima, I_{mp}

Características técnicas y valores del panel elegido:

- $Rendimiento = 17{,}42\,\%$
- $V_{oc} = 46{,}2V$
- $I_{sc} = 9{,}48A$
- $P_{mp} = 340W$
- $V_{mp} = 37{,}9V$
- $I_{mp} = 8{,}97A$
- $N_s = 72$
- $K_v = -143\frac{mV}{^\circ C}$
- $K_i = 5{,}02\frac{mA}{^\circ C}$
- $TONC = 0{,}0288\frac{^\circ Cm^2}{W}$
- $Superficie = 1{,}95m^2$
- $R_{s,1diodo} = 0{,}295\Omega$
- $R_{9,1diodo} = 581{,}4\Omega$
- $R_{s,2diodo} = 0{,}291\Omega$
- $R_{9,2diodo} = 604{,}7\Omega$

4.2. Simulación bajo condiciones diversas

Una vez seleccionado el módulo e identificados sus parámetros, se procederá a generar simulaciones de diversas condiciones en las que se pueda encontrar el módulo fotovoltaico, a fin de observar el comportamiento del simulador ante diferentes valores de irradiancia y temperatura respecto a los valores reales obtenidos en laboratorio.
Primero observaremos el comportamiento de los módulos a las dos condiciones de funcionamiento de laboratorio más comunes, que son las condiciones de test estándar (STC, *Standard Test Conditions*, 1000 $\frac{W}{m^2}$ y 25 $^{\circ}C$) y el punto nominal de funcionamiento (NOCT, *Nominal Operation Cell Temperature*, 800 $\frac{W}{m^2}$ y 20 $^{\circ}C$ de temperatura ambiente, con vientos de 1 $\frac{m}{s}$, lo que equivale a aproximadamente 30.5 $^{\circ}C$ de temperatura de célula en este caso). Una vez simulados estos puntos de funcionamiento, se simularán diferentes condiciones de irradiancia a la misma temperatura (1000, 800, 600, 400 y 200 $\frac{W}{m^2}$ a 25 $^{\circ}C$) y diferentes temperaturas a la misma irradiancia (5, 25, 45 y 65 $^{\circ}C$ a 1000 $\frac{W}{m^2}$). Las dos primeras condiciones de simulación se usarán para comparar los resultados con los valores reales y evaular su fiabilidad, y los resultados a diferentes irradiancias y temperaturas se usarán para realizar una comparación entre los dos modelos de simulación.

A continuación se representarán los resultados obtenidos en las diferentes simulaciones:

- Condiciones de test estándar (STC):

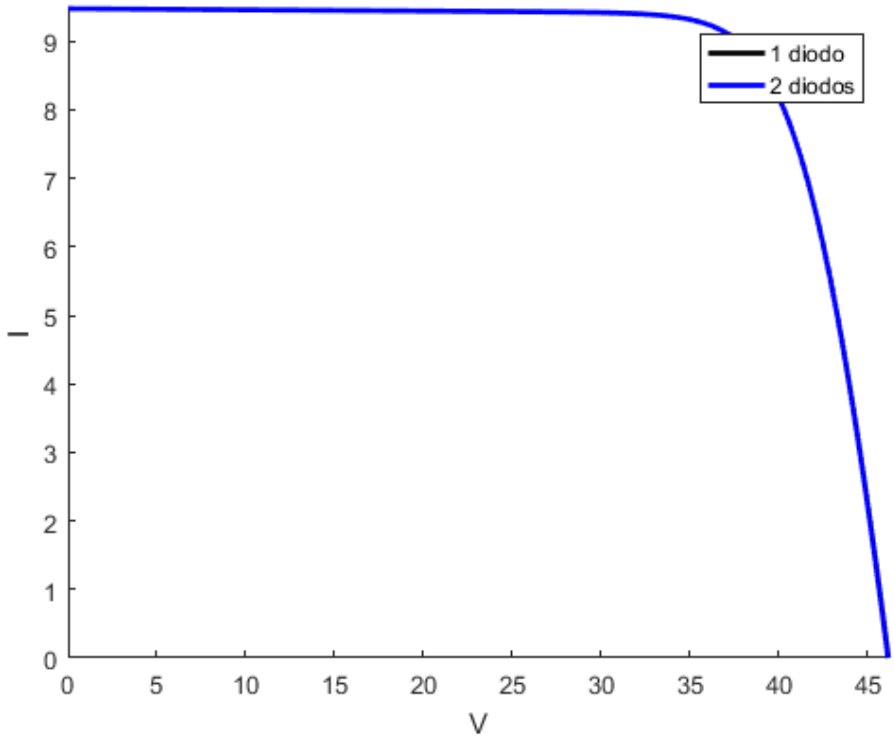


Figura 4.1 : comparación de modelos en condiciones de test estándar (STC)

- Modelo de 1 diodo: para éste modelo, se han obtenido los siguientes errores frente a los valores reales:
  - V_{oc} : 0,033 %
  - I_{sc} : 0,051 %
  - P_{mp} : 0,067 %
  - V_{mp} : 0,221 %
  - I_{mp} : 0,277 %
- Modelo de 2 diodos: para éste modelo, se han obtenido los siguientes errores frente a los valores reales:
  - V_{oc} : 0,094 %
  - I_{sc} : 0,048 %
  - P_{mp} : 0,068 %
  - V_{mp} : 0,209 %
  - I_{mp} : 0,266 %

- Condiciones de funcionamiento nominal (NOCT):

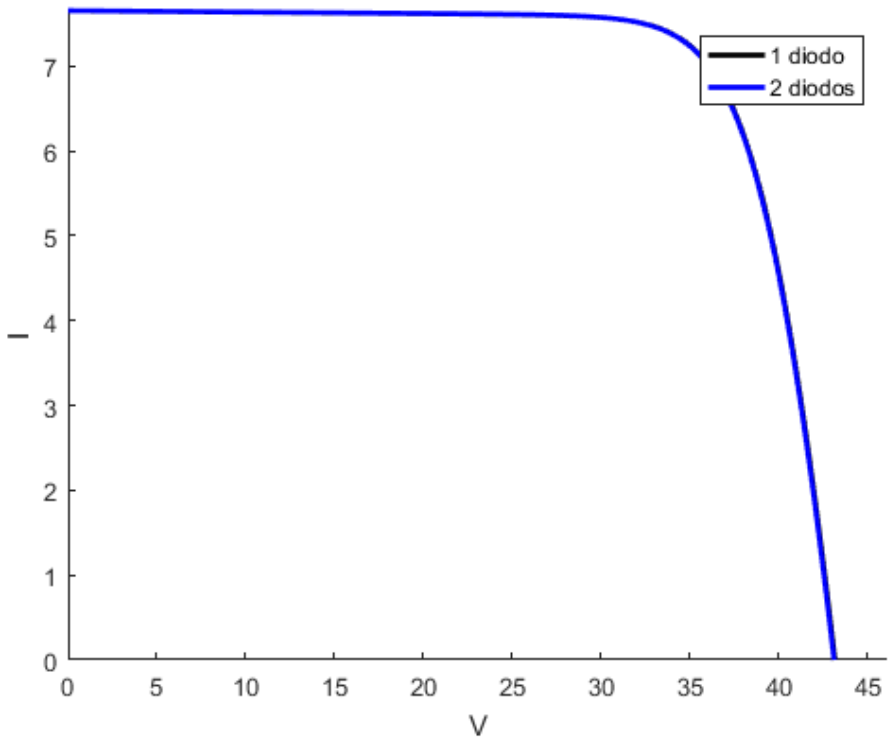


Figura 4.2 : comparación de modelos en condiciones de operación nominales (NOCT)

- Modelo de 1 diodo: para éste modelo, se han obtenido los siguientes errores frente a los valores reales:
  - V_{oc} : 0,394 %
  - I_{sc} : 0,099 %
  - P_{mp} : 1,490 %
  - V_{mp} : 1,365 %
  - I_{mp} : 0,170 %
- Modelo de 2 diodos: para éste modelo, se han obtenido los siguientes errores frente a los valores reales:
  - V_{oc} : 0,280 %
  - I_{sc} : 0,097 %
  - P_{mp} : 1,378 %
  - V_{mp} : 1,259 %
  - I_{mp} : 0,164 %

- Diferentes condiciones de irradiancia a temperatura constante (25 $^{\circ}C$):
  - Modelo de 1 diodo:

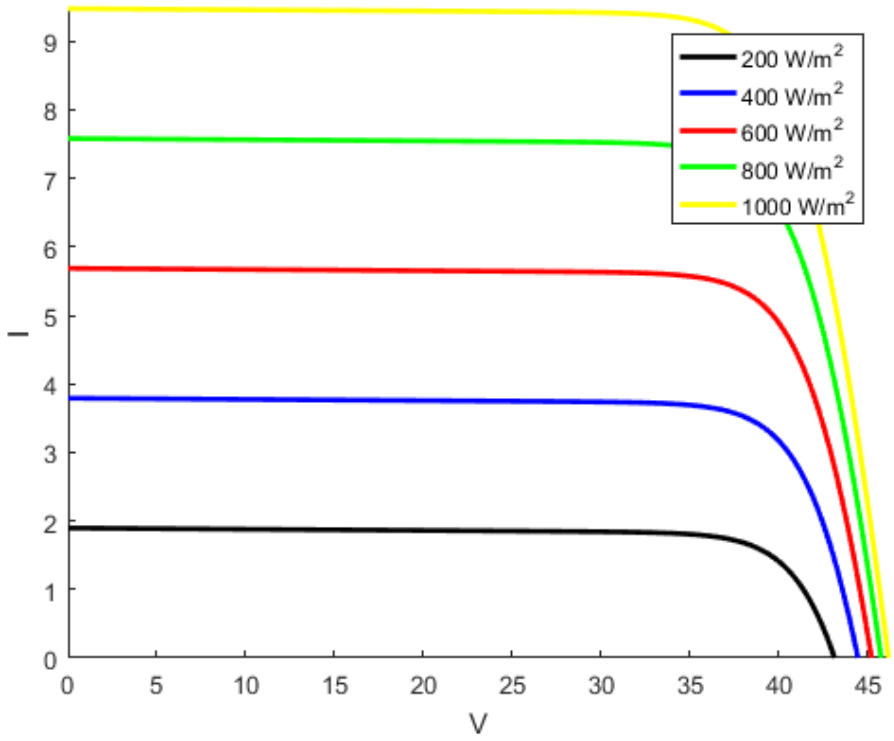


Figura 4.3 : comparación de resultados en irradiancia con el modelo de 1 diodo

  - Modelo de 2 diodos:

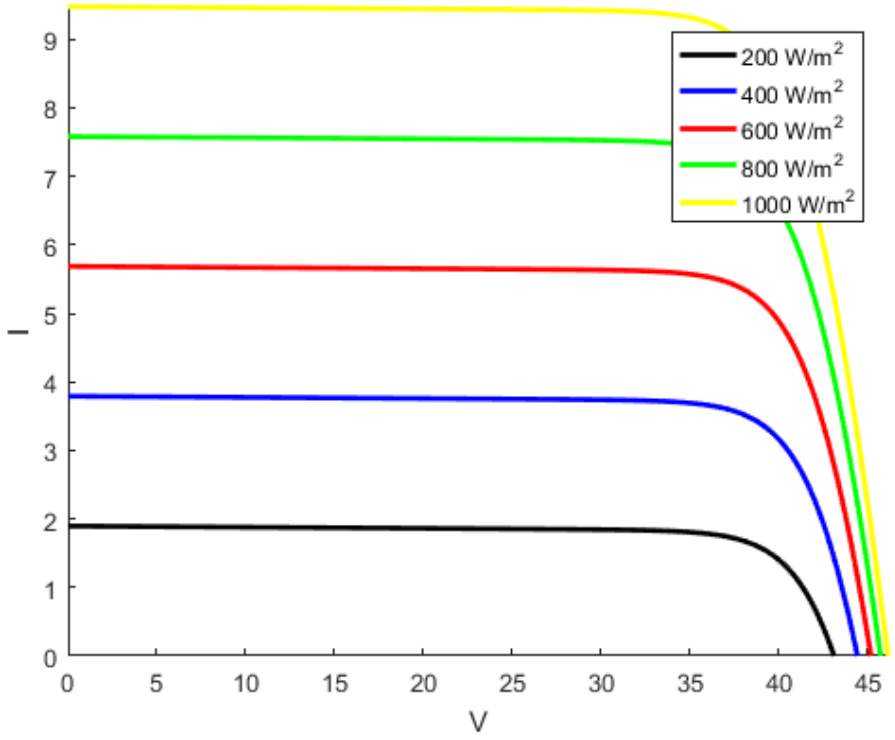


Figura 4.4 : comparación de resultados en irradiancia con el modelo de 2 diodos

- $Irradiancia = 1000\frac{W}{m^2}$: para ésta irradiancia, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,061 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,001 %
  - V_{mp} : 0,012 %
  - I_{mp} : 0,011 %
- $Irradiancia = 800\frac{W}{m^2}$: para ésta irradiancia, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,064 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,018 %
  - V_{mp} : 0,034 %
  - I_{mp} : 0,016 %
- $Irradiancia = 600\frac{W}{m^2}$: para ésta irradiancia, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:

- V_{oc} : 0,066 %
- I_{sc} : 0,003 %
- P_{mp} : 0,031 %
- V_{mp} : 0,055 %
- I_{mp} : 0,024 %

- $Irradiancia = 400\frac{W}{m^2}$: para ésta irradiancia, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,072 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,036 %
  - V_{mp} : 0,080 %
  - I_{mp} : 0,043 %
- $Irradiancia = 200\frac{W}{m^2}$: para ésta irradiancia, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,080 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,004 %
  - V_{mp} : 0,106 %
  - I_{mp} : 0,102 %

- Diferentes condiciones de temperatura a irradiancia constante (1000 $\frac{W}{m^2}$):
  - Modelo de 1 diodo:

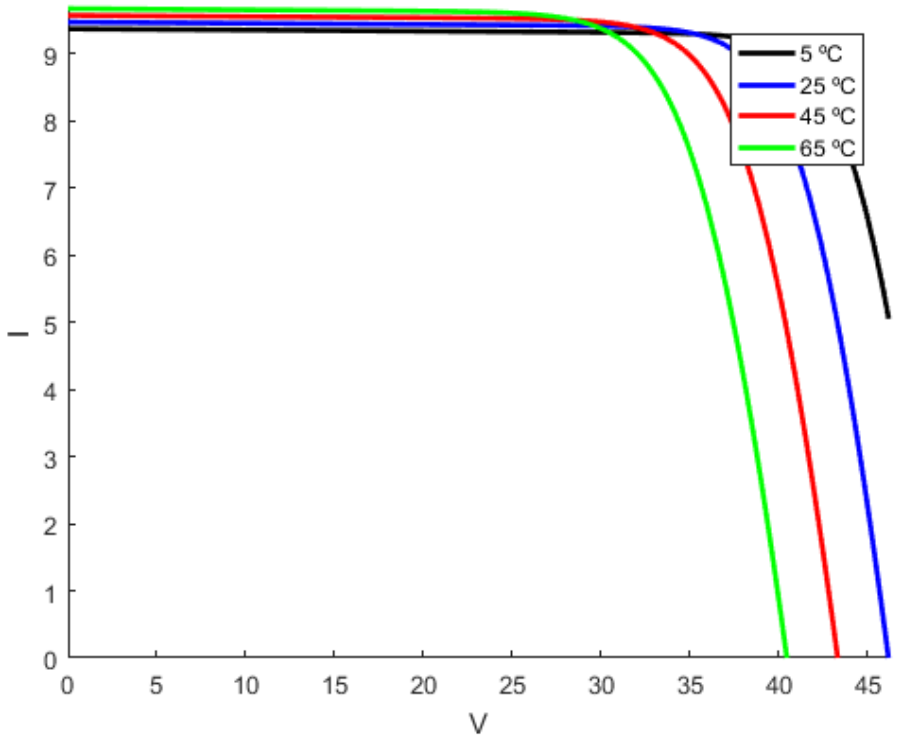


Figura 4.5 : comparación de resultados en temperatura con el modelo de 1 diodo

- Modelo de 2 diodos:

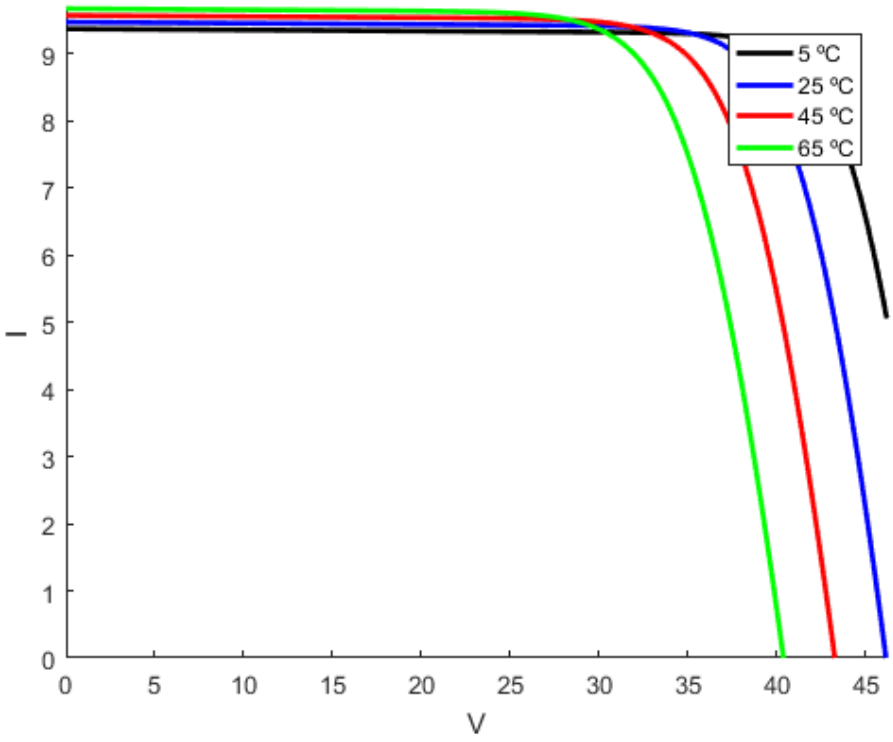


Figura 4.6 : comparación de resultados en temperatura con el modelo de 2 diodos

- $Temperatura = 5°C$: para ésta temperatura, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,000 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,056 %
  - V_{mp} : 0,030 %
  - I_{mp} : 0,025 %
- $Temperatura = 25°C$: para ésta temperatura, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,061 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,001 %
  - V_{mp} : 0,012 %
  - I_{mp} : 0,011 %
- $Temperatura = 45°C$: para ésta temperatura, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:

- V_{oc} : 0,115 %
- I_{sc} : 0,003 %
- P_{mp} : 0,090 %
- V_{mp} : 0,012 %
- I_{mp} : 0,012 %

- $Temperatura = 65°C$: para ésta temperatura, se han obtenido los siguientes errores en referencia al modelo de 2 diodos:
  - V_{oc} : 0,205 %
  - I_{sc} : 0,003 %
  - P_{mp} : 0,267 %
  - V_{mp} : 0,216 %
  - I_{mp} : 0,051 %

4.3. Resumen de resultados

Como se puede observar por los resultados y las gráficas representadas, los dos modelos tienen una gran precisión respecto a los valores reales del módulo, con un máximo del 1.490 % de error relativo en la potencia máxima con el modelo de 1 diodo y un 1.378 % con el modelo de 2 diodos, siendo ambos los mayores errores cometidos por cada modelo respecto a los valores reales. Observando la tendencia, se podría deducir que el error cometido aumentaría aun más con irradiancias o temperaturas que varíen demasiado de las condiciones de test estándar (STC).
Los dos modelos son bastante precisos, y no se diferencian en más del 0.267 % entre los valores obtenidos. Esto se debe a que ambos modelos tienen el añadido al circuito equivalente de las resistencias en serie y paralelo, que brindan la mayor diferencia de precisión respecto a un modelo sin resistencias. Aun así, aunque las diferencias son ínfimas, respecto a los resultados reales, se puede observar que el modelo de 2 diodos es más preciso en casi todas las circunstancias, pero siempre es más preciso respecto al punto de máxima potencia, el cual, en el caso de este trabajo, es el más importante.

Capítulo 5

PreciSol: aplicación simulador de módulos fotovoltaicos con interfaz gráfica

Una vez desarrollados los simuladores para los dos modelos y las herramientas necesarias para su uso (calculadores de resistencias), se ha decidido realizar una aplicación con interfaz gráfica mediante las herramientas proporcionadas por MATLAB, para ofrecer una herramienta de fácil utilización para simular diferentes módulos con datos de diferentes localizaciones.

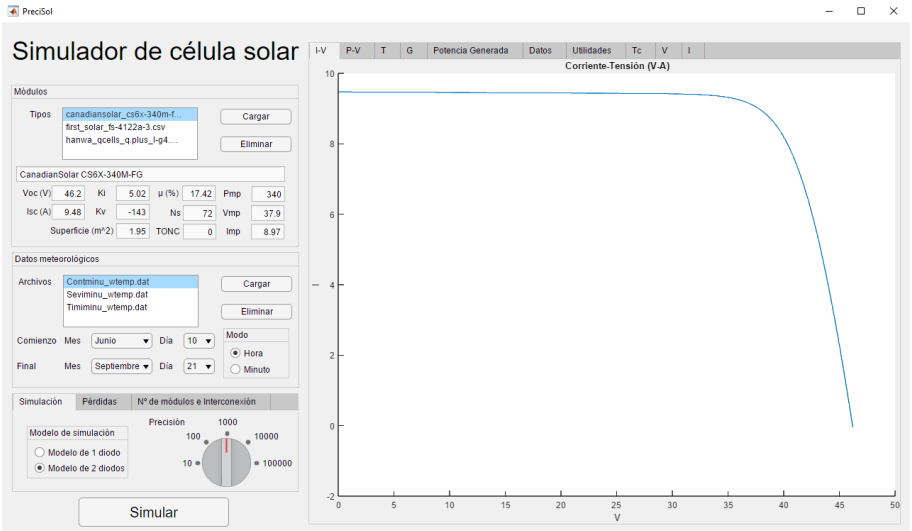


Figura 5.1 : ventana principal del Software PreciSol

Está comletamente desarrollada en lenguaje MATLAB, dividida en varios ficheros, uno siendo la aplicación principal con todo el código fuente (.mlapp, *MATLAB*

application), dos archivos de modelo gráfico de SIMULINK (.slx) y el resto pequeños scripts de MATLAB (.m) que aportan las funcionalidades necesarias a la aplicación y hacen el código más modular. Sus funcionalidades como simulador de módulos fotovoltaicos abarcan todo el ámbito de trabajo; como se ha expuesto en puntos anteriores, es capaz de calcular la potencia de una instalación determinada, con el voltaje siempre situado en el punto de máxima potencia. No abarca elementos como el seguidor de MPP, convertidor CC/CC, inversor, acumuladores, o cualquier elemento posterior al módulo.

Esta aplicación, debido a la necesidad del uso de los modelos de simulación mediante la herramienta SIMULINK, no se puede compilar como un ejecutable de aplicación completamente autónomo, por lo que se debe ejecutar siempre en el contexto de MATLAB; se debe tener abierto MATLAB para ejecutar el programa.

5.1. Aplicación principal: precisol.mlapp

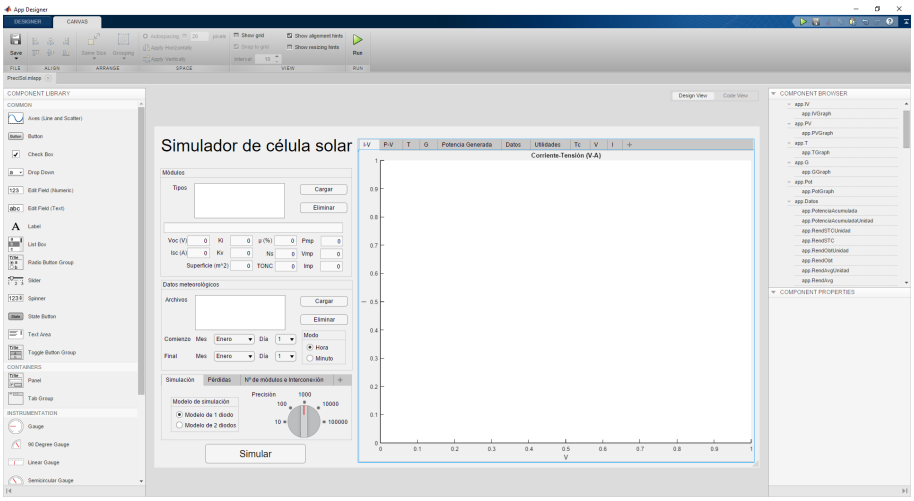


Figura 5.2 : entorno de desarrollo de la herramienta App Designer

Contiene todo el código correspondiente al simulador y el diseño y código de la interfaz gráfica.
A la hora de desarrollar el interfaz gráfico, MATLAB presenta dos opciones:

- GUIDE: herramienta de desarrollo de interfaces gráficas “figure-based” con soporte completo para gráficos 2D y 3D. Características:
    - Soporte de figuras: usa la función “figure” y las propiedades de “Figure”

- Soporte de ejes: usa la función “axis” y las propiedades de “Axis” para usar todas la funcionalidad gráfica de MATLAB
- Soporte de estructura: código es una serie de funciones locales para llamadas y funciones de utilidad
- Editabilidad del código: todo el código es editable
- Acceso a componentes y configuración: usa funciones “get” y “set”
- Configuración de llamadas (callbacks): usa la propiedad “Callback” de “uicontrols”
- Argumentos de llamadas: “hObject”, “eventdata” y una estructura de “handles”
- Compartición de datos: se usa la propiedad “UserData”, “guidata” o “setappdata”
- Creación de componentes: se usa la función “uicontrols” y las propiedades de “uicontrols”

- App Designer (MLAPP): herramienta de desarrollo de interfaces gráficas “uifigure-based” elementos adicionales y gráficos limitados a 2D. Características:
  - Soporte de figuras: usa la función “uifigure” y las propiedades de “UI Figure”
  - Soporte de ejes: usa la función “uiaxis” y las propiedades de “UI Axis” para representar gráficos. Representa la mayoría de gráficos 2-D
  - Soporte de estructura: el código es una clase de MATLAB que contiene los contenidos de la aplicación, llamadas, funciones de utilidad y propiedades para manejar y compartir datos
  - Editabilidad del código: sólo son editables las llamadas, las funciones de utilidad y las propiedades definidas por el usuario
  - Acceso a componentes y configuración: usa notación de puntos
  - Configuración de llamadas (callbacks): usa llamadas de acción específica
  - Argumentos de llamadas: se usan datos de aplicación y de evento (en caso de necesitarse)
  - Compartición de datos: usa propiedades de MATLAB creadas por el usuario

- Creación de componentes: usa una función específica de una función y sus propiedades correspondientes

Aun teniendo en cuenta la experiencia con la herramienta GUIDE, se ha decidido utilizar la herramienta *App Designer*, debido a que es una herramienta muy fácil de aprender, con buenas ayudas al desarrollo (organización de llamadas (callbacks), facilidad de añadir propiedades compartidas por llamadas, etc) y con elementos de diseño del interfaz muy útiles en el caso de esta aplicación (como el uso de pestañas para organizar el contenido), sin mencionar los aspectos estéticos adicionales, como los selectores de rueda (selector de precisión) y la capacidad de redimensionar la aplicación a pantalla completa.

Para describir el funcionamiento del programa, se hará una descripción de los elementos editables: el diseño de la interfaz gráfica, las propiedades (variables de compartición de datos entre llamadas) y las llamadas.

- Diseño de la interfaz gráfica: a la hora de realizar la interfaz, se ha puesto como objetivo la sencillez de utilización y gran superficie de visualización de gráficos, el segundo siendo una de las mayores razones para elegir la herramienta *App Designer*, ya que se pueden situar diferentes ejes en diferentes pestañas, ahorrando espacio. A continuación se describirán las funciones de cada elemento visible en el interfaz:

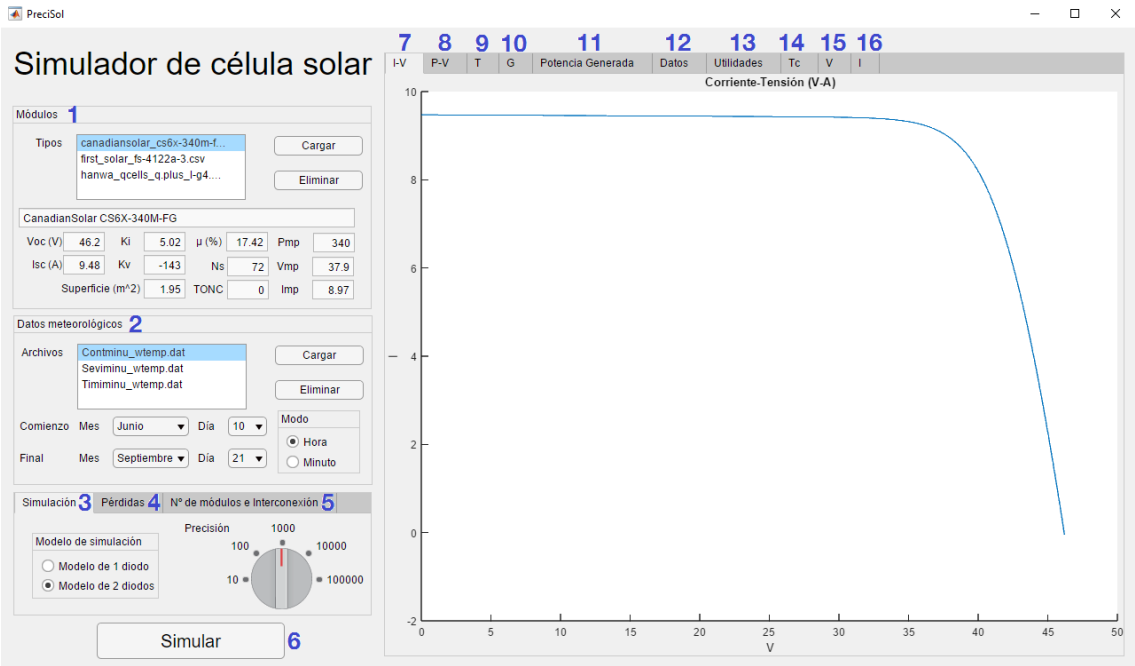


Figura 5.3 : página principal del software Precisol, numerado

1. Carga del módulo y visualización de los datos del mismo: permite cargar datos de módulos y la eliminación de los mismos, a una lista en la que se pueden seleccionar, guardados en un formato personalizado de extensión .csv (*comma separated values*).

Dato	Valor
Fabricante y modelo	0
rend	-
Voc	-
Isc	-
Pmp	-
Vmp	-
Ns	-
Kv	-
Ki	-
Rs_1	-
Rp_1	-
Rs_2	-
Rp_2	-
TONC	-
surface	-

Cuadro 5.1 formato del fichero de datos de módulo (.csv)

Viene además provisto de varios cuadros de valores en los que se puede visualizar los datos del modelo, desde el nombre y fabricante hasta la potencia del módulo y sus especificaciones matemáticas.

2. Carga de datos meteorológicos: permite cargar los datos meteorológicos y la eliminación de los mismos, a una lista en la que se pueden seleccionar, guardados en el formato “standard minute” de Meteonorm. En este cuadro también se encuentra el cuadro de selectores “Modo”, que se encarga del truncado de datos (cada minuto o cada hora), y los selectores de fecha de comienzo y de final. Envía un mensaje de error si se intenta simular con una fecha de comienzo mayor que la de final.
3. Simulación: contiene la información de la simulación y el modelo usado; un cuadro de selección del modelo usado (1 diodo o 2 diodos) y la precisión en puntos de la curva I-V calculada por el modelo de simulación. Cuanta más precisión, más tiempo de cálculo[1].
4. Pérdidas:

[1]Varía muy drásticamente entre 1000 y 10000 puntos, y más aún entre 10000 y 100000. Es menos notable entre 10 y 1000, debido a que el mayor tiempo consumido es por MATLAB al abrir una y otra vez el modelo por cada punto de datos meteorológicos calculados.

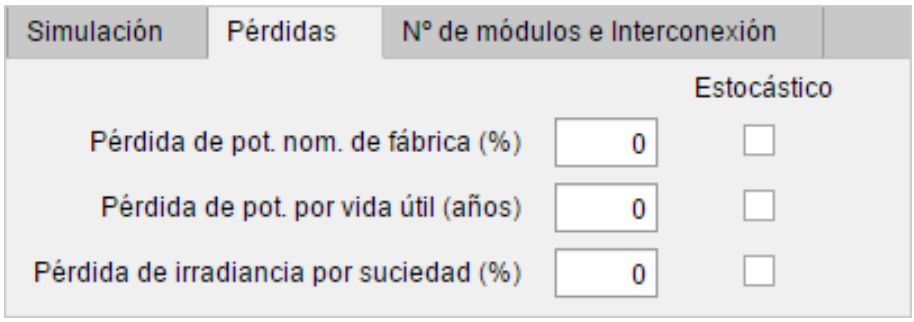


Figura 5.4 : pestaña de pérdidas del software PreciSol

Tiene tres campos para rellenar las pérdidas que se considerarán a la hora de simular. Se puede seleccionar el cuadro “Estocástico”, el cual dará un valor aleatorio para cada uno de los tres términos.

5. N°C de módulos e interconexión:

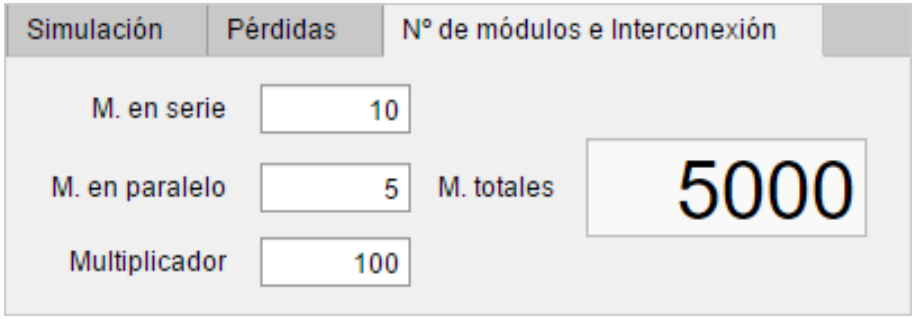


Figura 5.5 : pestaña de N°C de módulos e interconexión del software PreciSol

Contiene tres cuadros numéricos editables, en los cuales se introducen el número de paneles en serie, en paralelo y el multiplicador de paneles en dicha configuración, útil para considerar pequeñas secciones a las que se añadirán los inversores, convertidores CC/CC, etc; y un cuadro no editable que representa el número total de módulos en la instalación.

6. Botón de simular: ejecuta la rutina de simulación con todas las opciones seleccionadas anteriormente. Al terminar la simulación ofrece la opción de guardar los resultados en un fichero de datos.
7. Pestaña de curva I-V del módulo seleccionado en condiciones estándar (STC).
8. Pestaña de curva P-V del módulo seleccionado en condiciones estándar (STC).

9. Pestaña de la temperatura en $^{\circ}C$ del fichero de datos meteorológicos seleccionado en el periodo de tiempo determinado.
10. Pestaña de la irradiancia en $\frac{W}{m^2}$ del fichero de datos meteorológicos seleccionado en el periodo de tiempo determinado.
11. Pestaña de la potencia generada por los módulos seleccionados a lo largo del tiempo en el periodo de tiempo determinado.
12. Pestañas de datos:

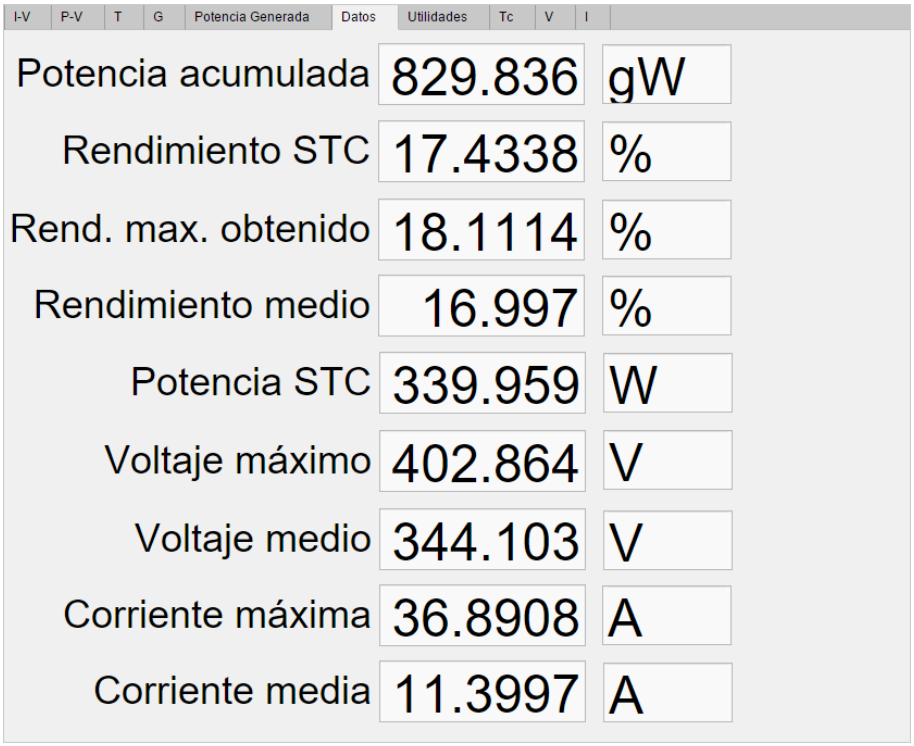


Figura 5.6 : pestaña de datos del software PreciSol

Contiene datos numéricos tales como la potencia acumulada en el periodo de tiempo seleccionado, el rendimiento medio, etc.

13. Pestaña de utilidades:

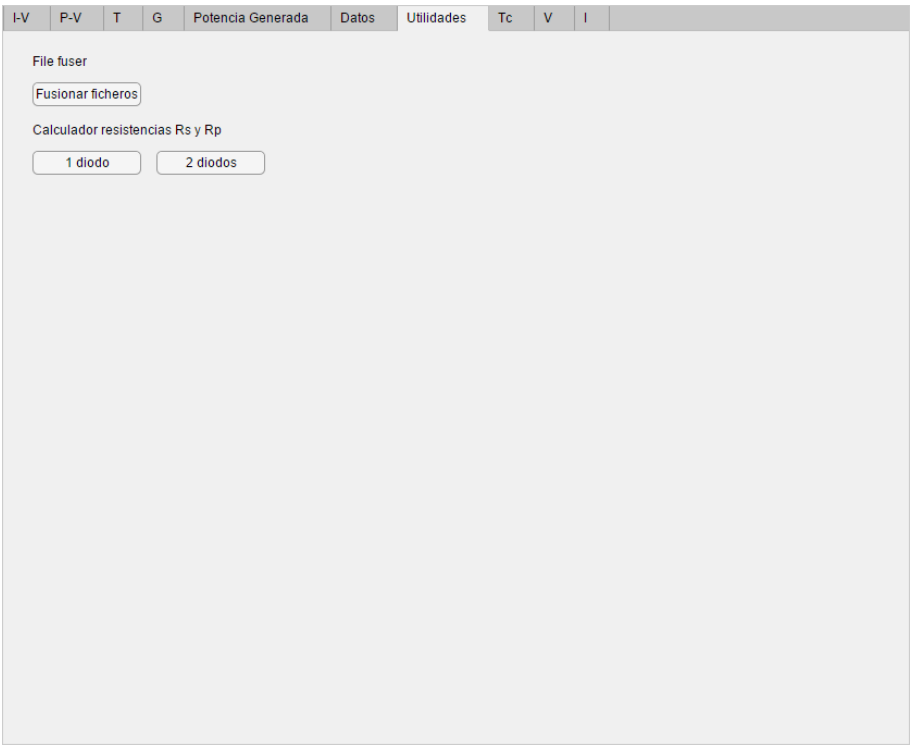


Figura 5.7 : pestaña de utilidades del software PreciSol

Contiene funciones útiles para el simulador, como el cálculo de resistencias o el "fusionado" de ficheros "standard" con temperatura con ficheros "standard minute" sin columna de temperatura de Meteonorm.

14. Pestaña de temperatura del módulo: calculada según los datos meteorológicos y el TONC del módulo, representa la temperatura del módulo en $^{\circ}C$ a lo largo del tiempo y en el intervalo de tiempo seleccionado.
15. Pestaña de tensión del conjunto de módulos en serie.
16. Pestaña de corriente del conjunto de módulos en paralelo.

Código disponible en la sección de anexos, "Aplicación principal", *precisol.mlapp*.

5.2. Funciones principales del simulador

Aquí se describirán las funciones principales del simulador, esenciales para la ejecución de la simulación, como la implementación de los modelos, los calculadores iterativos y la función de llamada a los modelos.

- sim_1_diode.slx: como se explico en la sección de implementación del modelo de 1 diodo 3.4, se ha codificado en un fichero gráfico de SIMULINK, el cual toma los datos del espacio de trabajo actual, realiza los cálculos pertinentes y los devuelve al mismo espacio de trabajo.

- sim_2_diode.slx: como se explico en la sección de implementación del modelo de 2 diodos 3.4, se ha codificado en un fichero gráfico de SIMULINK, el cual toma los datos del espacio de trabajo actual, realiza los cálculos pertinentes y los devuelve al mismo espacio de trabajo.

- fast_diode.m: script de MATLAB que contiene las llamadas a los dos modelos de SIMULINK desarrollados, seleccionados mediante argumento de función. Completamente descrita en la sección de implementación de los modelos de simulación 3.4.

  Código disponible en la sección de anexos, “Funciones principales del simulador”, *fast_diode.m*.

- fast_1_diode_resget.m: script de MATLAB que, utilizando el modelo de SIMULINK de 1 diodo, calcula las resistencias del módulo descrito en los argumentos de forma iterativa para el modelo de 1 diodo. Completamente descrita en la sección de implementación de los modelos de simulación 3.4.

  Código disponible en la sección de anexos, “Funciones principales del simulador”, *fast_1_diode_resget.m*.

- fast_1_diode_resget.m: script de MATLAB que, utilizando el modelo de SIMULINK de 2 diodos, calcula las resistencias del módulo descrito en los argumentos de forma iterativa para el modelo de 2 diodos. Completamente descrita en la sección de implementación de los modelos de simulación 3.4.

  Código disponible en la sección de anexos, “Funciones principales del simulador”, *fast_2_diode_resget.m*.

5.3. Funciones de utilidad

Aquí se describirán las funciones utilizadas por la aplicación principal del simulador, útiles para la modularización y compactación del código.

- read_modulefile.m: función de lectura de fichero de descripción de módulo, lee todos los datos de la ruta especificada y los devuelve al llamador de la función.
  Datos de entrada:

  - filename: dirección completa del fichero .csv de datos del módulo

  Datos de salida:

  - model_name: fabricante y modelo del módulo
  - rend: rendimiento del módulo
  - Voc: tensión de circuito abierto del módulo
  - Isc: corriente de cortocircuito del módulo
  - Pmp: potencia máxima del módulo
  - Imp: corriente de pot. máxima del módulo
  - Vmp: tensión de pot. máxima del módulo
  - Ns: número de células en el módulo
  - Kv: coeficiente de temp. respecto a la tensión de circcuito abierto (V_{oc})
  - Ki: coeficiente de temp. respecto a la corriente de cortocircuito (I_{sc})
  - Rs: vector de resistencias en serie de los modelos de 1 y 2 diodos
  - Rp: vector de resistencias en paralelo de los modelos de 1 y 2 diodos
  - TONC: tempeartura nominal de operación, en
  - surface: superficie del módulo, en m^2

  Código disponible en la sección de anexos, “Funciones de utilidad”, *read_modulefile.m*.

- meteonorm_file.m: función de lectura de fichero de datos meteorológicos, formato “standard minute” de Meteonorm, lee los datos de temperatura ambiente (T_a) e irradiancia global horizontal (G_{Gh}) y los devuelve al llamador de la función.
  Datos de entrada:

  - filename: dirección completa del fichero .dat de datos meteorológicos

  Datos de salida:

  - G_Gh: vector con todos los datos de irradiancia global horizontal durante un año minuto a minuto (525600 elementos)

- Ta: vector con todos los datos de temperatura ambiente durante un año minuto a minuto (525600 elementos)

Código disponible en la sección de anexos, "Funciones de utilidad", *meteonorm_file.m*.

- file_fuser.m: "fusionador" de ficheros "standard" con ficheros "standard minute". Debido a que Meteonorm 6.0 no exporta los datos "standard minute" con la columna de temperatura, ha de tomarse la temperatura de un fichero "standard" de la misma localización con las mismas opciones seleccionadas, para tomar la columna de la temperatura ambiente, con datos por cada hora, e interpolarla a cada minuto, para después introducir la nueva columna al fichero "standard minute". Esto lo consigue leyendo por completo el fichero "standard minute", para posteriormente leer el fichero "standard" y tomar sólo la columna de la temperatura ambiente, la cual se somete a un proceso de interpolación lineal para crear un nuevo vector de tamaño 525600 elementos, compatible con el formato "standard minute". Una vez tenemos todo el fichero en minutos y la columna de temperaturas interpolada, se guarda en un fichero diferente, con el mismo nombre del fichero "standard minute", añadiéndole "_wtemp" al nombre del fichero. Éstos serán los ficheros de datos meteorológicos utilizados por la aplicación del simulador.
  No posee argumentos de entrada ni datos de salida, debido a que los ficheros a fusionar se seleccionan en la ejecución del script con cuadros de diálogo de selección de ficheros y la salida es simplemente el fichero completo.

  Código disponible en la sección de anexos, "Funciones de utilidad", *file_fuser.m*.

- get_date_index.m: función que devuelve un índice según el mes y el día seleccionado que corresponde al último elemento del día de los vectores de datos meteorológicos.
  Datos de entrada:

  - n_month: número del mes
  - day: número del día

  Datos de salida:

  - date_index: número correspondiente al último elemento del día y mes seleccionados

Código disponible en la sección de anexos, “Funciones de utilidad”, *get_date_index.m*.

- get_days_month.m: ésta función devuelve el número correspondiente al número de mes proporcionado. Útil para la función de obtención de índices y para la aplicación central del simulador.
  Datos de entrada:

  - n_month: número del mes

  Datos de salida:

  - n_days: número días que tiene el mes seleccionado

  Código disponible en la sección de anexos, “Funciones de utilidad”, *get_days_month.m*.

- get_month.m: ésta función devuelve la cadena de caracteres correspondiente para cada número de mes en español. Útil solo para la aplicación central del simulador.
  Datos de entrada:

  - n_month: número del mes

  Datos de salida:

  - month: cadena de caracteres con el nombre del mes seleccionado en español

  Código disponible en la sección de anexos, “Funciones de utilidad”, *get_month.m*.

- get_month_num.m: ésta función devuelve el número de mes correspondiente a la cadena de caracteres proporcionada. Útil solo para la aplicación central del simulador, para la obtención del número del mes a partir de un valor de un menú desplegable.
  Datos de entrada:

  - month: cadena de caracteres con el nombre del mes en español

  Datos de salida:

  - n_month: número del mes seleccionado

  Código disponible en la sección de anexos, “Funciones de utilidad”, *get_month_num.m*.

- max_p_point.m: Función encargada de obtener el voltaje e intensidad correspondientes al punto de máxima potencia. Datos de entrada:

- I: array de datos de intensidad del dato simulado
- V: array de datos de voltaje del dato simulado

Datos de salida:

- I_max: corriente correspondiente al punto de máxima potencia del dato simulado
- V_max: tensión correspondiente al punto de máxima potencia del dato simulado

Código disponible en la sección de anexos, "Funciones de utilidad", *max_p_point.m*.

- progressbar.m: función desarrollada por Steve Hoelzer [10], como una aplicación de barra de progreso extremadamente sencilla con capacidad de ofrecer una experiencia de usuario rica. Características:
  - Posibilidad de añadir barras de progreso a scripts .m con una sola línea de código
  - Soporta múltiples barras de progreso en una sola figura para observar progreso de bucles anidados
  - Etiquetas opcionales en las barras
  - La barra se cierra automáticamente cuando la tarea se completa
  - Solo puede existir una ventana de barra de progreso para no amontonarlas en el escritorio
  - El tiempo estimado de finalización es fiable incluso si la barra de progreso se cierra
  - Tiempo de ejecución mínimo. No ralentiza el código
  - Colores aleatorios

  Muy útil para observar los progresos de fusión de ficheros y la simulación en curso.

  Código disponible en la sección de anexos, "Funciones de utilidad", *progressbar.m*.

- save_data.m: ésta función tiene como objetivo guardar en un fichero .dat los resultados obtenidos en la simulación, en la siguiente estructura:

Irradiancia	Temp. ambiente	Temp. de módulo	Potencia	Tensión	Corriente
G_{Gh}	T_a	T_c	P	V	I

Cuadro 5.2 formato del fichero de datos de simulación (.dat)

Datos de entrada:

- G_Gh: irradiancia (con pérdidas aplicadas)
- T_a: temperatura ambiente
- T_c: temperatura del módulo
- P: potencia (con pérdidas aplicadas)
- V_pmax: tensión de punto de máxima potencia
- I_pmax: corriente de punto de máxima potencia
- start_month: número con el mes de comienzo de la simulación
- start_day: día de comienzo de la simulación
- end_month: número con el mes de finalización de la simulación
- end_day: día de finalización de la simulación
- M_series: N$^{\circ}C$ de módulos en serie
- M_parallel: N$^{\circ}C$ de módulos en paralelo
- M_total: N$^{\circ}C$ total de módulos

El script no tiene datos de salida, son el propio fichero.
Comienza por preguntar la carpeta en la que se desea guardar el fichero, y una vez obtenido el directorio, genera el fichero y lo guarda en la carpeta seleccionada.
Formato de la llamada:

$$save\_data(G\_Gh, T\_a, T\_c, P, V\_pmax, I\_pmax, start\_month, ...start\_day, end\_month, end$$

Formato de nombre de fichero: *'sim_' + *start_mon (3 letras)* + *start_day* + '_' + *end_mon (3 letras)* + *end_day* + '_s' + *M_serie* + 'p' + *M_parallel* + 't' + *M_total* + '.dat'*
Ejemplo: *sim_Jun10_Sep21_s10p5t5000*.

Código disponible en la sección de anexos, “Funciones de utilidad”, *save_data.m*.

Capítulo 6

Simulaciones de plantas reales

Ya desarrollada la aplicación, se procederá a realizar una prueba de la misma, simulando diferentes instalaciones reales en diferentes localizaciones, y mostrando los resultados de las mismas, a la vista de poder realizar comparaciones con resultados reales en producción.

6.1. Villajimena II, Enerpal, Valladolid

En ésta sección se simulará la huerta solar fotovoltaica Villajimena II, de 400kW de potencia, ejecutada por Enerpal [24].
Ésta huerta está situada en la localidad de Fuentes de Valdepero (Palencia), por lo que se utilizarán los datos meteorológicos obtenidos para Valladolid, siendo la estación meteorológica más cercana. Características de la central:

- Potencia: 400kW
- Paneles utilizados: Solara SM7000, 180Wp
- Inversores: Xantrex GT 100 E
- Dividida en 4 instalaciones
- Paneles por instalación:
    - 17 módulos en serie (rama)
    - 34 ramas conectadas en paralelo
    - 578 módulos por instalación
- Total: 2312 módulos a través de las 4 instalaciones

Debido a que no se dispone de las especificaciones técnicas de los módulos solares, se ha elegido un módulo de potencia pico equivalente (180Wp), del cual se dispone de todos los datos técnicos necesarios. En éste caso se ha elegido el módulo de JR Solar JRM180M.
Con todos los datos obtenidos, se pasa a simular ésta huerta solar con la aplicación PreciSol.

Invierno

- Meses de invierno: para esta simulación se realizará una simulación de los meses de Enero y Febrero en Valladolid, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

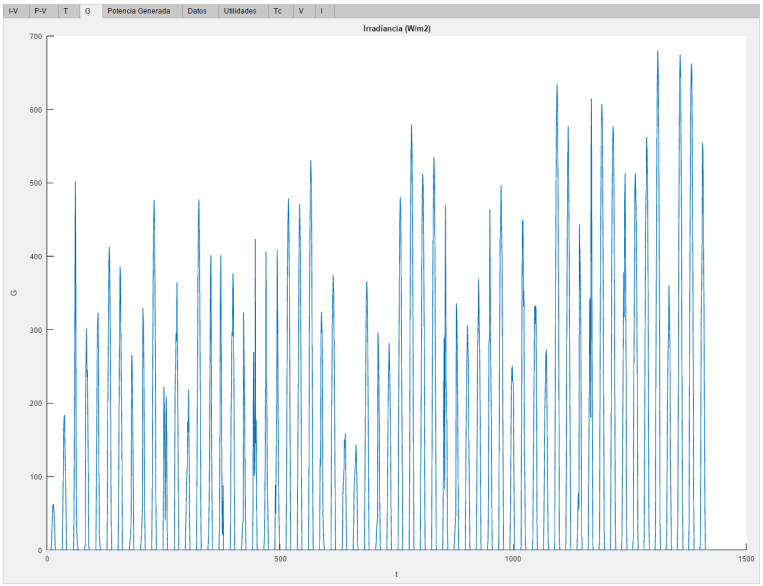


Figura 6.1 : irradiancia en la huerta de Villajimena II, Enero y Febrero

  - Temperatura ambiente

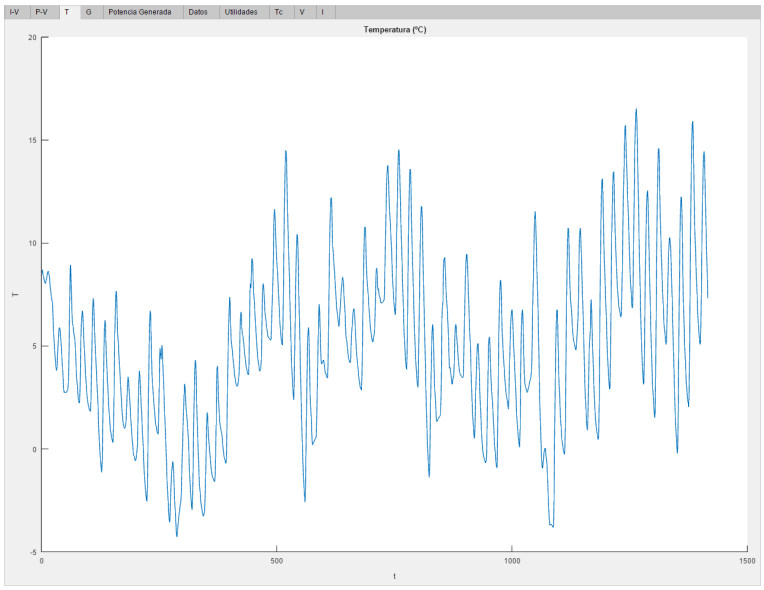


Figura 6.2 : temperatura ambiente en la huerta de Villajimena II, Enero y Febrero

- Potencia generada

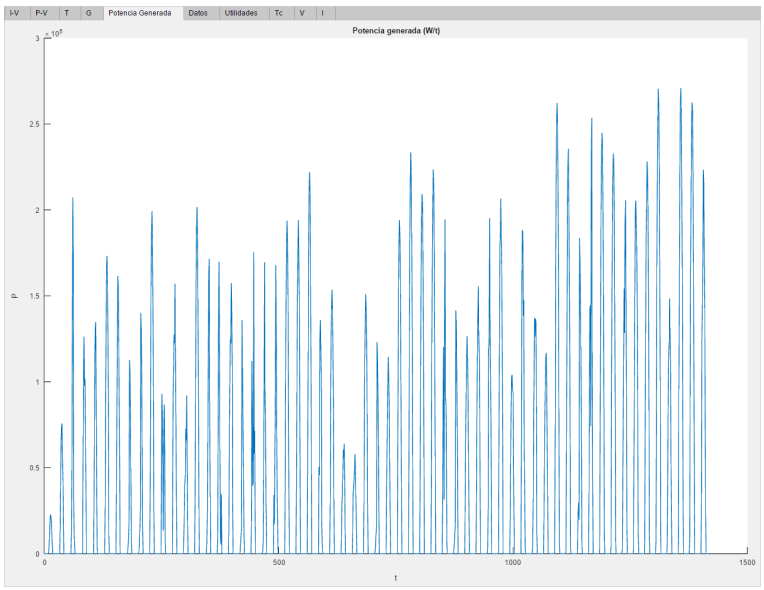


Figura 6.3 : potencia generada en la huerta de Villajimena II, Enero y Febrero

- Temperatura de módulo

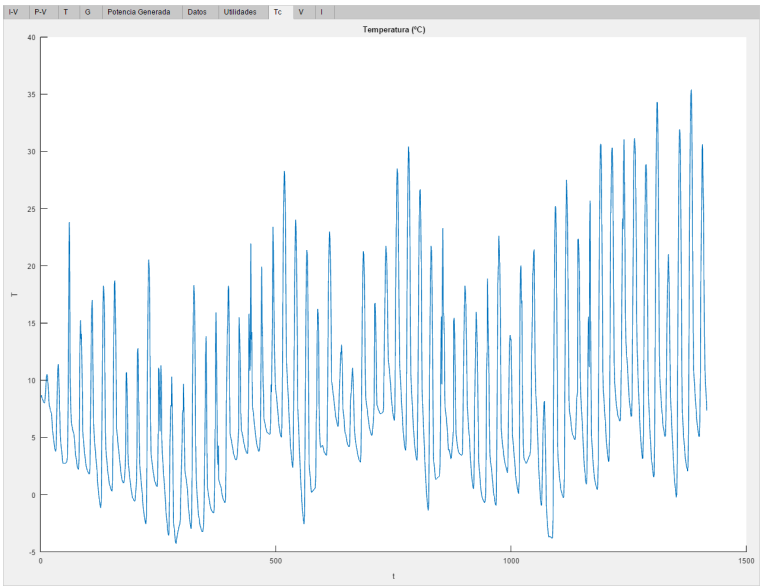


Figura 6.4 : temperatura de módulo en la huerta de Villajimena II, Enero y Febrero

- Voltaje en P_{mp}

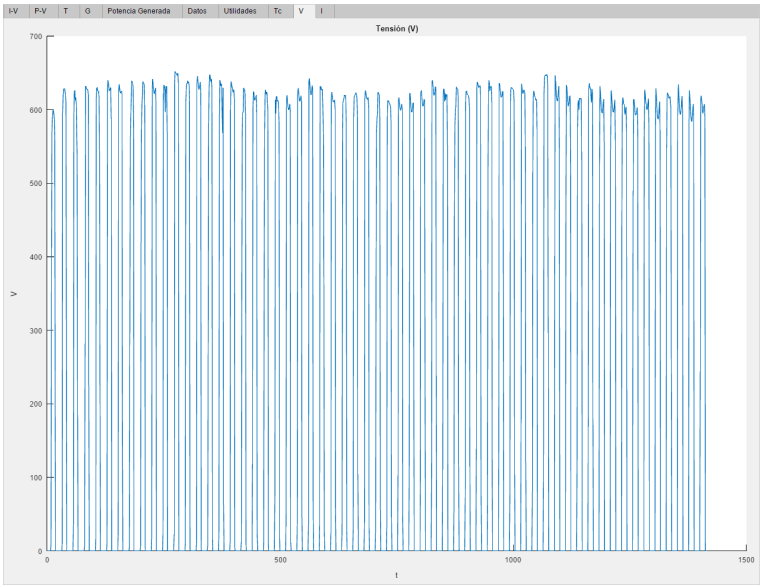


Figura 6.5 : voltaje de rama en la huerta de Villajimena II, Enero y Febrero

- Intensidad en P_{mp}

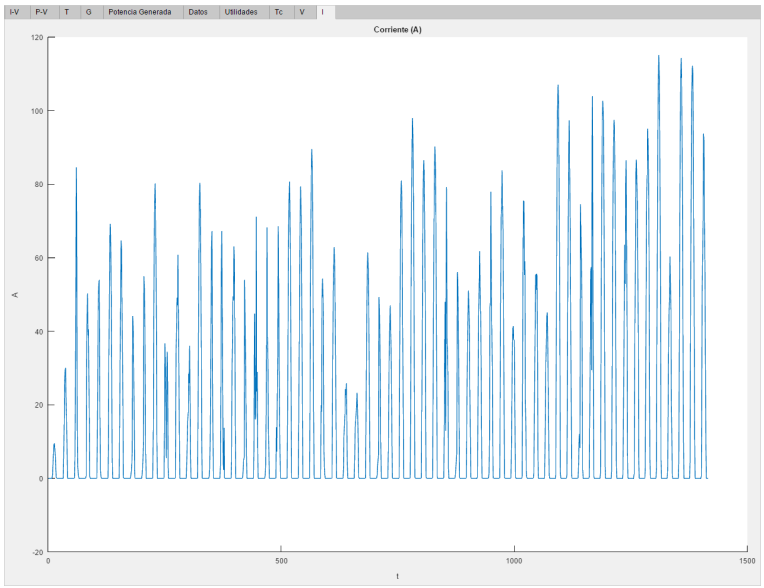


Figura 6.6 : corriente de instalación en la huerta de Villajimena II, Enero y Febrero

Datos numéricos:

- Potencia acumulada: 54.14 GW
- Rendimiento STC: 15.99 %
- Rendimiento máximo obtenido: 16.59 %
- Rendimiento medio: 15.69 %
- Potencia STC: 180 W
- Voltaje máximo: 652.27 V
- Voltaje medio: 535.91 V
- Corriente máxima: 115.13 A
- Corriente media: 34.48 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (3 de Enero en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

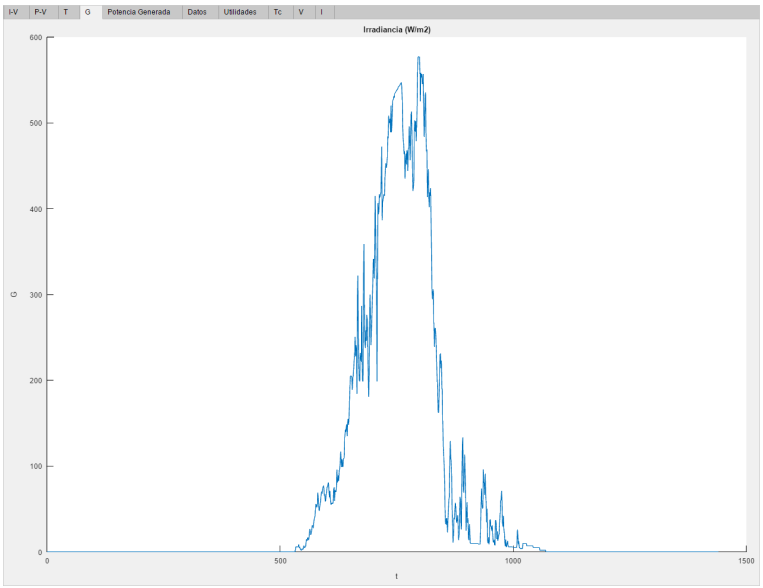


Figura 6.7 : irradiancia en la huerta de Villajimena II, 3 de Enero

- Temperatura ambiente

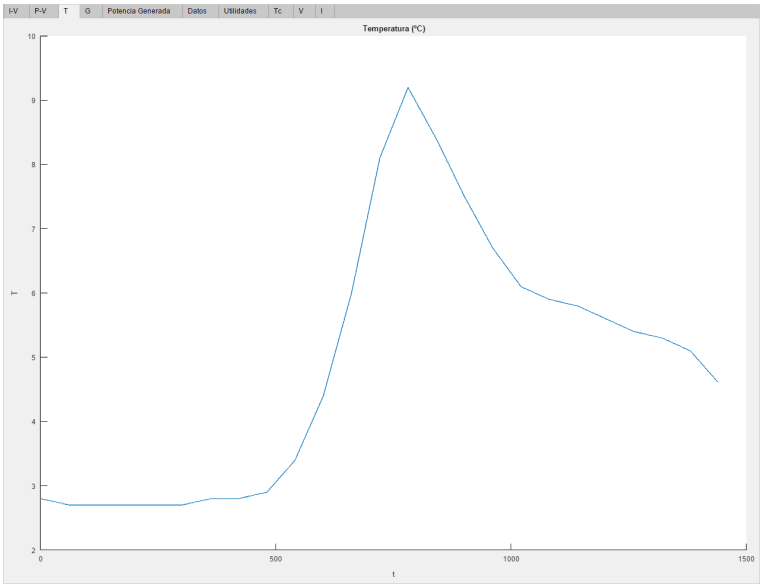


Figura 6.8 : temperatura ambiente en la huerta de Villajimena II, 3 de Enero

- Potencia generada

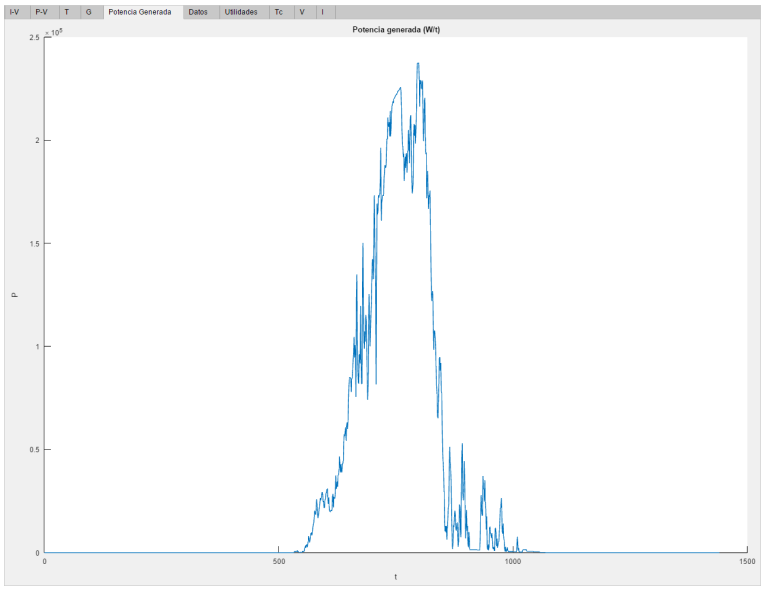


Figura 6.9 : potencia generada en la huerta de Villajimena II, 3 de Enero

- Temperatura de módulo

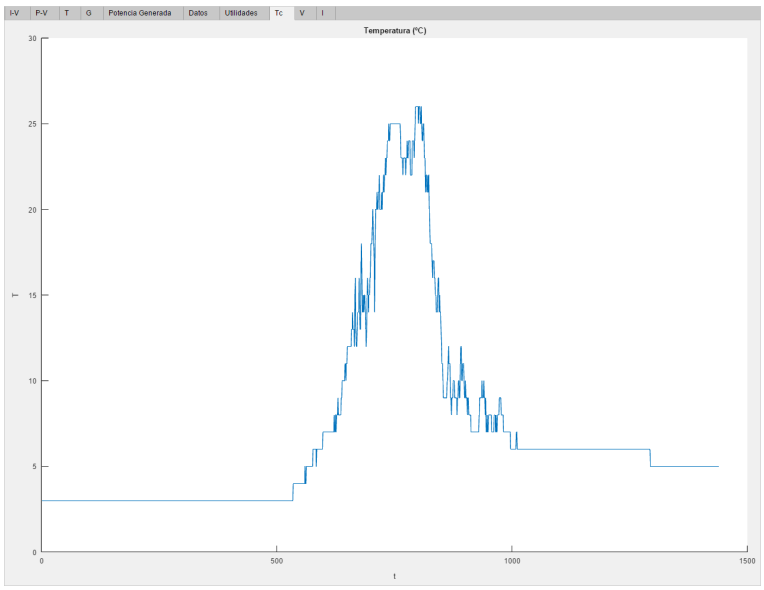


Figura 6.10 : temperatura de módulo en la huerta de Villajimena II, 3 de Enero

- Voltaje en P_{mp}

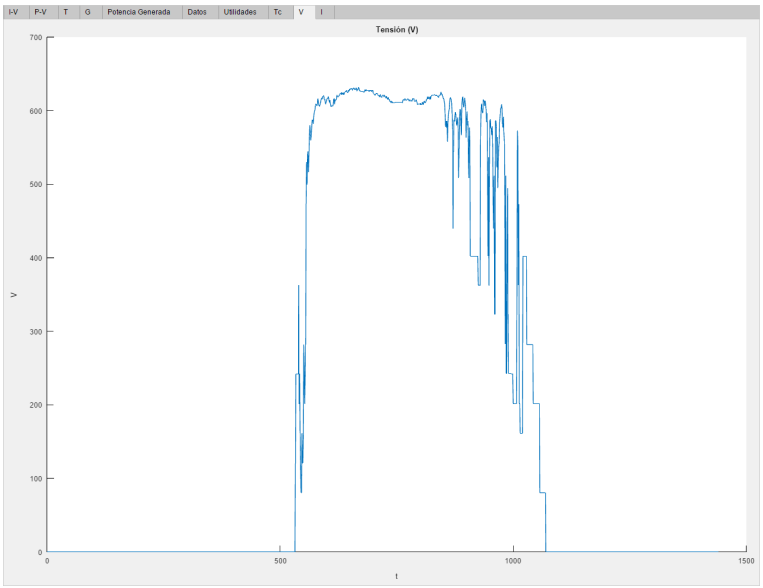


Figura 6.11 : voltaje de rama en la huerta de Villajimena II, 3 de Enero

- Intensidad en P_{mp}

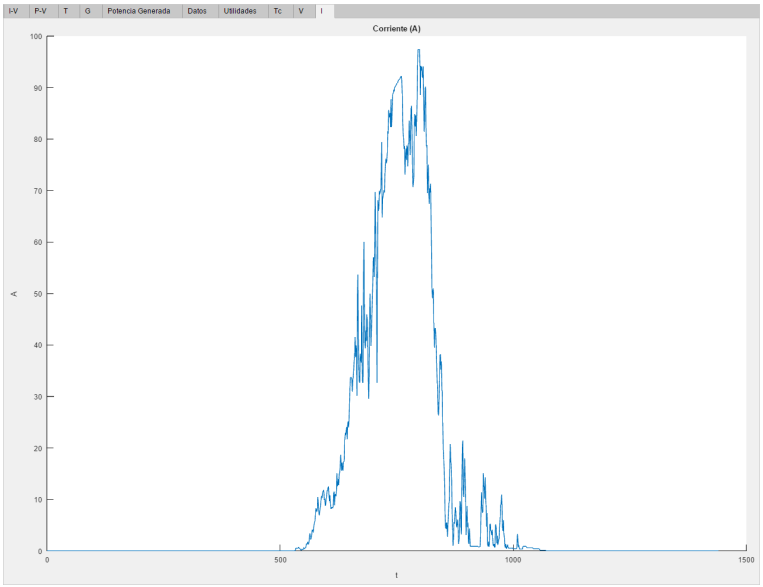


Figura 6.12 : corriente de instalación en la huerta de Villajimena II, 3 de Enero

Datos numéricos:

- Potencia acumulada: 605.99 MW
- Rendimiento STC: 15.99 %

- Rendimiento máximo obtenido: 16.11 %
- Rendimiento medio: 15.54 %
- Potencia STC: 180 W
- Voltaje máximo: 631.37 V
- Voltaje medio: 518.58 V
- Corriente máxima: 97.43 A
- Corriente media: 27.565 A

Verano

- Meses de verano: para esta simulación se realizará una simulación de los meses de Junio y Julio en Valladolid, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

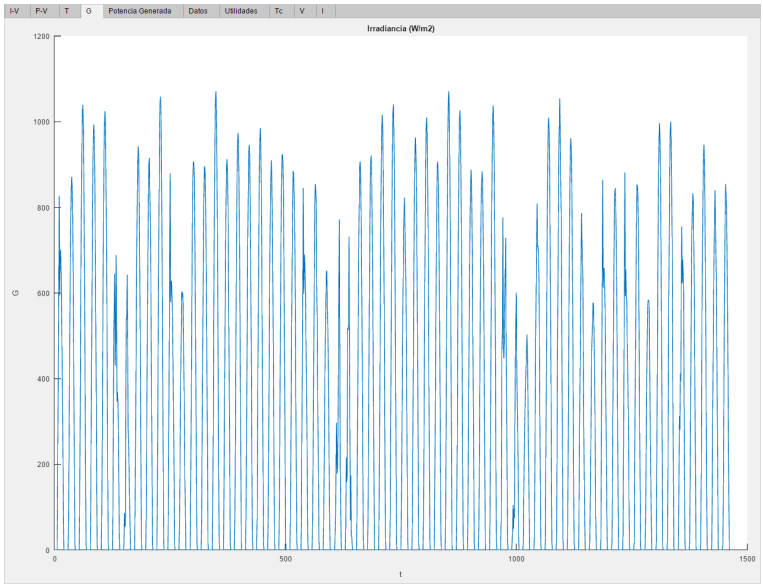


Figura 6.13 : irradiancia en la huerta de Villajimena II, Junio y Julio

  - Temperatura ambiente

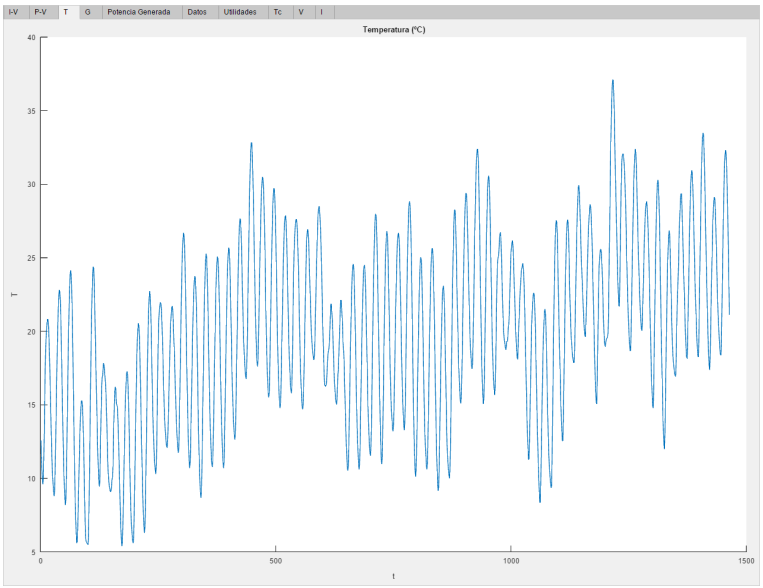


Figura 6.14 : temperatura ambiente en la huerta de Villajimena II, Junio y Julio

- Potencia generada

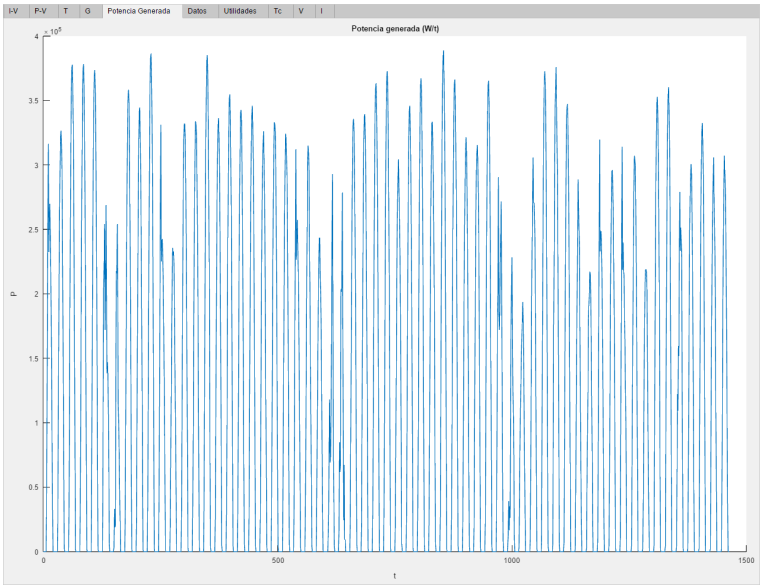


Figura 6.15 : potencia generada en la huerta de Villajimena II, Junio y Julio

- Temperatura de módulo

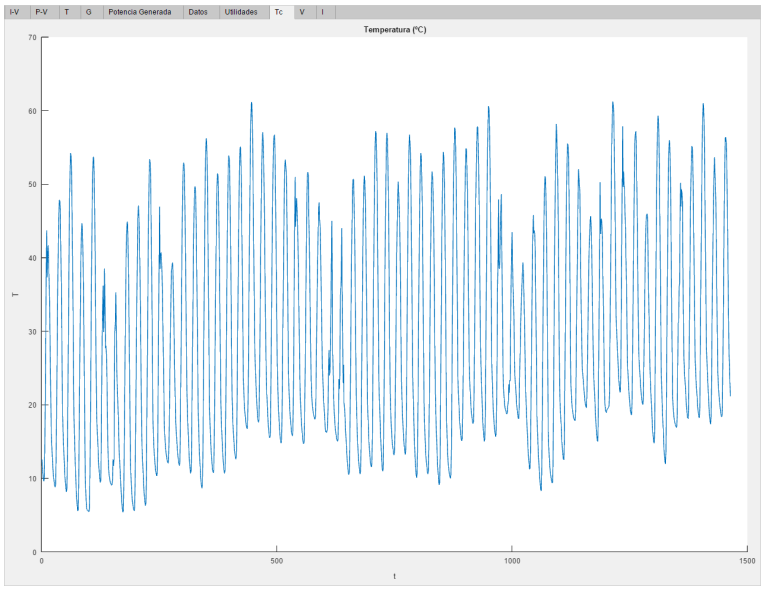


Figura 6.16 : temperatura de módulo en la huerta de Villajimena II, Junio y Julio

- Voltaje en P_{mp}

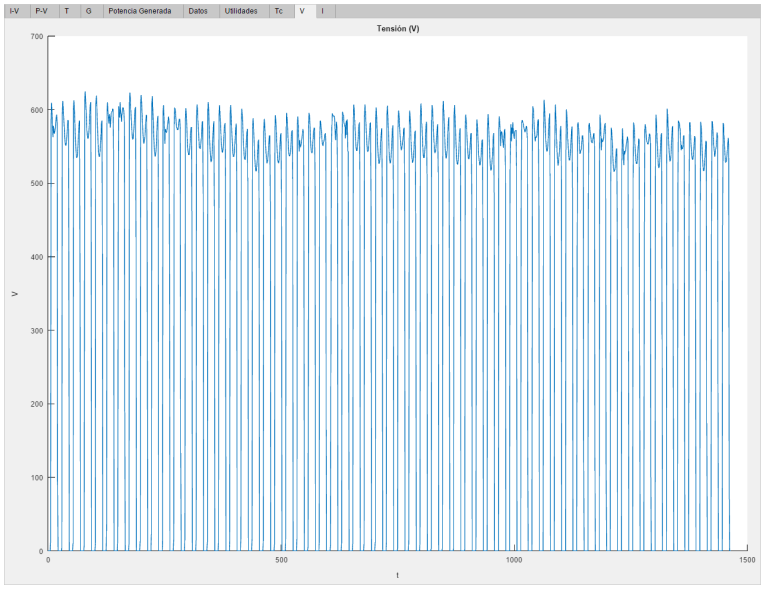


Figura 6.17 : voltaje de rama en la huerta de Villajimena II, Junio y Julio

- Intensidad en P_{mp}

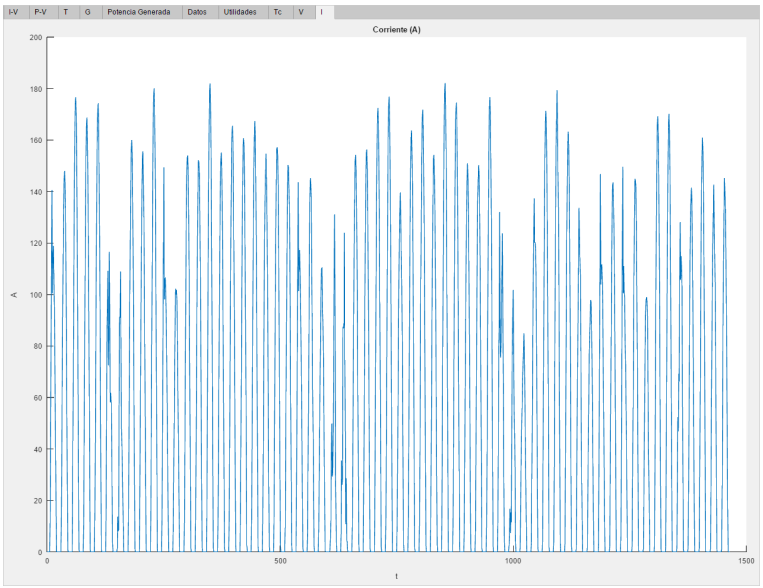


Figura 6.18 : corriente de instalación en la huerta de Villajimena II, Junio y Julio

Datos numéricos:

- Potencia acumulada: 168.37 gW
- Rendimiento STC: 15.99 %
- Rendimiento máximo obtenido: 15.91 %
- Rendimiento medio: 14.45 %
- Potencia STC: 180 W
- Voltaje máximo: 624.653 V
- Voltaje medio: 528.84 V
- Corriente máxima: 182.15 A
- Corriente media: 79.12 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (3 de Junio en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

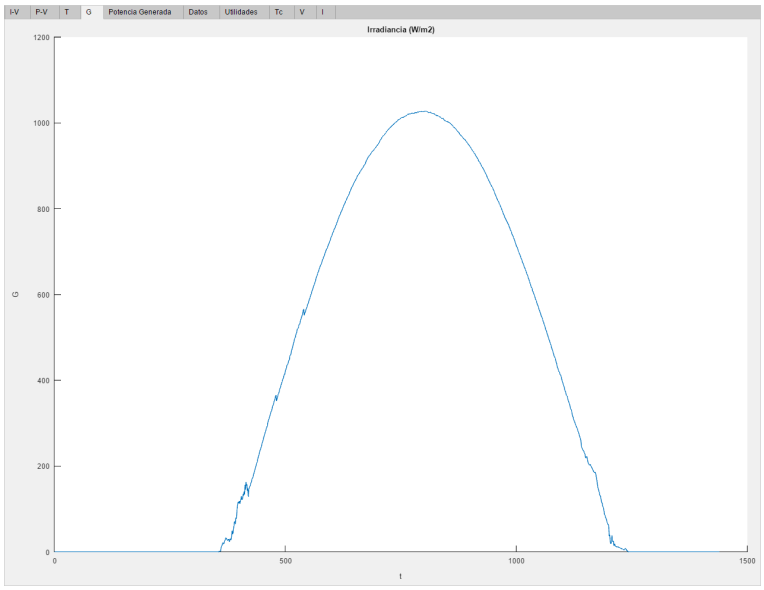


Figura 6.19 : irradiancia en la huerta de Villajimena II, 3 de Junio

- Temperatura ambiente

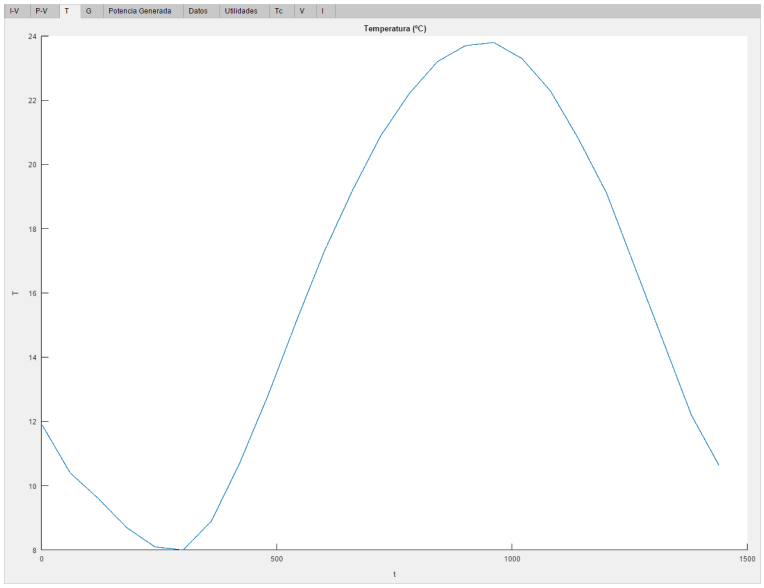


Figura 6.20 : temperatura ambiente en la huerta de Villajimena II, 3 de Junio

- Potencia generada

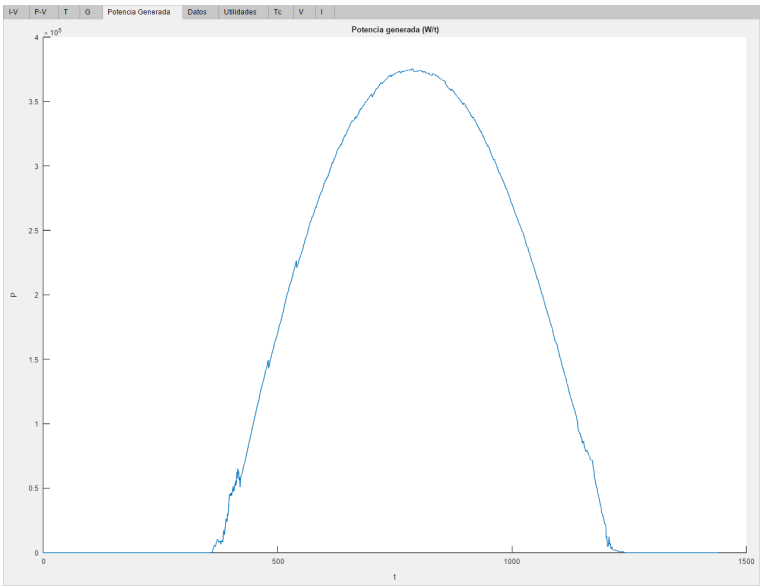


Figura 6.21 : potencia generada en la huerta de Villajimena II, 3 de Junio

- Temperatura de módulo

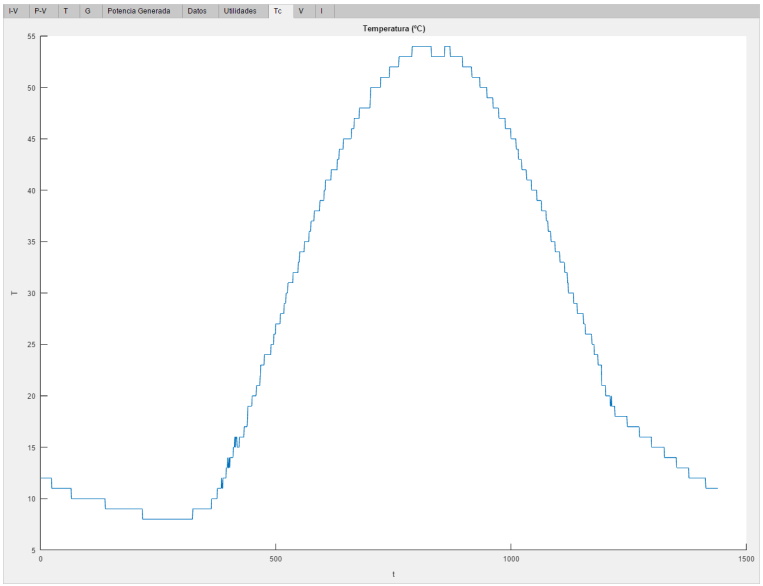


Figura 6.22 : temperatura de módulo en la huerta de Villajimena II, 3 de Junio

- Voltaje en P_{mp}

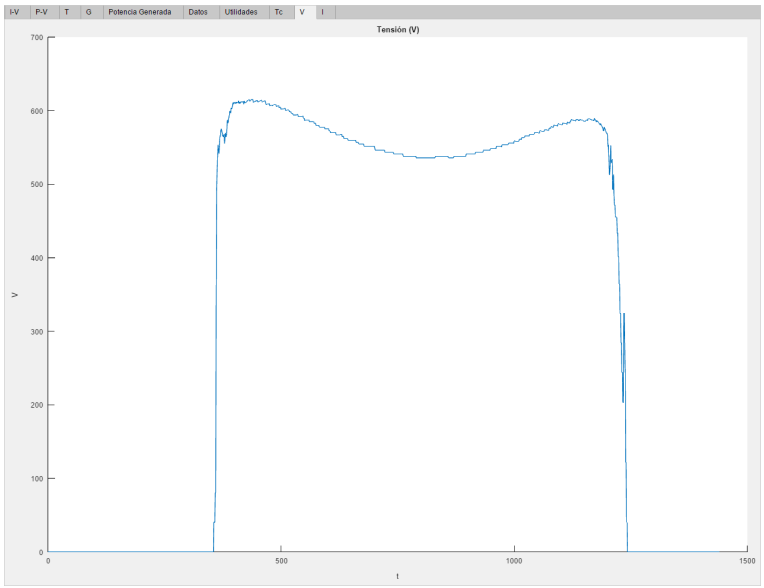


Figura 6.23 : voltaje de rama en la huerta de Villajimena II, 3 de Junio

- Intensidad en P_{mp}

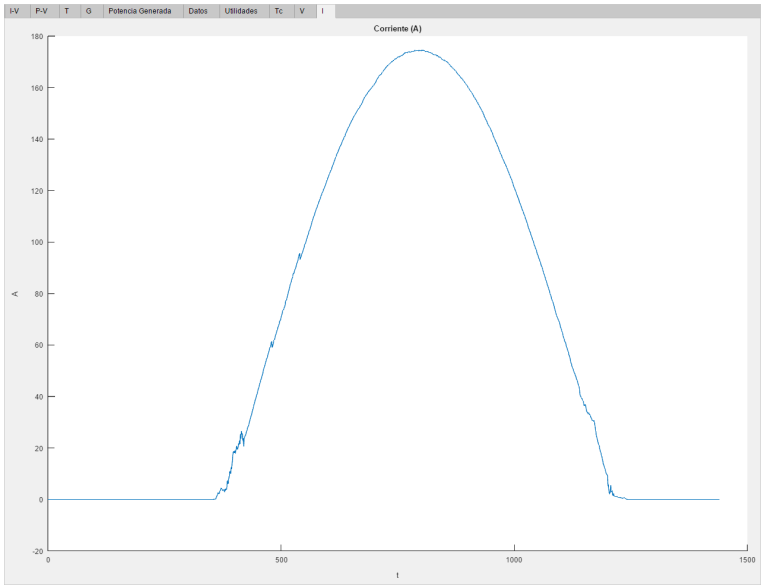


Figura 6.24 : corriente de instalación en la huerta de Villajimena II, 3 de Junio

Datos numéricos:

- Potencia acumulada: 3.37 gW
- Rendimiento STC: 15.99 %

- Rendimiento máximo obtenido: 15.75 %
- Rendimiento medio: 14.51 %
- Potencia STC: 180 W
- Voltaje máximo: 615.7 V
- Voltaje medio: 556.12 V
- Corriente máxima: 174.45 A
- Corriente media: 102.07 A

6.2. Parque solar Cortijo del Cura, Málaga

En ésta sección se simulará el parque solar Cortijo del Cura [26].
Ésta huerta está situada en la localidad de Casabermeja, por lo que se utilizarán los datos meteorológicos obtenidos para Sevilla, siendo la estación meteorológica más cercana.

Debido a que no se disponen de datos detallados del proyecto, se supondrá la interconexión de los módulos, para ajustarse al número de módulos proporcionado, y se elegirá un panel que concuerde con la potencia especificada. Características de la central:

- Dividida en 18 instalaciones
- Paneles por instalación:
  - 17 módulos en serie (rama)
  - 30 ramas conectadas en paralelo
  - 510 módulos por instalación
- Total: 9180 módulos a través de las 18 instalaciones

Debido a que no se dispone de las especificaciones técnicas de los módulos solares, se ha elegido un módulo de potencia pico equivalente (220Wp), del cual se dispone de todos los datos técnicos necesarios. En éste caso se ha elegido el módulo Siliken SLK60P6L, que cumple con la especificación de potencia pico.

Con todos los datos obtenidos, se pasa a simular ésta huerta solar con la aplicación PreciSol.

Invierno

- Meses de invierno: para esta simulación se realizará una simulación de los meses de Enero y Febrero en Sevilla, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

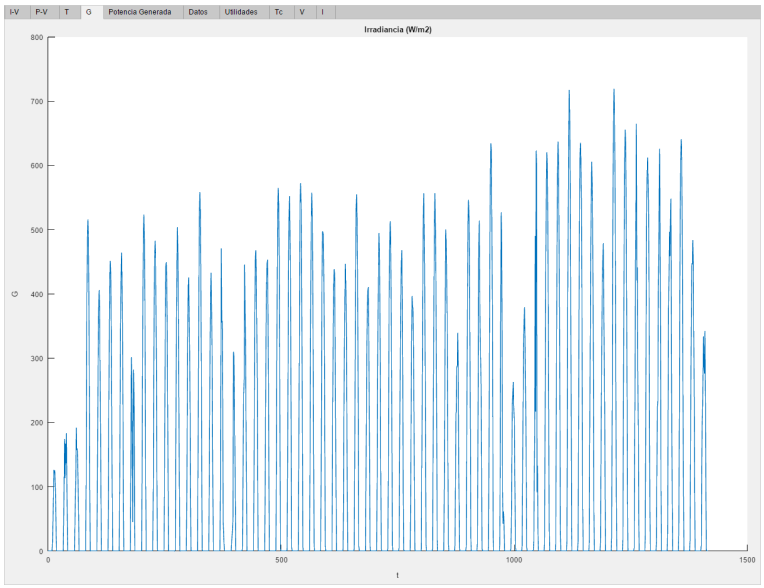


Figura 6.25 : irradiancia en el parque Cortijo del Cura, Enero y Febrero

  - Temperatura ambiente

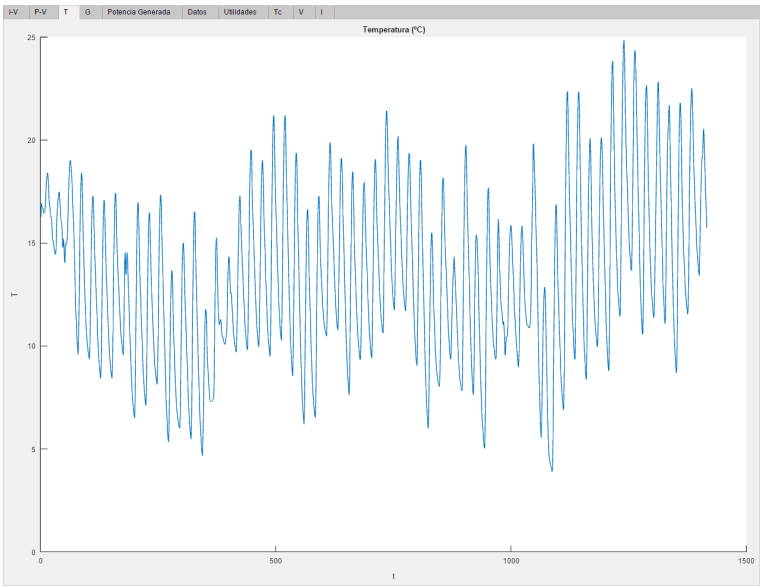


Figura 6.26 : temperatura ambiente en el parque Cortijo del Cura, Enero y Febrero

- Potencia generada

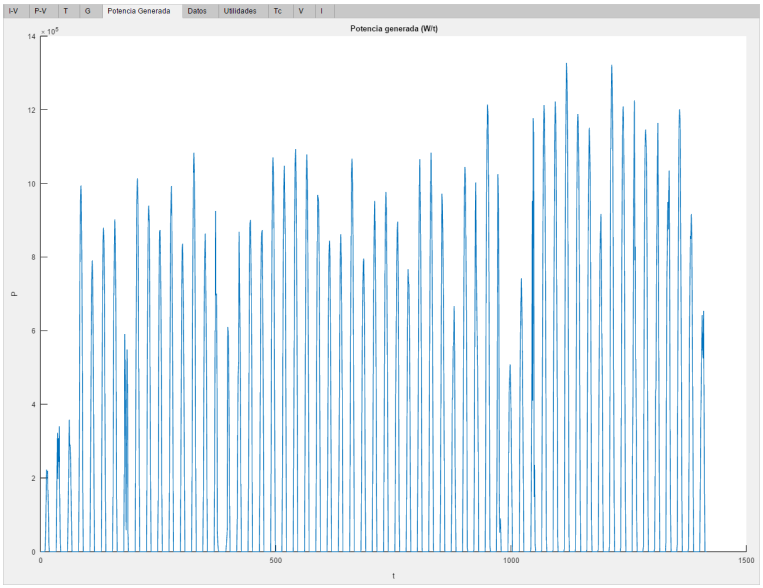


Figura 6.27 : potencia generada en el parque Cortijo del Cura, Enero y Febrero

- Temperatura de módulo

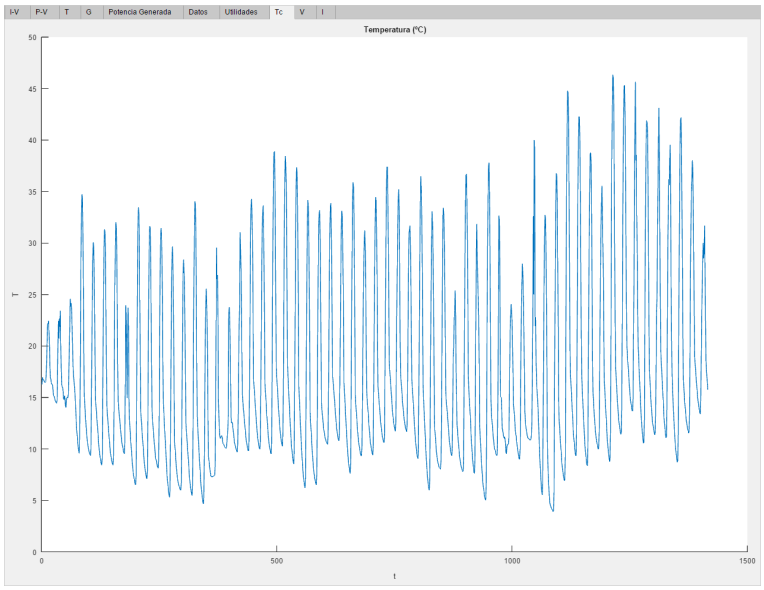


Figura 6.28 : temperatura de módulo en el parque Cortijo del Cura, Enero y Febrero

- Voltaje en P_{mp}

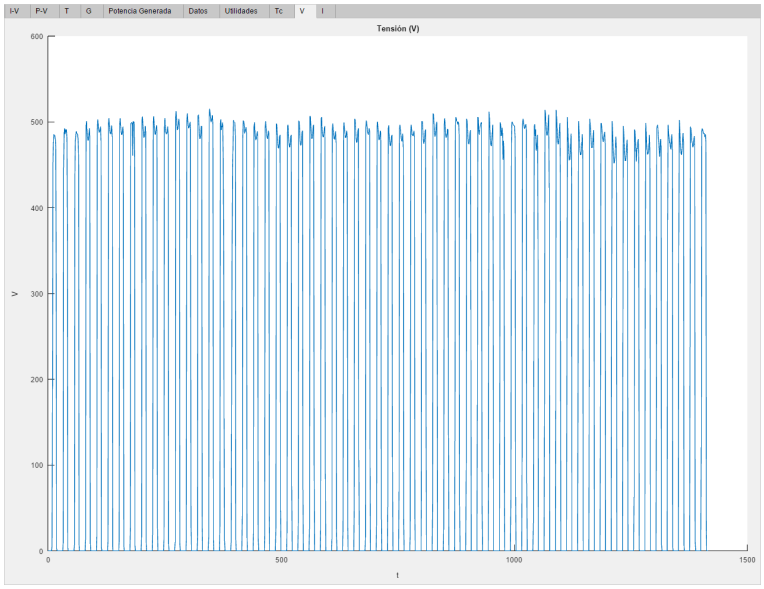


Figura 6.29 : voltaje de rama en el parque Cortijo del Cura, Enero y Febrero

- Intensidad en P_{mp}

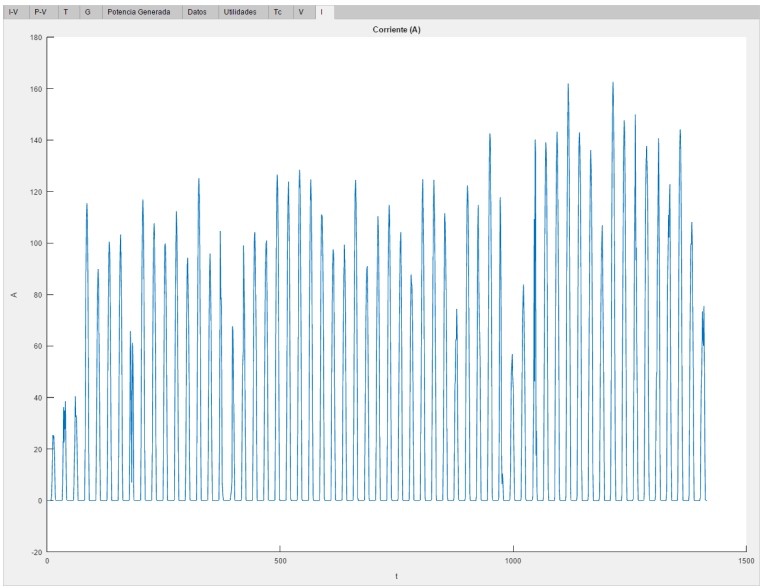


Figura 6.30 : corriente de instalación en el parque Cortijo del Cura, Enero y Febrero

Datos numéricos:

- Potencia acumulada: 328.18 gW
- Rendimiento STC: 13.59 %
- Rendimiento máximo obtenido: 13.47 %
- Rendimiento medio: 12.79 %
- Potencia STC: 220.17 W
- Voltaje máximo: 515.34 V
- Voltaje medio: 413.10 V
- Corriente máxima: 162.48 A
- Corriente media: 57.31 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (14 de Enero en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

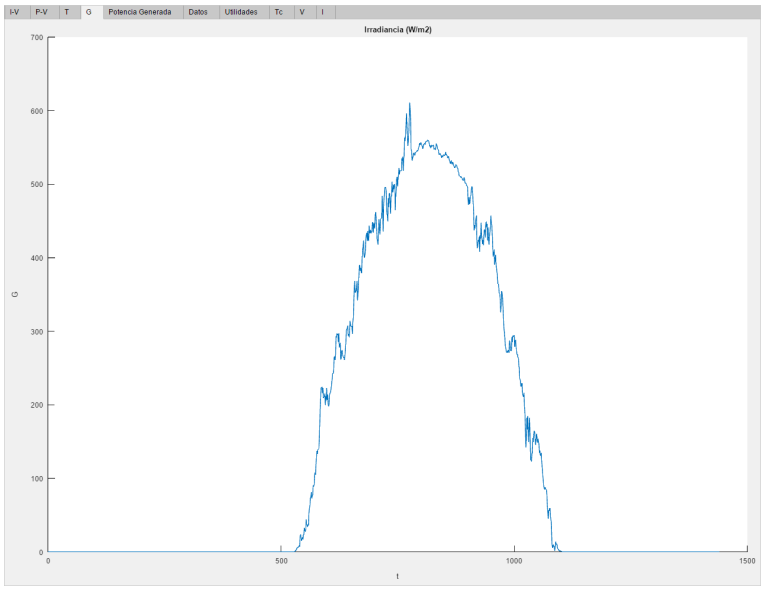


Figura 6.31 : irradiancia en el parque Cortijo del Cura, 14 de Enero

- Temperatura ambiente

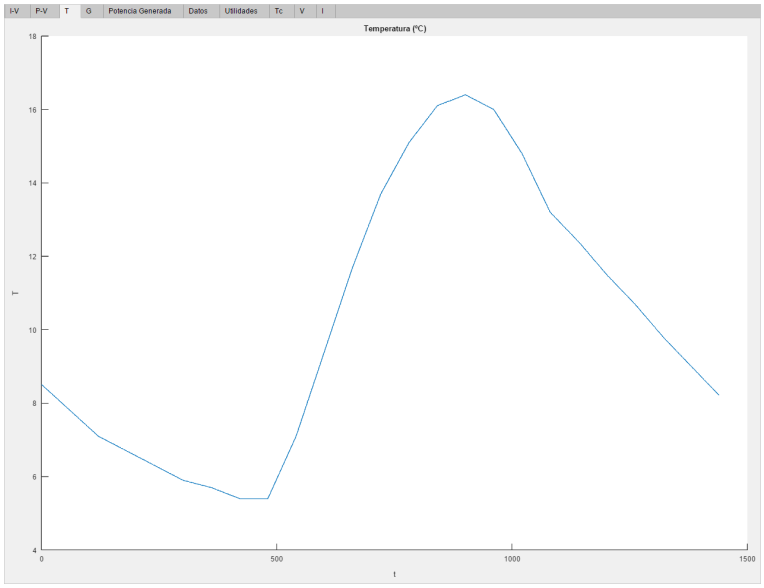


Figura 6.32 : temperatura ambiente en el parque Cortijo del Cura, 14 de Enero

- Potencia generada

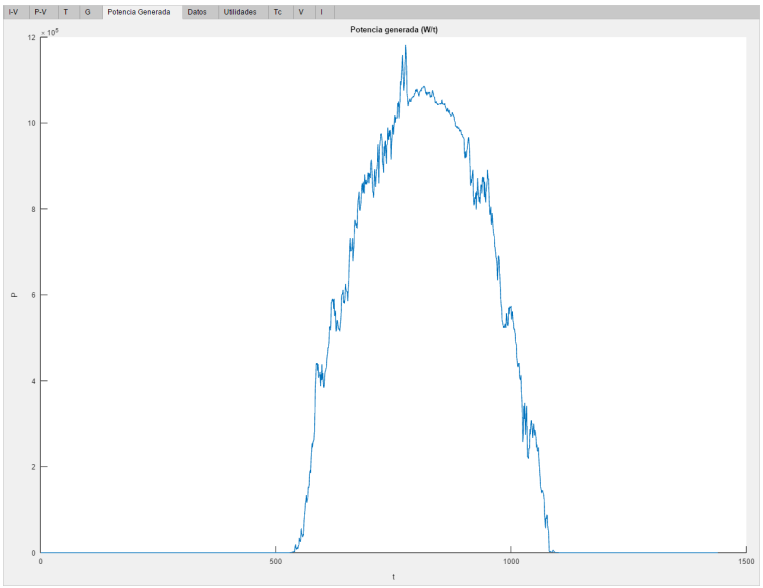


Figura 6.33 : potencia generada en el parque Cortijo del Cura, 14 de Enero

- Temperatura de módulo

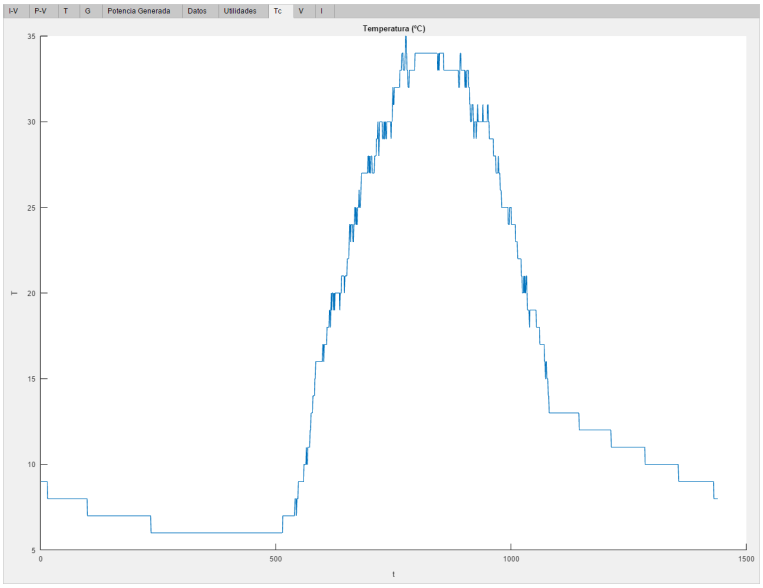


Figura 6.34 : temperatura de módulo en el parque Cortijo del Cura, 14 de Enero

- Voltaje en P_{mp}

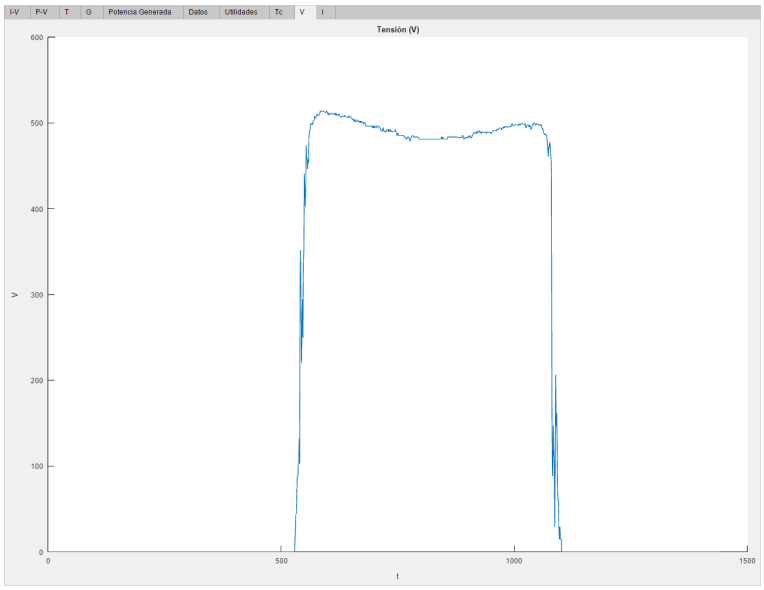


Figura 6.35 : voltaje de rama en el parque Cortijo del Cura, 14 de Enero

- Intensidad en P_{mp}

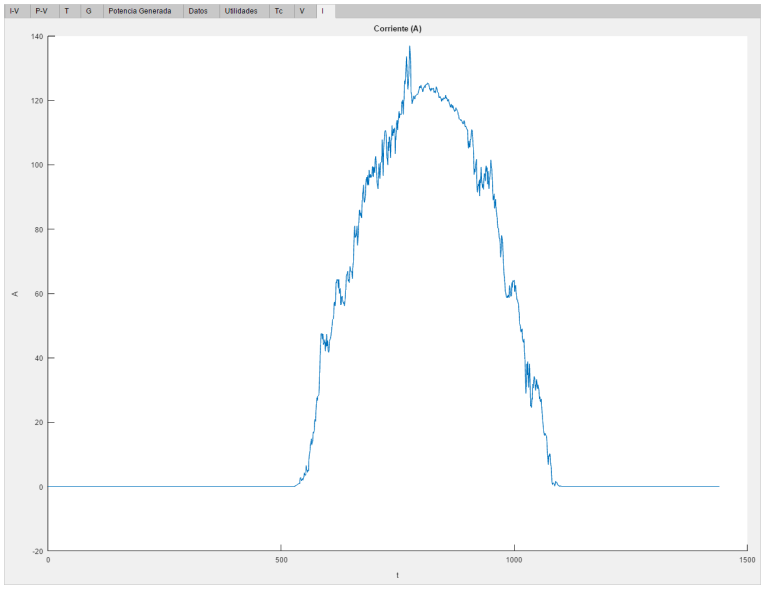


Figura 6.36 : corriente de instalación en el parque Cortijo del Cura, 14 de Enero

Datos numéricos:

- Potencia acumulada: 6.33 gW
- Rendimiento STC: 13.59 %

- Rendimiento máximo obtenido: 13.41 %
- Rendimiento medio: 13.05 %
- Potencia STC: 220.17 W
- Voltaje máximo: 514.09 V
- Voltaje medio: 465.71 V
- Corriente máxima: 137.04 A
- Corriente media: 75.33 A

Verano

- Meses de verano: para esta simulación se realizará una simulación de los meses de Junio y Julio en Sevilla, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

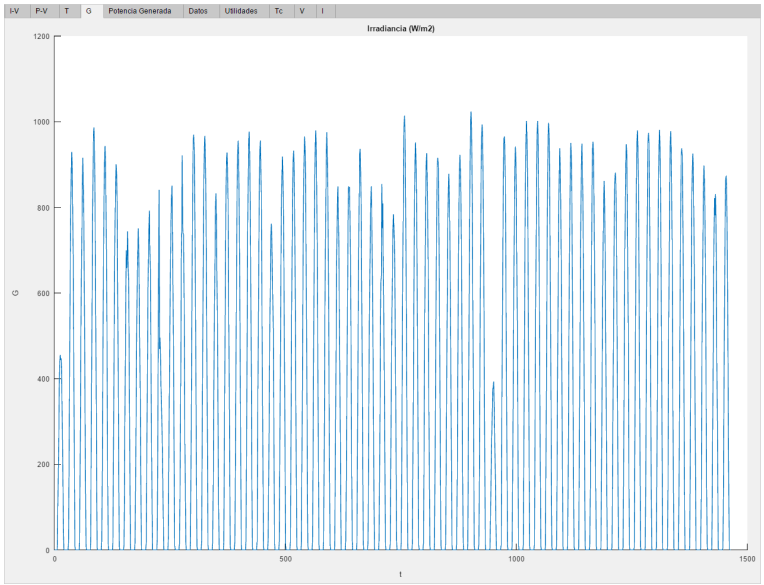


Figura 6.37 : irradiancia en el parque Cortijo del Cura, Junio y Julio

  - Temperatura ambiente

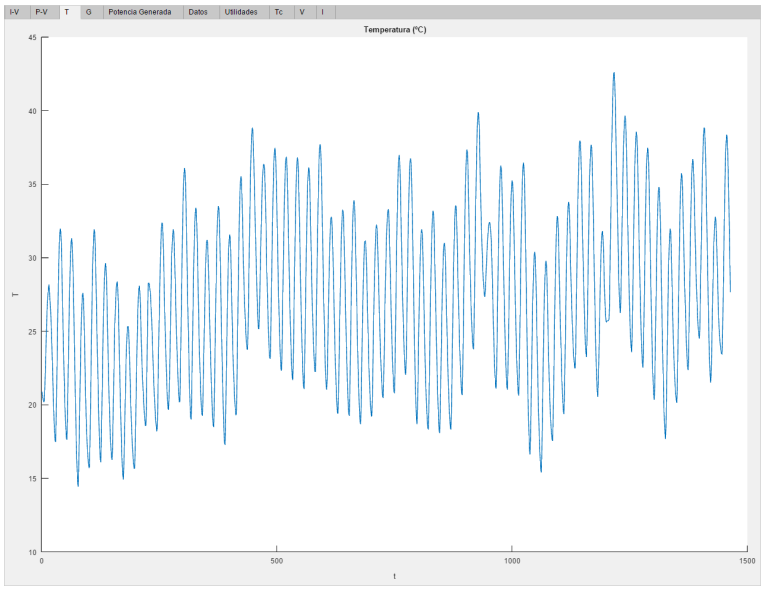


Figura 6.38 : temperatura ambiente en el parque Cortijo del Cura, Junio y Julio

- Potencia generada

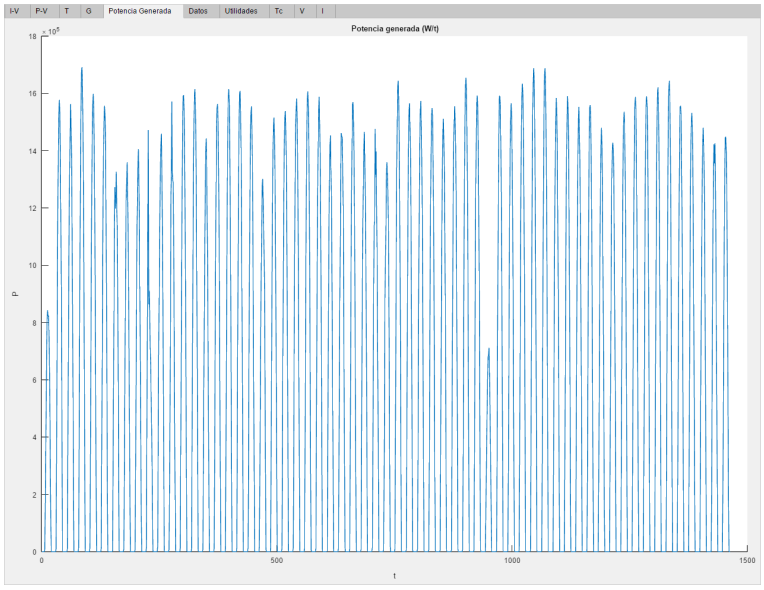


Figura 6.39 : potencia generada en el parque Cortijo del Cura, Junio y Julio

- Temperatura de módulo

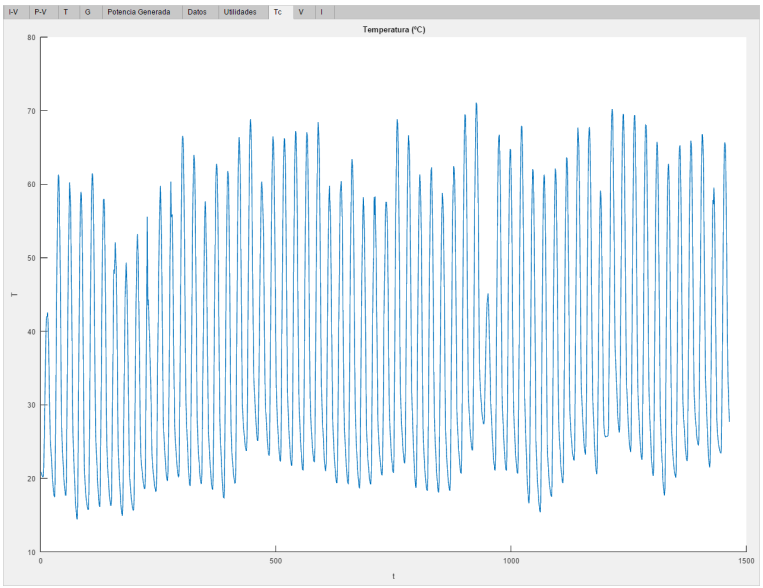


Figura 6.40 : temperatura de módulo en el parque Cortijo del Cura, Junio y Julio

- Voltaje en P_{mp}

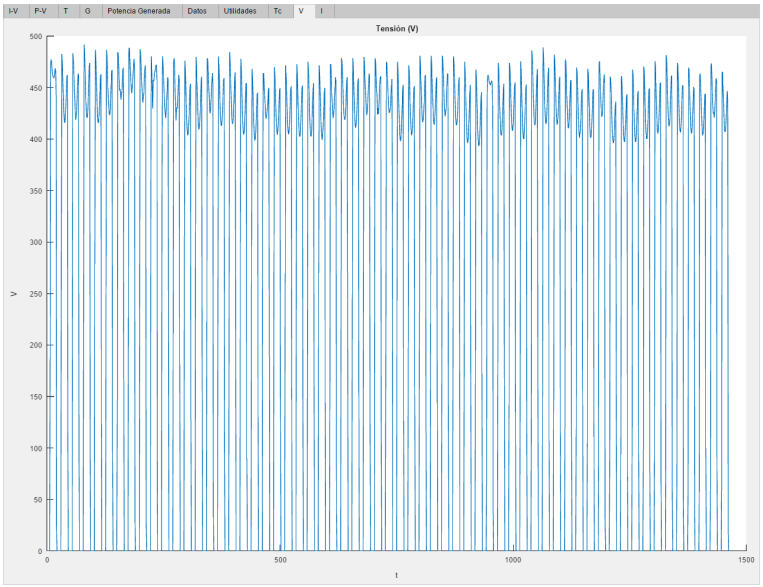


Figura 6.41 : voltaje de rama en el parque Cortijo del Cura, Junio y Julio

- Intensidad en P_{mp}

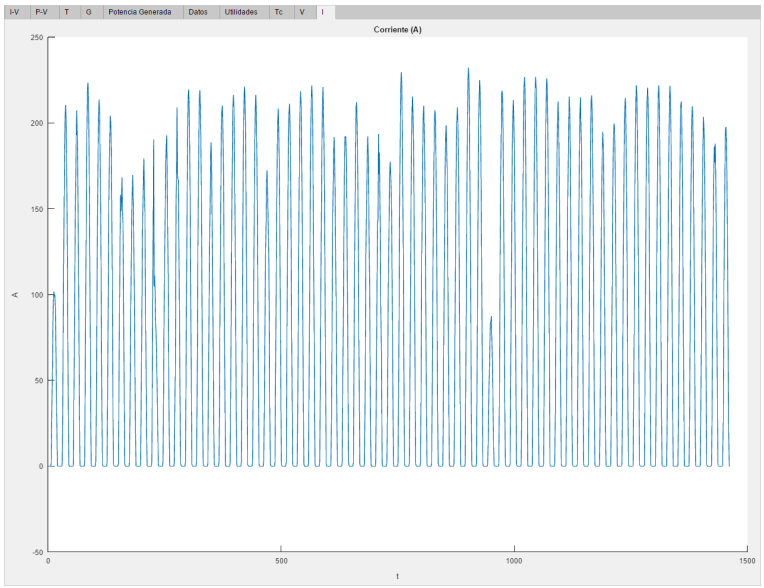


Figura 6.42 : corriente de instalación en el parque Cortijo del Cura, Junio y Julio

Datos numéricos:

- Potencia acumulada: 796.14 gW
- Rendimiento STC: 13.59 %
- Rendimiento máximo obtenido: 12.94 %
- Rendimiento medio: 11.68 %
- Potencia STC: 220.17 W
- Voltaje máximo: 491.63 V
- Voltaje medio: 403 V
- Corriente máxima: 232.03 A
- Corriente media: 112.32 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (4 de Julio en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

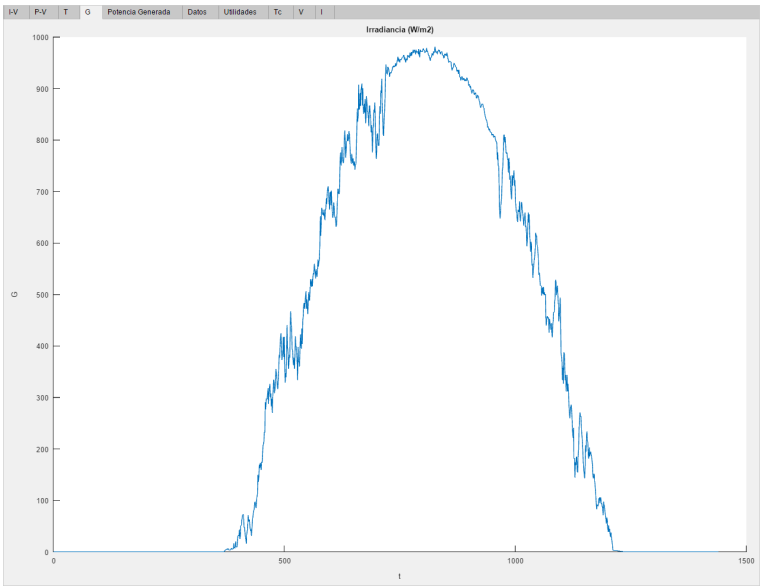


Figura 6.43 : irradiancia en el parque Cortijo del Cura, 4 de Junio

- Temperatura ambiente

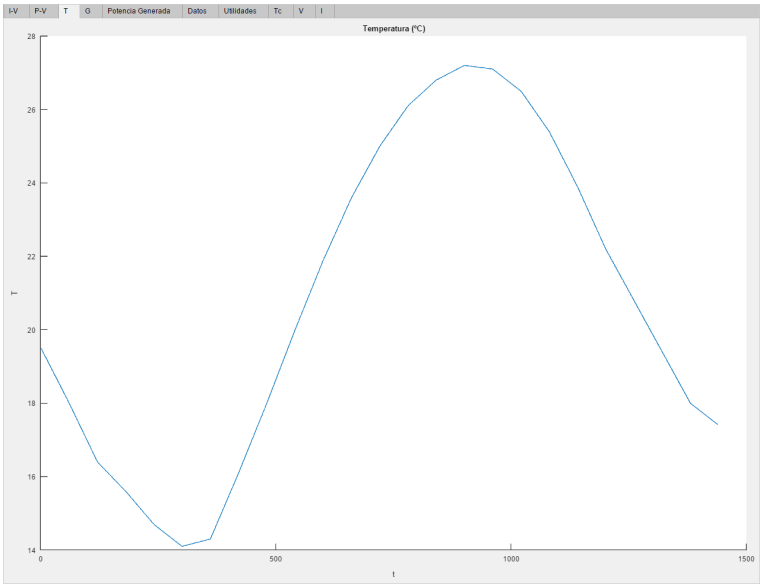


Figura 6.44 : temperatura ambiente en el parque Cortijo del Cura, 4 de Junio

- Potencia generada

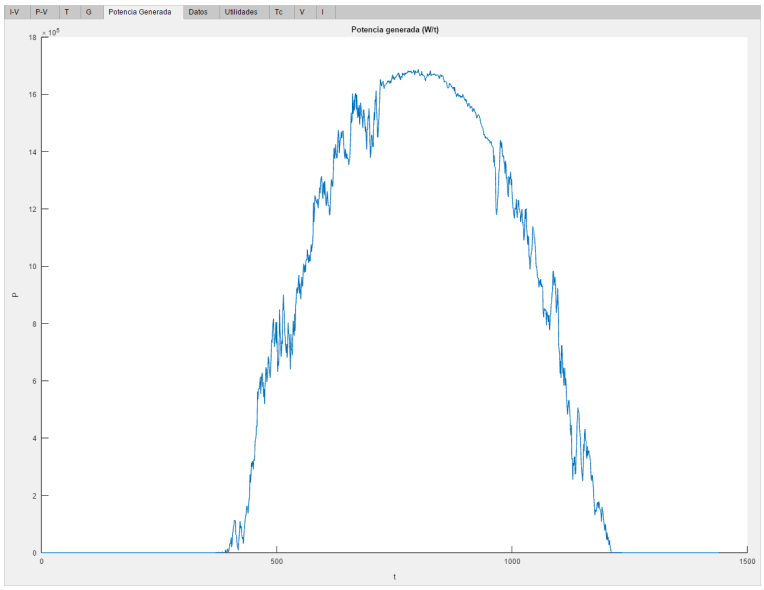


Figura 6.45 : potencia generada en el parque Cortijo del Cura, 4 de Junio

- Temperatura de módulo

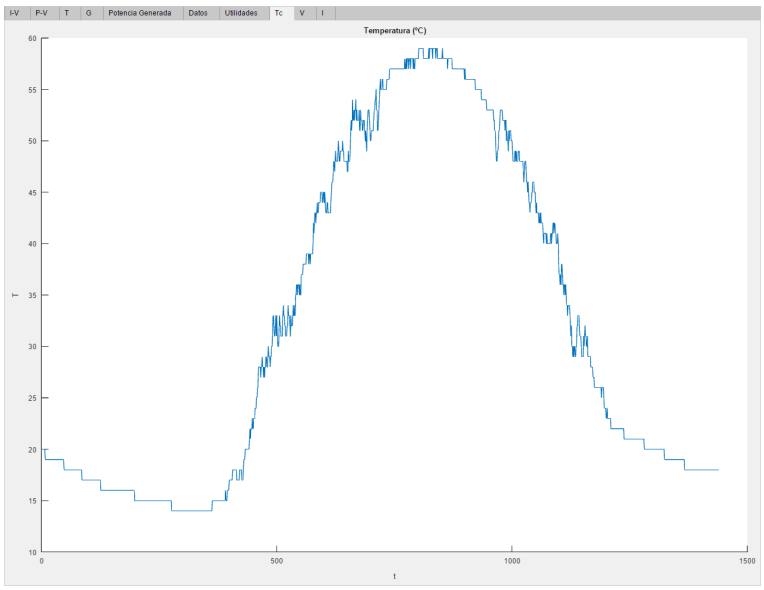


Figura 6.46 : temperatura de módulo en el parque Cortijo del Cura, 4 de Junio

- Voltaje en P_{mp}

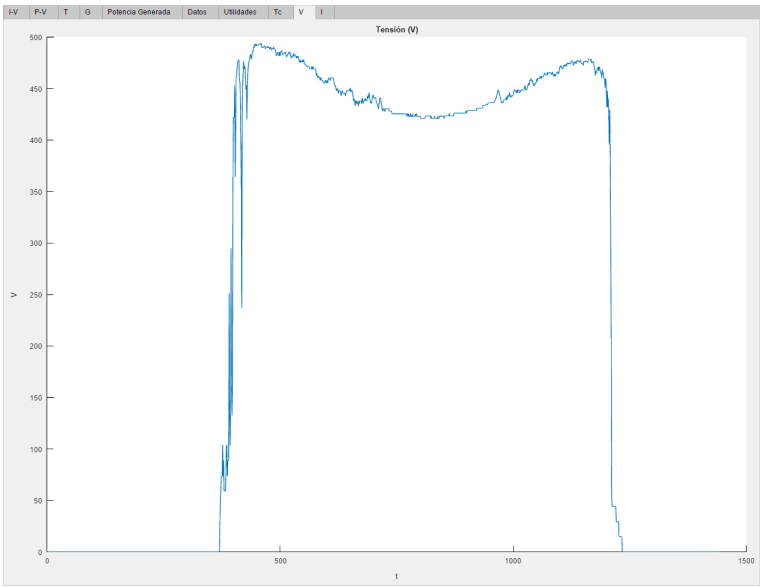


Figura 6.47 : voltaje de rama en el parque Cortijo del Cura, 4 de Junio

- Intensidad en P_{mp}

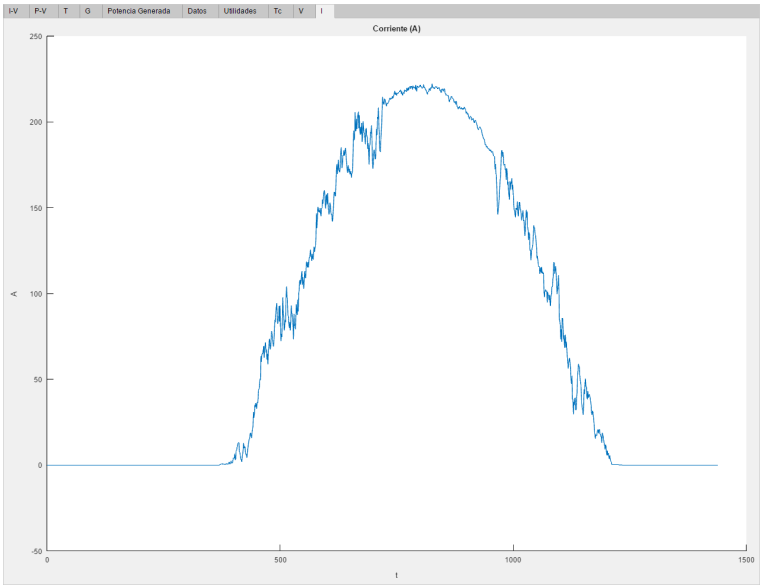


Figura 6.48 : corriente de instalación en el parque Cortijo del Cura, 4 de Junio

Datos numéricos:

- Potencia acumulada: 14.68 gW
- Rendimiento STC: 13.59 %

- Rendimiento máximo obtenido: 13.05 %
- Rendimiento medio: 12.02 %
- Potencia STC: 220.17 W
- Voltaje máximo: 494.13 V
- Voltaje medio: 427.83 V
- Corriente máxima: 222.05 A
- Corriente media: 128.22 A

6.3. Instalación solar en Frechen, Alemania

En ésta sección se simulará la instalación solar de Frechen [28].
Ésta huerta está situada en la localidad de Frechen, por lo que se utilizarán los datos meteorológicos obtenidos para Colonia, siendo la estación meteorológica más cercana.

Debido a que no se disponen de datos detallados del proyecto, se supondrá la interconexión de los módulos, para ajustarse al número de módulos proporcionado, y se elegirá un panel que concuerde con la potencia especificada. Características de la central:

- Dividida en 23 partes
- Paneles por instalación:
  - 4 módulos en serie (rama)
  - 40 ramas conectadas en paralelo
  - 80 módulos por instalación
- Total: 3680 módulos

En las especificaciones técnicas del proyecto, se especifica el módulo usado, un Schüco MPE-90-AL-01.

Con todos los datos obtenidos, se pasa a simular ésta huerta solar con la aplicación PreciSol.

Invierno

- Meses de invierno: para esta simulación se realizará una simulación de los meses de Enero y Febrero en Colonia, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

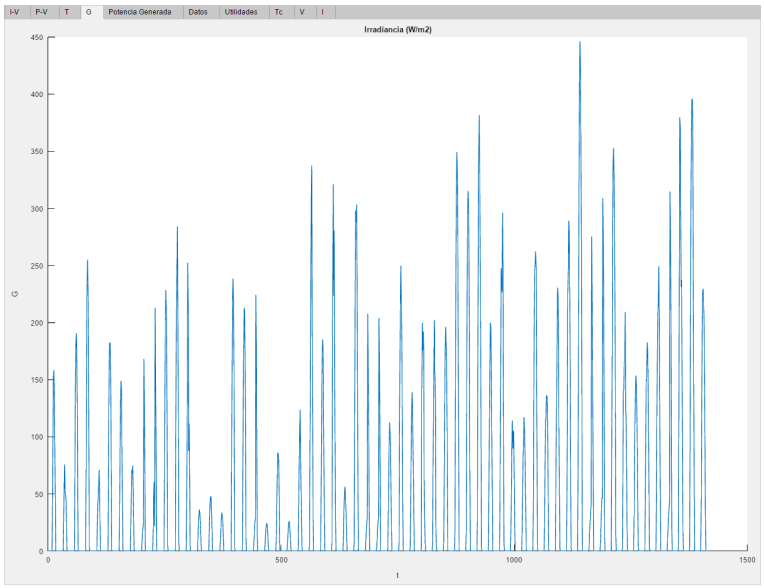


Figura 6.49 : irradiancia en la instalación de Frechen, Enero y Febrero

  - Temperatura ambiente

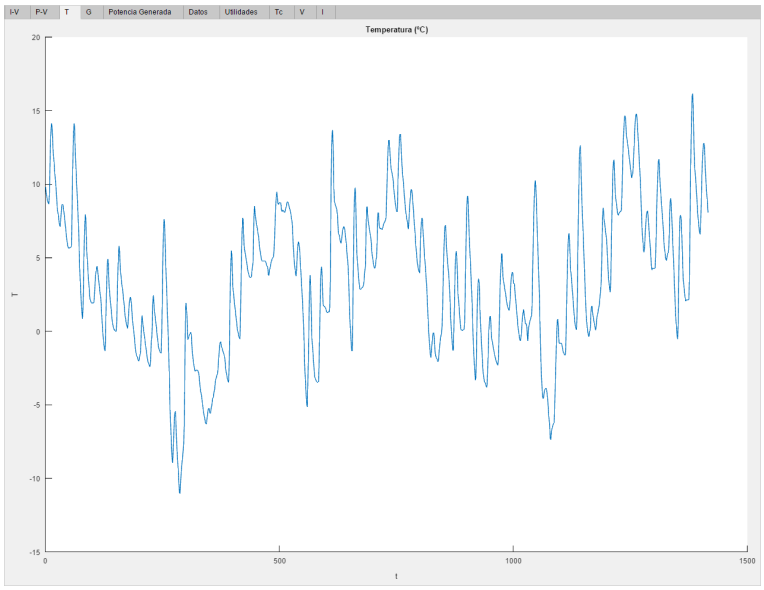


Figura 6.50 : temperatura ambiente en la instalación de Frechen, Enero y Febrero

- Potencia generada

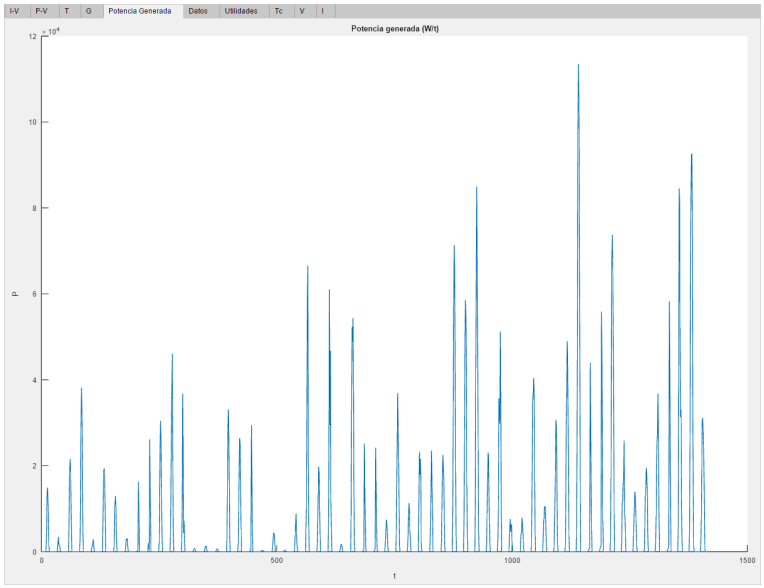


Figura 6.51 : potencia generada en la instalación de Frechen, Enero y Febrero

- Temperatura de módulo

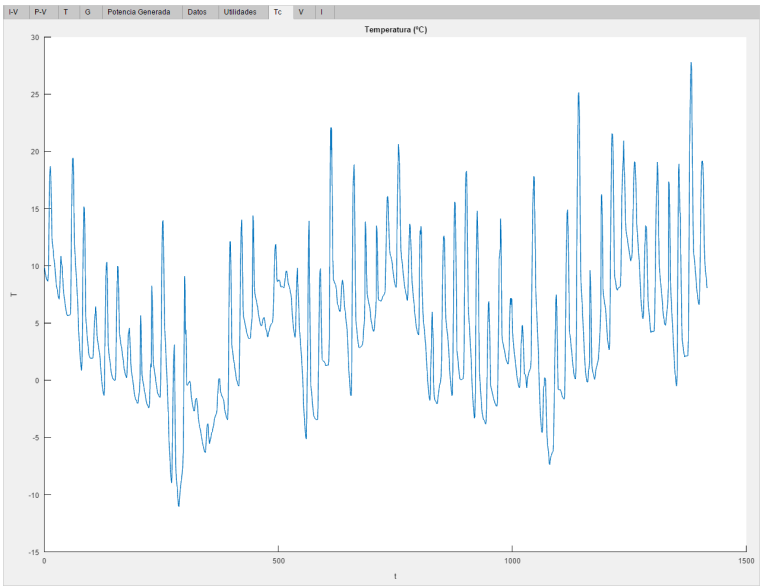


Figura 6.52 : temperatura de módulo en la instalación de Frechen, Enero y Febrero

- Voltaje en P_{mp}

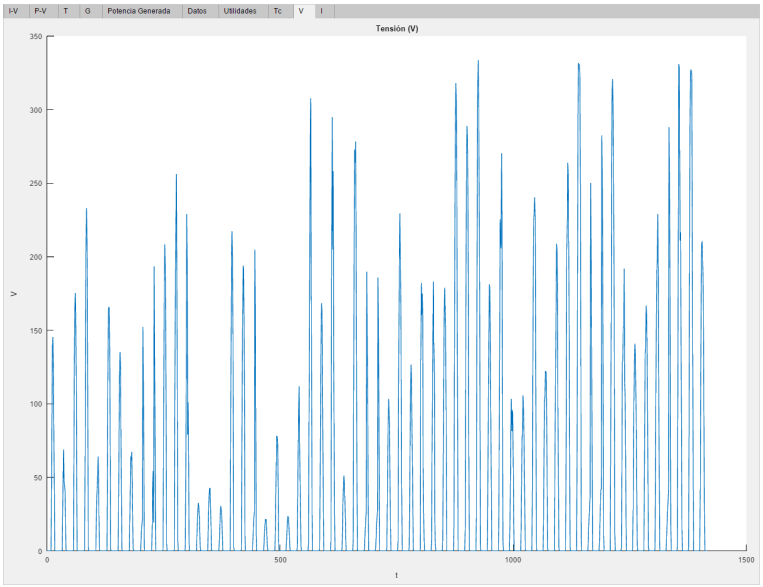


Figura 6.53 : voltaje de rama en la instalación de Frechen, Enero y Febrero

- Intensidad en P_{mp}

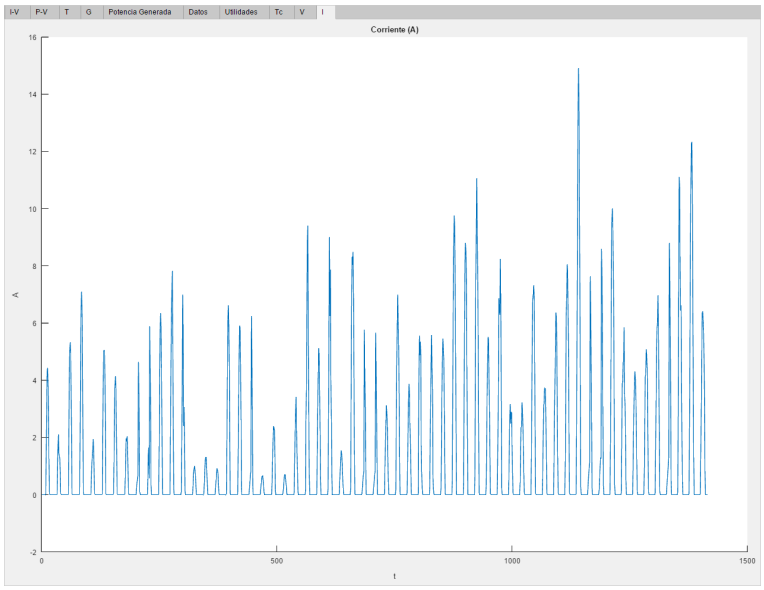


Figura 6.54 : corriente de instalación en la instalación de Frechen, Enero y Febrero

Datos numéricos:

- Potencia acumulada: 7.09 gW
- Rendimiento STC: 6.29 %
- Rendimiento máximo obtenido: 4.83 %
- Rendimiento medio: 2.2 %
- Potencia STC: 89.93 W
- Voltaje máximo: 333.62 V
- Voltaje medio: 94.91 V
- Corriente máxima: 14.91 A
- Corriente media: 2.93 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (28 de Febrero en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

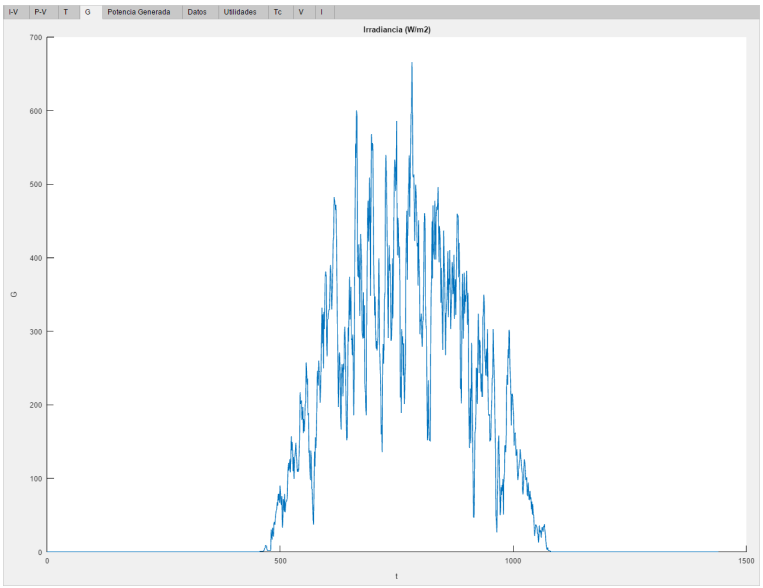


Figura 6.55 : irradiancia en la instalación de Frechen, 28 de Febrero

- Temperatura ambiente

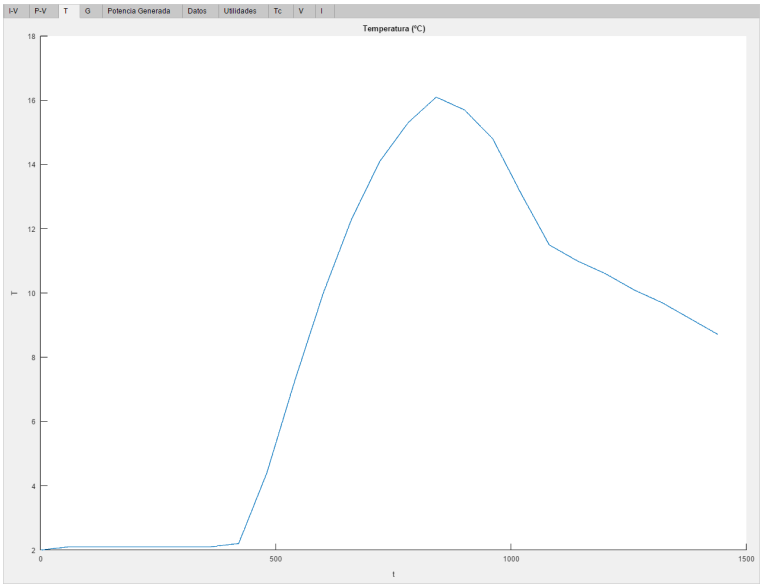


Figura 6.56 : temperatura ambiente en la instalación de Frechen, 28 de Febrero

- Potencia generada

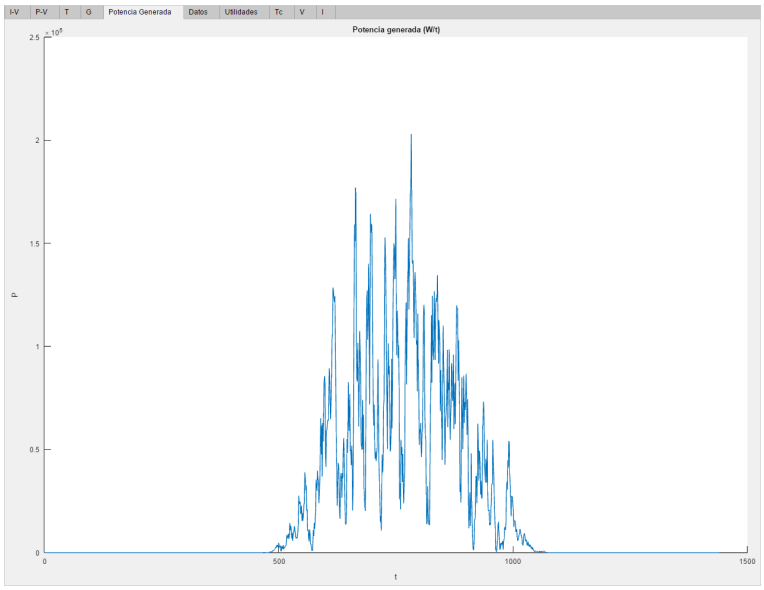


Figura 6.57 : potencia generada en la instalación de Frechen, 28 de Febrero

- Temperatura de módulo

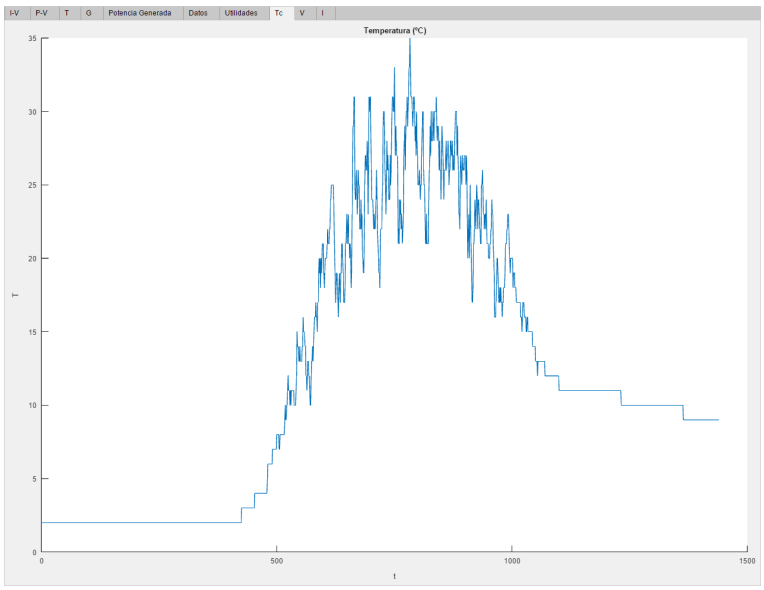


Figura 6.58 : temperatura de módulo en la instalación de Frechen, 28 de Febrero

- Voltaje en P_{mp}

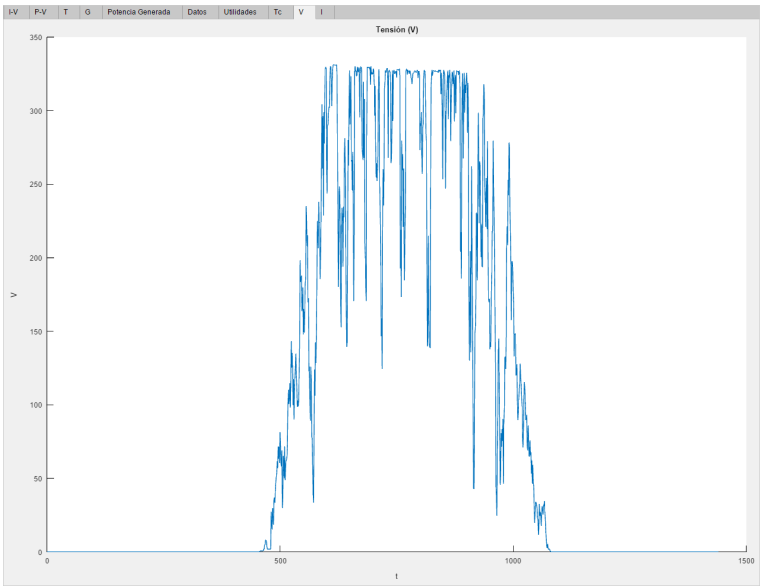


Figura 6.59 : voltaje de rama en la instalación de Frechen, 28 de Febrero

- Intensidad en P_{mp}

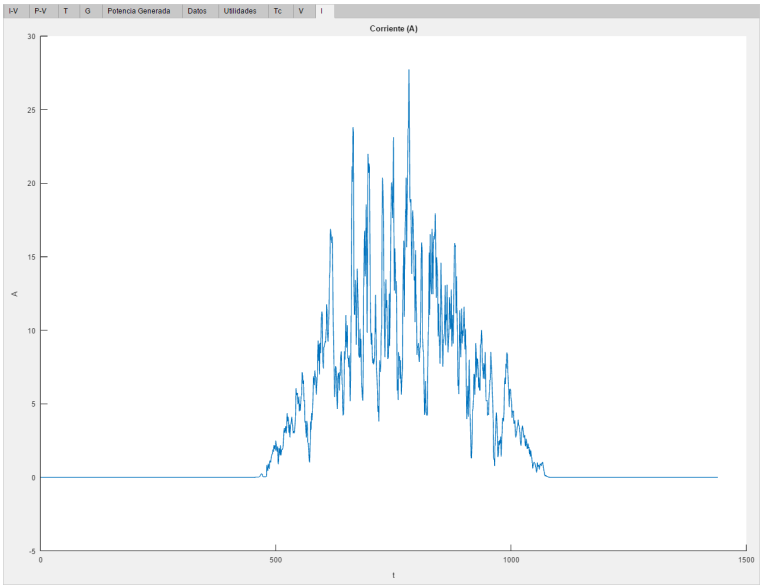


Figura 6.60 : corriente de instalación en la instalación de Frechen, 28 de Febrero

Datos numéricos:

- Potencia acumulada: 484.78 kW
- Rendimiento STC: 6.29 %

- Rendimiento máximo obtenido: 5.79 %
- Rendimiento medio: 3.67 %
- Potencia STC: 89.93 W
- Voltaje máximo: 331.31 V
- Voltaje medio: 202.23 V
- Corriente máxima: 27.74 A
- Corriente media: 7.38 A

Verano

- Meses de verano: para esta simulación se realizará una simulación de los meses de Junio y Julio en Colonia, seleccionando el modo de datos por hora.

  Gráficas:

  - Irradiancia

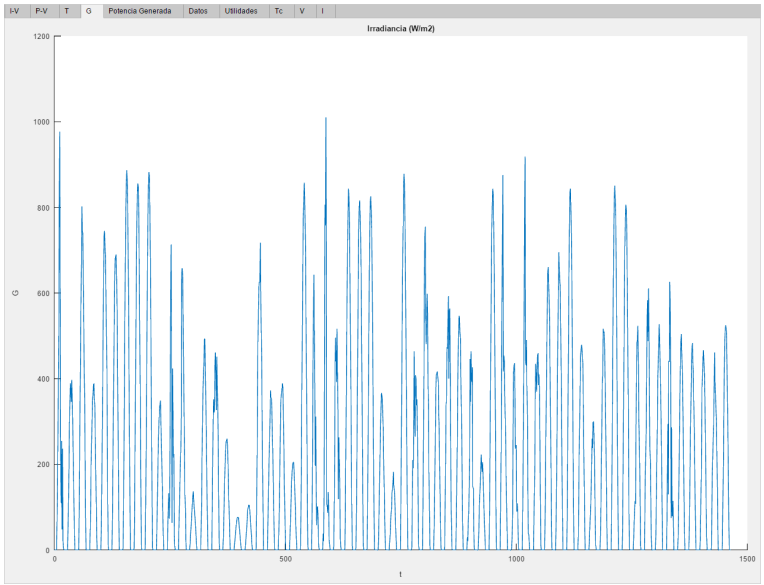


Figura 6.61 : irradiancia en la instalación de Frechen, Junio y Julio

  - Temperatura ambiente

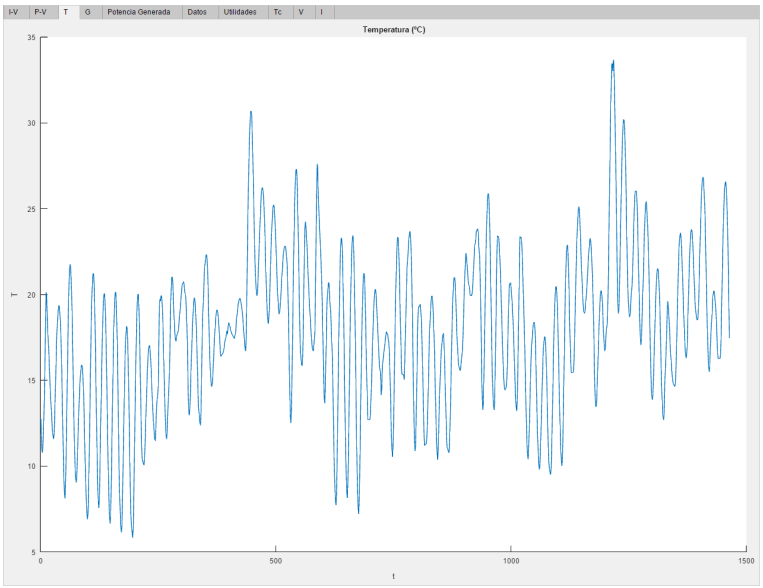


Figura 6.62 : temperatura ambiente en la instalación de Frechen, Junio y Julio

- Potencia generada

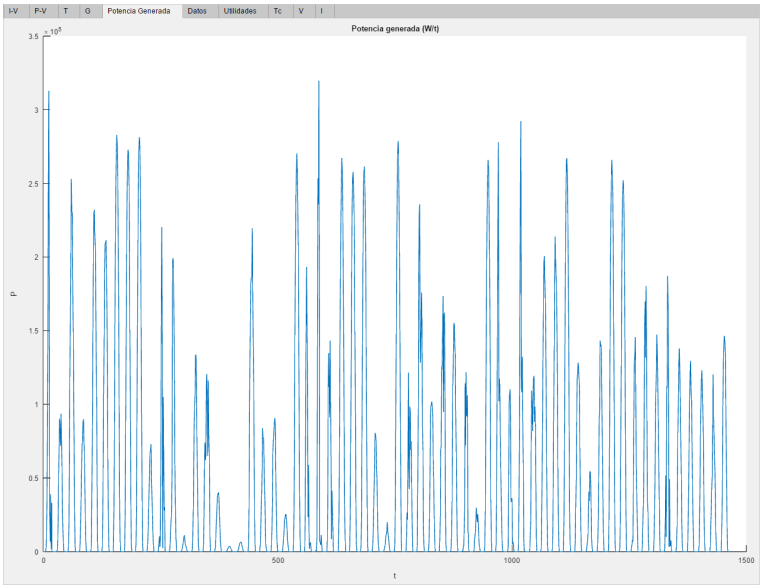


Figura 6.63 : potencia generada en la instalación de Frechen, Junio y Julio

- Temperatura de módulo

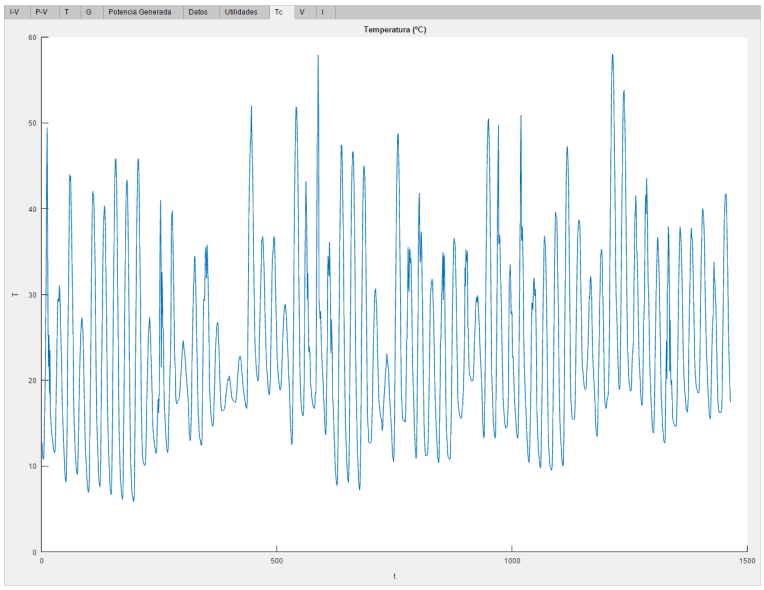


Figura 6.64 : temperatura de módulo en la instalación de Frechen, Junio y Julio

- Voltaje en P_{mp}

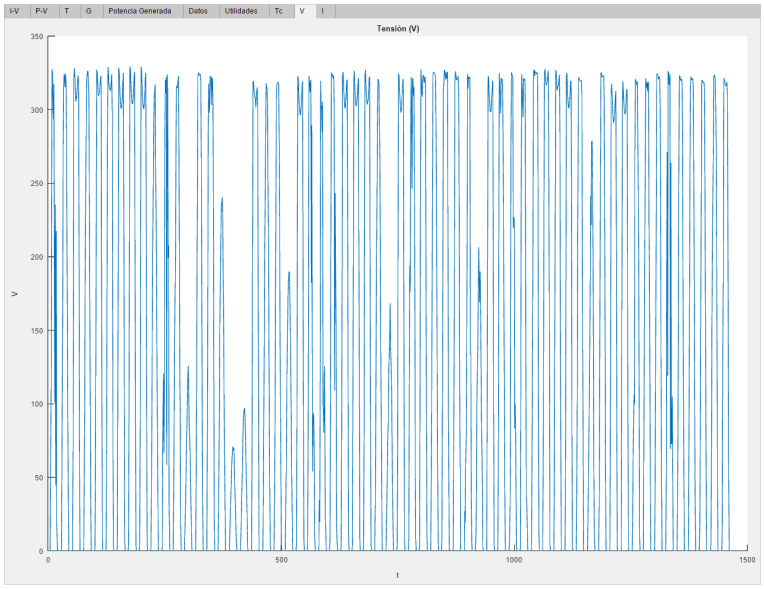


Figura 6.65 : voltaje de rama en la instalación de Frechen, Junio y Julio

- Intensidad en P_{mp}

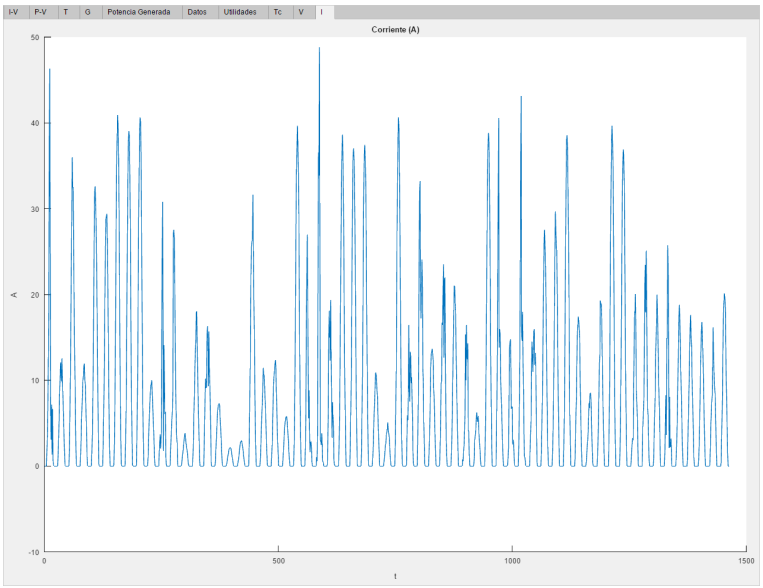


Figura 6.66 : corriente de instalación en la instalación de Frechen, Junio y Julio

Datos numéricos:

- Potencia acumulada: 70.37 gW
- Rendimiento STC: 6.29 %
- Rendimiento máximo obtenido: 6.08 %
- Rendimiento medio: 4.54 %
- Potencia STC: 89.93 W
- Voltaje máximo: 329.39 V
- Voltaje medio: 191.99 V
- Corriente máxima: 48.83 A
- Corriente media: 10.25 A

- Día tipo: para ésta simulación se ha seleccionado un día con una irradiancia razonable (1 de Junio en este caso), y se ha seleccionado el modo de datos por minuto.

  Gráficas:

  - Irradiancia

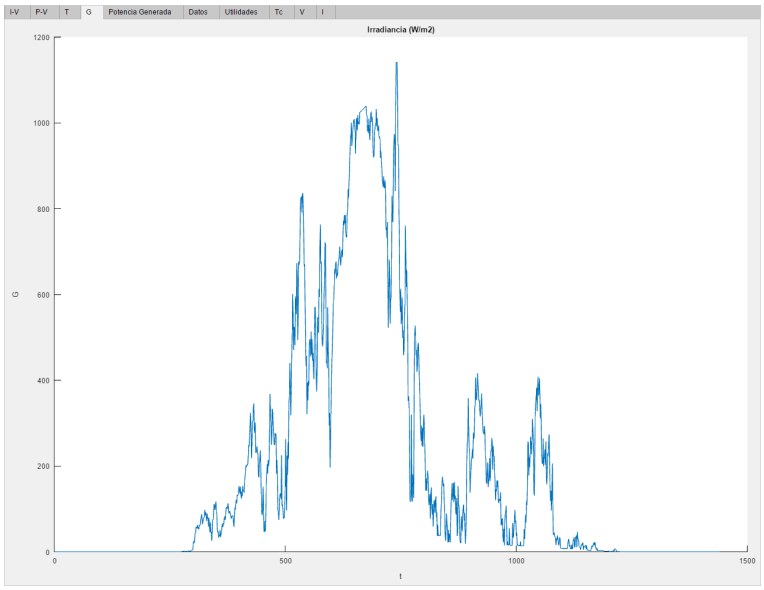


Figura 6.67 : irradiancia en la instalación de Frechen, 1 de Junio

- Temperatura ambiente

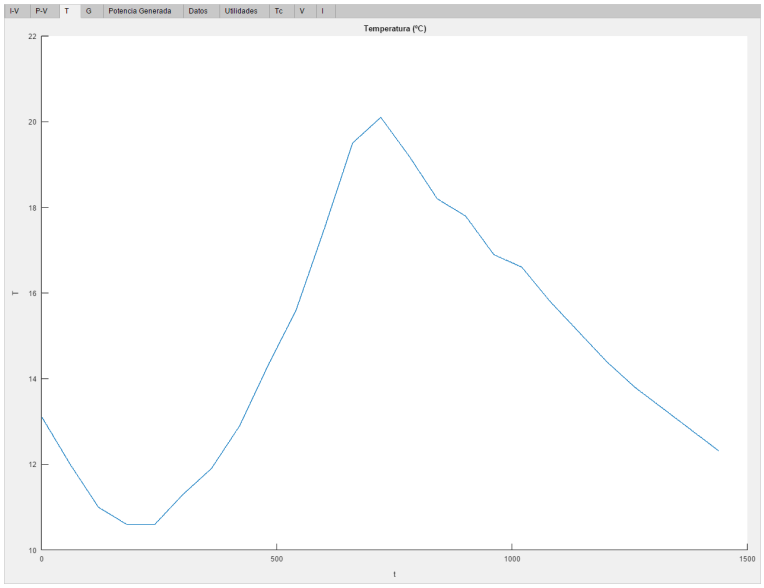


Figura 6.68 : temperatura ambiente en la instalación de Frechen, 1 de Junio

- Potencia generada

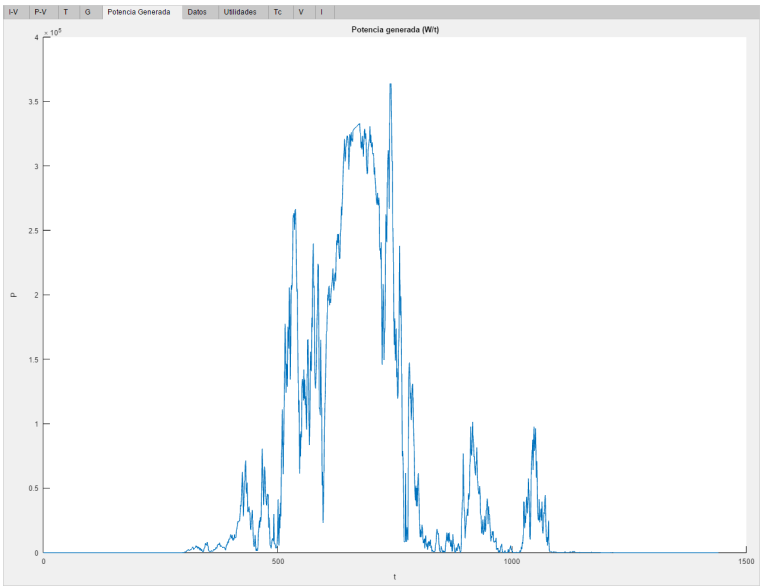


Figura 6.69 : potencia generada en la instalación de Frechen, 1 de Junio

- Temperatura de módulo

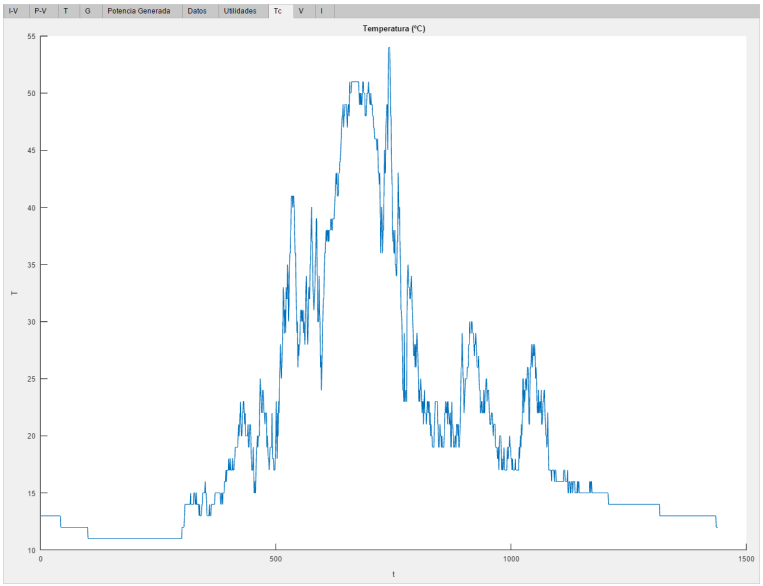


Figura 6.70 : temperatura de módulo en la instalación de Frechen, 1 de Junio

- Voltaje en P_{mp}

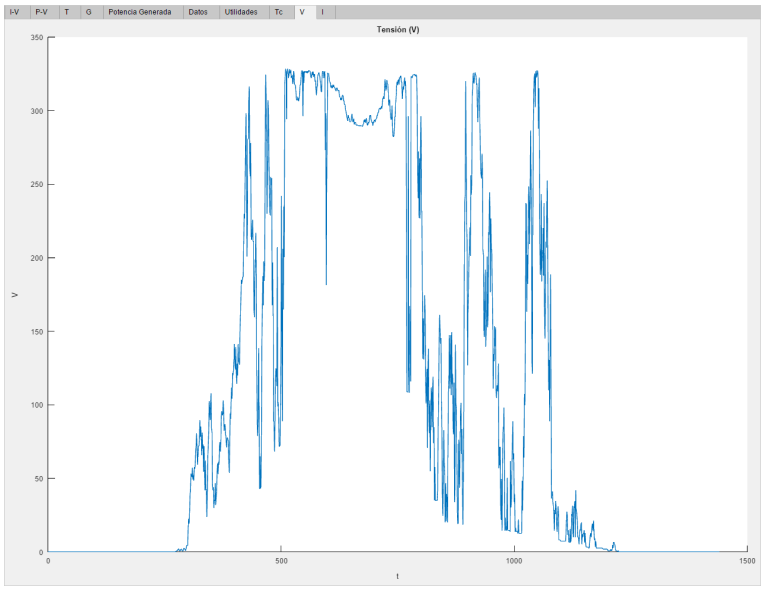


Figura 6.71 : voltaje de rama en la instalación de Frechen, 1 de Junio

- Intensidad en P_{mp}

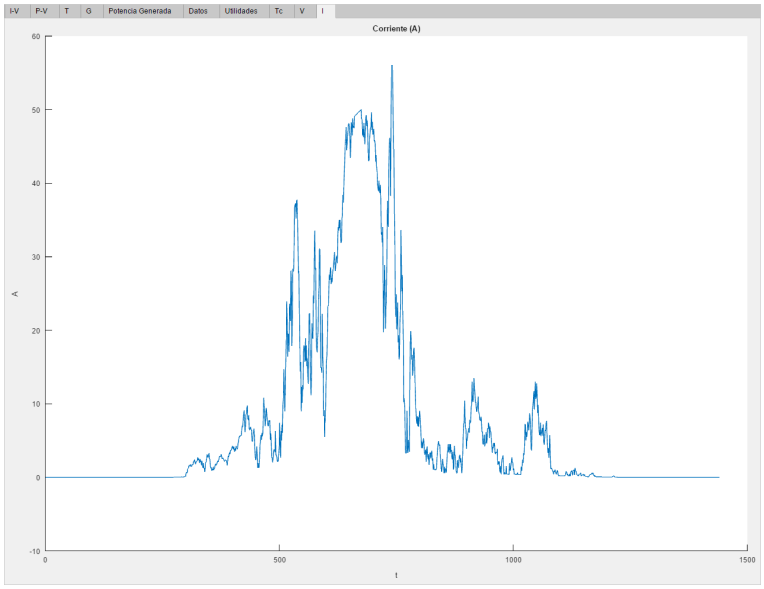


Figura 6.72 : corriente de instalación en la instalación de Frechen, 1 de Junio

Datos numéricos:

- Potencia acumulada: 1.14 gW
- Rendimiento STC: 6.29 %

- Rendimiento máximo obtenido: 6.1 %
- Rendimiento medio: 4.79 %
- Potencia STC: 89.93 W
- Voltaje máximo: 328.23 V
- Voltaje medio: 164.06 V
- Corriente máxima: 56.01 A
- Corriente media: 11.17 A

Capítulo 7

Conclusiones

Como ya se dijo en el resumen del estudio del estado del arte y de mercado, ésta tecnología está creciendo mucho más rápido que el resto de energías renovables, y cada vez hay tecnologías que satisfacen las necesidades de las diferentes y numerosas aplicaciones de la energía fotovoltaica.

Desarrollando el simulador, se han ido observando diferentes métodos de cálculo y de simulación; la inmensa mayoría, y exceptuando los simuladores de alta precisión a nivel físico, suelen ser bastante imprecisos, utilizando ecuaciones sencillas, muchos de ellos basándose en cálculos de pérdidas realizados sobre la potencia máxima en condiciones de test estándar (STC). Debido a que se deseaba un modelo capaz de simular el comportamiento completo de una célula y la curva I-V completa, se decidió por buscar un modelo matemático o circuito equivalente capaz de cumplir esta tarea con el compromiso de obtener una simulación precisa y fiel a los resultados reales, sin incurrir en tiempos de cálculo extremadamente largos. Éste compromiso se vio realizado en los modelos de 1 y 2 diodos expuestos en este trabajo [18], el cual es capaz de realizar cálculos razonablemente rápidos con el modelo de 2 diodos, sin incurrir en largos cálculos iterativos, más allá del cálculo inicial de las resistencias en serie y paralelo.
Como se ha podido observar en el análisis de resultados, los modelos implementados tienen una gran precisión a niveles de funcionamiento nominales, y una buena flexibilidad de valores, incurriendo en tiempos de cálculo de unos pocos segundos utilizando una alta precisión, de 100000 puntos.

Una vez cumplidos los objetivos primordiales del trabajo, representados por los capítulos de análisis del estado del arte, desarrollo y testeo del simulador, se decidió

desarrollar una aplicación más accesible, de fácil uso, con el objetivo de crear un simulador sencillo pero preciso para uso didáctico, con posibilidad de expansión para su uso con más módulos de los previstos y sets de datos meteorológicos para cualquier lugar (siempre que se use Meteonorm 6.0 y las utilidades previstas por PreciSol), además de fijar una plataforma para posibles futuros desarrollos, como la añadidura de los siguientes elementos en una instalación fotovoltaica. Una vez implementados dichos módulos adicionales, se podría utilizar a nivel ingenieril para hacer análisis provisionales de instalaciones completas y la posibilidad de usar los resultados para un estudio de viabilidad. Todos estos objetivos adicionales, debido a la actual extensión del trabajo, consumición de tiempo estimada por los mismos y alejamiento de los objetivos primordiales, han debido omitirse.

Bibliografía

[1] Flisom AG. *Flisom: Flexible PV from Lab to Fab*. Flisom AG., 2014.

[2] Fayçal Djeffalc Boumedième Benyoucefd Jean-Pierre Charles Ali Cheknanea, Hikmat S. Hilalb. An equivalent circuit approach to organic solar cell modelling. *Elsevier*, 2008.

[3] Mariusz Prorok Andrzej Dziedzic Tadeusz Z˙danowicz Barbara Werner, Wlodzimierz Kolodenny. Electrical modeling of cigs thin-film solar cells working in natural conditions. *Elsevier*, 2011.

[4] Solar Frontier CIS. Página web de solar frontier cis. `www.solar-frontier.com/`. Accedida: 2017-07-07.

[5] Diario de avisos. "Hallan en aguas de Canarias el mayor yacimiento mundial de telurio". `http://diariodeavisos.elespanol.com/2017/04/hallan-aguas-canarias-mayor-yacimiento-mundial-telurio/`. Accedida: 2017-07-07.

[6] EPFL. Página web de École polytechniqie fédérale de laussane. `https://www.epfl.ch/`. Accedida: 2017-07-07.

[7] EPSJ. Página web de escuela politécnica superior de jaén. `http://www.meteonorm.com/`. Accedida: 2017-08-23.

[8] Chopra et al. *Progress in Photovoltaics*. Chopra et al., 2004.

[9] Union Española Fotovoltaica. Informes anuales de la unef. `https://unef.es/`. Accedida: 2017-07-07.

[10] Steve Hoelzer. Página de mathworks de progressbar.m, desarrollado por steve hoelzer. `https://es.mathworks.com/matlabcentral/fileexchange/6922-progressbar`. Accedida: 2017-08-25.

[11] Energy Informative. "The Real Lifespan of Solar Panels". http://energyinformative.org/lifespan-solar-panels/. Accedida: 2017-08-23.

[12] Naoki Inoue and Yanhui Zou. *Physics of Solid State Ionics: Physical properties of perovskite-type lithium ionic conductor*. Takashi Sakuma and Haruyuki Takahashi (Eds.), 2006.

[13] Fraunhofer ISE. Página web de fraunhofer-institut für solar energiesysteme ise. https://www.ise.fraunhofer.de/. Accedida: 2017-07-07.

[14] Fraunhofer ISE. *Fraunhofer ISE Photovoltaics Report*. Fraunhofer ISE, 2014.

[15] ISFH. Página web del instituto para la energía solar fotovoltaica. http://www.isfh.de/institut_solarforschung/. Accedida: 2017-08-23.

[16] M. Norbert Fisch Dirk Christoffers Fritz Pfisterer Dieter Meissner Joachim Nitsch Joachim Luther, Michael Nast. *Solar Technology*. Wiley-VCH, 2002.

[17] K. Norrman F.C. Krebs Jørgensen, M. *Stability/degradation of polymer solar cells*. Solar Energy Materials and Solar Cells, 2008.

[18] Hamed Taheri Kashif Ishaque, Zainal Salam. Simple, fast and accurate two-diode model for photovoltaic modules. *Elsevier*, 2010.

[19] KRICT. Página web de korea research institute of chemical technology. http://english.krict.re.kr/eng/. Accedida: 2017-07-07.

[20] Asociación Lacecal. *Curso sobre Diseño e Instalación de Sistemas de Energía Fotovoltaica*. Asociación Lacecal, 2014.

[21] MathWorks. Página web de mathworks, creadores de matlab y simulink. https://es.mathworks.com/. Accedida: 2017-07-07.

[22] Meteonorm. Página web de meteonorm. http://www.meteonorm.com/. Accedida: 2017-08-23.

[23] Chandima Gomes Mohd Amran Radzi Mohammad Ismael Rezadad Shahrooz Hajighorbani Mohammad Reza Maghami, Hashim Hizam. Power loss due to soiling on solar panel: A review. *Elsevier*, 2016.

[24] IES María Moliner. *Visita a la huerta solar Villajimena II - 400kW Enerpal - IES María Moliner*. IES María Moliner, 2017.

[25] K. Morgan, D. V.; Board. *An Introduction To Semiconductor Microtechnology*. Chichester, West Sussex, England: John Wiley & Sons, 2nd ed. edition, 1991.

[26] El Mundo. "Un nuevo parque solar suministrará energía para un millar de viviendas". http://www.elmundo.es/elmundo/2009/06/24/andalucia_malaga/1245865941.html. Accedida: 2017-10-01.

[27] NREL. Página web de laboratorio nacional de energías renovables (nrel). https://www.nrel.gov/. Accedida: 2017-07-07.

[28] AIC Projects. "Dachinstallation Frechen". http://www.aic-projects.com/referenzen_freiflaechenanlagen_dachanlage_frechen.html. Accedida: 2017-10-01.

[29] PV-Tech. Investigaciónes de mercado de pv-tech. https://www.pv-tech.org/. Accedida: 2017-07-07.

[30] PVPMC. Página web de pv_lib. https://pvpmc.sandia.gov/applications/pv_lib-toolbox/. Accedida: 2017-08-23.

[31] PVSyst. Página web de pvsyst. http://www.pvsyst.com/. Accedida: 2017-08-23.

[32] REN21. Estudios anuales de energías renovables ren21. http://www.ren21.net/. Accedida: 2017-07-07.

[33] C. Sah. *Fundamentals of solid-state electronics*. World Scientific Publishing Co. Pvt. Ltd., Singapur, 1991.

[34] Sanyo. Tecnología solar sanyo-panasonic hit. https://eu-solar.panasonic.net/en/hit-heterojunction-sanyo-panasonic.htm. Accedida: 2017-07-07.

[35] P. Singh and N.M. Ravindra. *Temperature Dependence of Solar Cell Performance — an Analysis*. Solar Energy Materials and Solar Cells, 2012.

[36] Era Solar. Revista era solar.

[37] First solar. Página web de first solar. http://www.firstsolar.com/. Accedida: 2017-07-07.

[38] Solexel. Página web beamreach solar, antigua solexel (actualmente en bancarrota). http://www.beamreachsolar.com/. Accedida: 2017-07-07.

[39] IHS Technology. Investigaciónes de mercado ihs. https://technology.ihs.com/Categories/450474/solar-demand-industry. Accedida: 2017-07-07.

[40] UNINST. Página web de ulsan national institute of science and technology. http://www.unist.ac.kr/. Accedida: 2017-07-07.

[41] UST. Página web de hong kong university of science and technology. http://www.ust.hk/. Accedida: 2017-07-07.

[42] P. Singh Wiley-VCH, Weinheim 2006 and N.M. Ravindra. *Organic Electronics: Materials, Manufacturing and Applications*. Hagen Klauk (Ed.), 2012.

[43] X. Deng y E. A. Schiff. *Handbook of Photovoltaic Science and Engineering: "Amorphous Silicon-based Solar Cells"*. A. Luque and S. Hegedus, 2002.

[44] Kunio; Okabe Toru H. Yasuda, Kouji; Saegusa. *Production of Solar-grade Silicon by Halidothermic Reduction of Silicon Tetrachloride*. Metallurgical and Materials Transactions, 2010.

Apéndice A

Anexos

A.1. Análisis de los modelos implementados

- model_evaluation.m

```
% ------------------------------------------------------------------------
% Script "model_evaluation":
%     Script usado para comparar los resultados obtenidos por los
% simuladores entre s  y con los valores reales para obtener
% conclusiones sobre su funcionamiento y precisi n
% ------------------------------------------------------------------------

% Limpieza previa
clear;
close all;

% Carga de modelos y limpieza de memoria
load_system('sim_1_diode'); % 1 diodo
load_system('sim_2_diode'); % 2 diodos
precision = 100000;

% Datos de m dulo
% ------------------------------------------------------------------------
Voc_stc = 46.2;
Isc_stc = 9.48;
Pmp_stc = 340;
Vmp_stc = 37.9;
Imp_stc = 8.97;
Voc_noct = 43;
Isc_noct = 7.66;
Pmp_noct = 250;
Vmp_noct = 34.9;
Imp_noct = 7.16;
Ns = 72;
Kv = -143;
Ki = 5.02;
Rs = [0.295, 0.291];
```

```
Rp = [581.4, 604.7];
TONC = 0.0288;
% ------------------------------------------------------------------------


% Datos de pruebas
% ------------------------------------------------------------------------
G_stc = 1000;
Tc_stc = 25 + 273.15;
G_noct = 800;
Tc_noct = 43 + 273.15;
G_gtest = [200, 400, 600, 800, 1000];
Tc_gtest = 25 + 273.15;
G_ttest = 1000;
Tc_ttest = [5, 25, 45, 65] + 273.15;
% ------------------------------------------------------------------------

% Barra de progreso
progressbar(0);
progressbar('Progreso simulaci n');
n_sims = 22;

% Test STC
[I_stc_1, V_stc_1] = fast_diode(Voc_stc, Isc_stc, Ns, Kv, Ki, Rs, ...
    Rp, G_stc, Tc_stc, precision, false);
[I_stc_2, V_stc_2] = fast_diode(Voc_stc, Isc_stc, Ns, Kv, Ki, Rs, ...
    Rp, G_stc, Tc_stc, precision, true);
progressbar(1/11);
I_stc_sc_1_index = find(V_stc_1 <= 0);
V_stc_oc_1_index = find(I_stc_1 <= 0);
I_stc_sc_1 = I_stc_1(I_stc_sc_1_index(1));
V_stc_oc_1 = V_stc_1(V_stc_oc_1_index(1));
P_stc_1 = I_stc_1.*V_stc_1;
P_stc_mp_1 = max(P_stc_1);
I_V_stc_mp_1_index = find(P_stc_1 == P_stc_mp_1);
I_stc_mp_1 = I_stc_1(I_V_stc_mp_1_index);
V_stc_mp_1 = V_stc_1(I_V_stc_mp_1_index);
% disp(I_stc_sc_1);
% disp(V_stc_oc_1);
% disp(P_stc_mp_1);
% disp(I_stc_mp_1);
% disp(V_stc_mp_1);
I_stc_sc_2_index = find(V_stc_2 <= 0);
V_stc_oc_2_index = find(I_stc_2 <= 0);
I_stc_sc_2 = I_stc_2(I_stc_sc_2_index(1));
V_stc_oc_2 = V_stc_2(V_stc_oc_2_index(1));
P_stc_2 = I_stc_2.*V_stc_2;
P_stc_mp_2 = max(P_stc_2);
I_V_stc_mp_2_index = find(P_stc_2 == P_stc_mp_2);
I_stc_mp_2 = I_stc_2(I_V_stc_mp_2_index);
V_stc_mp_2 = V_stc_2(I_V_stc_mp_2_index);
% disp(I_stc_sc_2);
% disp(V_stc_oc_2);
% disp(P_stc_mp_2);
```

```
% disp(I_stc_mp_2);
% disp(V_stc_mp_2);

% Test NOCT
[I_noct_1, V_noct_1] = fast_diode(Voc_stc, Isc_stc, Ns, Kv, Ki, Rs, ...
    Rp, G_noct, Tc_noct, precision, false);
[I_noct_2, V_noct_2] = fast_diode(Voc_stc, Isc_stc, Ns, Kv, Ki, Rs, ...
    Rp, G_noct, Tc_noct, precision, true);
progressbar(2/11);
I_noct_sc_1_index = find(V_noct_1 <= 0);
V_noct_oc_1_index = find(I_noct_1 <= 0);
I_noct_sc_1 = I_noct_1(I_noct_sc_1_index(1));
V_noct_oc_1 = V_noct_1(V_noct_oc_1_index(1));
P_noct_1 = I_noct_1.*V_noct_1;
P_noct_mp_1 = max(P_noct_1);
I_V_noct_mp_1_index = find(P_noct_1 == P_noct_mp_1);
I_noct_mp_1 = I_noct_1(I_V_noct_mp_1_index);
V_noct_mp_1 = V_noct_1(I_V_noct_mp_1_index);
% disp(I_noct_sc_1);
% disp(V_noct_oc_1);
% disp(P_noct_mp_1);
% disp(I_noct_mp_1);
% disp(V_noct_mp_1);
I_noct_sc_2_index = find(V_noct_2 <= 0);
V_noct_oc_2_index = find(I_noct_2 <= 0);
I_noct_sc_2 = I_noct_2(I_noct_sc_2_index(1));
V_noct_oc_2 = V_noct_2(V_noct_oc_2_index(1));
P_noct_2 = I_noct_2.*V_noct_2;
P_noct_mp_2 = max(P_noct_2);
I_V_noct_mp_2_index = find(P_noct_2 == P_noct_mp_2);
I_noct_mp_2 = I_noct_2(I_V_noct_mp_2_index);
V_noct_mp_2 = V_noct_2(I_V_noct_mp_2_index);
% disp(I_noct_sc_2);
% disp(V_noct_oc_2);
% disp(P_noct_mp_2);
% disp(I_noct_mp_2);
% disp(V_noct_mp_2);

% An lisis de puntos generados por el simulador
i_v_length = length(I_stc_1);
gtest_length = length(G_gtest);
ttest_length = length(Tc_ttest);

% Test irradiancia variable
I_gtest_1 = zeros([gtest_length, i_v_length]);
V_gtest_1 = zeros([gtest_length, i_v_length]);
I_gtest_2 = zeros([gtest_length, i_v_length]);
V_gtest_2 = zeros([gtest_length, i_v_length]);
I_gtest_sc_1 = zeros([gtest_length, 1]);
V_gtest_oc_1 = zeros([gtest_length, 1]);
P_gtest_mp_1 = zeros([gtest_length, 1]);
I_gtest_mp_1 = zeros([gtest_length, 1]);
V_gtest_mp_1 = zeros([gtest_length, 1]);
I_gtest_sc_2 = zeros([gtest_length, 1]);
```

```
V_gtest_oc_2 = zeros([gtest_length, 1]);
P_gtest_mp_2 = zeros([gtest_length, 1]);
I_gtest_mp_2 = zeros([gtest_length, 1]);
V_gtest_mp_2 = zeros([gtest_length, 1]);
for i = 1:length(G_gtest)
    [I_gtest_1(i, :), V_gtest_1(i, :)] = fast_diode(Voc_stc, Isc_stc, ...
        Ns, Kv, Ki, Rs, Rp, G_gtest(i), Tc_gtest, precision, false);
    [I_gtest_2(i, :), V_gtest_2(i, :)] = fast_diode(Voc_stc, Isc_stc, ...
        Ns, Kv, Ki, Rs, Rp, G_gtest(i), Tc_gtest, precision, true);
    progressbar((i+2)/11);
    I_gtest_sc_1_index = find(V_gtest_1(i, :) <= 0);
    V_gtest_oc_1_index = find(I_gtest_1(i, :) <= 0);
    I_gtest_sc_1(i) = I_gtest_1(i, I_gtest_sc_1_index(1));
    V_gtest_oc_1(i) = V_gtest_1(i, V_gtest_oc_1_index(1));
    P_gtest_1 = I_gtest_1(i, :).*V_gtest_1(i, :);
    P_gtest_mp_1(i) = max(P_gtest_1);
    I_V_gtest_mp_1_index = find(P_gtest_1 == P_gtest_mp_1(i));
    I_gtest_mp_1(i) = I_gtest_1(i, I_V_gtest_mp_1_index);
    V_gtest_mp_1(i) = V_gtest_1(i, I_V_gtest_mp_1_index);
%     disp(I_gtest_sc_1(i));
%     disp(V_gtest_oc_1(i));
%     disp(P_gtest_mp_1(i));
%     disp(I_gtest_mp_1(i));
%     disp(V_gtest_mp_1(i));
    I_gtest_sc_2_index = find(V_gtest_2(i, :) <= 0);
    V_gtest_oc_2_index = find(I_gtest_2(i, :) <= 0);
    I_gtest_sc_2(i) = I_gtest_2(i, I_gtest_sc_2_index(1));
    V_gtest_oc_2(i) = V_gtest_2(i, V_gtest_oc_2_index(1));
    P_gtest_2 = I_gtest_2(i, :).*V_gtest_2(i, :);
    P_gtest_mp_2(i) = max(P_gtest_2);
    I_V_gtest_mp_2_index = find(P_gtest_2 == P_gtest_mp_2(i));
    I_gtest_mp_2(i) = I_gtest_2(i, I_V_gtest_mp_2_index);
    V_gtest_mp_2(i) = V_gtest_2(i, I_V_gtest_mp_2_index);
%     disp(I_gtest_sc_2(i));
%     disp(V_gtest_oc_2(i));
%     disp(P_gtest_mp_2(i));
%     disp(I_gtest_mp_2(i));
%     disp(V_gtest_mp_2(i));
end

% Test temperatura variable
I_ttest_1 = zeros([ttest_length, i_v_length]);
V_ttest_1 = zeros([ttest_length, i_v_length]);
I_ttest_2 = zeros([ttest_length, i_v_length]);
V_ttest_2 = zeros([ttest_length, i_v_length]);
I_ttest_sc_1 = zeros([ttest_length, 1]);
V_ttest_oc_1 = zeros([ttest_length, 1]);
P_ttest_mp_1 = zeros([ttest_length, 1]);
I_ttest_mp_1 = zeros([ttest_length, 1]);
V_ttest_mp_1 = zeros([ttest_length, 1]);
I_ttest_sc_2 = zeros([ttest_length, 1]);
V_ttest_oc_2 = zeros([ttest_length, 1]);
P_ttest_mp_2 = zeros([ttest_length, 1]);
I_ttest_mp_2 = zeros([ttest_length, 1]);
```

```
V_ttest_mp_2 = zeros([ttest_length, 1]);
for i = 1:length(Tc_ttest)
    [I_ttest_1(i, :), V_ttest_1(i, :)] = fast_diode(Voc_stc, Isc_stc, ...
        Ns, Kv, Ki, Rs, Rp, G_ttest, Tc_ttest(i), precision, false);
    [I_ttest_2(i, :), V_ttest_2(i, :)] = fast_diode(Voc_stc, Isc_stc, ...
    Ns, Kv, Ki, Rs, Rp, G_ttest, Tc_ttest(i), precision, true);
    progressbar((i+7)/11);
    I_ttest_sc_1_index = find(V_ttest_1(i, :) <= 0);
    V_ttest_oc_1_index = find(I_ttest_1(i, :) <= 0);
    % Fix----------------------------------------------------------------
    if isempty(V_ttest_oc_1_index)
        V_ttest_oc_1_index(1) = length(I_ttest_1(i, :));
    end
    % Fix----------------------------------------------------------------
    I_ttest_sc_1(i) = I_ttest_1(i, I_ttest_sc_1_index(1));
    V_ttest_oc_1(i) = V_ttest_1(i, V_ttest_oc_1_index(1));
    P_ttest_1 = I_ttest_1(i, :).*V_ttest_1(i, :);
    P_ttest_mp_1(i) = max(P_ttest_1);
    I_V_ttest_mp_1_index = find(P_ttest_1 == P_ttest_mp_1(i));
    I_ttest_mp_1(i) = I_ttest_1(i, I_V_ttest_mp_1_index);
    V_ttest_mp_1(i) = V_ttest_1(i, I_V_ttest_mp_1_index);
%     disp(I_ttest_sc_1(i));
%     disp(V_ttest_oc_1(i));
%     disp(P_ttest_mp_1(i));
%     disp(I_ttest_mp_1(i));
%     disp(V_ttest_mp_1(i));
    I_ttest_sc_2_index = find(V_ttest_2(i, :) <= 0);
    V_ttest_oc_2_index = find(I_ttest_2(i, :) <= 0);
    % Fix----------------------------------------------------------------
    if isempty(V_ttest_oc_2_index)
        V_ttest_oc_2_index(1) = length(I_ttest_2(i, :));
    end
    % Fix----------------------------------------------------------------
    I_ttest_sc_2(i) = I_ttest_2(i, I_ttest_sc_2_index(1));
    V_ttest_oc_2(i) = V_ttest_2(i, V_ttest_oc_2_index(1));
    P_ttest_2 = I_ttest_2(i, :).*V_ttest_2(i, :);
    P_ttest_mp_2(i) = max(P_ttest_2);
    I_V_ttest_mp_2_index = find(P_ttest_2 == P_ttest_mp_2(i));
    I_ttest_mp_2(i) = I_ttest_2(i, I_V_ttest_mp_2_index);
    V_ttest_mp_2(i) = V_ttest_2(i, I_V_ttest_mp_2_index);
%     disp(I_ttest_sc_2(i));
%     disp(V_ttest_oc_2(i));
%     disp(P_ttest_mp_2(i));
%     disp(I_ttest_mp_2(i));
%     disp(V_ttest_mp_2(i));
end

% Finalizaci n de simulaciones
progressbar(1);


% Comparaciones Gr ficas
C = {'k','b','r','g','y'};
```

```
% Figura 1: STC
figure(1);
hold on;
plot(V_stc_1, I_stc_1, 'color', C{1}, 'LineWidth', 2);
xlabel('V');
ylabel('I');
xlim([0 inf]);
ylim([0 inf]);
plot(V_stc_2, I_stc_2, 'color', C{2}, 'LineWidth', 2);
xlabel('V');
ylabel('I');
xlim([0 inf]);
ylim([0 inf]);
legend('1 diodo', '2 diodos');
hold off;

% Figura 2: NOCT
figure(2);
hold on;
plot(V_noct_1, I_noct_1, 'color', C{1}, 'LineWidth', 2);
xlabel('V');
ylabel('I');
xlim([0 inf]);
ylim([0 inf]);
plot(V_noct_2, I_noct_2, 'color', C{2}, 'LineWidth', 2);
xlabel('V');
ylabel('I');
xlim([0 inf]);
ylim([0 inf]);
legend('1 diodo', '2 diodos');
hold off;

% Figura 3 y 4: Comparativa irradiancias
% Modelo de 1 diodo
figure(3);
hold on;
for i = 1:length(G_gtest)
    plot(V_gtest_1(i, :), I_gtest_1(i, :), 'color', C{i}, ...
        'LineWidth', 2);
    xlabel('V')
    ylabel('I')
    xlim([0 inf])
    ylim([0 inf])
end
legend('200 W/m^2', '400 W/m^2', '600 W/m^2', '800 W/m^2', '1000 W/m^2');
hold off;
% Modelo de 2 diodos
figure(4);
hold on;
for i = 1:length(G_gtest)
    plot(V_gtest_2(i, :), I_gtest_2(i, :), 'color', C{i}, ...
        'LineWidth', 2);
    xlabel('V')
    ylabel('I')
```

```
    xlim([0 inf])
    ylim([0 inf])
end
legend('200 W/m^2', '400 W/m^2', '600 W/m^2', '800 W/m^2', '1000 W/m^2');
hold off;

% Figura 5 y 6: Comparativa irradiancias
% Modelo de 1 diodo
figure(5);
hold on;
for i = 1:length(Tc_ttest)
    plot(V_ttest_1(i, :), I_ttest_1(i, :), 'color', C{i}, ...
        'LineWidth', 2);
    xlabel('V')
    ylabel('I')
    xlim([0 inf])
    ylim([0 inf])
end
legend('5 ºC', '25 ºC', '45 ºC', '65 ºC');
hold off;
% Modelo de 2 diodos
figure(6);
hold on;
for i = 1:length(Tc_ttest)
    plot(V_ttest_2(i, :), I_ttest_2(i, :), 'color', C{i}, ...
        'LineWidth', 2);
    xlabel('V')
    ylabel('I')
    xlim([0 inf])
    ylim([0 inf])
end
legend('5 ºC', '25 ºC', '45 ºC', '65 ºC');
hold off;

% Escritura con resultados de precisi n
file_path = uigetdir(pwd);
filename = strcat(file_path, '\model_evaluation_results.txt');

fid = fopen(filename, 'wt');
fprintf(fid, 'Resultados STC\r\n');
fprintf(fid, '\tModelo de 1 diodo:\r\n');
error = abs(((Voc_stc-V_stc_oc_1)/Voc_stc)*100);
fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
error = abs(((Isc_stc-I_stc_sc_1)/Isc_stc)*100);
fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
error = abs(((Pmp_stc-P_stc_mp_1)/Pmp_stc)*100);
fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
error = abs(((Vmp_stc-V_stc_mp_1)/Vmp_stc)*100);
fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
error = abs(((Imp_stc-I_stc_mp_1)/Imp_stc)*100);
fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
fprintf(fid, '\tModelo de 2 diodos:\r\n');
error = abs(((Voc_stc-V_stc_oc_2)/Voc_stc)*100);
fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
```

```
error = abs(((Isc_stc-I_stc_sc_2)/Isc_stc)*100);
fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
error = abs(((Pmp_stc-P_stc_mp_2)/Pmp_stc)*100);
fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
error = abs(((Vmp_stc-V_stc_mp_2)/Vmp_stc)*100);
fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
error = abs(((Imp_stc-I_stc_mp_2)/Imp_stc)*100);
fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
fprintf(fid, '\r\nResultados NOCT\r\n');
fprintf(fid, '\tModelo de 1 diodo:\r\n');
error = abs(((Voc_noct-V_noct_oc_1)/Voc_noct)*100);
fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
error = abs(((Isc_noct-I_noct_sc_1)/Isc_noct)*100);
fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
error = abs(((Pmp_noct-P_noct_mp_1)/Pmp_noct)*100);
fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
error = abs(((Vmp_noct-V_noct_mp_1)/Vmp_noct)*100);
fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
error = abs(((Imp_noct-I_noct_mp_1)/Imp_noct)*100);
fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
fprintf(fid, '\tModelo de 2 diodos:\r\n');
error = abs(((Voc_noct-V_noct_oc_2)/Voc_noct)*100);
fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
error = abs(((Isc_noct-I_noct_sc_2)/Isc_noct)*100);
fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
error = abs(((Pmp_noct-P_noct_mp_2)/Pmp_noct)*100);
fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
error = abs(((Vmp_noct-V_noct_mp_2)/Vmp_noct)*100);
fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
error = abs(((Imp_noct-I_noct_mp_2)/Imp_noct)*100);
fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
fprintf(fid, '\r\n\r\n----COMPARATIVAS (respecto modelo de 2 diodos)----\r\n');
fprintf(fid, '\r\nResultados comparativa en irradiancia\r\n');
for i = 1:length(G_gtest)
    fprintf(fid, '\tPara %d W/m^2:\r\n', G_gtest(i));
    error = abs(((V_gtest_oc_1(i)-V_gtest_oc_2(i))/V_gtest_oc_2(i))*100);
    fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
    error = abs(((I_gtest_sc_1(i)-I_gtest_sc_2(i))/I_gtest_sc_2(i))*100);
    fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
    error = abs(((P_gtest_mp_1(i)-P_gtest_mp_2(i))/P_gtest_mp_2(i))*100);
    fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
    error = abs(((V_gtest_mp_1(i)-V_gtest_mp_2(i))/V_gtest_mp_2(i))*100);
    fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
    error = abs(((I_gtest_mp_1(i)-I_gtest_mp_2(i))/I_gtest_mp_2(i))*100);
    fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
end
fprintf(fid, '\r\nResultados comparativa en temperatura:\r\n');
for i = 1:length(Tc_ttest)
    fprintf(fid, '\tPara %d ºC: \r\n', Tc_ttest(i)-273.15);
    error = abs(((V_ttest_oc_1(i)-V_ttest_oc_2(i))/V_ttest_oc_2(i))*100);
    fprintf(fid, '\t\tVoc: %f%% de error\r\n', error);
    error = abs(((I_ttest_sc_1(i)-I_ttest_sc_2(i))/I_ttest_sc_2(i))*100);
    fprintf(fid, '\t\tIsc: %f%% de error\r\n', error);
    error = abs(((P_ttest_mp_1(i)-P_ttest_mp_2(i))/P_ttest_mp_2(i))*100);
```

```
        fprintf(fid, '\t\tPmp: %f%% de error\r\n', error);
        error = abs(((V_ttest_mp_1(i)-V_ttest_mp_2(i))/V_ttest_mp_2(i))*100);
        fprintf(fid, '\t\tVmp: %f%% de error\r\n', error);
        error = abs(((I_ttest_mp_1(i)-I_ttest_mp_2(i))/I_ttest_mp_2(i))*100);
        fprintf(fid, '\t\tImp: %f%% de error\r\n', error);
    end

    fclose(fid);
```

- model_evaluation_results.txt

```
Resultados STC
        Modelo de 1 diodo:
                Voc: 0.033000% de error
                Isc: 0.050714% de error
                Pmp: 0.067076% de error
                Vmp: 0.221087% de error
                Imp: 0.276675% de error
        Modelo de 2 diodos:
                Voc: 0.094000% de error
                Isc: 0.048100% de error
                Pmp: 0.068261% de error
                Vmp: 0.208897% de error
                Imp: 0.265727% de error

Resultados NOCT
        Modelo de 1 diodo:
                Voc: 0.393674% de error
                Isc: 0.099149% de error
                Pmp: 1.490090% de error
                Vmp: 1.364653% de error
                Imp: 0.170227% de error
        Modelo de 2 diodos:
                Voc: 0.279786% de error
                Isc: 0.096536% de error
                Pmp: 1.377792% de error
                Vmp: 1.258751% de error
                Imp: 0.164037% de error


----COMPARATIVAS (respecto modelo de 2 diodos)----

Resultados comparativa en irradiancia
        Para 200 W/m^2:
                Voc: 0.080368% de error
                Isc: 0.002615% de error
                Pmp: 0.003804% de error
                Vmp: 0.106326% de error
                Imp: 0.102413% de error
        Para 400 W/m^2:
                Voc: 0.071740% de error
                Isc: 0.002615% de error
                Pmp: 0.036131% de error
                Vmp: 0.079561% de error
                Imp: 0.043396% de error
        Para 600 W/m^2:
                Voc: 0.066438% de error
                Isc: 0.002615% de error
                Pmp: 0.031023% de error
                Vmp: 0.054701% de error
                Imp: 0.023665% de error
        Para 800 W/m^2:
                Voc: 0.063635% de error
```

```
                Isc: 0.002615% de error
                Pmp: 0.017635% de error
                Vmp: 0.033996% de error
                Imp: 0.016355% de error
        Para 1000 W/m^2:
                Voc: 0.061057% de error
                Isc: 0.002615% de error
                Pmp: 0.001186% de error
                Vmp: 0.012165% de error
                Imp: 0.010977% de error

Resultados comparativa en temperatura:
        Para 5  C:
                Voc: 0.000000% de error
                Isc: 0.002615% de error
                Pmp: 0.055622% de error
                Vmp: 0.030431% de error
                Imp: 0.025199% de error
        Para 25  C:
                Voc: 0.061057% de error
                Isc: 0.002615% de error
                Pmp: 0.001186% de error
                Vmp: 0.012165% de error
                Imp: 0.010977% de error
        Para 45  C:
                Voc: 0.115300% de error
                Isc: 0.002615% de error
                Pmp: 0.101308% de error
                Vmp: 0.089765% de error
                Imp: 0.011532% de error
        Para 65  C:
                Voc: 0.204787% de error
                Isc: 0.002613% de error
                Pmp: 0.266856% de error
                Vmp: 0.216238% de error
                Imp: 0.050509% de error
```

A.2. Aplicación principal

- precisol.mlapp

```
classdef PreciSol < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        PreciSolApp             matlab.ui.Figure                    % PreciSol
        Graficas                matlab.ui.container.TabGroup        % I-V, P...
        IV                      matlab.ui.container.Tab             % I-V
        IVGraph                 matlab.ui.control.UIAxes            % Corrie...
        PV                      matlab.ui.container.Tab             % P-V
        PVGraph                 matlab.ui.control.UIAxes            % Potenc...
        T                       matlab.ui.container.Tab             % T
        TGraph                  matlab.ui.control.UIAxes            % Temper...
        G                       matlab.ui.container.Tab             % G
        GGraph                  matlab.ui.control.UIAxes            % Irradi...
        Pot                     matlab.ui.container.Tab             % Potenc...
        PotGraph                matlab.ui.control.UIAxes            % Potenc...
        Datos                   matlab.ui.container.Tab             % Datos
        LabelNumericEditField2  matlab.ui.control.Label             % Potenc...
        PotenciaAcumulada       matlab.ui.control.NumericEditField % [-Inf ...
        PotenciaAcumuladaUnidad matlab.ui.control.EditField
        RendSTCUnidad           matlab.ui.control.EditField
        Label4                  matlab.ui.control.Label             % Rendim...
        RendSTC                 matlab.ui.control.NumericEditField % [-Inf ...
        RendObtUnidad           matlab.ui.control.EditField
        Label5                  matlab.ui.control.Label             % Rend. ...
        RendObt                 matlab.ui.control.NumericEditField % [-Inf ...
        RendAvgUnidad           matlab.ui.control.EditField
        Label6                  matlab.ui.control.Label             % Rendim...
        RendAvg                 matlab.ui.control.NumericEditField % [-Inf ...
        PotSTCUnidad            matlab.ui.control.EditField
        Label22                 matlab.ui.control.Label             % Potenc...
        PotSTC                  matlab.ui.control.NumericEditField % [-Inf ...
        Label28                 matlab.ui.control.Label             % Voltaj...
        VoltMax                 matlab.ui.control.NumericEditField % [-Inf ...
        Label29                 matlab.ui.control.Label             % Corrie...
        IntMax                  matlab.ui.control.NumericEditField % [-Inf ...
        Label30                 matlab.ui.control.Label             % Voltaj...
        VoltAvg                 matlab.ui.control.NumericEditField % [-Inf ...
        Label31                 matlab.ui.control.Label             % Corrie...
        IntAvg                  matlab.ui.control.NumericEditField % [-Inf ...
        VoltMaxUnidad           matlab.ui.control.EditField
        VoltAvgUnidad           matlab.ui.control.EditField
        IntMaxUnidad            matlab.ui.control.EditField
        IntAvgUnidad            matlab.ui.control.EditField
        Utilidades              matlab.ui.container.Tab             % Utilid...
        Label20                 matlab.ui.control.Label             % File f...
        FileFuser               matlab.ui.control.Button            % Fusion...
        Label23                 matlab.ui.control.Label             % Calcul...
        CalcRes1                matlab.ui.control.Button            % 1 diodo
```

```
CalcRes2                matlab.ui.control.Button           % 2 diodos
TempModulo              matlab.ui.container.Tab            % Tc
TcGraph                 matlab.ui.control.UIAxes           % Temper...
TensionPmax             matlab.ui.container.Tab            % V
TensGraph               matlab.ui.control.UIAxes           % Tensi ...
CorrientePmax           matlab.ui.container.Tab            % I
CorrGraph               matlab.ui.control.UIAxes           % Corrie...
Simular                 matlab.ui.control.Button           % Simular
Label                   matlab.ui.control.Label            % Simula...
MeteoData               matlab.ui.container.Panel          % Datos ...
CargarData              matlab.ui.control.Button           % Cargar
EliminarData            matlab.ui.control.Button           % Eliminar
LabelListBox2           matlab.ui.control.Label            % Archivos
ListaData               matlab.ui.control.ListBox
LabelDropDown2          matlab.ui.control.Label            % Mes
ComienzoMes             matlab.ui.control.DropDown         % Enero,...
Label3                  matlab.ui.control.Label            % D a
ComienzoDia             matlab.ui.control.DropDown         % 1, 2, ...
FechaMode               matlab.ui.container.ButtonGroup    % Modo
ModoHora                matlab.ui.control.RadioButton      % Hora
ModoMinuto              matlab.ui.control.RadioButton      % Minuto
Label16                 matlab.ui.control.Label            % Mes
FinalMes                matlab.ui.control.DropDown         % Enero,...
Label17                 matlab.ui.control.Label            % D a
FinalDia                matlab.ui.control.DropDown         % 1, 2, ...
Label18                 matlab.ui.control.Label            % Comienzo
Label19                 matlab.ui.control.Label            % Final
Modulos                 matlab.ui.container.Panel          % M dulos
CargarModulos           matlab.ui.control.Button           % Cargar
EliminarModulos         matlab.ui.control.Button           % Eliminar
Label2                  matlab.ui.control.Label            % Tipos
ListaModulos            matlab.ui.control.ListBox
ModeloModulo            matlab.ui.control.EditField
LabelNumericEditField4  matlab.ui.control.Label            % Voc (V)
ModuloVoc               matlab.ui.control.NumericEditField % [-Inf ...
Label7                  matlab.ui.control.Label            % Isc (A)
ModuloIsc               matlab.ui.control.NumericEditField % [-Inf ...
Label8                  matlab.ui.control.Label            % Ki
ModuloKi                matlab.ui.control.NumericEditField % [-Inf ...
Label9                  matlab.ui.control.Label            % Kv
ModuloKv                matlab.ui.control.NumericEditField % [-Inf ...
Label10                 matlab.ui.control.Label            % Ns
ModuloNs                matlab.ui.control.NumericEditField % [-Inf ...
Label11                 matlab.ui.control.Label            % TONC
ModuloTONC              matlab.ui.control.NumericEditField % [-Inf ...
Label12                 matlab.ui.control.Label            % Vmp
ModuloVmp               matlab.ui.control.NumericEditField % [-Inf ...
Label13                 matlab.ui.control.Label            % Imp
ModuloImp               matlab.ui.control.NumericEditField % [-Inf ...
Label14                 matlab.ui.control.Label            % ? (%)
ModuloRend              matlab.ui.control.NumericEditField % [-Inf ...
Label15                 matlab.ui.control.Label            % Pmp
ModuloPmp               matlab.ui.control.NumericEditField % [-Inf ...
Label21                 matlab.ui.control.Label            % Superf...
```

```
    ModuloSup               matlab.ui.control.NumericEditField % [-Inf ...
    DatosSimulacion         matlab.ui.container.TabGroup       % Simula...
    DatosBasicos            matlab.ui.container.Tab            % Simula...
    LabelDiscreteKnob       matlab.ui.control.Label            % Precisi n
    Precision               matlab.ui.control.DiscreteKnob     % 10, 10...
    Modelo                  matlab.ui.container.ButtonGroup    % Modelo...
    Diodo1Modelo            matlab.ui.control.RadioButton      % Modelo...
    Diodo2Modelo            matlab.ui.control.RadioButton      % Modelo...
    DatosPerdidas           matlab.ui.container.Tab            % P rdidas
    LabelNumericEditField5  matlab.ui.control.Label            % P rdid...
    PerFab                  matlab.ui.control.NumericEditField % [0 10]
    Label24                 matlab.ui.control.Label            % P rdid...
    PerVid                  matlab.ui.control.NumericEditField % [0 25]
    Label25                 matlab.ui.control.Label            % P rdid...
    PerSuc                  matlab.ui.control.NumericEditField % [0 100]
    Label26                 matlab.ui.control.Label            % Estoc ...
    PerFabRand              matlab.ui.control.CheckBox
    PerVidRand              matlab.ui.control.CheckBox
    PerSucRand              matlab.ui.control.CheckBox
    Tab                     matlab.ui.container.Tab            % Nº de ...
    LabelNumericEditField   matlab.ui.control.Label            % M. en ...
    ModulosSerie            matlab.ui.control.NumericEditField % [1 Inf]
    Label27                 matlab.ui.control.Label            % M. en ...
    ModulosParalelo         matlab.ui.control.NumericEditField % [1 Inf]
    LabelNumericEditField6  matlab.ui.control.Label            % M. tot...
    NumeroModulos           matlab.ui.control.NumericEditField % [-Inf ...
    Label32                 matlab.ui.control.Label            % Multip...
    MultiplicadorModulos    matlab.ui.control.NumericEditField % [1 Inf]
end


properties (Access = private)
    DataFileNames = cell(0) % Lista con los nombres de los ficheros de datos
     ↪  meteorol gicos cargados
    DataFilePaths = cell(0) % Lista con las rutas de los ficheros de datos
     ↪  meteorol gicos cargados
    ModuleFileNames = cell(0) % Lista con los nombres de los ficheros de
     ↪  par metros de m dulo cargados
    ModuleFilePaths = cell(0) % Lista con las rutas de los ficheros de
     ↪  par metros de m dulo cargados
    G_Gh = 0 % Variable vector con los datos de irradiancia globobtenidos del
     ↪  fichero de dat. meteo.
    Ta = 0 % Variable vector con los datos de temp. ambiente obtenidos del
     ↪  fichero de dat. meteo.
    ModuleName = '' % Variable con el nombre del m dulo cargado
    Rend = 0 % Var. con el rendimiento del m dulo cargado
    Voc = 0 % Var. con la tens. de cir. abierto del m dulo cargado
    Isc = 0 % Var. con la corr. de cortocircuito del m dulo cargado
    Pmp = 0 % Var. con la potencia m x. del m dulo cargado
    Imp = 0 % Var. con la corriente de P.M. del m dulo cargado
    Vmp = 0 % Var. con la tensi n de P.M. del m dulo cargado
    Ns = 0 % Var. con el nº de c lulas del m dulo cargado
    Kv = 0 % Var. con el coef. de temp. sobre Voc del m dulo cargado
    Ki = 0 % Var. con el coef. de temp. sobre Isc del m dulo cargado
```

```
        Rs = 0 % Var. vector con la res. en serie del m dulo cargado para los dos
         ↪  modelos
        Rp = 0 % Var. vector con la res. en paralelo del m dulo cargado para los
         ↪  dos modelos
        TONC = 0 % Var. con la Temperatura de operaci n nominal del m dulo cargado
        Sup = 0 % Var. con la superficie del m dulo cargado
    end


    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            % Carga de los dos modelos de SIMULINK de 1 diodo y
            % dos diodos
            load_system('sim_1_diode'); % 1 diodo
            load_system('sim_2_diode'); % 2 diodos
        end

        % CargarData button pushed function
        function CargarDataButtonPushed(app)
            % Seleccionar fichero de datos
            [aux_name, aux_path] = uigetfile('.dat');
            % A adimos un hueco m s a la lista de nombres de ficheros
            % de datos
            aux_size = size(app.DataFileNames);
            aux_size = aux_size(2);
            app.DataFileNames{aux_size + 1} = [];
            app.DataFilePaths{aux_size + 1} = [];
            % Escribimos en los nuevos elementos
            aux_size = size(app.DataFileNames);
            aux_size = aux_size(2);
            app.DataFileNames{aux_size} = aux_name;
            app.DataFilePaths{aux_size} = aux_path;
            % Actualizado de la lista gr fica
            app.ListaData.Items = app.DataFileNames;
        end

        % EliminarData button pushed function
        function EliminarDataButtonPushed(app)
            % Detecci n de elemento a eliminar
            to_delete = app.ListaData.Value;
            % Determinamos tama o de la lista
            aux_size = size(app.DataFileNames);
            aux_size = aux_size(2);
            % Encontrar nombre de la lista a eliminar
            index_mat = strfind(app.DataFileNames, to_delete);
            for i = 1:length(index_mat)
                if (index_mat{i} == 1)
                    index_aux = i;
                    break;
                end
            end
            % Borrado del elemento de la lista
```

```
    app.DataFileNames = {app.DataFileNames{1:index_aux-1}, ...
        app.DataFileNames{index_aux+1:aux_size}};
    app.DataFilePaths = {app.DataFilePaths{1:index_aux-1}, ...
        app.DataFilePaths{index_aux+1:aux_size}};
    % Actualizado de la lista gr fica
    app.ListaData.Items = app.DataFileNames;
end

% CargarModulos button pushed function
function CargarModulosButtonPushed(app)
    % Seleccionar fichero de datos de m dulo
    [aux_name, aux_path] = uigetfile('.csv');
    % A adimos un hueco m s a la lista de nombres de ficheros
    % de datos de m dulo
    aux_size = size(app.ModuleFileNames);
    aux_size = aux_size(2);
    app.ModuleFileNames{aux_size + 1} = [];
    app.ModuleFilePaths{aux_size + 1} = [];
    % Escribimos en los nuevos elementos
    aux_size = size(app.ModuleFileNames);
    aux_size = aux_size(2);
    app.ModuleFileNames{aux_size} = aux_name;
    app.ModuleFilePaths{aux_size} = aux_path;

    app.ListaModulos.Items = app.ModuleFileNames;
end

% EliminarModulos button pushed function
function EliminarModulosButtonPushed(app)
    % Detecci n de elemento a eliminar
    to_delete = app.ListaModulos.Value;
    % Determinamos tama o de la lista
    aux_size = size(app.ModuleFileNames);
    aux_size = aux_size(2);
    % Encontrar nombre de la lista a eliminar
    index_mat = strfind(app.ModuleFileNames, to_delete);
    for i = 1:length(index_mat)
        if (index_mat{i} == 1)
            index_aux = i;
            break;
        end
    end
    % Borrado del elemento de la lista
    app.ModuleFileNames = {app.ModuleFileNames{1:index_aux-1}, ...
        app.ModuleFileNames{index_aux+1:aux_size}};
    app.ModuleFilePaths = {app.ModuleFilePaths{1:index_aux-1}, ...
        app.ModuleFilePaths{index_aux+1:aux_size}};
    % Actualizado de la lista gr fica
    app.ListaModulos.Items = app.ModuleFileNames;
end

% Simular button pushed function
function SimularButtonPushed(app)
    % ------------------------------------------------
```

```
% Funci n asociada al boton "Simular", ejecuta los
% modelos correspondientes para los datos
% meteorol gicos seleccionados, utilizando los
% par metros de m dulo seleccionado, as  como el
% resto de opci nes, como n mero de paneles,
% precisi n y modelo utilizado
% ------------------------------------------------

% Selecci n del modelo de simulaci n
if app.Diodo1Modelo.Value == 1 % 1 diodo
    module_mode = false; % Modo
elseif app.Diodo2Modelo.Value == 1 % 2 diodos
    module_mode = true;
end

% Creaci n de los  ndices de comienzo y final de simulaci n:
% Comienzo
n_month_start = get_month_num(app.ComienzoMes.Value);
n_day_start = str2double(app.ComienzoDia.Value);
date_index_start = get_date_index(n_month_start, n_day_start) - 1439;
% Final
n_month_end = get_month_num(app.FinalMes.Value);
n_day_end = str2double(app.FinalDia.Value);
date_index_end = get_date_index(n_month_end, n_day_end);

if date_index_start > date_index_end ||...
        length(app.G_Gh) == 0 ||...
        strcmp(app.ModuleName, '') % Detecci n de errores
    if date_index_start > date_index_end % Fechas incorrectas
        prompt='Fecha de comienzo mayor que la final';
    elseif length(app.G_Gh) == 0 % Datos meteo. no cargados
        prompt='Datos meteorol gicos no cargados';
    elseif strcmp(app.ModuleName, '') % Datos de mod. no cargados
        prompt='Datos de m dulo no cargados';
    end
    name = 'ERROR';
    warndlg(prompt, name);
else
    % Selecci n de truncado de datos
    if app.ModoHora.Value == 1 % Datos en hora
        t = date_index_start:60:date_index_end;
        pot_divider = 1;
    else % Datos en minutos
        t = date_index_start:date_index_end;
        pot_divider = 60;
    end

    % Comienzo de ejecuci n de la simulaci n
    progressbar(0)
    progressbar('Progreso de la simulaci n')
    % Inicializaci n de vectores
    P = 0;
    I_pmax = 0;
    V_pmax = 0;
```

```
data_T = 0;
data_G = 0;
data_Tc = 0;
i_aux = 1;
for i = t
    % Truncado de datos
    if app.ModoHora.Value == 1
        Ta_aux = sum(app.Ta(i:i+60))/60;
        G_Gh_aux = sum(app.G_Gh(i:i+60))/60;
    else
        Ta_aux = app.Ta(i);
        G_Gh_aux = app.G_Gh(i);
    end
    % Aplicaci n de p rdidas por suciedad
    G_Gh_aux = G_Gh_aux*(1 - app.PerSuc.Value/100);
    % Temperatura de c lula
    Tc = 273.15 + Ta_aux + app.TONC*G_Gh_aux;
    % Ejecuci n de la simulaci n de un punto
    [I, V] = fast_diode(app.Voc, app.Isc, app.Ns, ...
        app.Kv, app.Ki, app.Rs, app.Rp, G_Gh_aux, ...
        Tc, str2double(app.Precision.Value), module_mode);
    % C lculo de la potencia m xima y tension y corriente en Pmax
    [I_aux, V_aux] = max_p_point(I, V);
    V_aux = V_aux*app.ModulosSerie.Value;
    I_aux = I_aux*app.ModulosParalelo.Value;
    I_pmax(i_aux) = I_aux;
    V_pmax(i_aux) = V_aux;
    P(i_aux) = max(I.*V) * app.NumeroModulos.Value;
    % Guardado de datos meteorol gicos truncados
    data_T(i_aux) = Ta_aux;
    data_G(i_aux) = G_Gh_aux;
    data_Tc(i_aux) = Tc - 273.15;
    % Avance de barra de progreso e  ndice auxiliar
    progressbar((1/length(t))*i_aux);
    i_aux = i_aux + 1;
end
progressbar(1) % Cierre de barra de progreso

% Aplicaci n de p rdidas a la potencia
for i = 1:length(P)
    P(i) = P(i)*(1 - app.PerFab.Value/100);
    P(i) = P(i)*(1 - (app.PerVid.Value*0.8)/100);
end

% REPRESENTACI N DE DATOS GR FICOS
t = 1:i_aux-1;
plot(app.PotGraph, t, P); % Potencia Generada
plot(app.TGraph, t, data_T); % Temp. ambiente
plot(app.TcGraph, t, data_Tc); % Temp. de m dulo
plot(app.GGraph, t, data_G); % Irradiancia
plot(app.TensGraph, t, V_pmax); % Tensi n Pmax
plot(app.CorrGraph, t, I_pmax); % Corriente Pmax

% C LCULO Y REPRESENTACI N DE DATOS NUM RICOS
```

```
% Pot. acumulada
pot_acumulada = sum(P)/pot_divider;
i_aux = 0;
while pot_acumulada > 1000 && i_aux < 5
    pot_acumulada = pot_acumulada/1000;
    i_aux = i_aux + 1;
end
app.PotenciaAcumulada.Value = pot_acumulada;
switch i_aux
    case 0;
        app.PotenciaAcumuladaUnidad.Value = 'W';
    case 1;
        app.PotenciaAcumuladaUnidad.Value = 'kW';
    case 2;
        app.PotenciaAcumuladaUnidad.Value = 'gW';
    case 3;
        app.PotenciaAcumuladaUnidad.Value = 'tW';
    case 4;
        app.PotenciaAcumuladaUnidad.Value = 'pW';
    case 5;
        app.PotenciaAcumuladaUnidad.Value = 'eW';
end
% Rendimiento medio
rend = sum(P)/(sum(data_G)*app.Sup)*100;
app.RendAvg.Value = rend/app.NumeroModulos.Value;
app.RendAvgUnidad.Value = '%';
% Rendimiento máximo obtenido
rend = P./(data_G*app.Sup);
app.RendObt.Value = max(rend)*100/app.NumeroModulos.Value;
app.RendObtUnidad.Value = '%';
% Rendimiento STC dado por el fabricante
[I, V] = fast_diode(app.Voc, app.Isc, app.Ns, app.Kv, app.Ki, ...
    app.Rs, app.Rp, 1000, 298, 1000, module_mode);
P_stc = I.*V;
app.PotSTC.Value = max(P_stc);
app.PotSTCUnidad.Value = 'W';
rend = max(P_stc)/(1000*app.Sup)*100;
app.RendSTC.Value = rend;
app.RendSTCUnidad.Value = '%';
% Voltaje máximo
app.VoltMax.Value = max(V_pmax);
app.VoltMaxUnidad.Value = 'V';
% Intensidad máxima
app.IntMax.Value = max(I_pmax);
app.IntMaxUnidad.Value = 'A';
% Voltaje e intensidad medias
V_total = 0;
I_total = 0;
count = 0;
for i = 1:length(V_pmax)
    if V_pmax(i) > 0
       V_total = V_total + V_pmax(i);
       I_total = I_total + I_pmax(i);
       count = count + 1;
```

```
            end
        end
        app.VoltAvg.Value = V_total/count;
        app.VoltAvgUnidad.Value = 'V';
        app.IntAvg.Value = I_total/count;
        app.IntAvgUnidad.Value = 'A';
        % Representaci n de curvas caracter sticas
        % simuladas del m dulo
        plot(app.IVGraph, V, I);
        plot(app.PVGraph, V, P_stc);

        % Opci n de guardar datos en fichero
        quest = '¿Deseas guardar los resultados en un fichero?';
        name = 'Guardar datos';
        respuesta = questdlg(quest, name, 'Si', 'No', 'No');
        if strcmp(respuesta, 'Si')
            % Guardado en fichero
            save_data(data_G, data_T, data_Tc, P, V_pmax, ...
                I_pmax, n_month_start, n_day_start, ...
                n_month_end, n_day_end, ...
                app.ModulosSerie.Value, ...
                app.ModulosParalelo.Value, ...
                app.NumeroModulos.Value);
            prompt = 'Datos guardados!';
            name = 'Atencion';
        else
            prompt = 'Datos no guardados!';
            name = 'Atencion';
        end
        % Mensaje con el resultado de la operaci n
        warndlg(prompt, name);
    end
end

% ListaData value changed function
function ListaDataValueChanged(app)
    % Detecci n del fichero seleccionado
    data_to_read = app.ListaData.Value;
    % Busca  ndice en la lista de ficheros cargados
    index_mat = strfind(app.DataFileNames, data_to_read);
    for i = 1:length(index_mat)
        if (index_mat{i} == 1)
            index_aux = i;
            break;
        end
    end
    % Generaci n ruta del fichero
    data_filename = strcat(app.DataFilePaths{index_aux}, ...
        app.DataFileNames{index_aux});
    % Carga del fichero en memoria
    [app.G_Gh, app.Ta] = meteonorm_file(data_filename);
end

% ListaModulos value changed function
```

```
function ListaModulosValueChanged(app)
    % Detecci n del m dulo a usar
    module_to_read = app.ListaModulos.Value;
    % Busca  ndice en la lista de ficheros cargados
    index_mat = strfind(app.ModuleFileNames, module_to_read);
    for i = 1:length(index_mat)
        if (index_mat{i} == 1)
            index_aux = i;
            break;
        end
    end
    % Generaci n ruta del fichero
    module_filename = strcat(app.ModuleFilePaths{index_aux}, ...
        app.ModuleFileNames{index_aux});
    % Carga del fichero en memoria
    [app.ModuleName, app.Rend, app.Voc, app.Isc, app.Pmp, app.Imp, ...
        app.Vmp, app.Ns, app.Kv, app.Ki, app.Rs, app.Rp, ...
        app.TONC, app.Sup] = read_modulefile(module_filename);
    disp(app.ModuleName);
    % Visualizaci n de los datos en memoria
    app.ModeloModulo.Value = app.ModuleName;
    app.ModuloRend.Value = app.Rend;
    app.ModuloVoc.Value = app.Voc;
    app.ModuloIsc.Value = app.Isc;
    app.ModuloPmp.Value = app.Pmp;
    app.ModuloImp.Value = app.Imp;
    app.ModuloVmp.Value = app.Vmp;
    app.ModuloNs.Value = app.Ns;
    app.ModuloKv.Value = app.Kv;
    app.ModuloKi.Value = app.Ki;
    app.ModuloSup.Value = app.Sup;
end

% ComienzoMes value changed function
function ComienzoMesValueChanged(app, event)
    % Obtenci n del n mero de mes
    n_month = get_month_num(app.ComienzoMes.Value);
    % Obtenci n del n º de d as del mes
    n_days = get_days_month(n_month);
    % Creaci n de la lista de d as
    day_list = cell(0);
    for i = 1:n_days
        day_list{i} = num2str(i);
    end
    % Guardado de la lista de d as, selecci n del primero
    app.ComienzoDia.Items = day_list;
    app.ComienzoDia.Value = app.ComienzoDia.Items{1};
end

% FinalMes value changed function
function FinalMesValueChanged(app, event)
    % Obtenci n del n mero de mes
    n_month = get_month_num(app.FinalMes.Value);
    % Obtenci n del n º de d as del mes
```

```
    n_days = get_days_month(n_month);
    % Creaci n de la lista de d as
    day_list = cell(0);
    for i = 1:n_days
        day_list{i} = num2str(i);
    end
    % Guardado de la lista de d as, selecci n del primero
    app.FinalDia.Items = day_list;
    app.FinalDia.Value = app.FinalDia.Items{1};
end

% FileFuser button pushed function
function FileFuserButtonPushed(app)
    % Ejecuci n de la utilidad "file_fuser"
    file_fuser();
end

% CalcRes1 button pushed function
function CalcRes1ButtonPushed(app)
    % Generaci n de ventana de petici n de datos
    prompt = {'Voc:', 'Isc:', 'Vmp::', 'Imp:', 'Ns', ...
        'Kv:', 'Ki:'};
    dlg_title = 'Valores m dulo';
    num_lines = 1;
    defaultans = {'46.2', '9.48', '37.9', '8.97', '72', ...
        '-143', '5.02'};
    values = inputdlg(prompt,dlg_title,num_lines,defaultans);
    % Guardado de datos en memoria
    Voc_aux = str2double(values{1});
    Isc_aux = str2double(values{2});
    Vmp_aux = str2double(values{3});
    Imp_aux = str2double(values{4});
    Ns_aux = str2double(values{5});
    Kv_aux = str2double(values{6});
    Ki_aux = str2double(values{7});
    % Ejecuci n del script de c lculo de resistencias
    fast_1_diode_resget(Voc_aux, Isc_aux, Vmp_aux, Imp_aux, ...
        Ns_aux, Kv_aux, Ki_aux);
end

% CalcRes2 button pushed function
function CalcRes2ButtonPushed(app)
     % Generaci n de ventana de petici n de datos
    prompt = {'Voc:', 'Isc:', 'Vmp::', 'Imp:', 'Ns', 'Kv:', ...
        'Ki:'};
    dlg_title = 'Valores m dulo';
    num_lines = 1;
    defaultans = {'46.2', '9.48', '37.9', '8.97', '72', ...
        '-143', '5.02'};
    values = inputdlg(prompt,dlg_title,num_lines,defaultans);
    % Guardado de datos en memoria
    Voc_aux = str2double(values{1});
    Isc_aux = str2double(values{2});
    Vmp_aux = str2double(values{3});
```

```
    Imp_aux = str2double(values{4});
    Ns_aux = str2double(values{5});
    Kv_aux = str2double(values{6});
    Ki_aux = str2double(values{7});
    % Ejecuci n del script de c lculo de resistencias
    fast_2_diode_resget(Voc_aux, Isc_aux, Vmp_aux, Imp_aux, ...
        Ns_aux, Kv_aux, Ki_aux);
end

% PerFabRand value changed function
function PerFabRandValueChanged(app)
    % Generaci n de valor aletorio para
    % p rdidas de potencia de f brica
    value = app.PerFabRand.Value;
    if value == true
        app.PerFab.Editable = 'off';
        app.PerFab.Value = round((10*rand(1, 1, ...
            'double'))*100)/100;
    else
        app.PerFab.Editable = 'on';
    end
end

% PerVidRand value changed function
function PerVidRandValueChanged(app)
    % Generaci n de valor aletorio para
    % p rdidas de potencia de ciclo de vida
    value = app.PerVidRand.Value;
    if value == true
        app.PerVid.Editable = 'off';
        app.PerVid.Value = randi(25);
    else
        app.PerVid.Editable = 'on';
    end
end

% PerSucRand value changed function
function PerSucRandValueChanged(app)
    % Generaci n de valor aletorio para
    % p rdidas de irradiancia por suciedad
    value = app.PerSucRand.Value;
    if value == true
        app.PerSuc.Editable = 'off';
        app.PerSuc.Value = round((100*rand(1, 1, ...
            'double'))*100)/100;
    else
        app.PerSuc.Editable = 'on';
    end
end

% ModulosSerie value changed function
function ModulosSerieValueChanged(app)
    app.NumeroModulos.Value = app.ModulosSerie.Value *...
        app.ModulosParalelo.Value *...
```

```
            app.MultiplicadorModulos.Value;
        end

        % ModulosParalelo value changed function
        function ModulosParaleloValueChanged(app)
            app.NumeroModulos.Value = app.ModulosSerie.Value *...
                app.ModulosParalelo.Value *...
                app.MultiplicadorModulos.Value;
        end

        % MultiplicadorModulos value changed function
        function MultiplicadorModulosValueChanged(app)
            app.NumeroModulos.Value = app.ModulosSerie.Value *...
                app.ModulosParalelo.Value *...
                app.MultiplicadorModulos.Value;
        end
    end

    % App initialization and construction
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create PreciSolApp
            app.PreciSolApp = uifigure;
            app.PreciSolApp.Position = [100 100 1280 720];
            app.PreciSolApp.Name = 'PreciSol';
            setAutoResize(app, app.PreciSolApp, true)

            % Create Graficas
            app.Graficas = uitabgroup(app.PreciSolApp);
            app.Graficas.Units = 'pixels';
            app.Graficas.Position = [432 12 835 680];

            % Create IV
            app.IV = uitab(app.Graficas);
            app.IV.Units = 'pixels';
            app.IV.Title = 'I-V';

            % Create IVGraph
            app.IVGraph = uiaxes(app.IV);
            title(app.IVGraph, 'Corriente-Tensi n (V-A)');
            xlabel(app.IVGraph, 'V');
            ylabel(app.IVGraph, 'I');
            app.IVGraph.Position = [1 2 833 656];

            % Create PV
            app.PV = uitab(app.Graficas);
            app.PV.Units = 'pixels';
            app.PV.Title = 'P-V';

            % Create PVGraph
            app.PVGraph = uiaxes(app.PV);
```

```
title(app.PVGraph, 'Potencia (W)');
xlabel(app.PVGraph, 'P');
ylabel(app.PVGraph, 'I');
app.PVGraph.Position = [1 2 833 656];

% Create T
app.T = uitab(app.Graficas);
app.T.Units = 'pixels';
app.T.Title = 'T';

% Create TGraph
app.TGraph = uiaxes(app.T);
title(app.TGraph, 'Temperatura (ºC)');
xlabel(app.TGraph, 't');
ylabel(app.TGraph, 'T');
app.TGraph.Position = [1 2 833 656];

% Create G
app.G = uitab(app.Graficas);
app.G.Units = 'pixels';
app.G.Title = 'G';

% Create GGraph
app.GGraph = uiaxes(app.G);
title(app.GGraph, 'Irradiancia (W/m2)');
xlabel(app.GGraph, 't');
ylabel(app.GGraph, 'G');
app.GGraph.Position = [1 2 833 656];

% Create Pot
app.Pot = uitab(app.Graficas);
app.Pot.Units = 'pixels';
app.Pot.Title = 'Potencia Generada';

% Create PotGraph
app.PotGraph = uiaxes(app.Pot);
title(app.PotGraph, 'Potencia generada (W/t)');
xlabel(app.PotGraph, 't');
ylabel(app.PotGraph, 'P');
app.PotGraph.Position = [1 1 833 657];

% Create Datos
app.Datos = uitab(app.Graficas);
app.Datos.Units = 'pixels';
app.Datos.Title = 'Datos';

% Create LabelNumericEditField2
app.LabelNumericEditField2 = uilabel(app.Datos);
app.LabelNumericEditField2.HorizontalAlignment = 'right';
app.LabelNumericEditField2.FontSize = 36;
app.LabelNumericEditField2.Position = [10 594 327 47];
app.LabelNumericEditField2.Text = 'Potencia acumulada';

% Create PotenciaAcumulada
```

```
app.PotenciaAcumulada = uieditfield(app.Datos, 'numeric');
app.PotenciaAcumulada.ValueDisplayFormat = '%11.6g';
app.PotenciaAcumulada.Editable = 'off';
app.PotenciaAcumulada.FontSize = 48;
app.PotenciaAcumulada.Position = [344 587 190 57];

% Create PotenciaAcumuladaUnidad
app.PotenciaAcumuladaUnidad = uieditfield(app.Datos, 'text');
app.PotenciaAcumuladaUnidad.Editable = 'off';
app.PotenciaAcumuladaUnidad.FontSize = 48;
app.PotenciaAcumuladaUnidad.Position = [550 588 119 55];

% Create RendSTCUnidad
app.RendSTCUnidad = uieditfield(app.Datos, 'text');
app.RendSTCUnidad.Editable = 'off';
app.RendSTCUnidad.FontSize = 48;
app.RendSTCUnidad.Position = [550 517 119 55];

% Create Label4
app.Label4 = uilabel(app.Datos);
app.Label4.HorizontalAlignment = 'right';
app.Label4.FontSize = 36;
app.Label4.Position = [52 523 285 47];
app.Label4.Text = 'Rendimiento STC';

% Create RendSTC
app.RendSTC = uieditfield(app.Datos, 'numeric');
app.RendSTC.ValueDisplayFormat = '%11.6g';
app.RendSTC.Editable = 'off';
app.RendSTC.FontSize = 48;
app.RendSTC.Position = [344 516 190 57];

% Create RendObtUnidad
app.RendObtUnidad = uieditfield(app.Datos, 'text');
app.RendObtUnidad.Editable = 'off';
app.RendObtUnidad.FontSize = 48;
app.RendObtUnidad.Position = [550 445 119 55];

% Create Label5
app.Label5 = uilabel(app.Datos);
app.Label5.HorizontalAlignment = 'right';
app.Label5.FontSize = 36;
app.Label5.Position = [4 451 333 47];
app.Label5.Text = 'Rend. max. obtenido';

% Create RendObt
app.RendObt = uieditfield(app.Datos, 'numeric');
app.RendObt.ValueDisplayFormat = '%11.6g';
app.RendObt.Editable = 'off';
app.RendObt.FontSize = 48;
app.RendObt.Position = [344 444 190 57];

% Create RendAvgUnidad
app.RendAvgUnidad = uieditfield(app.Datos, 'text');
```

```
app.RendAvgUnidad.Editable = 'off';
app.RendAvgUnidad.FontSize = 48;
app.RendAvgUnidad.Position = [550 375 119 55];

% Create Label6
app.Label6 = uilabel(app.Datos);
app.Label6.HorizontalAlignment = 'right';
app.Label6.FontSize = 36;
app.Label6.Position = [26 381 311 47];
app.Label6.Text = 'Rendimiento medio';

% Create RendAvg
app.RendAvg = uieditfield(app.Datos, 'numeric');
app.RendAvg.ValueDisplayFormat = '%11.6g';
app.RendAvg.Editable = 'off';
app.RendAvg.FontSize = 48;
app.RendAvg.Position = [344 374 190 57];

% Create PotSTCUnidad
app.PotSTCUnidad = uieditfield(app.Datos, 'text');
app.PotSTCUnidad.Editable = 'off';
app.PotSTCUnidad.FontSize = 48;
app.PotSTCUnidad.Position = [551 304 119 55];

% Create Label22
app.Label22 = uilabel(app.Datos);
app.Label22.HorizontalAlignment = 'right';
app.Label22.FontSize = 36;
app.Label22.Position = [115 310 223 47];
app.Label22.Text = 'Potencia STC';

% Create PotSTC
app.PotSTC = uieditfield(app.Datos, 'numeric');
app.PotSTC.ValueDisplayFormat = '%11.6g';
app.PotSTC.Editable = 'off';
app.PotSTC.FontSize = 48;
app.PotSTC.Position = [345 303 190 57];

% Create Label28
app.Label28 = uilabel(app.Datos);
app.Label28.HorizontalAlignment = 'right';
app.Label28.FontSize = 36;
app.Label28.Position = [91 238 247 47];
app.Label28.Text = 'Voltaje m ximo';

% Create VoltMax
app.VoltMax = uieditfield(app.Datos, 'numeric');
app.VoltMax.ValueDisplayFormat = '%11.6g';
app.VoltMax.Editable = 'off';
app.VoltMax.FontSize = 48;
app.VoltMax.Position = [345 231 190 57];

% Create Label29
app.Label29 = uilabel(app.Datos);
```

```
app.Label29.HorizontalAlignment = 'right';
app.Label29.FontSize = 36;
app.Label29.Position = [53 96 285 47];
app.Label29.Text = 'Corriente m xima';

% Create IntMax
app.IntMax = uieditfield(app.Datos, 'numeric');
app.IntMax.ValueDisplayFormat = '%11.6g';
app.IntMax.Editable = 'off';
app.IntMax.FontSize = 48;
app.IntMax.Position = [345 89 190 57];

% Create Label30
app.Label30 = uilabel(app.Datos);
app.Label30.HorizontalAlignment = 'right';
app.Label30.FontSize = 36;
app.Label30.Position = [119 166 219 47];
app.Label30.Text = 'Voltaje medio';

% Create VoltAvg
app.VoltAvg = uieditfield(app.Datos, 'numeric');
app.VoltAvg.ValueDisplayFormat = '%11.6g';
app.VoltAvg.Editable = 'off';
app.VoltAvg.FontSize = 48;
app.VoltAvg.Position = [345 159 190 57];

% Create Label31
app.Label31 = uilabel(app.Datos);
app.Label31.HorizontalAlignment = 'right';
app.Label31.FontSize = 36;
app.Label31.Position = [81 26 257 47];
app.Label31.Text = 'Corriente media';

% Create IntAvg
app.IntAvg = uieditfield(app.Datos, 'numeric');
app.IntAvg.ValueDisplayFormat = '%11.6g';
app.IntAvg.Editable = 'off';
app.IntAvg.FontSize = 48;
app.IntAvg.Position = [345 19 190 57];

% Create VoltMaxUnidad
app.VoltMaxUnidad = uieditfield(app.Datos, 'text');
app.VoltMaxUnidad.Editable = 'off';
app.VoltMaxUnidad.FontSize = 48;
app.VoltMaxUnidad.Position = [551 233 119 55];

% Create VoltAvgUnidad
app.VoltAvgUnidad = uieditfield(app.Datos, 'text');
app.VoltAvgUnidad.Editable = 'off';
app.VoltAvgUnidad.FontSize = 48;
app.VoltAvgUnidad.Position = [551 159 119 55];

% Create IntMaxUnidad
app.IntMaxUnidad = uieditfield(app.Datos, 'text');
```

```
app.IntMaxUnidad.Editable = 'off';
app.IntMaxUnidad.FontSize = 48;
app.IntMaxUnidad.Position = [550 90 119 55];

% Create IntAvgUnidad
app.IntAvgUnidad = uieditfield(app.Datos, 'text');
app.IntAvgUnidad.Editable = 'off';
app.IntAvgUnidad.FontSize = 48;
app.IntAvgUnidad.Position = [551 19 119 55];

% Create Utilidades
app.Utilidades = uitab(app.Graficas);
app.Utilidades.Units = 'pixels';
app.Utilidades.Title = 'Utilidades';

% Create Label20
app.Label20 = uilabel(app.Utilidades);
app.Label20.Position = [27 621 50 15];
app.Label20.Text = 'File fuser';

% Create FileFuser
app.FileFuser = uibutton(app.Utilidades, 'push');
app.FileFuser.ButtonPushedFcn = createCallbackFcn(app,
↪   @FileFuserButtonPushed);
app.FileFuser.Position = [27 587 100 22];
app.FileFuser.Text = 'Fusionar ficheros';

% Create Label23
app.Label23 = uilabel(app.Utilidades);
app.Label23.Position = [27 558 174 15];
app.Label23.Text = 'Calculador resistencias Rs y Rp';

% Create CalcRes1
app.CalcRes1 = uibutton(app.Utilidades, 'push');
app.CalcRes1.ButtonPushedFcn = createCallbackFcn(app,
↪   @CalcRes1ButtonPushed);
app.CalcRes1.Position = [27 524 100 22];
app.CalcRes1.Text = '1 diodo';

% Create CalcRes2
app.CalcRes2 = uibutton(app.Utilidades, 'push');
app.CalcRes2.ButtonPushedFcn = createCallbackFcn(app,
↪   @CalcRes2ButtonPushed);
app.CalcRes2.Position = [141 524 100 22];
app.CalcRes2.Text = '2 diodos';

% Create TempModulo
app.TempModulo = uitab(app.Graficas);
app.TempModulo.Units = 'pixels';
app.TempModulo.Title = 'Tc';

% Create TcGraph
app.TcGraph = uiaxes(app.TempModulo);
title(app.TcGraph, 'Temperatura (ºC)');
```

```
xlabel(app.TcGraph, 't');
ylabel(app.TcGraph, 'T');
app.TcGraph.Position = [1 2 833 656];

% Create TensionPmax
app.TensionPmax = uitab(app.Graficas);
app.TensionPmax.Units = 'pixels';
app.TensionPmax.Title = 'V';

% Create TensGraph
app.TensGraph = uiaxes(app.TensionPmax);
title(app.TensGraph, 'Tensi n (V)');
xlabel(app.TensGraph, 't');
ylabel(app.TensGraph, 'V');
app.TensGraph.Position = [1 2 833 656];

% Create CorrientePmax
app.CorrientePmax = uitab(app.Graficas);
app.CorrientePmax.Units = 'pixels';
app.CorrientePmax.Title = 'I';

% Create CorrGraph
app.CorrGraph = uiaxes(app.CorrientePmax);
title(app.CorrGraph, 'Corriente (A)');
xlabel(app.CorrGraph, 't');
ylabel(app.CorrGraph, 'A');
app.CorrGraph.Position = [1 2 833 656];

% Create Simular
app.Simular = uibutton(app.PreciSolApp, 'push');
app.Simular.ButtonPushedFcn = createCallbackFcn(app,
↪   @SimularButtonPushed);
app.Simular.FontSize = 20;
app.Simular.Position = [108 9 212 41];
app.Simular.Text = 'Simular';

% Create Label
app.Label = uilabel(app.PreciSolApp);
app.Label.FontSize = 36;
app.Label.Position = [14 646 405 58];
app.Label.Text = 'Simulador de c lula solar';

% Create MeteoData
app.MeteoData = uipanel(app.PreciSolApp);
app.MeteoData.BorderType = 'line';
app.MeteoData.Title = 'Datos meteorol gicos';
app.MeteoData.FontName = 'Helvetica';
app.MeteoData.FontUnits = 'pixels';
app.MeteoData.FontSize = 12;
app.MeteoData.Units = 'pixels';
app.MeteoData.Position = [14 205 405 191];

% Create CargarData
app.CargarData = uibutton(app.MeteoData, 'push');
```

```
app.CargarData.ButtonPushedFcn = createCallbackFcn(app,
 ↪   @CargarDataButtonPushed);
app.CargarData.Position = [294 134 100 22];
app.CargarData.Text = 'Cargar';

% Create EliminarData
app.EliminarData = uibutton(app.MeteoData, 'push');
app.EliminarData.ButtonPushedFcn = createCallbackFcn(app,
 ↪   @EliminarDataButtonPushed);
app.EliminarData.Position = [294 95 100 22];
app.EliminarData.Text = 'Eliminar';

% Create LabelListBox2
app.LabelListBox2 = uilabel(app.MeteoData);
app.LabelListBox2.HorizontalAlignment = 'right';
app.LabelListBox2.Position = [8 141 47 15];
app.LabelListBox2.Text = 'Archivos';

% Create ListaData
app.ListaData = uilistbox(app.MeteoData);
app.ListaData.Items = {};
app.ListaData.ValueChangedFcn = createCallbackFcn(app,
 ↪   @ListaDataValueChanged);
app.ListaData.Position = [71 84 191 74];
app.ListaData.Value = {};

% Create LabelDropDown2
app.LabelDropDown2 = uilabel(app.MeteoData);
app.LabelDropDown2.HorizontalAlignment = 'right';
app.LabelDropDown2.Position = [73 58 23 15];
app.LabelDropDown2.Text = 'Mes';

% Create ComienzoMes
app.ComienzoMes = uidropdown(app.MeteoData);
app.ComienzoMes.Items = {'Enero', 'Febrero', 'Marzo', 'Abril',
 ↪   'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre',
 ↪   'Noviembre', 'Diciembre'};
app.ComienzoMes.ValueChangedFcn = createCallbackFcn(app,
 ↪   @ComienzoMesValueChanged, true);
app.ComienzoMes.Position = [111 54 86 22];
app.ComienzoMes.Value = 'Enero';

% Create Label3
app.Label3 = uilabel(app.MeteoData);
app.Label3.HorizontalAlignment = 'right';
app.Label3.Position = [206 58 20 15];
app.Label3.Text = 'D a';

% Create ComienzoDia
app.ComienzoDia = uidropdown(app.MeteoData);
app.ComienzoDia.Items = {'1', '2', '3', '4', '5', '6', '7', '8', '9',
 ↪   '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
 ↪   '20', '21', '22', '23', '24', '25', '26', '27', '28', '29',
 ↪   '30', '31'};
```

```
app.ComienzoDia.Position = [241 54 44 22];
app.ComienzoDia.Value = '1';

% Create FechaMode
app.FechaMode = uibuttongroup(app.MeteoData);
app.FechaMode.BorderType = 'line';
app.FechaMode.Title = 'Modo';
app.FechaMode.FontName = 'Helvetica';
app.FechaMode.FontUnits = 'pixels';
app.FechaMode.FontSize = 12;
app.FechaMode.Units = 'pixels';
app.FechaMode.Position = [297 12 93 71];

% Create ModoHora
app.ModoHora = uiradiobutton(app.FechaMode);
app.ModoHora.Text = 'Hora';
app.ModoHora.Position = [9 26 46 16];
app.ModoHora.Value = true;

% Create ModoMinuto
app.ModoMinuto = uiradiobutton(app.FechaMode);
app.ModoMinuto.Text = 'Minuto';
app.ModoMinuto.Position = [9 3 56 16];

% Create Label16
app.Label16 = uilabel(app.MeteoData);
app.Label16.HorizontalAlignment = 'right';
app.Label16.Position = [73 22 23 15];
app.Label16.Text = 'Mes';

% Create FinalMes
app.FinalMes = uidropdown(app.MeteoData);
app.FinalMes.Items = {'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo',
↪   'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre',
↪   'Noviembre', 'Diciembre'};
app.FinalMes.ValueChangedFcn = createCallbackFcn(app,
↪   @FinalMesValueChanged, true);
app.FinalMes.Position = [111 18 86 22];
app.FinalMes.Value = 'Enero';

% Create Label17
app.Label17 = uilabel(app.MeteoData);
app.Label17.HorizontalAlignment = 'right';
app.Label17.Position = [206 22 20 15];
app.Label17.Text = 'D a';

% Create FinalDia
app.FinalDia = uidropdown(app.MeteoData);
app.FinalDia.Items = {'1', '2', '3', '4', '5', '6', '7', '8', '9',
↪   '10', '11', '12', '13', '14', '15', '16', '17', '18', '19',
↪   '20', '21', '22', '23', '24', '25', '26', '27', '28', '29',
↪   '30', '31'};
app.FinalDia.Position = [241 18 44 22];
app.FinalDia.Value = '1';
```

```
% Create Label18
app.Label18 = uilabel(app.MeteoData);
app.Label18.Position = [6 58 56 15];
app.Label18.Text = 'Comienzo';

% Create Label19
app.Label19 = uilabel(app.MeteoData);
app.Label19.Position = [6 22 27 15];
app.Label19.Text = 'Final';

% Create Modulos
app.Modulos = uipanel(app.PreciSolApp);
app.Modulos.BorderType = 'line';
app.Modulos.Title = 'M dulos';
app.Modulos.FontName = 'Helvetica';
app.Modulos.FontUnits = 'pixels';
app.Modulos.FontSize = 12;
app.Modulos.Units = 'pixels';
app.Modulos.Position = [13 403 405 229];

% Create CargarModulos
app.CargarModulos = uibutton(app.Modulos, 'push');
app.CargarModulos.ButtonPushedFcn = createCallbackFcn(app,
↪   @CargarModulosButtonPushed);
app.CargarModulos.Position = [294 172 100 22];
app.CargarModulos.Text = 'Cargar';

% Create EliminarModulos
app.EliminarModulos = uibutton(app.Modulos, 'push');
app.EliminarModulos.ButtonPushedFcn = createCallbackFcn(app,
↪   @EliminarModulosButtonPushed);
app.EliminarModulos.Position = [294 133 100 22];
app.EliminarModulos.Text = 'Eliminar';

% Create Label2
app.Label2 = uilabel(app.Modulos);
app.Label2.HorizontalAlignment = 'right';
app.Label2.Position = [25 179 30 15];
app.Label2.Text = 'Tipos';

% Create ListaModulos
app.ListaModulos = uilistbox(app.Modulos);
app.ListaModulos.Items = {};
app.ListaModulos.ValueChangedFcn = createCallbackFcn(app,
↪   @ListaModulosValueChanged);
app.ListaModulos.Position = [71 122 191 74];
app.ListaModulos.Value = {};

% Create ModeloModulo
app.ModeloModulo = uieditfield(app.Modulos, 'text');
app.ModeloModulo.Editable = 'off';
app.ModeloModulo.Position = [6 91 379 22];
```

```
% Create LabelNumericEditField4
app.LabelNumericEditField4 = uilabel(app.Modulos);
app.LabelNumericEditField4.HorizontalAlignment = 'right';
app.LabelNumericEditField4.Position = [15 68 40 15];
app.LabelNumericEditField4.Text = 'Voc (V)';

% Create ModuloVoc
app.ModuloVoc = uieditfield(app.Modulos, 'numeric');
app.ModuloVoc.Editable = 'off';
app.ModuloVoc.Position = [56 64 47 22];

% Create Label7
app.Label7 = uilabel(app.Modulos);
app.Label7.HorizontalAlignment = 'right';
app.Label7.Position = [21 42 34 15];
app.Label7.Text = 'Isc (A)';

% Create ModuloIsc
app.ModuloIsc = uieditfield(app.Modulos, 'numeric');
app.ModuloIsc.Editable = 'off';
app.ModuloIsc.Position = [56 38 47 22];

% Create Label8
app.Label8 = uilabel(app.Modulos);
app.Label8.HorizontalAlignment = 'right';
app.Label8.Position = [112 68 20 15];
app.Label8.Text = 'Ki';

% Create ModuloKi
app.ModuloKi = uieditfield(app.Modulos, 'numeric');
app.ModuloKi.Editable = 'off';
app.ModuloKi.Position = [147 64 47 22];

% Create Label9
app.Label9 = uilabel(app.Modulos);
app.Label9.HorizontalAlignment = 'right';
app.Label9.Position = [112 42 20 15];
app.Label9.Text = 'Kv';

% Create ModuloKv
app.ModuloKv = uieditfield(app.Modulos, 'numeric');
app.ModuloKv.Editable = 'off';
app.ModuloKv.Position = [147 38 47 22];

% Create Label10
app.Label10 = uilabel(app.Modulos);
app.Label10.HorizontalAlignment = 'right';
app.Label10.Position = [218 40 20 15];
app.Label10.Text = 'Ns';

% Create ModuloNs
app.ModuloNs = uieditfield(app.Modulos, 'numeric');
app.ModuloNs.Editable = 'off';
app.ModuloNs.Position = [241 36 47 22];
```

```
% Create Label11
app.Label11 = uilabel(app.Modulos);
app.Label11.HorizontalAlignment = 'right';
app.Label11.Position = [203 14 34 15];
app.Label11.Text = 'TONC';

% Create ModuloTONC
app.ModuloTONC = uieditfield(app.Modulos, 'numeric');
app.ModuloTONC.Editable = 'off';
app.ModuloTONC.Position = [241 10 47 22];

% Create Label12
app.Label12 = uilabel(app.Modulos);
app.Label12.HorizontalAlignment = 'right';
app.Label12.Position = [297 40 25 15];
app.Label12.Text = 'Vmp';

% Create ModuloVmp
app.ModuloVmp = uieditfield(app.Modulos, 'numeric');
app.ModuloVmp.Editable = 'off';
app.ModuloVmp.Position = [337 36 47 22];

% Create Label13
app.Label13 = uilabel(app.Modulos);
app.Label13.HorizontalAlignment = 'right';
app.Label13.Position = [302 14 20 15];
app.Label13.Text = 'Imp';

% Create ModuloImp
app.ModuloImp = uieditfield(app.Modulos, 'numeric');
app.ModuloImp.Editable = 'off';
app.ModuloImp.Position = [337 10 47 22];

% Create Label14
app.Label14 = uilabel(app.Modulos);
app.Label14.HorizontalAlignment = 'right';
app.Label14.Position = [206 68 29 15];
app.Label14.Text = '? (%)';

% Create ModuloRend
app.ModuloRend = uieditfield(app.Modulos, 'numeric');
app.ModuloRend.Editable = 'off';
app.ModuloRend.Position = [240 64 47 22];

% Create Label15
app.Label15 = uilabel(app.Modulos);
app.Label15.HorizontalAlignment = 'right';
app.Label15.Position = [298 67 25 15];
app.Label15.Text = 'Pmp';

% Create ModuloPmp
app.ModuloPmp = uieditfield(app.Modulos, 'numeric');
app.ModuloPmp.Editable = 'off';
```

```
app.ModuloPmp.Position = [338 63 47 22];

% Create Label21
app.Label21 = uilabel(app.Modulos);
app.Label21.HorizontalAlignment = 'right';
app.Label21.Position = [54 15 89 15];
app.Label21.Text = 'Superficie (m^2)';

% Create ModuloSup
app.ModuloSup = uieditfield(app.Modulos, 'numeric');
app.ModuloSup.Editable = 'off';
app.ModuloSup.Position = [147 11 47 22];

% Create DatosSimulacion
app.DatosSimulacion = uitabgroup(app.PreciSolApp);
app.DatosSimulacion.Units = 'pixels';
app.DatosSimulacion.Position = [14 58 404 139];

% Create DatosBasicos
app.DatosBasicos = uitab(app.DatosSimulacion);
app.DatosBasicos.Units = 'pixels';
app.DatosBasicos.Title = 'Simulaci n';

% Create LabelDiscreteKnob
app.LabelDiscreteKnob = uilabel(app.DatosBasicos);
app.LabelDiscreteKnob.HorizontalAlignment = 'center';
app.LabelDiscreteKnob.Position = [191.5 91 51 15];
app.LabelDiscreteKnob.Text = 'Precisi n';

% Create Precision
app.Precision = uiknob(app.DatosBasicos, 'discrete');
app.Precision.Items = {'10', '100', '1000', '10000', '100000'};
app.Precision.Position = [267 7 69 69];
app.Precision.Value = '1000';

% Create Modelo
app.Modelo = uibuttongroup(app.DatosBasicos);
app.Modelo.BorderType = 'line';
app.Modelo.Title = 'Modelo de simulaci n';
app.Modelo.FontName = 'Helvetica';
app.Modelo.FontUnits = 'pixels';
app.Modelo.FontSize = 12;
app.Modelo.Units = 'pixels';
app.Modelo.Position = [20 19 143 74];

% Create Diodo1Modelo
app.Diodo1Modelo = uiradiobutton(app.Modelo);
app.Diodo1Modelo.Text = 'Modelo de 1 diodo';
app.Diodo1Modelo.Position = [10 27 121 16];
app.Diodo1Modelo.Value = true;

% Create Diodo2Modelo
app.Diodo2Modelo = uiradiobutton(app.Modelo);
app.Diodo2Modelo.Text = 'Modelo de 2 diodos';
```

```
app.Diodo2Modelo.Position = [10 5 127 16];

% Create DatosPerdidas
app.DatosPerdidas = uitab(app.DatosSimulacion);
app.DatosPerdidas.Units = 'pixels';
app.DatosPerdidas.Title = 'P rdidas';

% Create LabelNumericEditField5
app.LabelNumericEditField5 = uilabel(app.DatosPerdidas);
app.LabelNumericEditField5.HorizontalAlignment = 'right';
app.LabelNumericEditField5.Position = [37 71 192 15];
app.LabelNumericEditField5.Text = 'P rdida de pot. nom. de f brica
↪   (%)';

% Create PerFab
app.PerFab = uieditfield(app.DatosPerdidas, 'numeric');
app.PerFab.Limits = [0 10];
app.PerFab.Position = [244 67 46 22];

% Create Label24
app.Label24 = uilabel(app.DatosPerdidas);
app.Label24.HorizontalAlignment = 'right';
app.Label24.Position = [42 42 187 15];
app.Label24.Text = 'P rdida de pot. por vida  til (a os)';

% Create PerVid
app.PerVid = uieditfield(app.DatosPerdidas, 'numeric');
app.PerVid.Limits = [0 25];
app.PerVid.Position = [244 38 46 22];

% Create Label25
app.Label25 = uilabel(app.DatosPerdidas);
app.Label25.HorizontalAlignment = 'right';
app.Label25.Position = [12 13 217 15];
app.Label25.Text = 'P rdida de irradiancia por suciedad (%)';

% Create PerSuc
app.PerSuc = uieditfield(app.DatosPerdidas, 'numeric');
app.PerSuc.Limits = [0 100];
app.PerSuc.Position = [244 9 46 22];

% Create Label26
app.Label26 = uilabel(app.DatosPerdidas);
app.Label26.Position = [306 95 62 15];
app.Label26.Text = 'Estoc stico';

% Create PerFabRand
app.PerFabRand = uicheckbox(app.DatosPerdidas);
app.PerFabRand.ValueChangedFcn = createCallbackFcn(app,
↪   @PerFabRandValueChanged);
app.PerFabRand.Text = '';
app.PerFabRand.Position = [328 70 22 16];

% Create PerVidRand
```

```
app.PerVidRand = uicheckbox(app.DatosPerdidas);
app.PerVidRand.ValueChangedFcn = createCallbackFcn(app,
↪   @PerVidRandValueChanged);
app.PerVidRand.Text = '';
app.PerVidRand.Position = [328 41 22 16];

% Create PerSucRand
app.PerSucRand = uicheckbox(app.DatosPerdidas);
app.PerSucRand.ValueChangedFcn = createCallbackFcn(app,
↪   @PerSucRandValueChanged);
app.PerSucRand.Text = '';
app.PerSucRand.Position = [328 12 22 16];

% Create Tab
app.Tab = uitab(app.DatosSimulacion);
app.Tab.Units = 'pixels';
app.Tab.Title = 'Nº de m dulos e Interconexi n';

% Create LabelNumericEditField
app.LabelNumericEditField = uilabel(app.Tab);
app.LabelNumericEditField.HorizontalAlignment = 'right';
app.LabelNumericEditField.Position = [38 85 60 15];
app.LabelNumericEditField.Text = 'M. en serie';

% Create ModulosSerie
app.ModulosSerie = uieditfield(app.Tab, 'numeric');
app.ModulosSerie.ValueChangedFcn = createCallbackFcn(app,
↪   @ModulosSerieValueChanged);
app.ModulosSerie.Limits = [1 Inf];
app.ModulosSerie.Position = [113 81 67 22];
app.ModulosSerie.Value = 1;

% Create Label27
app.Label27 = uilabel(app.Tab);
app.Label27.HorizontalAlignment = 'right';
app.Label27.Position = [20 50 78 15];
app.Label27.Text = 'M. en paralelo';

% Create ModulosParalelo
app.ModulosParalelo = uieditfield(app.Tab, 'numeric');
app.ModulosParalelo.ValueChangedFcn = createCallbackFcn(app,
↪   @ModulosParaleloValueChanged);
app.ModulosParalelo.Limits = [1 Inf];
app.ModulosParalelo.Position = [113 46 67 22];
app.ModulosParalelo.Value = 1;

% Create LabelNumericEditField6
app.LabelNumericEditField6 = uilabel(app.Tab);
app.LabelNumericEditField6.HorizontalAlignment = 'right';
app.LabelNumericEditField6.Position = [192 50 52 15];
app.LabelNumericEditField6.Text = 'M. totales';

% Create NumeroModulos
app.NumeroModulos = uieditfield(app.Tab, 'numeric');
```

```
            app.NumeroModulos.ValueDisplayFormat = '%11.10g';
            app.NumeroModulos.Editable = 'off';
            app.NumeroModulos.FontSize = 36;
            app.NumeroModulos.Position = [258 35 125 44];
            app.NumeroModulos.Value = 1;

            % Create Label32
            app.Label32 = uilabel(app.Tab);
            app.Label32.HorizontalAlignment = 'right';
            app.Label32.Position = [28 17 70 15];
            app.Label32.Text = 'Multiplicador';

            % Create MultiplicadorModulos
            app.MultiplicadorModulos = uieditfield(app.Tab, 'numeric');
            app.MultiplicadorModulos.ValueChangedFcn = createCallbackFcn(app,
            ↪   @MultiplicadorModulosValueChanged);
            app.MultiplicadorModulos.Limits = [1 Inf];
            app.MultiplicadorModulos.Position = [113 13 67 22];
            app.MultiplicadorModulos.Value = 1;
        end
    end

    methods (Access = public)

        % Construct app
        function app = PreciSol()

            % Create and configure components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.PreciSolApp)

            % Execute the startup function
            runStartupFcn(app, @startupFcn)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.PreciSolApp)
        end
    end
end
```

A.3. Funciones principales del simulador

- fast_diode.m

```
function [I, V] = fast_diode(Voc, Isc, Ns, Kv, Ki, Rs, Rp, G, T, ...
    n_points, mode)
% -------------------------------------------------------------------------
% Funci n "fast_diode":
%     Funci n simple que convierte los datos proporcionados necesarios
% para el modelo en formato "timeseries" y as  prepararlos para el
% espacio de trabajo de SIMULINK, herramienta con la que se ha
% implementado los modelos, los cuales toman los datos del espacio de
% MATLAB actual. Permite seleccionar el uso de los dos tipos de modelo
% de c lula solar que se han desarrollado.
% -------------------------------------------------------------------------

% Convertimos los datos proporcionados a "timeseries" con los nombres
% correspondientes leidos por los modelos de simulink
Voc_in = timeseries(Voc);
Isc_in = timeseries(Isc);
Ns_in = timeseries(Ns);
Kv_in = timeseries(Kv);
Ki_in = timeseries(Ki);
G_in = timeseries(G);
T_in = timeseries(T);

% Se determinan las opciones para el modelo de SIMULINK de forma externa:
%     -Tama o m ximo de paso
%     -Selecci n del espacio de MATLAB actual para la lectura de
%     par metros
options = simset('MaxStep', num2str(Voc/n_points), 'SrcWorkspace', ...
    'current');
set_param('sim_2_diode','AlgebraicLoopSolver','LineSearch'); % Evita
    % problemas con la resoluci n de cantidades cero o muy peque as
% Selecci n de modo
if mode == true
    Rs_in = timeseries(Rs(2));
    Rp_in = timeseries(Rp(2));
    sim('sim_2_diode', [0, Voc], options);
else
    Rs_in = timeseries(Rs(1));
    Rp_in = timeseries(Rp(1));
    sim('sim_1_diode', [0, Voc], options);
end

% Devoluci n de datos
I = I_out.Data;
V = V_out.Data;
```

- fast_1_diode_resget.m

```
function fast_1_diode_resget(Voc, Isc, Vmp,  Imp, Ns, Kv, Ki)
% ------------------------------------------------------------------------
% Funci n "fast_1_diode_resget":
%     Funci n encargada de la obtenci n de las resistencias serie y
% paralelo correspondientes al modelo de 1 diodo. Difiere ligeramente
% del diagrama de flujo descrito en la memoria (resistencia paralelo
% de comienzo y condici n de detenci n de b squeda iterativa).
% ------------------------------------------------------------------------

% Definici n de constantes y asignaci n de variables
Tref = 298;
Gref = 1000;
T = Tref;
G = Gref;

% Aproximaci n iinicial de Rs y Rp
prompt = 'Aproximaci n inicial de Rs:';
dlg_title = 'Initial Approx.';
num_lines = 1;
defaultans = {'0'};
Rs = inputdlg(prompt,dlg_title,num_lines,defaultans);
Rs = str2double(Rs{1});
Rp = inf;

% Asignamos todas las constantes y variables y las convertivos a objetos
% "tiemseries", compatibles para la lectura del espacio de trabajo de
% SIMULINK
Voc_in = timeseries(Voc);
Isc_in = timeseries(Isc);
Ns_in = timeseries(Ns);
Kv_in = timeseries(Kv);
Ki_in = timeseries(Ki);
Rs_in = timeseries(Rs);
Rp_in = timeseries(Rp);
T_in = timeseries(T);
G_in = timeseries(G);

% Ejecuci n preliminar
options = simset('SrcWorkspace','current');
sim('sim_1_diode', [0, Voc], options);

% Obtenci n de datos preliminares
I = I_out.Data;
V = V_out.Data;
Io = Io_out.Data(1);
Vt = Vt_out.Data(1);
Ipv = Isc;

% C lculo preliminar de diferencia de potencias
Pc = V.*I;
Pmpc = max(Pc);
Pmpe = Vmp*Imp;
```

```
if Pmpc < Pmpe
    prompt = 'Sobreestimaci n del valor de resistencia en serie';
    name = 'ERROR';
    warndlg(prompt, name);
else
    % Aviso de c lculo
    msgbox('Calculando resistencia, puede tardar varios minutos...');

    % C lculo iterativo:
    while (Pmpc > Pmpe)
        Rs = Rs + 0.001; % Aumento de Rs por cada iteraci n
        Rp = (Vmp+Imp*Rs)/(Ipv - Io*(exp((Vmp + Imp*Rs)/Vt) - 1) - Imp);
        Rs_in = timeseries(Rs);
        Rp_in = timeseries(Rp);
        sim('sim_1_diode', [0, Voc], options);
        I = I_out.Data;
        V = V_out.Data;
        Pc = V.*I;
        Pmpc = max(Pc);
    end

    % Final de c lculo y ventana de informaci n
    msg = {strcat('Rs:  ', num2str(Rs), ' Ohm'), ...
        strcat('Rp:  ', num2str(Rp), ' Ohm')};
    msgbox(msg);
end
```

- fast_2_diode_resget.m

```
function fast_2_diode_resget(Voc, Isc, Vmp,  Imp, Ns, Kv, Ki)
% ------------------------------------------------------------------------
% Funci n "fast_2_diode_resget":
%     Funci n encargada de la obtenci n de las resistencias serie y
% paralelo correspondientes al modelo de 2 diodos. Difiere ligeramente
% del diagrama de flujo descrito en la memoria (resistencia paralelo
% de comienzo y condici n de detenci n de b squeda iterativa).
% ------------------------------------------------------------------------

% Definici n de constantes y asignaci n de variables
Tref = 298;
Gref = 1000;
T = Tref;
G = Gref;

% Aproximaci n iinicial de Rs y Rp
prompt = 'Aproximaci n inicial de Rs:';
dlg_title = 'Initial Approx.';
num_lines = 1;
defaultans = {'0'};
Rs = inputdlg(prompt,dlg_title,num_lines,defaultans);
Rs = str2double(Rs{1});
Rp = inf;

% Asignamos todas las constantes y variables y las convertivos a objetos
% "tiemseries", compatibles para la lectura del espacio de trabajo de
% SIMULINK
Voc_in = timeseries(Voc);
Isc_in = timeseries(Isc);
Ns_in = timeseries(Ns);
Kv_in = timeseries(Kv);
Ki_in = timeseries(Ki);
Rs_in = timeseries(Rs);
Rp_in = timeseries(Rp);
T_in = timeseries(T);
G_in = timeseries(G);

% Ejecuci n preliminar
options = simset('SrcWorkspace','current');
sim('sim_2_diode', [0, Voc], options);

% Obtenci n de datos preliminares
I = I_out.Data;
V = V_out.Data;
Io = Io_out.Data(1);
Vt = Vt_out.Data(1);
Ipv = Isc;

% C lculo preliminar de diferencia de potencias
Pc = V.*I;
Pmpc = max(Pc);
Pmpe = Vmp*Imp;
```

```
if Pmpc < Pmpe
    prompt = 'Sobreestimaci n del valor de resistencia en serie';
    name = 'ERROR';
    warndlg(prompt, name);
else
    % Aviso de c lculo
    msgbox('Calculando resistencia, puede tardar varios minutos...');

    % C lculo iterativo:
    while (Pmpc > Pmpe)
        Rs = Rs + 0.001; % Aumento de Rs por cada iteraci n
        Rp = (Vmp+Imp*Rs)/(Ipv - Io*(exp((Vmp + Imp*Rs)/Vt) + ...
            exp((Vmp + Imp*Rs)/(1.2*Vt)) + 2) - Imp);
        Rs_in = timeseries(Rs);
        Rp_in = timeseries(Rp);
        sim('sim_2_diode', [0, Voc], options);
        I = I_out.Data;
        V = V_out.Data;
        Pc = V.*I;
        Pmpc = max(Pc);
    end

    % Final de c lculo y ventana de informaci n
    msg = {strcat('Rs:  ', num2str(Rs), ' Ohm'), ...
        strcat('Rp:  ', num2str(Rp), ' Ohm')};
    msgbox(msg);
end
```

A.4. Funciones de utilidad

- read_modulefile.m

```
function [model_name, rend, Voc, Isc, Pmp, Imp, Vmp, Ns, Kv, Ki, Rs, ...
    Rp, TONC, surface] = read_modulefile(filename)
% ------------------------------------------------------------------------
% Funci n "read_modulefile":
%     Funci n encargada de la lectura de los par metros correspondientes
% a un m dulo solar determinado. Los par metros se almacenan en un
% archivo de extensi n .csv: en la primera columna se nombran los
% par metros almacenados y en la segunda los valores de dichos
% par metros. Los datos correspondientes a un fichero de m dulo son:
%     -Fabricante e indentificador del m dulo
%     -Tensi n de circuito abierto (Voc) [Voltios, V]
%     -Corriente de cortocircuito (Isc) [Amperios, A]
%     -Potencia m xima (Pmp) [Vatios, W]
%     -Corriente de pot. m xima (Imp) [Amperios, A]
%     -Tensi n de pot. m xima (Vmp) [Voltios, V]
%     -N mero de c lulas por m dulo (Ns)
%     -Coeficiente de temp. respecto a Voc (Kv) [mV/ºC]
%     -Coeficiente de temp. respecto a Isc (Ki) [mA/ºC]
%     -Resistencias en serie para mod. de 1 y 2 diodos (Rs) [Ohmios]
%     -Resistencias en paralelo para mod. de 1 y 2 diodos (Rp) [Ohmios]
%     -Temperatura de operaci n nominal de la c lula (TONC) [ºC*m/W]
%     -Superficie total del panel [m^2]
% ------------------------------------------------------------------------

% Apertura y lectura del fichero, ignorando la primera fila
fid = fopen( filename, 'r' );
formatspec = '%s%f';
values = textscan(fid, formatspec, 'Delimiter', ',', ...
    'Headerlines', 1);
fclose(fid);

% Asignaci n de los valores leidos a las variables devueltas por la
% funci n
model_name = values{1, 1}{1}; % Nombre del modelo
rend = values{1, 2}(2); % Rendimiento de la c lula
Voc = values{1, 2}(3); % Voltaje de cortocircuito
Isc = values{1, 2}(4); % Corriente de cortocircuito
Pmp = values{1, 2}(5); % Potencia m xima
Imp = values{1, 2}(6); % Corriente de potencia m xima
Vmp = values{1, 2}(7); % Voltaje de potencia m xima
Ns = values{1, 2}(8); % N mero de c lulas por m dulo
Kv = values{1, 2}(9); % Coef. de temp. respecto a Voc
Ki = values{1, 2}(10); % Coef. de temp. respecto a Isc
Rs(1) = values{1, 2}(11); % Res. serie del modelo de 1 diodo
Rp(1) = values{1, 2}(12); % Res. paralelo del modelo de 1 diodo
Rs(2) = values{1, 2}(13); % Res. serie del modelo de 2 diodos
Rp(2) = values{1, 2}(14); % Res. paralelo del modelo de 2 diodos
TONC = values{1, 2}(15); % Temperatura de operaci n nominal
surface = values{1, 2}(16); % Superficie total del panel
```

- meteonorm_file.m

```
function [G_Bn, Ta] = meteonorm_file(filename)
% ------------------------------------------------------------------------
% Funci n "meteonorm_file":
%     Funci n encargada de la lectura de los datos relevantes del
% fichero "standard minute" proporcionados por la aplicaci n Meteonorm
% 6.0. Lee la radiaci n solar global horizontal por minuto y la
% temperatura ambiente por minuto.
% ------------------------------------------------------------------------

% Apertura del fichero
fid = fopen( filename, 'r' );
% Selecci n del formato de lectura, para leer s lo una columna
formatSpec = '%*d%*d%*d%*d%*d%*d%f%*d%*d%*d%*d%*d%*f'; % G_Gh
G_Bn = cell2mat(textscan(fid, formatSpec)).';
frewind(fid); % Rebobinamos  ndice de lectura del fichero

formatSpec = '%*d%*d%*d%*d%*d%*d%f%*d%*d%*d%*d%*d%f'; % Ta
Ta = cell2mat(textscan(fid, formatSpec)).';

% Cierre del fichero
fclose(fid);
```

- file_fuser.m

```
function fused_filename = file_fuser()
% ------------------------------------------------------------------------
% Funci n "file_fuser":
%     Esta funci n se encarga de solucionar un problema de Meteonorm,
% el cual no interpola a minutos la temperatura por horas, lo que deja
% una columna de datos vac a en el modelo de datos "standard minute".
% Se soluciona interpolando los datos "standard" a minutos mediante
% interpolaci n lineal, y agregando todos los datos obtenidos al fichero
% de minutos sinla columna de temperatura.
% Se implementa como una utilidad para la aplicaci n central del
% simulador.
% ------------------------------------------------------------------------

% Obtenci n de la ruta del fichero "standard"
[hour_filename, hour_path] = uigetfile('.dat');
if strcmp(hour_filename, '') % Si no se ha seleccionado nada
    % No se ejecuta nada, error
else
    % Obtenci n del fichero de "standard minute"
    [minute_filename, minute_path] = uigetfile('.dat');
    % Creaci n de la ruta y el nombre del fichero final completo
    city_name = minute_filename(1:4);
    fused_path = strcat(minute_path, city_name, 'minu_wtemp.dat');

    % Apertura y lectura del fichero "standard", obtenci n de los datos
    % de temperatura
    fid_hour = fopen(strcat(hour_path, hour_filename), 'r' );
    formatspec = '%*d%*d%*d%*d%*d%*d%*d%*d%*d%f';
    hour_temp = cell2mat(textscan(fid_hour, formatspec));
    fclose(fid_hour);

    % Interpolaci n de los datos
    minu_temp = 0;
    for i = 1:length(hour_temp)
        ini_val = hour_temp(i);
        if i == length(hour_temp);
            fin_val = ini_val - (hour_temp(i-1) - ini_val);
        else
            fin_val = hour_temp(i+1);
        end
        interval = (fin_val-ini_val)/60;
        for j = 1:60
           minu_temp((i-1)*60+j) = ini_val + interval*(j-1);
        end
    end
    clear hour_temp; % Limpiamos de memoria el vector de temp. sin interp.

    % Apertura y lectura del fichero "standard minute"
    fid_minu = fopen( strcat(minute_path, minute_filename), 'r' );
    formatspec = '%d%d%d%d%d%d%f%d%d%d%d%d';
    minu_matrix = textscan(fid_minu, formatspec);
    fclose(fid_minu);
```

```
    % A adimos a la matriz de los datos leidos el vector de temperaturas
    % interpoladas
    minu_matrix{13} = minu_temp.';

    % Creaci n de barra de progreso para la escritura del fichero
    progressbar(0)
    progressbar('Progreso de la fusi n');

    % Creaci n y apertura del fichero fusionado
    fid_fused = fopen(fused_path, 'wt');
    % Divisi n de datos a escribir, no se pueden guardar datos de
    % distinto formato en el mismo vector, se hace en 4 tandas
    formatspec_1 = '%d\t%d\t%d\t%d\t%d\t%d\t';
    formatspec_2 = '%0.1f\t';
    formatspec_3 = '%d\t%d\t%d\t%d\t%d\t';
    formatspec_4 = '%0.2f\r\n';
    % Escritura de las filas del fichero de datos fusionado
    for i = 1:length(minu_matrix{1})
        min_vec_1 = [minu_matrix{1}(i); minu_matrix{2}(i); ...
            minu_matrix{3}(i); minu_matrix{4}(i); ...
            minu_matrix{5}(i); minu_matrix{6}(i)];
        min_vec_2 = minu_matrix{7}(i);
        min_vec_3 = [minu_matrix{8}(i); minu_matrix{9}(i); ...
            minu_matrix{10}(i); minu_matrix{11}(i); ...
            minu_matrix{12}(i)];
        min_vec_4 = minu_matrix{13}(i);
        fprintf(fid_fused, formatspec_1, min_vec_1);
        fprintf(fid_fused, formatspec_2, min_vec_2);
        fprintf(fid_fused, formatspec_3, min_vec_3);
        fprintf(fid_fused, formatspec_4, min_vec_4);
        progressbar((1/length(minu_matrix{1}))*i);
    end
    % Cierre del fichero y cierre de la barra de progreso
    fclose(fid_fused);
    progressbar(1);
end
```

- get_date_index.m

```
function date_index = get_date_index(n_month, day)
% ------------------------------------------------------------------------
% Funci n "get_date_index":
%      sta funci n calcula el n mero de columna del fichero de datos de
% Meteonorm "standard minute" al que corresponde la fecha proporcionada,
% en forma de n mero de mes y n mero de d a.
% ------------------------------------------------------------------------

% Comprobaci n de errores en el mes
if n_month > 12 || n_month < 1
    date_index = -1;
else
    date_index = 0;

    if n_month == 1 % Si es enero
        date_index = 0;
    else % Para el resto de meses
        for i = 1:n_month-1
            date_index = date_index + get_days_month(i);
        end
    end
    % Comprobaci n de errores en el d a
    if day > get_days_month(n_month) || day < 1
        date_index = -1;
    else
        % Se a ade el n mero de d as y se multiplica por el n mero
        % de minutos de un d a
        date_index = (date_index + day)*1440;
    end
end
```

- get_days_month.m

```
function n_days = get_days_month(n_month)
% ------------------------------------------------------------------------
% Funci n "get_days_month":
%      sta funci n devuelve el n mero correspondiente al n mero de mes
% proporcionado.  til para la funci n de obtenci n de  ndices y para
% la aplicaci n central del simulador.
% ------------------------------------------------------------------------

% Se devuelve el n mero de d as correspondientes a cada mes
switch n_month
    case 1; % Enero
        n_days = 31;
    case 2; % Febrero (siempre se supone a o no bisiesto)
        n_days = 28;
    case 3; % Marzo
        n_days = 31;
    case 4; % Abril
        n_days = 30;
    case 5; % Mayo
        n_days = 31;
    case 6; % Junio
        n_days = 30;
    case 7; % Julio
        n_days = 31;
    case 8; % Agosto
        n_days = 31;
    case 9; % Septiembre
        n_days = 30;
    case 10; % Octubre
        n_days = 31;
    case 11; % Noviembre
        n_days = 30;
    case 12; % Diciembre
        n_days = 31;
    otherwise; % Mes err neo
        n_days = -1;
end
```

- get_month.m

```
function month = get_month(n_month)
% ------------------------------------------------------------------------
% Funci n "get_month":
%      sta funci n devuelve la cadena de caracteres correspondiente para
% cada n mero de mes en espa ol.  til solo para la aplicaci n central
% del simulador.
% ------------------------------------------------------------------------

% Selecci n segun el n mero del mes
switch n_month
    case 1;
        month = 'Enero';
    case 2;
        month = 'Febrero';
    case 3;
        month = 'Marzo';
    case 4;
        month = 'Abril';
    case 5;
        month = 'Mayo';
    case 6;
        month = 'Junio';
    case 7;
        month = 'Julio';
    case 8;
        month = 'Agosto';
    case 9;
        month = 'Septiembre';
    case 10;
        month = 'Octubre';
    case 11;
        month = 'Noviembre';
    case 12;
        month = 'Diciembre';
    otherwise; % Error
        month = 'default';
end
```

- get_month_num.m

```
function n_month = get_month_num(month)
% ------------------------------------------------------------------------
% Funci n "get_month_num":
%      sta funci n devuelve el n mero de mes correspondiente a la cadena
% de caracteres proporcionada.  til solo para la aplicaci n central
% del simulador, para la obtenci n del n mero del mes a partir de un
% valor de un men  desplegable
% ------------------------------------------------------------------------

% Obtenci n del n mero del mes segun la cadena de caracteres proporcionada
switch month
    case 'Enero';
        n_month = 1;
    case 'Febrero';
        n_month = 2;
    case 'Marzo';
        n_month = 3;
    case 'Abril';
        n_month = 4;
    case 'Mayo';
        n_month = 5;
    case 'Junio';
        n_month = 6;
    case 'Julio';
        n_month = 7;
    case 'Agosto';
        n_month = 8;
    case 'Septiembre';
        n_month = 9;
    case 'Octubre';
        n_month = 10;
    case 'Noviembre';
        n_month = 11;
    case 'Diciembre';
        n_month = 12;
    otherwise; % Error
        n_month = -1;
end
```

- max_p_point.m

```
function [I_max, V_max] = max_p_point(I, V)
% ------------------------------------------------------------------------
% Funci n "max_p_point":
%     Funci n encargada de obtener el voltaje e intensidad
% correspondientes al punto de m xima potencia.
% ------------------------------------------------------------------------

% Obtenemos el vector de potencia
P = I.*V;
P_max = max(P); % M xima potencia
%  ndice del pto de Pmax
index = find(P==P_max);
index = index(1);
I_max = I(index); % Corriente de m x. potencia
V_max = V(index); % Tensi n de m x- potencia
```

- progressbar.m

```
function progressbar(varargin)
% Description:
%   progressbar() provides an indication of the progress of some task using
% graphics and text. Calling progressbar repeatedly will update the figure and
% automatically estimate the amount of time remaining.
%   This implementation of progressbar is intended to be extremely simple to use
% while providing a high quality user experience.
%
% Features:
%   - Can add progressbar to existing m-files with a single line of code.
%   - Supports multiple bars in one figure to show progress of nested loops.
%   - Optional labels on bars.
%   - Figure closes automatically when task is complete.
%   - Only one figure can exist so old figures don't clutter the desktop.
%   - Remaining time estimate is accurate even if the figure gets closed.
%   - Minimal execution time. Won't slow down code.
%   - Randomized color. When a programmer gets bored...
%
% Example Function Calls For Single Bar Usage:
%   progressbar               % Initialize/reset
%   progressbar(0)            % Initialize/reset
%   progressbar('Label')      % Initialize/reset and label the bar
%   progressbar(0.5)          % Update
%   progressbar(1)            % Close
%
% Example Function Calls For Multi Bar Usage:
%   progressbar(0, 0)         % Initialize/reset two bars
%   progressbar('A', '')      % Initialize/reset two bars with one label
%   progressbar('', 'B')      % Initialize/reset two bars with one label
%   progressbar('A', 'B')     % Initialize/reset two bars with two labels
%   progressbar(0.3)          % Update 1st bar
%   progressbar(0.3, [])      % Update 1st bar
%   progressbar([], 0.3)      % Update 2nd bar
%   progressbar(0.7, 0.9)     % Update both bars
%   progressbar(1)            % Close
%   progressbar(1, [])        % Close
%   progressbar(1, 0.4)       % Close
%
% Notes:
%   For best results, call progressbar with all zero (or all string) inputs
% before any processing. This sets the proper starting time reference to
% calculate time remaining.
%   Bar color is choosen randomly when the figure is created or reset. Clicking
% the bar will cause a random color change.
%
% Demos:
%     % Single bar
%     m = 500;
%     progressbar % Init single bar
%     for i = 1:m
%       pause(0.01) % Do something important
%       progressbar(i/m) % Update progress bar
```

```
%     end
%
%     % Simple multi bar (update one bar at a time)
%     m = 4;
%     n = 3;
%     p = 100;
%     progressbar(0,0,0) % Init 3 bars
%     for i = 1:m
%         progressbar([],0) % Reset 2nd bar
%         for j = 1:n
%             progressbar([],[],0) % Reset 3rd bar
%             for k = 1:p
%                 pause(0.01) % Do something important
%                 progressbar([],[],k/p) % Update 3rd bar
%             end
%             progressbar([],j/n) % Update 2nd bar
%         end
%         progressbar(i/m) % Update 1st bar
%     end
%
%     % Fancy multi bar (use labels and update all bars at once)
%     m = 4;
%     n = 3;
%     p = 100;
%     progressbar('Monte Carlo Trials','Simulation','Component') % Init 3 bars
%     for i = 1:m
%         for j = 1:n
%             for k = 1:p
%                 pause(0.01) % Do something important
%                 % Update all bars
%                 frac3 = k/p;
%                 frac2 = ((j-1) + frac3) / n;
%                 frac1 = ((i-1) + frac2) / m;
%                 progressbar(frac1, frac2, frac3)
%             end
%         end
%     end
%
% Author:
%   Steve Hoelzer
%
% Revisions:
% 2002-Feb-27   Created function
% 2002-Mar-19   Updated title text order
% 2002-Apr-11   Use floor instead of round for percentdone
% 2002-Jun-06   Updated for speed using patch (Thanks to waitbar.m)
% 2002-Jun-19   Choose random patch color when a new figure is created
% 2002-Jun-24   Click on bar or axes to choose new random color
% 2002-Jun-27   Calc time left, reset progress bar when fractiondone == 0
% 2002-Jun-28   Remove extraText var, add position var
% 2002-Jul-18   fractiondone input is optional
% 2002-Jul-19   Allow position to specify screen coordinates
% 2002-Jul-22   Clear vars used in color change callback routine
% 2002-Jul-29   Position input is always specified in pixels
```

```
% 2002-Sep-09   Change order of title bar text
% 2003-Jun-13   Change 'min' to 'm' because of built in function 'min'
% 2003-Sep-08   Use callback for changing color instead of string
% 2003-Sep-10   Use persistent vars for speed, modify titlebarstr
% 2003-Sep-25   Correct titlebarstr for 0% case
% 2003-Nov-25   Clear all persistent vars when percentdone = 100
% 2004-Jan-22   Cleaner reset process, don't create figure if percentdone = 100
% 2004-Jan-27   Handle incorrect position input
% 2004-Feb-16   Minimum time interval between updates
% 2004-Apr-01   Cleaner process of enforcing minimum time interval
% 2004-Oct-08   Seperate function for timeleftstr, expand to include days
% 2004-Oct-20   Efficient if-else structure for sec2timestr
% 2006-Sep-11   Width is a multiple of height (don't stretch on widescreens)
% 2010-Sep-21   Major overhaul to support multiple bars and add labels
%

persistent progfig progdata lastupdate

% Get inputs
if nargin > 0
    input = varargin;
    ninput = nargin;
else
    % If no inputs, init with a single bar
    input = {0};
    ninput = 1;
end

% If task completed, close figure and clear vars, then exit
if input{1} == 1
    if ishandle(progfig)
        delete(progfig) % Close progress bar
    end
    clear progfig progdata lastupdate % Clear persistent vars
    drawnow
    return
end

% Init reset flag
resetflag = false;

% Set reset flag if first input is a string
if ischar(input{1})
    resetflag = true;
end

% Set reset flag if all inputs are zero
if input{1} == 0
    % If the quick check above passes, need to check all inputs
    if all([input{:}] == 0) && (length([input{:}]) == ninput)
        resetflag = true;
    end
end
```

```
% Set reset flag if more inputs than bars
if ninput > length(progdata)
    resetflag = true;
end

% If reset needed, close figure and forget old data
if resetflag
    if ishandle(progfig)
        delete(progfig) % Close progress bar
    end
    progfig = [];
    progdata = []; % Forget obsolete data
end

% Create new progress bar if needed
if ishandle(progfig)
else % This strange if-else works when progfig is empty (~ishandle() does not)

    % Define figure size and axes padding for the single bar case
    height = 0.03;
    width = height * 8;
    hpad = 0.02;
    vpad = 0.25;

    % Figure out how many bars to draw
    nbars = max(ninput, length(progdata));

    % Adjust figure size and axes padding for number of bars
    heightfactor = (1 - vpad) * nbars + vpad;
    height = height * heightfactor;
    vpad = vpad / heightfactor;

    % Initialize progress bar figure
    left = (1 - width) / 2;
    bottom = (1 - height) / 2;
    progfig = figure(...
        'Units', 'normalized',...
        'Position', [left bottom width height],...
        'NumberTitle', 'off',...
        'Resize', 'off',...
        'MenuBar', 'none' );

    % Initialize axes, patch, and text for each bar
    left = hpad;
    width = 1 - 2*hpad;
    vpadtotal = vpad * (nbars + 1);
    height = (1 - vpadtotal) / nbars;
    for ndx = 1:nbars
        % Create axes, patch, and text
        bottom = vpad + (vpad + height) * (nbars - ndx);
        progdata(ndx).progaxes = axes( ...
            'Position', [left bottom width height], ...
            'XLim', [0 1], ...
            'YLim', [0 1], ...
```

```
            'Box', 'on', ...
            'ytick', [], ...
            'xtick', [] );
        progdata(ndx).progpatch = patch( ...
            'XData', [0 0 0 0], ...
            'YData', [0 0 1 1] );
        progdata(ndx).progtext = text(0.99, 0.5, '', ...
            'HorizontalAlignment', 'Right', ...
            'FontUnits', 'Normalized', ...
            'FontSize', 0.7 );
        progdata(ndx).proglabel = text(0.01, 0.5, '', ...
            'HorizontalAlignment', 'Left', ...
            'FontUnits', 'Normalized', ...
            'FontSize', 0.7 );
        if ischar(input{ndx})
            set(progdata(ndx).proglabel, 'String', input{ndx})
            input{ndx} = 0;
        end

        % Set callbacks to change color on mouse click
        set(progdata(ndx).progaxes, 'ButtonDownFcn', {@changecolor,
        ↪  progdata(ndx).progpatch})
        set(progdata(ndx).progpatch, 'ButtonDownFcn', {@changecolor,
        ↪  progdata(ndx).progpatch})
        set(progdata(ndx).progtext, 'ButtonDownFcn', {@changecolor,
        ↪  progdata(ndx).progpatch})
        set(progdata(ndx).proglabel, 'ButtonDownFcn', {@changecolor,
        ↪  progdata(ndx).progpatch})

        % Pick a random color for this patch
        changecolor([], [], progdata(ndx).progpatch)

        % Set starting time reference
        if ~isfield(progdata(ndx), 'starttime') ||
        ↪  isempty(progdata(ndx).starttime)
            progdata(ndx).starttime = clock;
        end
    end

    % Set time of last update to ensure a redraw
    lastupdate = clock - 1;

end

% Process inputs and update state of progdata
for ndx = 1:ninput
    if ~isempty(input{ndx})
        progdata(ndx).fractiondone = input{ndx};
        progdata(ndx).clock = clock;
    end
end

% Enforce a minimum time interval between graphics updates
myclock = clock;
```

```
if abs(myclock(6) - lastupdate(6)) < 0.01 % Could use etime() but this is faster
    return
end

% Update progress patch
for ndx = 1:length(progdata)
    set(progdata(ndx).progpatch, 'XData', ...
        [0, progdata(ndx).fractiondone, progdata(ndx).fractiondone, 0])
end

% Update progress text if there is more than one bar
if length(progdata) > 1
    for ndx = 1:length(progdata)
        set(progdata(ndx).progtext, 'String', ...
            sprintf('%1d%%', floor(100*progdata(ndx).fractiondone)))
    end
end

% Update progress figure title bar
if progdata(1).fractiondone > 0
    runtime = etime(progdata(1).clock, progdata(1).starttime);
    timeleft = runtime / progdata(1).fractiondone - runtime;
    timeleftstr = sec2timestr(timeleft);
    titlebarstr = sprintf('%2d%%    %s remaining', ...
        floor(100*progdata(1).fractiondone), timeleftstr);
else
    titlebarstr = ' 0%';
end
set(progfig, 'Name', titlebarstr)

% Force redraw to show changes
drawnow

% Record time of this update
lastupdate = clock;


% ------------------------------------------------------------------------------
function changecolor(h, e, progpatch) %#ok<INUSL>
% Change the color of the progress bar patch

% Prevent color from being too dark or too light
colormin = 1.5;
colormax = 2.8;

thiscolor = rand(1, 3);
while (sum(thiscolor) < colormin) || (sum(thiscolor) > colormax)
    thiscolor = rand(1, 3);
end

set(progpatch, 'FaceColor', thiscolor)


% ------------------------------------------------------------------------------
```

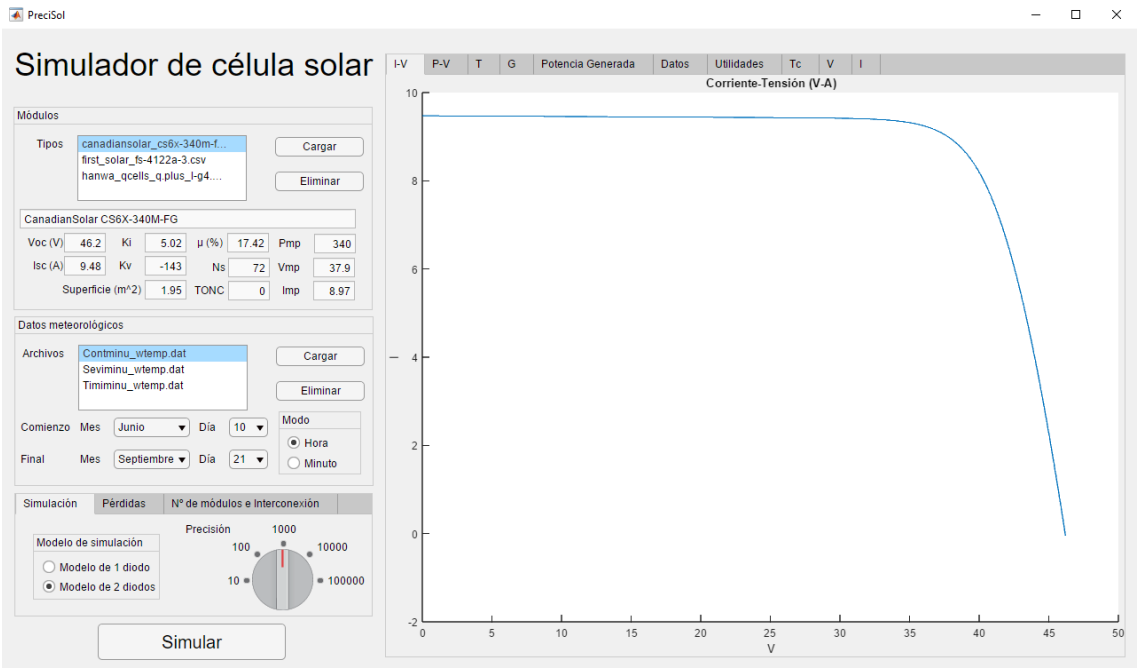
```
function timestr = sec2timestr(sec)
% Convert a time measurement from seconds into a human readable string.

% Convert seconds to other units
w = floor(sec/604800); % Weeks
sec = sec - w*604800;
d = floor(sec/86400); % Days
sec = sec - d*86400;
h = floor(sec/3600); % Hours
sec = sec - h*3600;
m = floor(sec/60); % Minutes
sec = sec - m*60;
s = floor(sec); % Seconds

% Create time string
if w > 0
    if w > 9
        timestr = sprintf('%d week', w);
    else
        timestr = sprintf('%d week, %d day', w, d);
    end
elseif d > 0
    if d > 9
        timestr = sprintf('%d day', d);
    else
        timestr = sprintf('%d day, %d hr', d, h);
    end
elseif h > 0
    if h > 9
        timestr = sprintf('%d hr', h);
    else
        timestr = sprintf('%d hr, %d min', h, m);
    end
elseif m > 0
    if m > 9
        timestr = sprintf('%d min', m);
    else
        timestr = sprintf('%d min, %d sec', m, s);
    end
else
    timestr = sprintf('%d sec', s);
end
```

- save_data.m

```
function save_data(G_Gh, T_a, T_c, P, V_pmax, I_pmax, start_month, start_day, ...
    end_month, end_day, M_series, M_parallel, M_total)
% ------------------------------------------------------------------------
% Funci n "file_fuser":
%     sta funci n tiene como objetivo guardar en un fichero .dat los
% resultados obtenidos en la simulaci n.
% ------------------------------------------------------------------------

% Otenci n del directorio de guardado
file_path = uigetdir(pwd);
% Obtenci n del nombre de fichero
start_month = get_month(start_month);
start_month = start_month(1:3);
start_day = num2str(start_day);
end_month = get_month(end_month);
end_month = end_month(1:3);
end_day = num2str(end_day);
M_series = num2str(M_series);
M_parallel = num2str(M_parallel);
M_total = num2str(M_total);
filename = strcat(file_path, '\sim_', start_month, start_day, '_', ...
    end_month, end_day, '_s', M_series, 'p', M_parallel, 't', M_total, ...
    '.dat');

% Inicio de barra de progreso
progressbar(0);
progressbar('Creando fichero...');

% Apertura de fichero de guardado
fid = fopen(filename, 'wt');
% Formato en columnas del fichero
formatspec = '%0.0f\t%0.1f\t%0.1f\t%0.0f\t%0.1f\t%0.1f\r\n';

% Escritura fila a fila
for i = 1:length(G_Gh)
    file_vec = [G_Gh(i); T_a(i); T_c(i); P(i); V_pmax(i); I_pmax(i)];
    fprintf(fid, formatspec, file_vec);
    progressbar((1/length(G_Gh))*i);
end

% Cierre de fichero y barra de progreso
fclose(fid);
progressbar(1);
```

Apéndice B

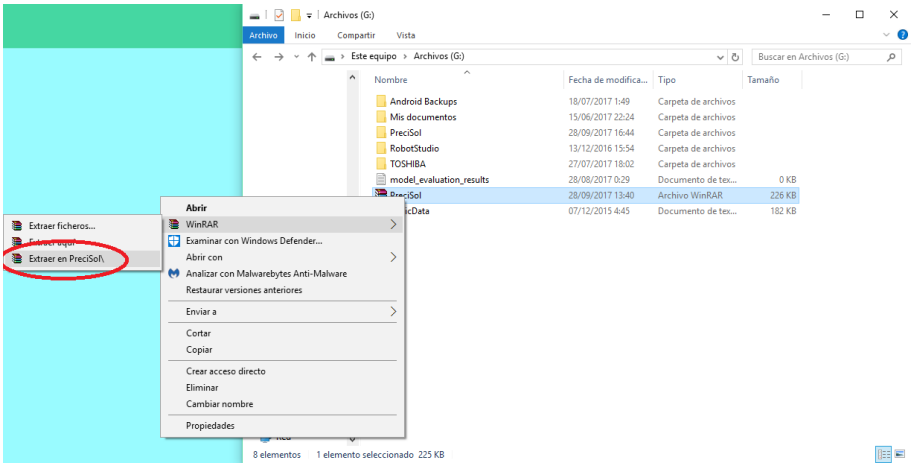
PreciSol: manual de usuario



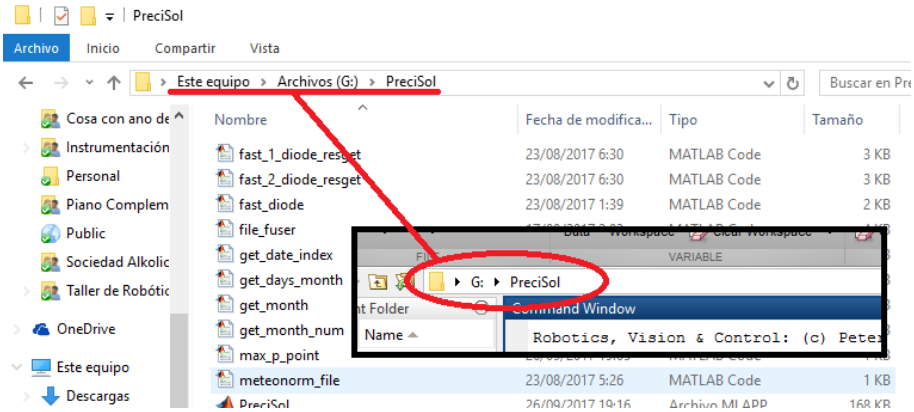
B.1. Instalación

Para la instalación, correcto funcionamiento y capacidad de edición de la aplicación, se necesita la versión 2016a de MATLAB o superior.
Pasos:

1. Instalar MATLAB 2016a o superior
2. Descomprimir los contenidos de la aplicación (16 archivos) en una carpeta definida por el usuario

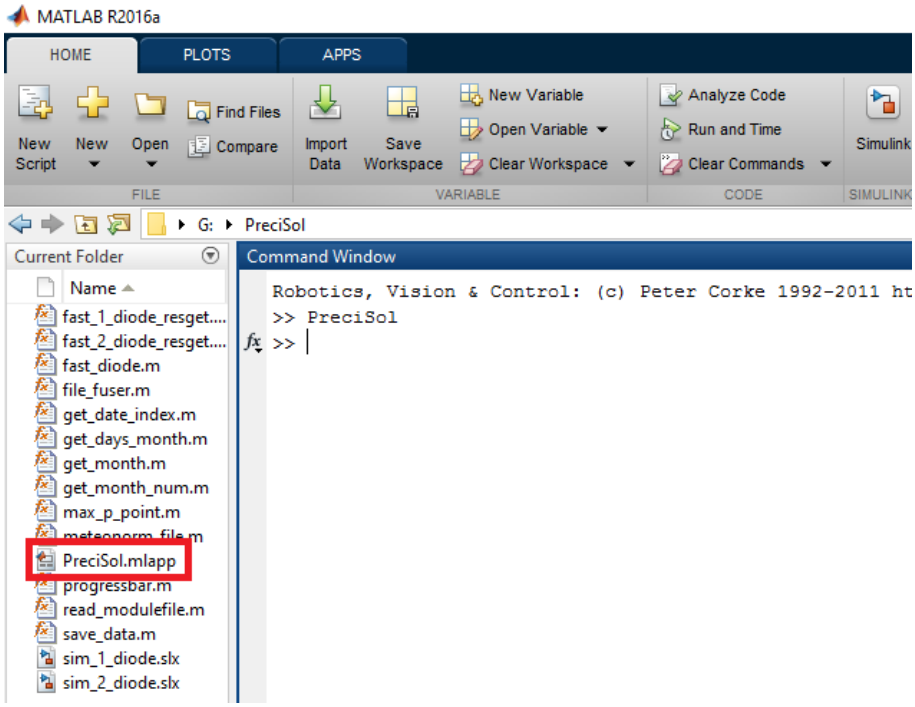


3. En el entorno de MATLAB, acceder a la carpeta donde se encuentran los archivos de la aplicación y utilizarla como carpeta actual

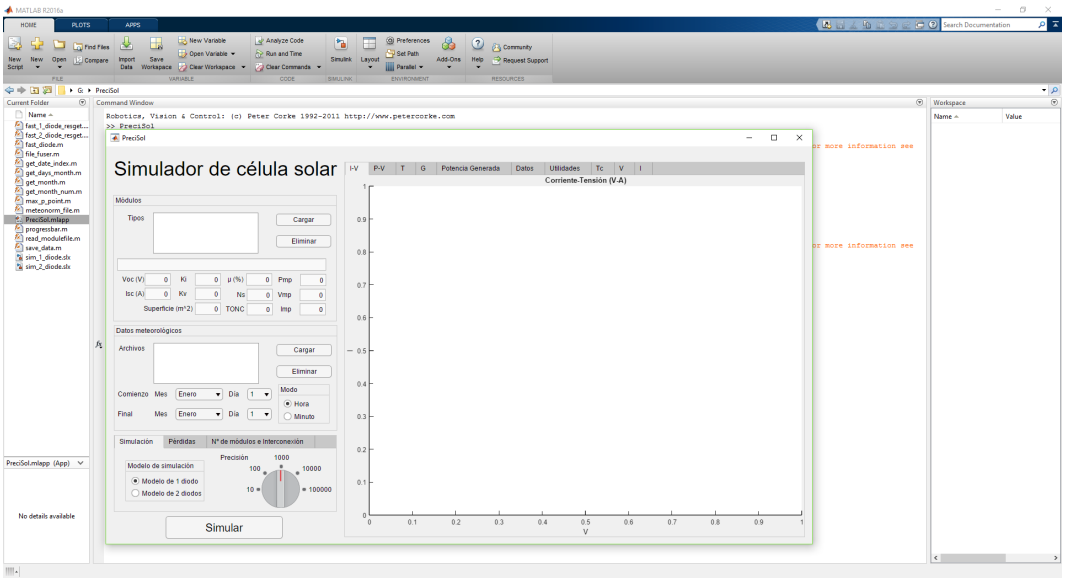


B.2. Ejecutando

Una vez hemos preparado los ficheros y MATLAB para utilizar el software, ejecutaremos el fichero “precisol.mlapp”, que contiene la aplicación principal. Para ello, podemos realizar doble clic en el archivo descrito.

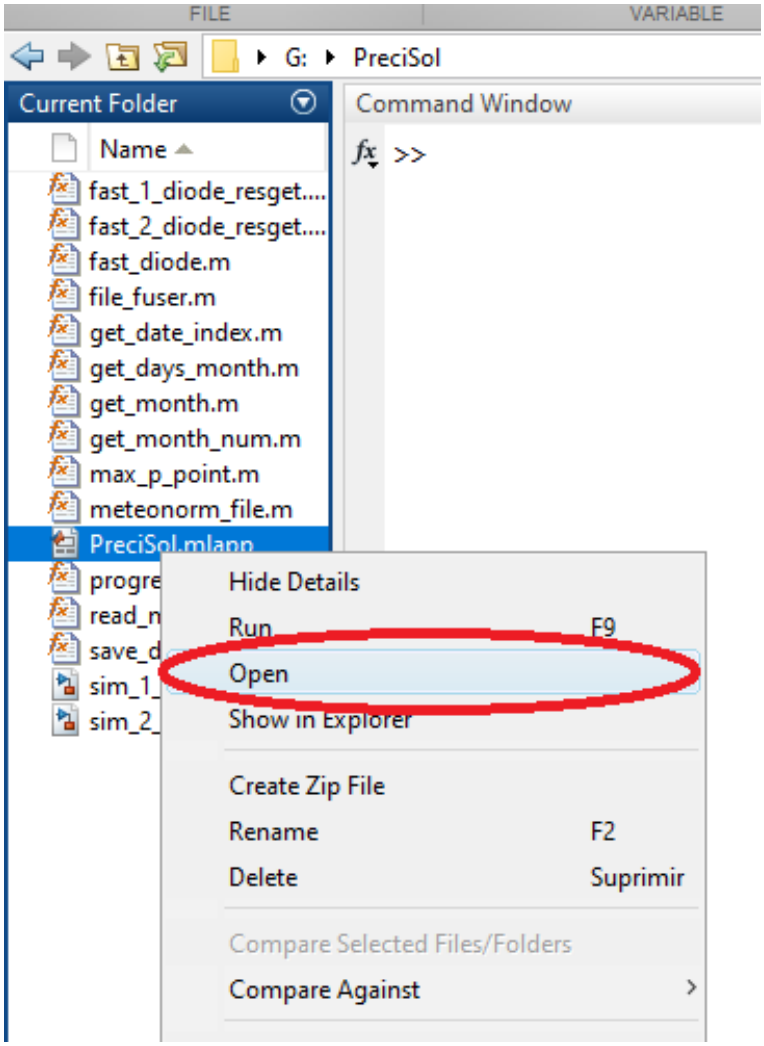


Después de un tiempo de carga, la ventana principal del software aparecerá en pantalla.

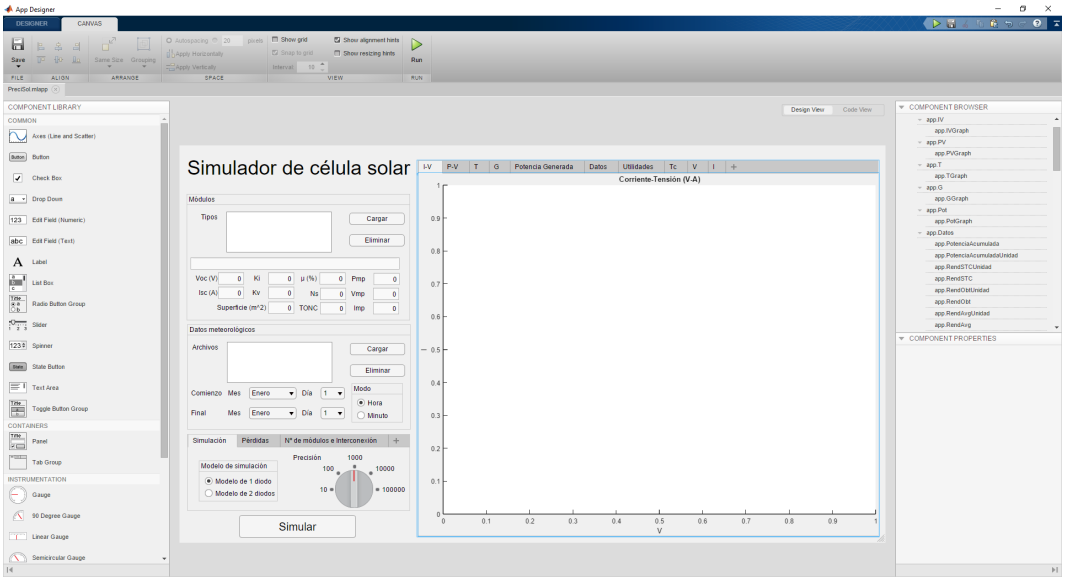


B.3. Edición del programa

Si se desea editar el código del programa principal, basta con abrir el programa desde el menú contextual, con la herramienta App Designer.



Una vez abierto, se puede editar tanto la interfaz gráfica como el código de las llamadas a funciones.



B.4. Simulación

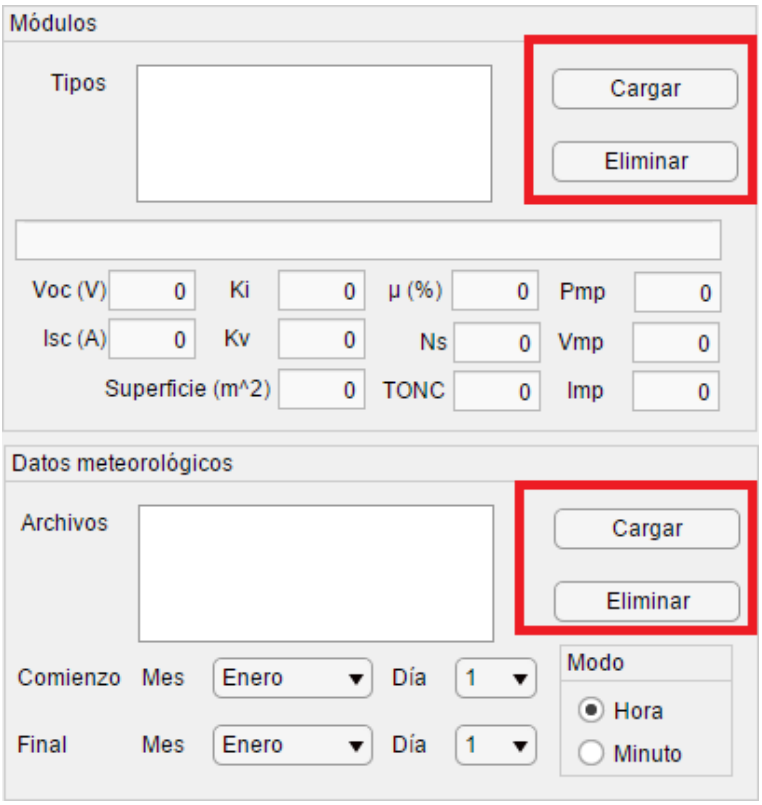
Una vez instalado y ejecutado el programa, se procederá a realizar una primera simulación. Para ello hemos de:

1. Cargar datos del y meteorológicos a simular
2. Ajuste de parámetros
    a) Fechas de comienzo y final de simulación
    b) Selección del modelo de simulación y precisión
    c) Determinación de pérdidas
    d) Introducción del número de módulos e interconexión
3. Presionar botón “Simular”

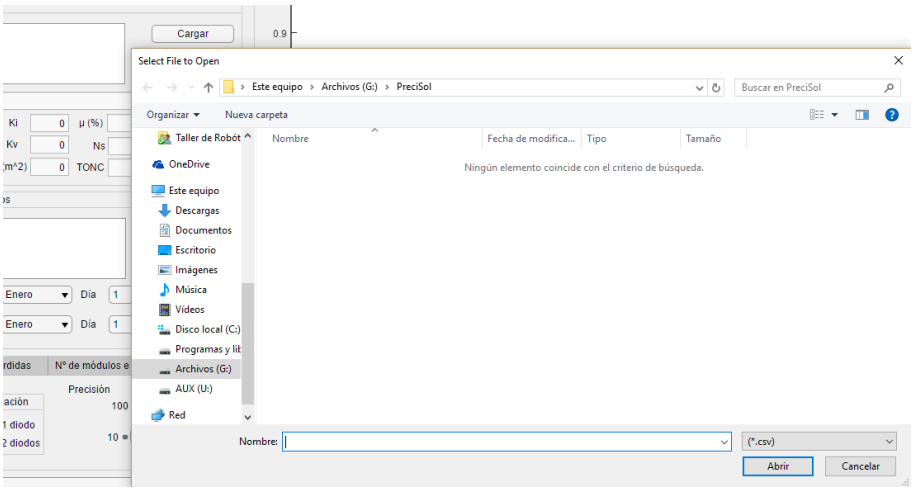
Carga de datos

Para configurar la simulación, se comienza por cargar los datos necesarios de módulo y meteorológicos, guardados en ficheros de formatos .csv (*comma separated value*s) y .dat, respectivamente.

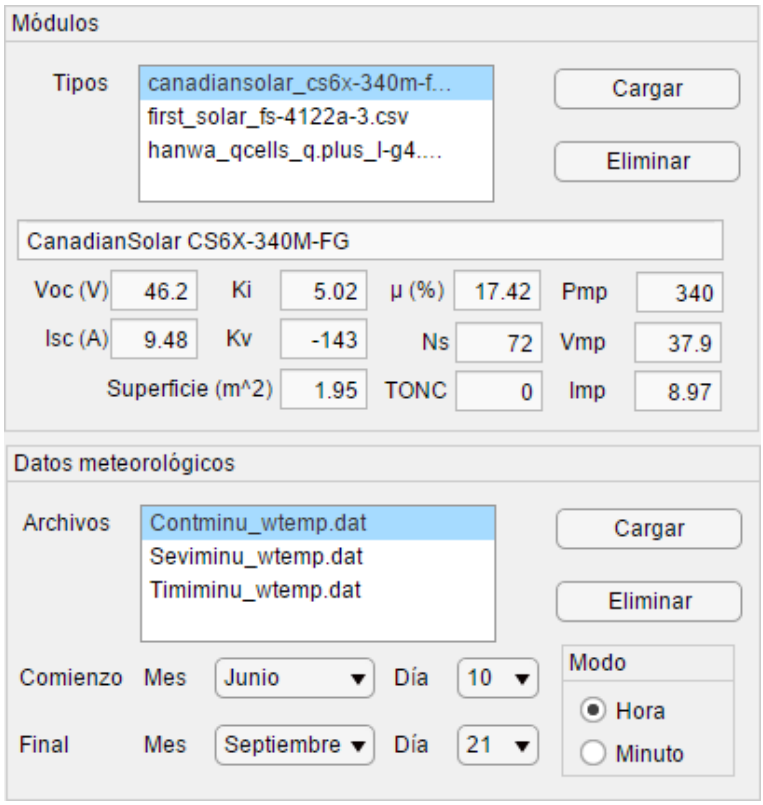
Para cargar y eliminar cada fichero del tipo correspondiente, se usan los dos botones situados a la derecha del cuadro donde se muestran los ficheros cargados.



Al hacer click en el botón de carga de alguno de los cuadros, se abre una ventana de selección de fichero, el cual aísla la posibilidad de seleccionar un fichero con el formato inadecuado.



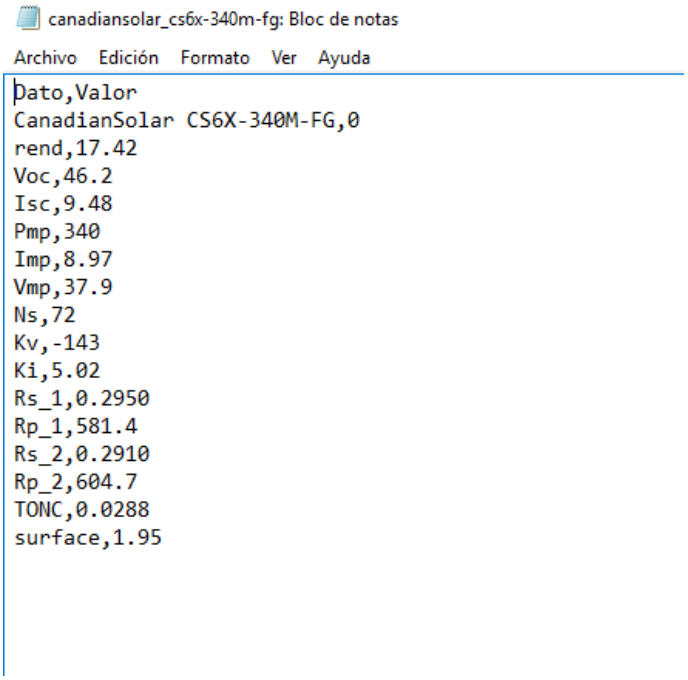
Una vez cargados los ficheros deseados, se pueden seleccionar haciendo clic en el nombre del fichero en el cuadro, y se puede eliminar el fichero seleccionado presionando el botón “Eliminar”.



- Creación de datos de módulo

  Los ficheros de datos de módulo son de formato .csv, y su estructura es de la siguiente manera:

Dato	Valor
Fabricante y modelo	0
rend	-
Voc	-
Isc	-
Pmp	-
Vmp	-
Ns	-
Kv	-
Ki	-
Rs_1	-
Rp_1	-
Rs_2	-
Rp_2	-
TONC	-
surface	-



Conociendo los parámetros del módulo, la creación de un fichero de datos de módulo es muy sencilla: simplemente se necesita rellenar cada dato con la estructura anteriormente descrita, y guardarlo en formato .csv, para poder ser cargado en el simulador.

- Creación de datos meteorológicos

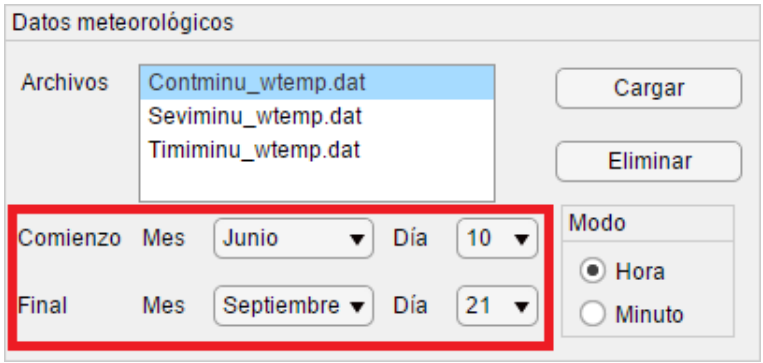
  La obtención de datos meteorológicos usados por éste programa es un tanto más compleja, ya que requiere de software adicional, Meteonorm 6.0. Debido a la incompatibilidad de este software con versiones de Windows posteriores a XP, se necesita de una máquina que posea éste sistema, o una máquina virtual que lo emule[1].

  Una vez en la aplicación, se han de sacar dos ficheros de datos por cada localización deseada: uno en formato “standard minute” y otro en “standard” (dato por cada hora), para posteriormente fusionarlos con la utilidad *File Fuser* ofrecida por la aplicación PreciSol y convertirlos en un fichero útil para la misma.

Ajuste de parámetros

Antes de realizar la simulación, han de seleccionarse los parámetros deseados para la simulación, que influirán en los resultados de las mismas.

- Fechas de comienzo y final de simulación: se seleccionan mediante los menús desplegables. La fecha de comienzo marca las 00:00 del día seleccionado, y la fecha de final marca las 23:59 del día seleccionado.



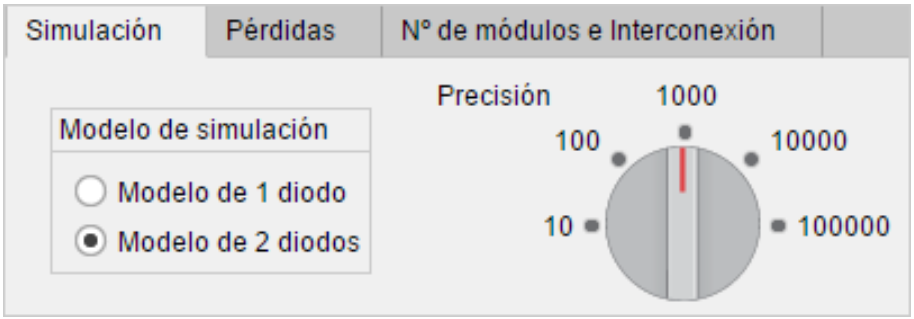
  EL programa lanza un mensaje de error al intentar simular en caso de que se haya seleccionado una fecha de comienzo posterior a la de finalización.

  En este cuadro se puede también seleccionar el modo de ejecución: hora o

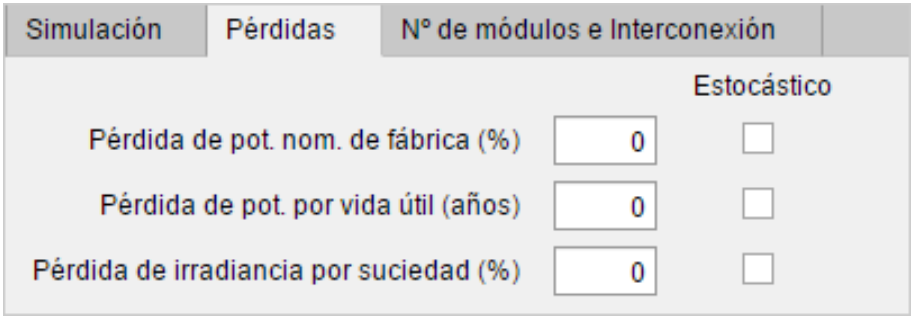
[1] En este caso se ha utilizado el software VirtualBox, configurando una máquina con los requisitos mínimos de la aplicación e instalando Meteonorm 6.0 en la misma. Los ficheros de datos meteorológicos se han sacado utilizando una unidad de almacenamiento USB

minuto. Dependiendo de éste parámetro, se utilizará el set completo de datos por cada minuto o sólo la media de cada dato por cada hora. En el primer caso, se realizan 60 veces más puntos que en el segundo, lo que aumenta el tiempo de simulación proporcionalmente.

- Selección del modelo de simulación y precisión: mediante el cuadro de botones y el dial discreto de selección de modelo y precisión respectivamente se establecen los valores para estos parámetros.

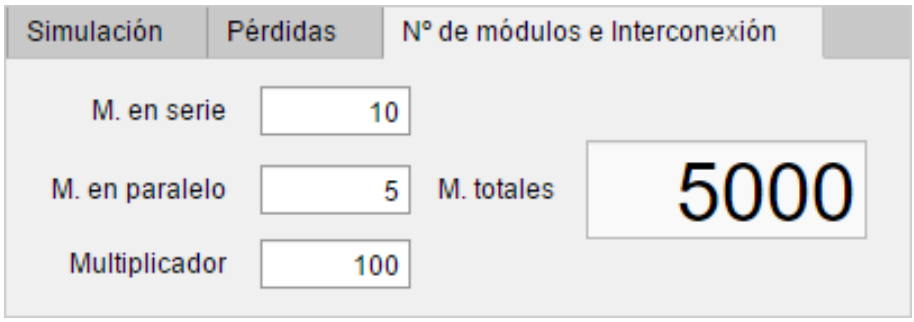


- Determinación de pérdidas: se introducen los valores deseados para las pérdidas en los cuadros, o se selecciona el botón “estocástico” situado al lado de cada cuadro editable en caso de desear un valor aleatorio para cada término de pérdidas.



Cada término está limitado mediante software, y no será posible introducir valores fuera de los límites establecidos.

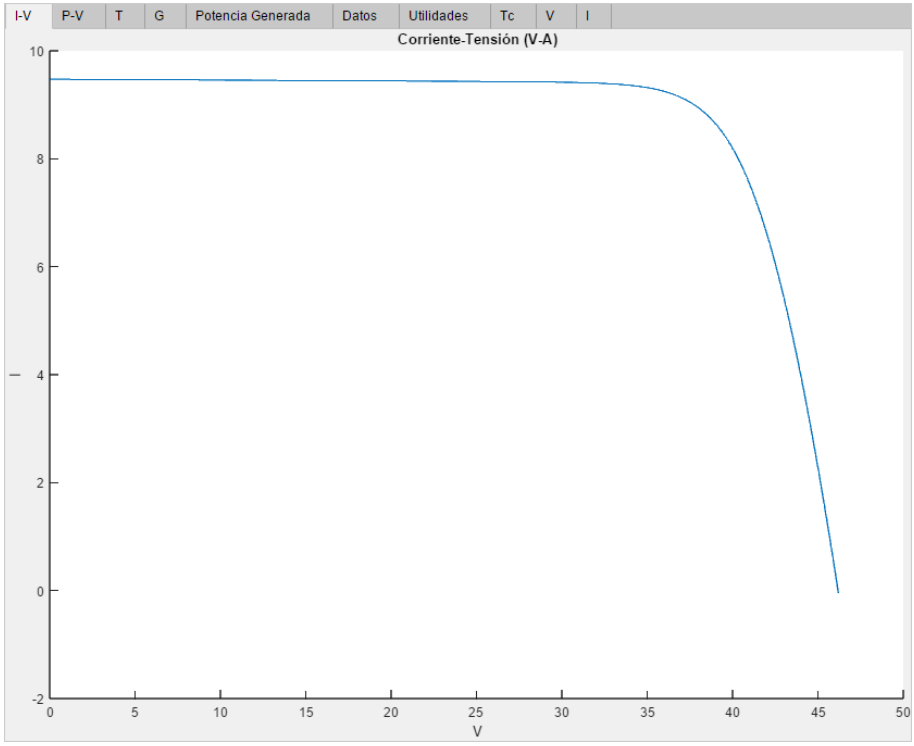
- Introducción del número de módulos e interconexión: aquí se introducen el número de módulos por cada rama serie, las ramas serie interconectadas en paralelo y el número de conjuntos de ramas conectadas en paralelo que se desean en la instalación (multiplicador).



Obtención de resultados

Una vez introducidos todos los parámetros necesarios, se pulsará el botón “Simular”, lo que, tras un tiempo de ejecución, generará una serie de resultados, que además podrán ser guardados en un fichero.

- Gráficas



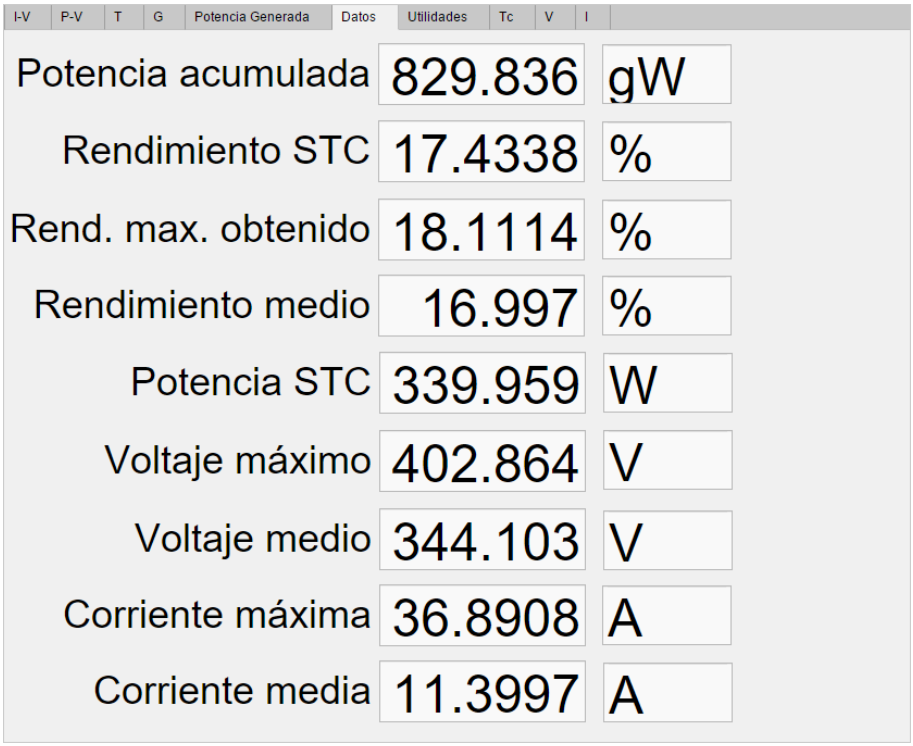
Se generan las siguientes gráficas:

- “I-V”: Curva I-V del módulo en condiciones STC

- “P-V”: Curva P-V del módulo en condiciones STC
- “T”: Gráfica de temperatura ambiental en la localización seleccionada durante el tiempo seleccionado
- “G”: Gráfica de irradiancia global horizontal en la localización seleccionada durante el tiempo seleccionado
- “Potencia generada”: Gráfica de potencia total generada (con pérdidas aplicadas) por la instalación simulada durante el tiempo seleccionado
- “Tc”: Temperatura de célula de los módulos bajo las condiciones ambientales seleecionadas en el tiempo establecido
- “V”: tensión en el punto de máxima potencia de los módulos durante el tiempo establecido
- “I”: corriente en el punto de máxima potencia de los módulos durante el tiempo establecido

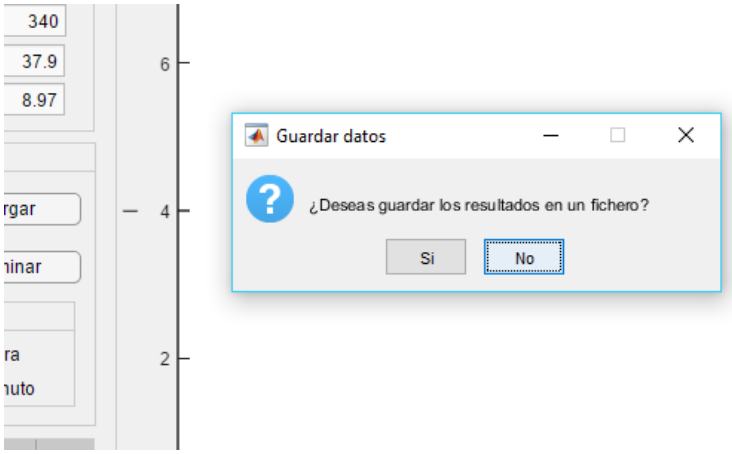
- Datos numéricos

  Ventana con una lista de datos numéricos útiles obtenidos de la simulación.



- Guardado de datos

Una vez terminada la simulación, una ventana emergente pregunta al usuario si se desea guardar los datos generados e la simulación. En caso afirmativo, se abrirá una ventana para seleccionar la carpeta de destino, y una vez seleccionada, se guardará dicho fichero, momento en el que se abrirá una ventana emergente notificando que el fichero se ha guardado de forma exitosa. En caso negativo, se abrirá una ventana emergente notificando que no se han guardado los datos de la simulación.

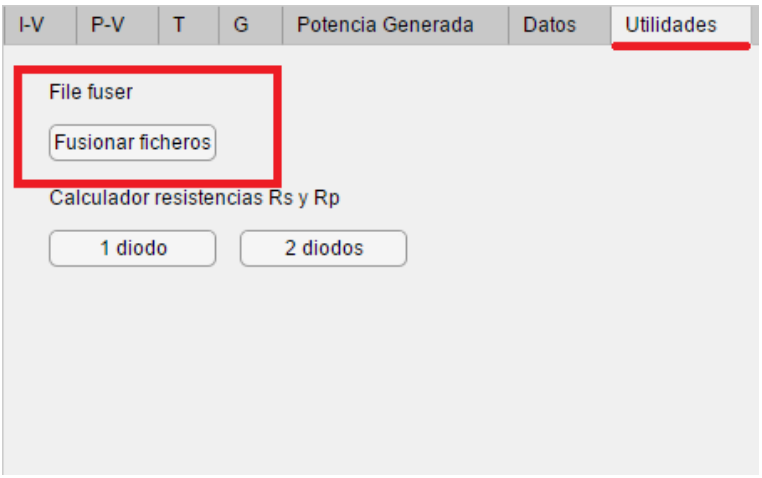


Uso de utilidades

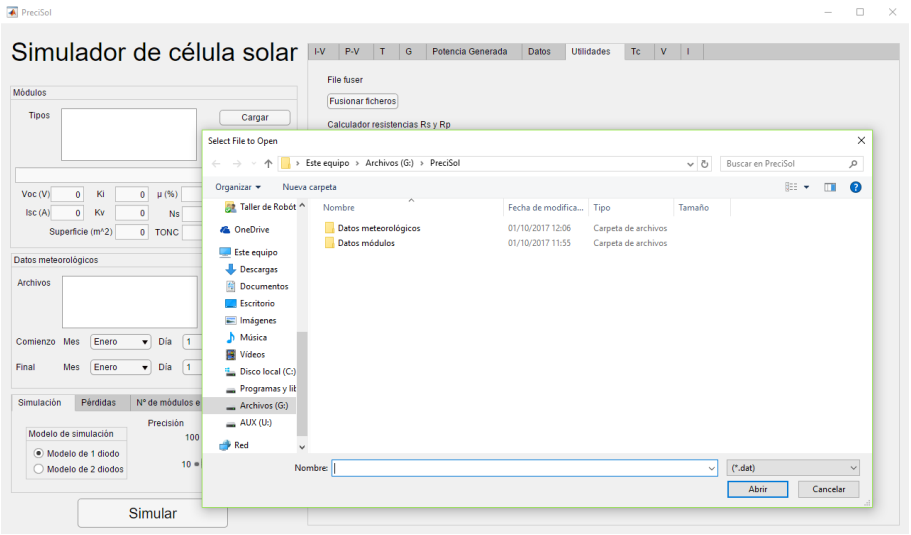
La aplicación viene provista de dos utilidades necesarias para la creación de ficheros de datos de módulo y meteorológicos.

- File fuser:

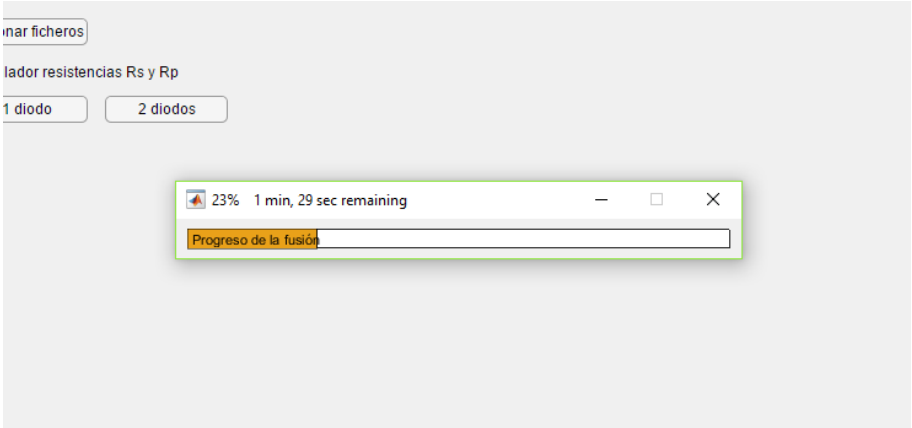
  Utilidad que permite crear un fichero unificado con los valores de irradiancias por minuto con los valores de temperaturas en horas.



Una vez se han obtenido los ficheros de datos meteorológicos en horas y minutos, se procede a la fusión de los mismos: al pulsar el botón se abrirá una ventana en la que se ha de seleccionar primero el fichero “standard”, y después un fichero “standard minute”.



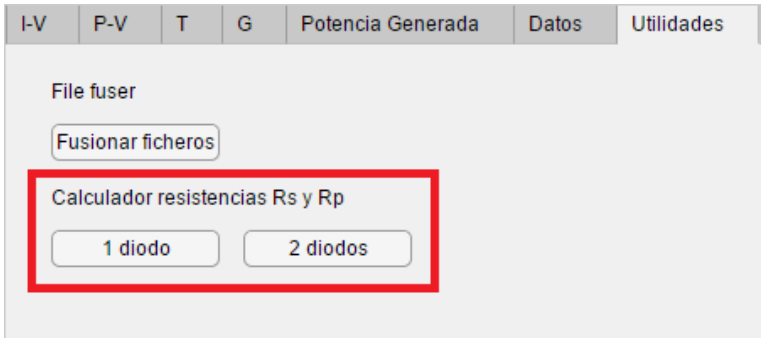
Una vez seleccionados los ficheros en el orden adecuado, la función comienza a fusionar los ficheros.



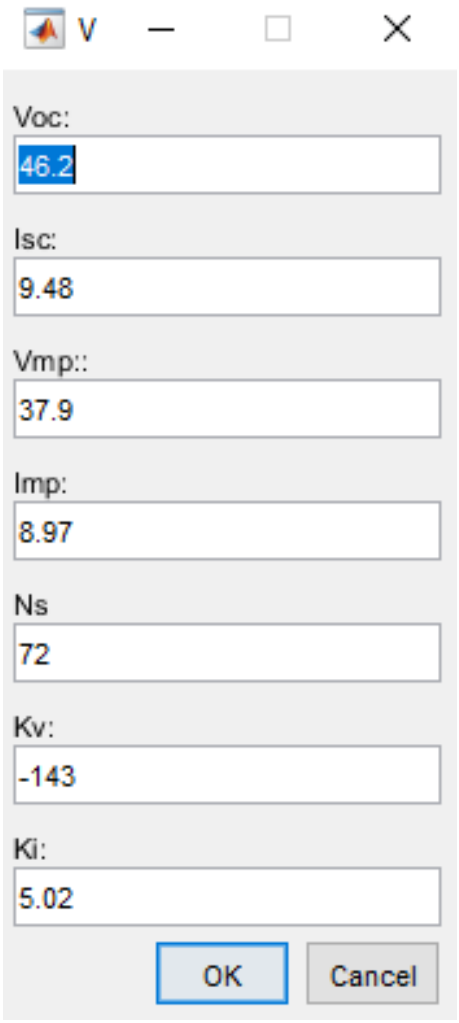
Una vez termine, creará un fichero con el nombre del fichero “standard minute” con el sufijo “_wtemp” en la misma carpeta donde se encuentre éste.

- Calculador de resistencias (modelos de 1 y 2 diodos):

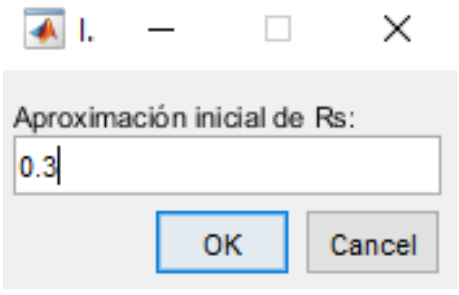
  Utilidad que permite calcular las resistencias en paralelo y serie del circuito equivalente de los modelos de 1 y 2 diodos (según el botón pulsado), dependiendo de los valores introducidos, correspondientes a un módulo determinado.



  Al pulsar el botón del modelo para el cual queremos calcular las resistencias, se abrirá una ventana emergente con cuadros de valores en los que se han de introducir los parámetros del módulo deseado.

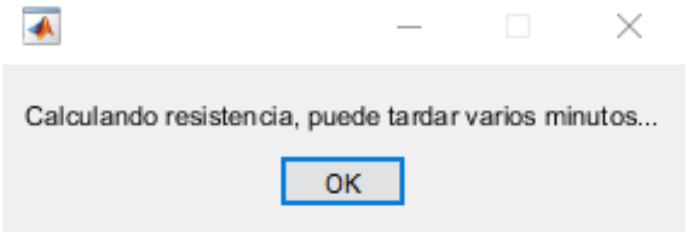


Una vez introducidos todos los valores, se pulsa el botón “Ok”, lo que abre otra ventana emergente con el cuadro de valor para seleccionar un valor de estimación inicial para la resistencia serie del modelo (R_s).



Ahora que se han determinado todos los parámetros para el cálculo, el programa abrirá una ventana emergente para notificar que se están calculando

las resistencias.



Al terminar, después de un tiempo de calculo que dependerá del valor alcanzado y que no puede ser calculado, se abrirá una ventana emergente con los valores de las resistencias obtenidos.