Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

Universidad de Valladolid
Escuela de Ingenierías Industriales

Grado en Ingeniería Electrónica Industrial y Automática

SISTEMA CERRADO CON PRODUCCIÓN ENERGÉTICA DE FORMA HIDRÁULICA

Autor:
García Álvaro, Marta

Tutor:
Arratia García, Óscar
Departamento de Matemática Aplicada

RESUMEN

Este trabajo trata de simular el funcionamiento de una central hidroeléctrica y de un sistema de centrales hidroeléctricas interconectadas. Para la implementación de las simulaciones se ha utilizado el lenguaje de programación Python. Para realizar los cálculos y comparar con valores reales se han modelado variables que afectan a la generación de energía y variables que definen, en cada momento, las necesidades de la población a la que las centrales dan servicio. El objetivo es disponer de una herramienta informática que, con valores reales, nos permita predecir si una instalación hidroeléctrica será capaz de abastecer una demanda prefijada, en qué grado puede paliar los efectos de un período de sequía o si fuese necesario recurrir a otros tipos de energías.

Palabras clave

Modelado de sistemas, energías renovables, central hidroeléctrica, Python, JupyterLab.

Abstract

This work tries to simulate the operation of a hydroelectric plant and a system of interconnected hydroelectric plants. The Python programming language has been used to implement the simulations. Some variables related to the energy generation and to the energy and water needs of the population have been modeled to carry out the computations and to compare the results against real data values. The main goal of this work is to develop a software tool to predict, using real data, wether a hydroelectric plant will be able to supply a predetermined demand, to what extent it can mitigate the effects of a period of drought or if it is necessary to look for other types of energy.

Keywords

Systems modeling, renewable energies, hydroelectric power plant, Python, JupyterLab.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1. INTRODUCCIÓN Y OBJETIVOS

En este trabajo pretendemos hacer un modelado base para la simulación de un sistema de centrales hidroeléctricas que abastece a una población genérica tanto de recursos hídricos como eléctricos.

El trabajo ha sido inspirado por mi experiencia personal, en la que he observado que la calidad de vida de las personas está fuertemente relacionada con los suministros de agua y electricidad y, en particular, me ha preocupado la forma en la que las sequías y el encarecimiento de los recursos afectaban a mi entorno.

La **memoria está organizada** de la siguiente manera:

En los capítulos 3, 4 y 5 se introducen algunos conceptos importantes para el posterior desarrollo de la Memoria. En el Capítulo 3 se habla de los dos recursos que se trata de controlar en el trabajo: por una parte, el agua, indicando los problemas que puede producir y haciendo hincapié en algunos conceptos fundamentales, y, por otro lado, la electricidad, presentando una introducción a las energías renovables y exponiendo la opinión al respecto de la IPCC (Intergovernmental Panel on Climate Change) y el acuerdo de París. En el Capítulo 4 se habla del clima, del cambio climático y de las medidas que las naciones están adoptando para tratar de evitarlo. Además, se enumeran algunos sistemas que se usan en la actualidad para combatir, de manera efectiva, algunos desastres naturales asociados al cambio climático. En el Capítulo 5 se describen los principales conceptos relativos a las centrales hidroeléctricas, incluyendo varias clasificaciones para los distintos tipos de centrales, entre los que se encuentran las centrales fluyentes, de derivación, de regulación y de bombeo, y algunos parámetros importantes para la generación de energía, como son la altura de la central y la frecuencia de salida de la potencia generada. En el Capítulo 6, se hablará de las herramientas informáticas que se han utilizado en la memoria. Estas herramientas son, principalmente, el lenguaje Python, los notebooks de Jupyter y la distribución Anaconda. En este capítulo, además, se realiza una pequeña introducción a las principales bibliotecas de Python empleadas en este trabajo.

El Capítulo 5 y el Capítulo 6 sirven como introducción al Capítulo 7, donde se desarrollan las funciones que modelan cada tipo de central (las centrales fluyentes, de derivación, de regulación y de bombeo) además, se modelan los caudales de entrada (las lluvias y caudales procedente de ríos) y la demanda energética e hídrica a través de distintos métodos: uso de datos reales, modelo senoidal y aproximaciones polinomiales trigonométrica. El capítulo concluye analizando los resultados obtenidos para las centrales diseñadas.

Finalmente, en el Capítulo 8 se presentan las funciones que se encargan de interconectar centrales, el principal objetivo de este trabajo. Se presentan las interconexiones entre centrales que deben estar situadas en el mismo río, pero con

distintas altitudes, con la posibilidad de tener en cuenta el tiempo que transcurre desde que el caudal de salida de la central superior llega hasta la central situada a una altura inferior del mismo río. También se presenta la interconexión por bombeo, en la cual se puede bombear el agua desde un sistema infinito (como podría ser el mar) o se bombea desde un sistema finito (en nuestro caso, un embalse de una central de bombeo o una central de regulación).

Capítulo 2. ESTADO DE LA CUESTIÓN

Este trabajo trata de abrir una perspectiva nueva a la hora de gestionar un sistema de energía eléctrica a base de agua, tratando de aportar una herramienta que ayude a la toma de decisiones y a la generación de hipótesis. Se puede asemejar al tema del que tratan las tesis doctorales (1) y (2), dado que se trata de dar una respuesta a una situación futura, pero, a diferencia de estos trabajos, más extensos, este trabajo de fin de grado se ha focalizado más en construir la herramienta para la creación de hipótesis de un sistema multicentro interconectado que en tratar de dar el resultado óptimo para cada situación.

Capítulo 3. RECURSOS ESENCIALES PARA EL DESARROLLO

Este capítulo trata de los recursos que se pretenden gestionar a lo largo del trabajo: la electricidad y el agua. Comenzaremos hablando del agua y de la importancia que tiene y, seguidamente, se darán algunas definiciones fundamentales para entender el contexto de las simulaciones que posteriormente realizaremos, Por último, se hablará de la electricidad.

3.1 EL AGUA

EL agua es un recurso indispensable para la vida. Todo ser vivo en nuestro planeta la necesita. Las plantas y los animales están constituidos, en una gran proporción, por moléculas de agua. No solo es esencial en su estado líquido y abundante, como la encontramos en los ríos o lago, también lo es humedeciendo el aire o impregnando la tierra para mantener con vida a los seres de nuestro planeta, para mantener la biodiversidad.

En nuestra sociedad actual el agua ya no solo sirve para cubrir las necesidades básicas, por ejemplo, beber o la higiene, como es el caso del resto de los seres vivos. Se usa para limpiar la casa, tener un jardín, para el arte o hasta para la diversión. Su correcta o incorrecta gestión puede producir un gran impacto ambiental, social y económico.

En la actualidad nos encontramos en un momento en el que tienen una gran importancia las decisiones que se tomen sobre medio ambiente y sostenibilidad. Como todos sabemos, nuestro medio ambiente está cambiando: la desertificación del suelo, la contaminación de las aguas, sequías e inundaciones donde antes no se producían. Nuestras decisiones pueden tanto agravar esta situación como contribuir a paliarla.

Las inundaciones y su antagonista, la sequía son uno de los mayores problemas que hay en relación con el agua y el cambio climático (3). Las Inundaciones producen daño económico y destrucción del entorno y la sequía la muerte de fauna y flora, además de malas cosechas y empeoramiento de la calidad de vida de los agricultores. En el ámbito estatal, provocan el encarecimiento de las materias primas y mayor necesidad de la importación de alimentos (4)

La gestión de los recursos hídricos es fundamental para su correcto aprovechamiento. Su explotación, sin una gestión adecuada, puede tener graves consecuencias que, en los casos más extremos, llegan a la desaparición de los acuíferos. Un buen ejemplo de esta situación lo tenemos en **Turquía** (5) (6), donde los agricultores recurren con frecuencia a los acuíferos debido a que es la fuente de obtención de agua más barata. Desde hace ya más de una década, la sobreexplotación de los acuíferos está produciendo hundimientos

en el terreno, como se puede ver en la Figura 1. En este país, ya en el año 2021, se habían contabilizado más de 150 socavones como este.

Figura 1: Socavón en los campos Turquía *(6)*.

En **España** también hay acuíferos con alta explotación y que se encuentran en alto riesgo, como podemos observar en la Figura 2, donde se especifica el estado de explotación de los acuíferos de nuestro país. No hace mucho, se han podido leer noticias sobre uno de estos acuíferos situado en el Parque Nacional de Doñana (7). El uso indiscriminado por los agricultores, como en el caso de Turquía citado anteriormente, y su mala gestión y control por parte de las distintas instituciones, está poniendo en peligro las aguas de este parque, pudiendo acabar secándose o provocando una filtración de las aguas desde el mar (8).

Otro grave acontecimiento es el que sucedió en el Parque Nacional de las Tablas de Daimiel (9), en Ciudad Real, que en 2009 sufrió una gran sequía producida por la sobreexplotación de los acuíferos de los años anteriores y el trasvase del Tajo que se empezó a realizar un año antes. Las Tablas de Daimiel es un parque de tablas fluviales, esto es, un terreno con baja pendiente por donde pasa un río que, en ciertas épocas del año, inunda la zona originando un humedal. Este enclave es un Parque Nacional desde 1973 y una reserva de la biosfera desde 1981 (10).

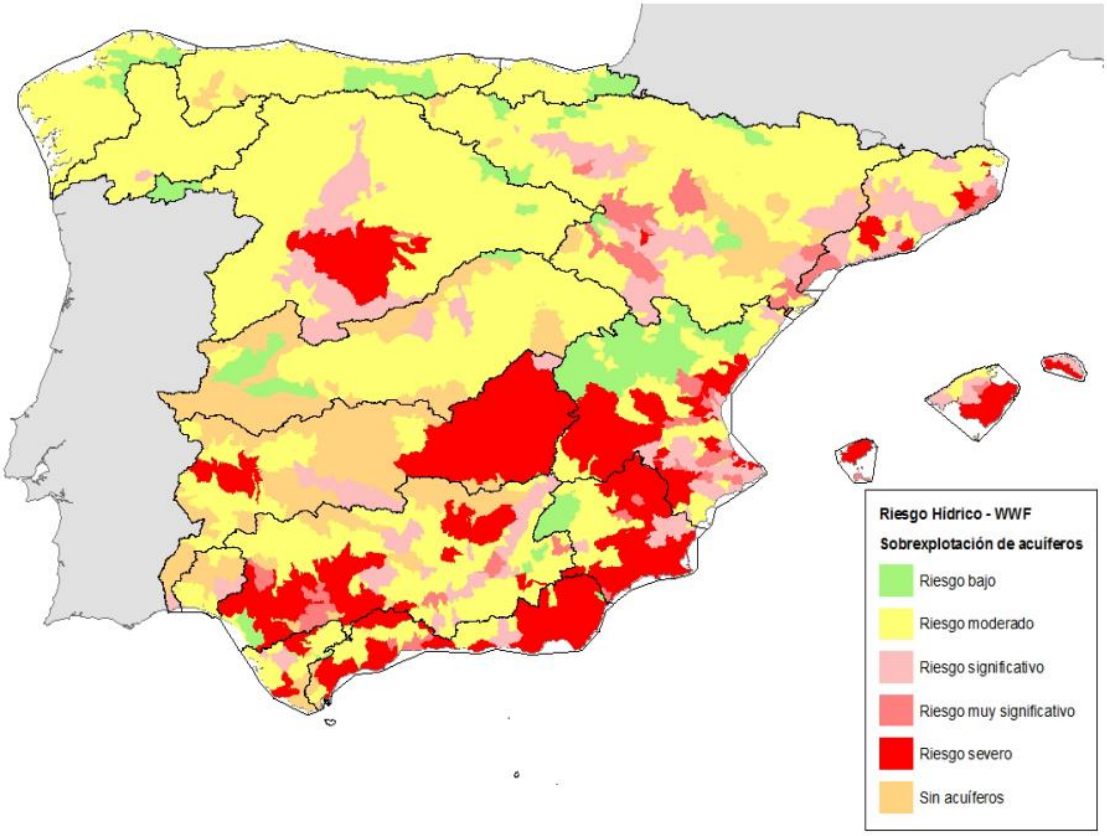


Figura 2: Sobreexplotación de los acuíferos.

3.1.1 Los ríos

En este trabajo los ríos juegan un papel fundamental. Ahora bien, un río ¿qué es exactamente?, ¿qué le hace a un río ser como es y en qué se diferencian unos de otros? A continuación, se presentan las definiciones y conceptos necesarios para responder a estas preguntas con precisión.

La Real Academia Española (RAE) proporciona la siguiente definición para la palabra **río**: "Corriente de agua continua y más o menos caudalosa que va a desembocar en otra, en un lago o en el mar" (11).

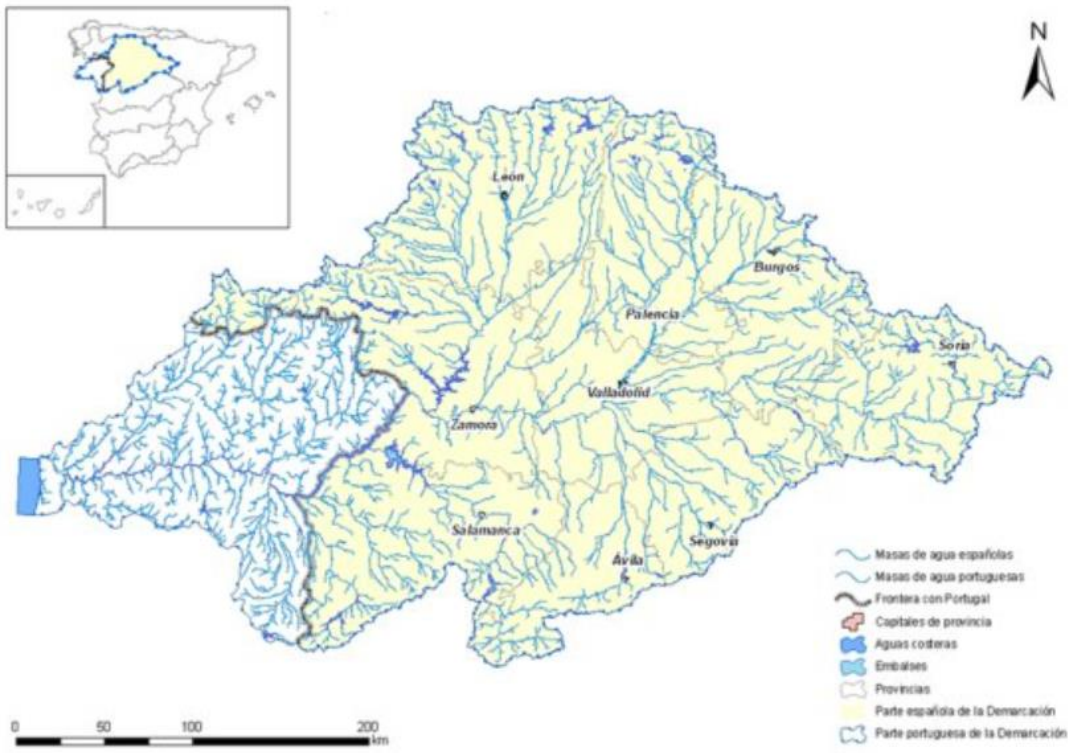


Figura 3: Cuenca hidrográfica del Duero.

El caudal del río puede deberse a tres tipos de procedencia o de **régimen**: pluvial, nival y glacial. Este tipo de procedencia influye en la estabilidad del caudal del río. Así, por ejemplo, los caudales pluviales son muy intermitentes, pudiendo llegar a secarse la cuenca del río en las estaciones más secas, mientras que la procedencia glacial permanece constante no solo durante el año sino a lo largo de años sucesivos. La procedencia nival, por su parte, suele producir un caudal máximo en primavera. Aunque lo más común es que se dé más de una de estas opciones a la vez. Así, por ejemplo, tenemos el río Duero, de régimen pluvio-nival, el río Paraná (América del Sur), de régimen pluvial y el río Rin (Alemania), de régimen nivo-glacial, con sus últimos tramos sumándoles el régimen pluvial de los afluentes.

Por otro lado, está el término **cuenca** que según la RAE es un "Territorio cuyas aguas afluyen todas a un mismo río, lago o mar".

Un aspecto importante de un río es su **cuenca hidrográfica,** que se refiere al camino que toman las aguas en una determinada zona. Según la RAE, una cuenca hidrográfica es una "Superficie de terreno cuya escorrentía superficial fluye en su totalidad a través de una serie de corrientes, ríos y eventualmente lagos hacia el mar por una única desembocadura, estuario o delta" (11). Por ejemplo, toda el agua que cae en la ciudad de Valladolid, que pertenece a la cuenca del Duero, termina llegando a dicho río (excepto, naturalmente, la que se evapore) a través de los afluentes que pasan por la ciudad: el río Pisuerga y el río Esgueva. La zona no se centra solo en una población o en el entorno alrededor de dicho río, sino que se extiende por todos los afluentes del río que desemboca en el mar. Así, continuando con el ejemplo anterior, tenemos que la cuenca hidrográfica del Duero en España sería prácticamente toda la meseta Norte, como se puede ver en la Figura 3. Esta cuenca es la más grande de España.

Muy parecido a lo anterior es lo que se denomina **cuenca fluvia**l de un río, que es el área que recoge todas las precipitaciones que acaban en el curso de dicho río (10).

Sobre los **mares** y **lagos** tenemos, según la RAE, la siguiente definición para mar: "1 Masa de agua salada que cubre la mayor parte de la superficie terrestre. 2. Cada una de las partes en que se considera dividido el mar. Mar Mediterráneo, Cantábrico. 3. Lago de cierta extensión. Mar Caspio, Muerto" (11). Para lago la definición es: "Gran masa permanente de agua depositada en depresiones del terreno" (11).

Ni el mar Caspio ni el mar Muerto tienen conexión con otros mares, son grandes masas de agua en una depresión en las que desembocan bastantes ríos. Estas masas de agua se denominan **lagos endorreicos.** Otro ejemplo relevante de este tipo de lagos lo constituye el lago Don Juan, en la Antártida. Endorreico viene de endorreísmo que según la RAE es un término geográfico que se define como "afluencia de las aguas de un territorio hacia el interior de este, sin desagüe al mar". En estos casos es de aplicación el concepto de **cuenca endorreica**, definido como una extensión de terreno cuya agua no desemboca en los océanos y que no está comunicada con ellos.

La importancia de estos conceptos para las simulaciones que construiremos más adelante radica en que son necesarios para entender algunas de las especificaciones que vamos a establecer en la descripción de los distintos modelos de centrales y embalses.

3.2 LA ELECTRICIDAD

La electricidad, según la RAE, es "1. f. Fís. Fuerza que se manifiesta por la atracción o repulsión entre partículas cargadas, originada por la existencia de electrones y protones.". Etimológicamente viene de eléctrico, que procede "del latín medieval electricus 'ambarino', del latín electrum 'ámbar', y este del griego ἤλεκτρον élektron, y el lat. -ĭcus '-ico'" (11).

La energía eléctrica, aun siendo algo con lo que llevamos viviendo menos de dos siglos, se ha convertido en un elemento indispensable para el ser humano y en un recurso fundamental para la mejora de la calidad de vida.

Los primeros datos históricos que tenemos sobre la electricidad datan de antes del comienzo del calendario cristiano. En el antiguo Egipto se percataron del poder de los rayos que se generaban en las tormentas y de las descargas que producían algunos peces del Nilo. Tales de Mileto, 600 años antes de Cristo, fue quien descubrió que el ámbar, después de frotarlo contra seda, atraía la paja y las pelusas. Pero no fue hasta el siglo XVI cuando William Gilbert realizó los primeros experimentos magnéticos y eléctricos. Desde el siglo XVII hasta el siglo XIX se siguieron realizando descubrimientos sobre la naturaleza de la electricidad: George Simón Ohm formuló las leyes de Ohm, que relacionaban el voltaje, la corriente y la resistencia, Coulomb desarrolló la Ley de Fuerzas Electrostáticas, Benjamín Franklin inventó el pararrayos, la ley de Ampere[1] que posteriormente Maxwell generalizó y, ya en el siglo XIX, Thomas Alva Edison y Nikola Tesla desarrollaron los primeros mecanismos de trasmisión continua y alterna de la electricidad, respectivamente. Así, en el siglo XIX se extendió su uso para la iluminación de las ciudades y de los hogares, dando comienzo consigo a la Segunda Revolución Industrial

La electricidad se produce mediante fuentes que se pueden clasificar en dos tipos: renovables y no renovables. En general, son preferibles las renovables por la independencia económica que otorgan al país que las utiliza, dado que no se necesita importar ni combustibles ni electricidad. Por otro lado, las fuentes no renovables, como es el caso del petróleo, el carbón y el uranio (en centrales nucleares), están constituidas por materias primas cuya obtención es contaminante y su compra puede complicarse debido a la inestabilidad política de muchos de los países que las poseen. Además, la producción de energía mediante estas materias primas también genera contaminación, ya sea del aire, radiactiva o de otros tipos.

3.2.1 Energías renovables

Las energías renovables constituyen un tópico sobre el que últimamente se habla con frecuencia. A continuación, se mostrarán algunas de las reflexiones que realizan algunas organizaciones internacionales al respecto.

El Grupo Intergubernamental de Expertos contra el Cambio Climático, conocido como IPCC (12) por sus siglas en inglés (Intergovernmental Panel on Climate Change), fue fundado en 1988 y se encarga de asesorar a las naciones sobre el cambio climático científico y técnico y también su efecto social y económico. Este grupo de expertos ha "observado cambios sin precedentes en el clima de nuestro planeta. Para evitar que esto vaya a más, la solución es reducir la emisión de gases de efecto invernadero, procedente de las actividades de combustibles fósiles'' (13). En la generación de energía eléctrica, objeto del que hablamos en este trabajo, serían el gas natural, el carbón y el petróleo y sus derivados.

[1] que relaciona el campo magnético con la corriente eléctrica estacionaria

En (14) encontramos el acuerdo de París, un acuerdo vinculante que busca no superar el aumento global de temperatura de 2 grados. Para alcanzar este objetivo propone buscar el equilibrio entre las emisiones y la absorción natural de los gases de efecto invernadero.

Las formas de producir energía eléctrica en España son las que se muestran en la Figura 4, que desglosa la proporción de energía total que genera cada una. En dicha figura las fuentes renovables aparecen desplazadas respecto al centro. Se puede observar que las renovables suponen un alto porcentaje del total y, en particular, destacan las de origen hidroeléctrico. Tal como dice (15), podemos obtener más energía solar, térmica y eólica de la que necesitamos. El único problema que tenemos es que son energías intermitentes. Es decir que no podemos obtenerlas en cualquier momento, pero este problema se puede solucionar con sistemas de almacenamiento de energía.

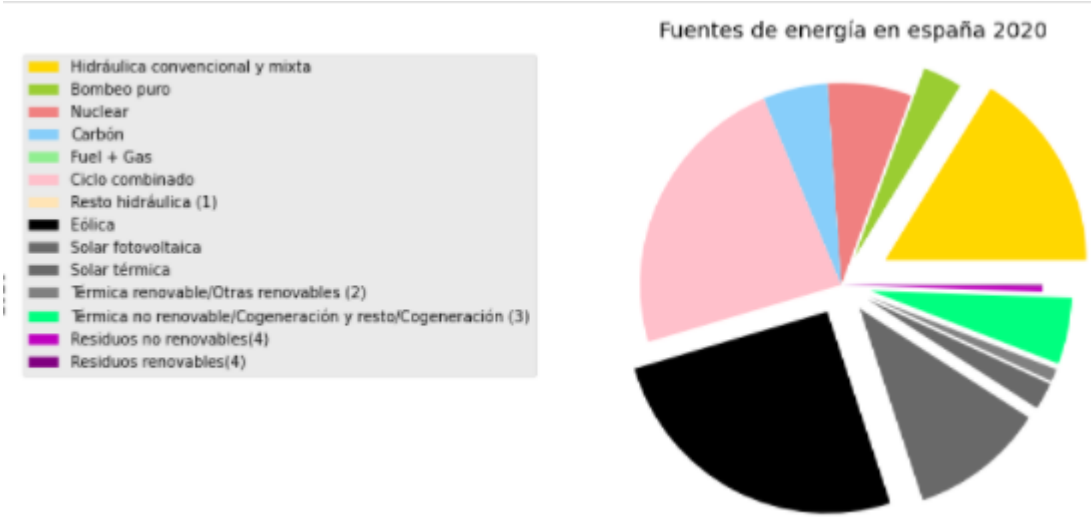


Figura 4: Fuentes de energía en España

Datos de la gráfica procedente de (46)

Ahora bien, ¿Qué entendemos por energías renovables?, la definición que nos da Iberdrola es:

"Las **energías renovables** son aquellas que provienen de fuentes naturales inagotables, bien porque contienen una enorme cantidad de energía —como el Sol o el viento— o porque son capaces de regenerarse en poco tiempo —como la biomasa—" (16).

A continuación, se presentan dos definiciones para las energías no renovables:

"La **energía no renovable**, es aquella en la que los recursos de suministro son limitados. El suministro proviene de la propia Tierra y, debido a que tarda millones de años en desarrollarse, es finito" (17).

"Las **energías no renovables** son aquellas que provienen de fuentes de las que hay una cantidad limitada. Así que, cuando se terminan, no se pueden reponer" (18).

Es decir, entendemos como energías no renovables a las formas de generación en las que los recursos pueden agotarse porque, generalmente, se consumen más rápido de lo que se producen. O, en otras palabras, las energías no renovables son las que proceden de una fuente de energía agotable.

Finalizamos el capítulo con una cita que pone de manifiesto el fuerte grado de dependencia de las fuentes de energía agotables que, desgraciadamente, tiene la sociedad actual: “Los combustibles fósiles son una fuente de energía que procede de la descomposición de materia orgánica de animales, plantas y microorganismos, y cuyo proceso de transformación tarda millones de años. Se clasifican en tres tipos -petróleo, carbón y gas natural-, y según las Naciones Unidas, comprenden el 80% de la demanda actual de energía primaria a nivel mundial” (19).

Capítulo 4. CAMBIO CLIMÁTICO Y RESPUESTA A DESASTRES NATURALES

El cambio climático causado por el hombre ya es un hecho como dicen algunas organizaciones intergubernamentales (12). En este capítulo se hablará del cambio climático natural, del producido por el hombre y de los mecanismos de reacción que han desarrollado algunas naciones para amortiguar el efecto de los desastres naturales y neutralizar o ralentizar el cambio climático (20).

4.1 EL CAMBIO CLIMÁTICO

En (21) podemos encontrar la siguiente definición: "El **clima** se puede entender como el estudio estadístico del tiempo". En ese mismo trabajo se habla del clima y del cambio climático que se está produciendo ahora. Que el clima de la Tierra se modifique a lo largo de los siglos es algo natural y que siempre ha pasado. Para predecir el clima de aquí a unos años los meteorólogos necesitan manejar un gran número de variables, las cuales no son todas predecibles. De forma natural, sin intervención del ser humano, al clima le afecta la variación de la posición de la Tierra a lo largo de su órbita alrededor del Sol, caracterizada por su excentricidad, la oblicuidad del eje de rotación terrestre y la precesión de dicho eje. Otras variables que afectan son: la deriva de los continentes, las corrientes oceánicas, la distribución planetaria de los continentes, los cambios en la actividad solar y las alteraciones en la composición atmosférica por causas naturales como la erupción volcánica o el impacto de cuerpos celestes.

Pero, si ha pasado siempre, ¿qué es lo preocupante? Lo preocupante es la velocidad con la que esto está ocurriendo en los últimos años y su correlación con los cambios de la composición atmosférica, cambios causados por la producción de CO_2 y otros gases que emitimos. El aumento de la temperatura asociado a estos cambios supone la desertificación del planeta y el aumento del nivel del mar a causa de la desglaciación de los polos.

La IPCC en su quinto informe (22), publicado en 2014, estableció que el aumento de temperatura en el periodo 1880-2012 había sido de 0,85 °C. Declaró que la zona más afectada era la Antártida. Su sexto informe saldrá para finales del año 2022 (23).

En las últimas décadas se han firmado acuerdos entre países y entre organizaciones para evitar o minimizar el cambio climático y tratar de preservar el medioambiente, como es el caso de la Conferencia de las Naciones Unidas sobre Medio Ambiente y el Desarrollo (CNUMAD) de 1992 en Río de Janeiro (Brasil). Aquí se estableció la Agenda 21, la Declaración sobre el Medio Ambiente y el Desarrollo y la Declaración de Principios de Bosques (20).

4.2 SISTEMAS DE PREVENCIÓN DE INUNDACIONES A NIVEL MUNDIAL

Para la prevención de los desastres naturales asociados a las inundaciones la Unión Europea y otros países han desarrollado distintos mecanismos con el fin último de minimizar los efectos que provocan.

En **Japón** se ha diseñado un método basado en tanques subterráneos capaces de absorber el agua de las inundaciones para paliarlas con mayor rapidez. Su desarrollo se ha visto impulsado por ser un país que sufre anualmente grandes tifones cuyo número ha aumentado en las últimas décadas. Esta tecnología se ha puesto en práctica en Tokio, que está cruzado por nueve ríos que, con cierta frecuencia, llegan a desbordarse en los meses de mayores precipitaciones (24), (25).

Figura 5: Tanque subterráneo en Japón (24)

En **China,** desde 2015, tratan de implantar un nuevo modelo de ciudades llamado ciudades esponja. Estas ciudades cuentan con un diseño que permite expandirse al río y maximiza la absorción de las aguas en caso de inundaciones (26), (27).

En **Europa** se planteó crear zonas donde, en caso de riadas, el exceso del caudal tenga espacio por donde extenderse sin producir daños y pueda ser absorbido por la tierra fácilmente. Un ejemplo de esta técnica lo constituye el jardín del Turia en Valencia, cuyo parque en el cauce puede evitar que la inundación llegue a los comercios y las viviendas.

Capítulo 5. PRESAS Y CENTRALES HIDROELÉCTRICAS

5.1 SITUACIÓN DE LAS PRESAS EN ESPAÑA

Dado que una de las fuentes de inspiración de este trabajo ha sido la situación de la generación de energía y de las centrales hidroeléctricas en España, vamos a empezar este capítulo hablando sobre este asunto.

España, a pesar de no ser uno de los países con mayor extensión de la Tierra, se encuentra entre los 5 países con más embalses, solo por detrás de China, Japón, Estados Unidos y Rusia. En concreto, en nuestro país hay más de 350 presas, en las cuales se pueden almacenar 54000 hectómetros cúbicos de agua.

Las primeras presas datan de los tiempos de los romanos. Entre ellas está la presa de Riofrío, en Segovia, donde se encuentra la toma de agua para el famoso acueducto de la ciudad y la presa de Muel, en Zaragoza, encima de la cual está situada la iglesia llamada de la Virgen de la Fuente, conocida por la decoración realizada por Goya en 1772. Posteriormente, no se produjo ningún cambio significativo hasta el siglo XX, cuando su número experimentó un gran aumento. La primera central hidroeléctrica en España se construyó en 1899 en Otero de Rey, a 10 km de Lugo (28) . Como ejemplo más reciente y novedoso estaría la central hidroeléctrica que se está construyendo en Canarias: un sistema de generación de energía cuya característica diferencial es la de estar intrínsecamente conectado a un sistema de depuración y filtrado del agua del mar. Este sistema suministrará agua al embalse superior y permitirá tener más recursos disponibles en cuanto a necesidades energéticas e hídricas se refiere.

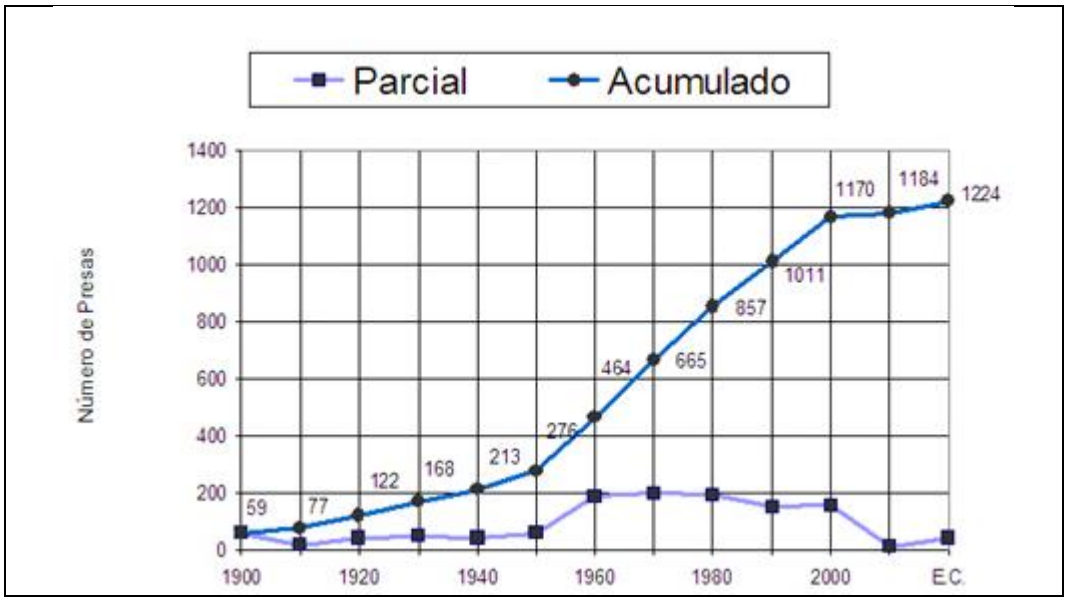


Figura 6: Evolución de número de presas en España *(29)*

En la Figura 6 podemos observar dos tendencias muy diferenciadas a lo largo del siglo XX: la de la primera mitad, con un crecimiento moderado del número de presas, y la de la segunda, periodo iniciado en la dictadura y continuado por los gobiernos posteriores, donde se acelera considerablemente la construcción de nuevas presas.

Factores importantes que han impulsado la construcción de embalses en España han sido: su clima mediterráneo, que la mayoría de los caudales de los ríos son de procedencia pluvial o pluvio-nival y, además, estos ríos poseen, como característica adicional, una temporada de estío. Así pues, Los embalses son una gran herramienta para mantener reservas de agua.

5.2 TIPOLOGÍA DE CENTRALES HIDROELÉCTRICAS

En este trabajo nos centraremos en las centrales hidroeléctricas. En concreto, en este apartado describiremos qué son, cómo son y otros datos relevantes.

Según la RAE, una **presa** se define como: "Muro grueso de piedra u otro material que se construye a través de un río, arroyo o canal, para almacenar el agua a fin de derivarla o regular su curso fuera del cauce" (11).

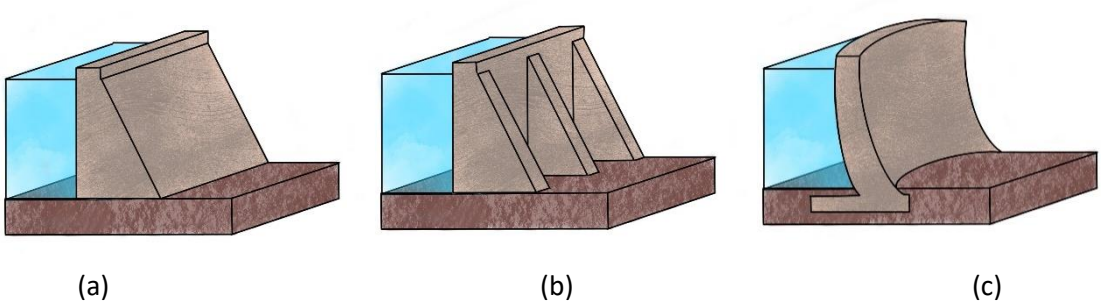
También según la RAE, un **embalse** es un "Gran depósito que se forma artificialmente, por lo común cerrando la boca de un valle mediante un dique o presa, y en el que se almacenan las aguas de un río o arroyo, a fin de utilizarlas en el riego de terrenos, en el abastecimiento de poblaciones, en la producción de energía eléctrica" (11).

Los países que más energía consumen son también grandes productores de energía mediante las centrales hidroeléctricas (30). Algunos ejemplos que ilustran esta situación son: las presas de las tres gargantas en China, cuya producción es de 22.500 MW, la presa de Taipu, entre Brasil y Paraguay, de 14.000 MW, la represa de Grand Coulee, en Estados Unidos, de 6.809 MW o la presa de Sayano-Shushenskaya, en Rusia, que produce 6.400 MW.

Hay varias formas de clasificar las centrales hidroeléctricas. En este apartado nos centraremos en dos clasificaciones: según el material con el que están construidas y según la relación existente entre el caudal que genera la energía (el que pasa por el generador) y el caudal natural del entorno:

Según los materiales con los que está construida tenemos:

1. Las presas de hormigón (31) (32), que a su vez pueden ser:

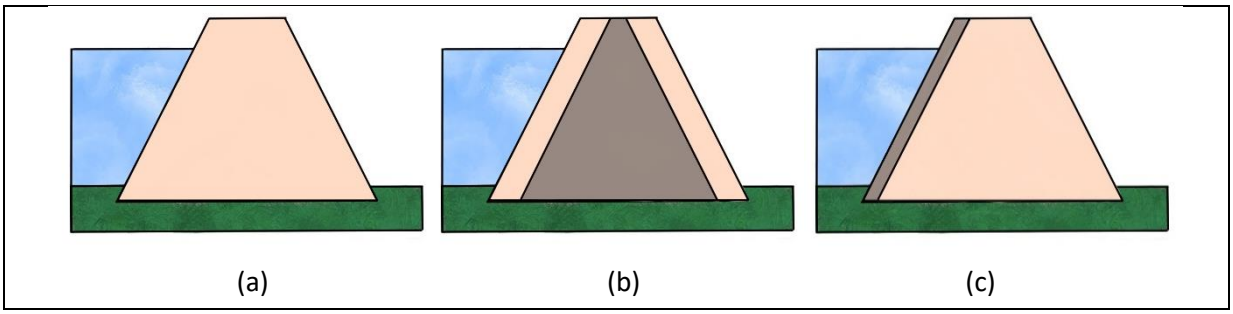


(a) (b) (c)

Figura 7: Clases de presas de hormigón: a) gravedad b) contrafuerte c) arco-bóveda

- **Presas de gravedad** (Figura 7.a): son presas de gran tamaño diseñadas para soportar la presión del agua y cuya construcción permite que cada sección de la presa sea estable e independiente del resto.
- **Presas aligeradas o de contrafuertes** (Figura 7.b): la presa de contrafuerte es un tipo de presa que, por su construcción, usa menos material gracias a, como su nombre indica, los contrafuertes que evitan que la presión del líquido desplace la presa.
- **Presas de arco** (Figura 7.c): este tipo de presa permite transmitir la fuerza de empuje a los lados laterales en los que se asienta. Este tipo de presas se colocan en valles estrechos.

2. Las presas de materiales sueltos (31) (32), que a su vez se subdividen en:



(a) (b) (c)

Figura 8: Presas de materiales sueltos (a) homogénea (b) de núcleo (c) de pantalla

- **Presas homogéneas** (Figura 8.a): Este tipo de presas emplean un material impermeable en la totalidad de su cuerpo. Este puede ser arcilla.
- **Presas de núcleo** (Figura 8.b): Constituidas por una parte externa de materiales permeables, denominado espaldones, y el núcleo. El núcleo está hecho con materiales impermeables y es tan alto como alta es la presa.
- **Presas de pantalla** (Figura 8.c): Esta presa está constituida casi en su totalidad por materiales permeables excepto una pantalla situada en el lado que da al cúmulo de aguas.

Dependiendo de la relación del caudal podemos diferenciar (33):

1. Las **centrales fluyentes**: donde su caudal es el mismo que el caudal del río.

2. Las **centrales de derivación**: cuyo caudal no tiene por qué ser el mismo que el caudal del río
3. Las **centrales de regulación** o de agua embalsada: Son las centrales que pueden proporcionar energía cuando se requiera y no solo dependiendo del caudal del río.
   - Las **centrales de bombeo**: las centrales de regulación, a su vez pueden ser de bombeo cuando se puede recircular el agua hacia arriba en momentos donde la demanda energética o la situación lo requiera.

En este trabajo utilizaremos solamente la clasificación de presas basada en el caudal, es decir crearemos modelos para las centrales fluyentes, de derivación y de regulación.

5.3 FUNCIONAMIENTO DE LAS CENTRALES HIDROELÉCTRICAS

Para explicar el funcionamiento de una central hidroeléctrica hay que tener en cuenta que la frecuencia suministrada a la red eléctrica depende de la velocidad de giro de las turbinas. El caudal que pasa por el generador y la altura del salto del agua son factores de los que depende la potencia que suministra el generador. Este caudal se puede controlar en los casos de las centrales de regulación y bombeo, a diferencia de lo que ocurre en las fluyentes y de derivación, donde no hay ningún tipo de control sobre dicho caudal.

Aunque la frecuencia de la potencia dependa del giro de la turbina, se puede modificar cambiando la relación entre el giro de la turbina y el giro del generador. Esto es lo que hace posible en centrales de caudal no regulable ajustar la frecuencia de potencia de salida.

Capítulo 6. HERRAMIENTAS INFORMÁTICAS USADAS EN LA MEMORIA

6.1 PYTHON

Cuando hablamos de Python nos referimos a uno de los lenguajes de programación dinámica más utilizado hoy en día. Se trata de un lenguaje de propósito general, esto significa que su utilización tiene múltiples fines como, por ejemplo, cálculos matemáticos, manejo y captura de datos, creación de aplicaciones…

La propiedad intelectual de este lenguaje pertenece a la fundación sin ánimo de lucro Python Software Foundation. Su misión es promover, proteger y seguir mejorando el lenguaje de programación. Como se ha indicado más arriba, Python es uno de los lenguajes de programación más empleados en el mundo debido, entre otras causas, a su rápida curva de aprendizaje y a su carácter libre (existe un enorme catálogo de librerías para este lenguaje que son de libre acceso).

Es interesante señalar que Python es un lenguaje de tipo scripting, esto quiere decir que la ejecución de los programas escritos en este lenguaje no requiere su compilación, lo que simplifica notablemente las tareas de programación. Sus inicios se remontan a las décadas de los años 80 y 90, cuando el programador holandés Guido Van Rossum trató de elaborar un lenguaje fácil de aprender y de implementar. Por otra parte, el carácter interpretado de Python hace que los programas escritos en este lenguaje sean, habitualmente, más lentos que los escritos en lenguajes compilados. No obstante, para hacer frente a este problema existe la variante CPython, que proporciona mejores rendimientos.

La primera versión operativa de Python fue la 0.9, publicada en 1994. Tras ella, llegó la versión 1.0, que incluía nuevas herramientas de programación funcional, como map y lambda. La versión, 1.6 tuvo algunos problemas con el tipo de licencia bajo la que se distribuía, hasta que la Free Software Foundation cambio Python a una licencia de software libre que lo hizo compatible con GPL[2] . La versión 2.0, publicada en octubre de 2000, incorpora la generación de listas, una de las características más importantes de este lenguaje de programación, y la posibilidad de hacer referencias cíclicas. La versión más moderna de Python corresponde a la serie 3.x, iniciada en 2008, resuelve los principales fallos de diseño detectados en las versiones anteriores y ha ido acumulando formas nuevas y redundantes de programar una misma tarea. En el momento de escribir esta memoria, la última versión publicada es la 3.10.4.

[2] GPL es una licencia desarrollada para la protección de softwares liberados

En mi caso, aprendí a programar con Pascal y poco después aprendí C++ y las diferencias con Python son bastante significativas. Pascal influyó en muchos lenguajes de programación actuales como C o Java, si bien su uso en estos momentos es muy reducido. Desde un punto de vista práctico, Pascal es un lenguaje bastante engorroso en comparación con las alternativas disponibles en la actualidad: al ser un lenguaje fuertemente estructurado, es necesario declarar al principio tanto las funciones como las variables y, en particular, estas últimas no pueden cambiar de tipo de dato. Además, la curva de aprendizaje de Pascal es mucho más lenta en comparación con la de Python. Por otra parte, es muy destacable la mayor velocidad de ejecución de C++, pero su gran problema es que no tiene la posibilidad de compilar por partes además de la menor cantidad de bibliotecas libres y, aunque también existe y es abundante, la cantidad de información disponible en internet para este lenguaje es mucho menor que la que se puede encontrar para Python.

La facilidad de aprendizaje de Python y su legibilidad hace que, en mi opinión, sea el lenguaje que resultaría más fácil aprender empezando desde cero. No obstante, en este trabajo la elección de Python no se basa tanto en su facilidad de aprendizaje como en la gran herramienta que suponen los cuadernos de Jupyter, de los cuales se hablará más adelante, y cuyo lenguaje de programación nativo es, precisamente Python. Un incentivo adicional para la elección de Python es la gran cantidad de recursos disponibles para la realización de cálculos complejos. Para finalizar y, a título personal, quiero señalar las facilidades que proporciona Python a la hora de depurar el código.

El lector interesado puede encontrar en (34) todo tipo de información acerca de Python.

6.2 MÓDULOS DE PYTHON UTILIZADOS

En esta sección vamos a describir, brevemente, los principales paquetes de Python, más allá de los módulos disponibles en la librería estándar, que se han utilizado en este trabajo.

6.2.1 NumPy

NumPy es una librería muy potente a la hora de realizar cálculos matemáticos y, de hecho, es la librería de referencia en Python para el cálculo numérico en los ámbitos técnico y científico. En Python podemos encontrar otras librerías para los cálculos, pero, en general tienen un alcance más limitado o, como es el caso de **SymPy**, están más orientadas hacia el cálculo simbólico.

La librería NumPy proporciona una gran cantidad de útiles matemáticos entre los que se encuentran funciones, constantes matemáticas y varios tipos de construcciones para el almacenamiento de datos. En particular, una de las construcciones más útiles de NumPy son los denominados arrays multidimensionales (objetos de tipo ndarray) que incorporan un gran número de utilidades que facilitan el trabajo con grandes volúmenes de información numérica. En este trabajo serán de gran importancia este tipo de arrays, ya que servirán para gestionar los datos del modelo de demanda, de caudales o los datos de salida.

En (35) se puede encontrar información adicional sobre esta biblioteca.

6.2.2 Matplotlib

Matplotlib es una librería para crear distintos tipos de gráficos en Python, principalmente en 2D, aunque también es posible generar imágenes 3D. En este trabajo lo que vamos a usar, principalmente, es el submódulo **pyplot**. Este módulo nos aportará las funciones necesarias para la representación de los datos obtenidos. Una característica de pyplot es que su uso se asemeja a la forma en la que se representan gráficamente los datos en MATLAB, lo que supone una ventaja para los usuarios que conocen este sistema.

Utilizaremos esta librería para analizar los datos de manera visual lo que permitirá interpretarlos de una manera sencilla.

Se puede encontrar información adicional sobre esta biblioteca en (36).

6.2.3 Pandas

Pandas es un módulo potente y flexible para el tratamiento de datos. En este trabajo las funciones más destacadas que vamos a usar son las que facilitan la importación de datos procedentes de archivos csv o Excel, entre otros, convirtiéndolos en dataframes y la obtención de máximos, mínimos y cantidades de tipo estadístico como medias y variancias. Con estos objetos vamos a diseñar los modelos y, posteriormente, los utilizaremos para comparar los valores del modelo con los valores reales

En (37) se pueden encontrar más datos sobre esta librería.

6.3 JUPYTER

Jupyter es un proyecto de código abierto que nace en 2014 como continuación del proyecto IPython, el cual está centrado en proporcionar una potente shell interactiva para la programación con Python y diversas herramientas para la visualización de datos. El proyecto Jupyter ha aprovechado toda la experiencia adquirida en la programación de IPython para diseñar un potente entorno de desarrollo de software basado en tecnología web y que, además, se puede usar con múltiples lenguajes, no solo Python. De hecho, el nombre del proyecto, inspirado en las anotaciones astronómicas de Galileo relativas al planeta Júpiter, también hace referencia a los tres lenguajes de programación a los que inicialmente daba soporte: **Ju**lia, **Py**thon y **R**. En la actualidad existen docenas de kernels, cada uno de los cuales hace posible la utilización de un lenguaje de programación en los entornos proporcionados por Jupyter (38).

El uso de Jupyter se basa en la utilización de unos archivos denominados cuadernos (notebooks en inglés) en los que es posible combinar, de forma muy sencilla, código ejecutable con texto y materiales gráficos. Los cuadernos de Jupyter facilitan la creación de documentos interactivos que se puede utilizar para ilustrar cómo funciona un programa mediante la inclusión de textos, enlaces e imágenes explicativas. También es posible utilizarlos para crear herramientas informáticas programadas con el fin de

resolver problemas concretos y este es, precisamente, el uso que se les dará en este trabajo.

En conclusión, la combinación de las capacidades matemáticas que proporcionan las librerías de Python con las posibilidades ofrecidas por los cuadernos de Jupyter proporciona una de las mejores herramientas existentes en la actualidad para desarrollar la programación matemática y la ejecución paso a paso que vamos a necesitar en este trabajo. De hecho, la calidad del entorno facilitado por el proyecto Jupyter es comparable, cuando no superior, a la proporcionada por sistemas como Matlab, Maple o Mathematica, con las ventajas adicionales de ser un software gratuito y de código abierto.

En (39) se puede encontrar más información sobre el proyecto Jupyter.

6.3.1 JupyterLab

Una de las finalidades del proyecto Jupyter es la de proporcionar una interfaz interactiva y multimedia para trabajar con el lenguaje de programación Python. Como se ha indicado anteriormente, la solución se basa en la utilización de archivos denominados "cuadernos" y la primera interfaz para trabajar con ellos, que data de 2015, se denominó "Jupyter Notebook". Esta interfaz proporcionaba utilidades básicas para la edición de los cuadernos, pero solo permitía abrir uno en cada pestaña del navegador web empleado. En los años posteriores se ampliaron las capacidades de Jupyter Notebook, en unos casos de forma nativa y en otros a través de extensiones programadas por miembros de la comunidad de usuarios.

La experiencia adquirida en la mejora de Jupyter Notebook condujo a la creación de una interfaz mucho más potente que se denominó "JupyterLab", cuya primera versión estable se anunció a principios de 2018 y que es la que actualmente se sigue desarrollando. En el momento de escribir este trabajo la versión más actual era la 3.4.2.

JupyterLab se diseñó desde cero con el objetivo de facilitar la extensibilidad del sistema, lo que ha permitido construir una interfaz extremadamente flexible en la que se puede combinar el trabajo simultáneo sobre varios cuadernos, consolas, ficheros con código fuente y otros objetos. Además, JupyterLab incorpora, de forma totalmente integrada en la interfaz, herramientas adicionales tales como el navegador de ficheros, un gestor de extensiones, un navegador de documentos y un depurador de código. Todas estas utilidades serán de gran ayuda para la creación de los cuadernos de Jupyter que generaremos en este trabajo.

El lector interesado puede encontrar en (40) información adicional sobre JupyterLab.

6.4 ANACONDA

La instalación de Python y Jupyter se puede realizar en todas las plataformas habituales (Windows, Mac, Linux) de varias formas diferentes. Cerraremos este capítulo con unos breves comentarios sobre la instalación basada en "Anaconda".

Anaconda es un paquete de software, creado y distribuido por la compañía del mismo nombre, que contiene todos los ingredientes necesarios para crear entornos técnicos de computación. En particular, incorpora las versiones de Python y de Jupyter más actualizadas. Desde el punto de vista práctico basta con descargar, desde la página web de Anaconda (41), el fichero correspondiente al sistema operativo que se desee y, a continuación, realizar la instalación de la manera habitual. Anaconda proporciona herramientas para mantener actualizados los paquetes que incluye o, si fuera necesario, instalar otros nuevos.

Los instaladores de Anaconda e información adicional se pueden encontrar en (41).

Capítulo 7. CENTRAL HIDROELÉCTRICA INDIVIDUAL

En este capítulo hablamos sobre el diseño de distintas clases de Python que nos ayudarán a simular cada central (7.1, 7.2, 7.3). La estructura elegida para la programación de estas clases nos facilitará, en el Capítulo 8, la interconexión de distintas centrales. Además, también se han desarrollado unos modelos de los caudales de entrada y de la demanda para poder ver el comportamiento de cada tipo de central en distintas situaciones (7.4, 7.5, 7.5.2).

7.1 MODELADO DE LAS CENTRALES INDIVIDUALES

Es este apartado vamos a explicar cómo se han programado las funciones principales, detallando su estructura. El sistema que describe cada central se ha analizado tratándolo de forma discretizada en la variable correspondiente al tiempo, y para realizar las simulaciones se emplean distintas clases y funciones.

7.1.1 Descripción de las clases utilizadas

Vamos a usar varias clases[3] para manejar la gran cantidad de atributos que se emplean en las simulaciones:

- **Presa**: en esta clase están implementados todos los atributos que definen la central, tales como la base, altura mínima, altura máxima, turbina (que a su vez es otro objeto), etc.
- **Turbina**: en esta clase se agrupan los atributos referidos a la generación de energía. En ella encontramos la función que modifica el caudal de la central, el caudal máximo y mínimo y el rendimiento.
- **Bomba:** esta clase posee todos los atributos que definirán los parámetros necesarios para el bombeo de la central reversible.
- **Aliviadero**: en esta clase se encuentran los parámetros necesarios para el modelado del aliviadero.

[3] Estas clases están desarrolladas en el módulo herramientas_presas de esta Memoria, situado en el

Anexo 6.2 Módulo de Herramientas Presas.

- **Necesidades**: en esta clase se encuentran las listas de la demanda energética y de agua.
- **Presa_Datos**: en esta clase se encuentran los atributos resultantes de la simulación y las funciones que calculan sus valores.

7.1.2 Cálculo de la potencia generada

Uno de los aspectos más importantes en el modelado de una central es el de la potencia generada, que está interrelacionado con la altura de la central y el caudal del generador.

Para aproximar el caudal y simplificar los cálculos se ha considerado que en intervalos temporales de una hora el caudal permanece constante.

Para modelar el sistema consideramos la ecuación de la energía de un flujo estacionario (ecuación1) (42):

$$\left(\frac{p_1}{\rho g}+\frac{V_1^2}{2g}+z_1\right)=\left(\frac{p_2}{\rho g}+\frac{V_2^2}{2g}+z_2\right)-h_q+h_s+h_v \qquad 1.$$

donde $h_q=\frac{q}{g}$ es la variación debida a la transferencia de calor, $h_s=\frac{w_s}{g}$ la variación debida al movimiento de las partes móviles, $h_v=\frac{w_v}{g}$ la variación debida al esfuerzo viscoso, $\frac{p_1}{\rho g}$ la carga/altura de presión, $\frac{V_1^2}{2g}$ la carga/altura de velocidad, g es la gravedad, V_1 es la velocidad inicial, V_2 es la velocidad de salida de la central, ρ es la densidad del agua y z1 y z2 son las alturas superior e inferior.

En nuestro sistema la ecuación 1 se reduce a la ecuación 2:

$$(z_1)=\left(\frac{V_2^2}{2g}+z_2\right)+h_s \rightarrow (z_1)-\left(\frac{V_2^2}{2g}+z_2\right)=h_s \qquad 2.$$

donde z_1 es la altura superior z_2 la altura inferior y V_2 la velocidad en m/s del caudal de salida, y h_s la altura de la turbina.

La potencia generada es el producto de la altura del sistema por la densidad, la gravedad, el caudal que pasa por la turbina y el rendimiento total del sistema.

La potencia del generador se calcula mediante la ecuación 3:

$$(h_s)\rho g\eta Q=P \ \left(w=\frac{Nm}{S}\right) \qquad 3.$$

donde η es el rendimiento, H_i La altura inicial, H_f la altura del fin del salto y P la potencia generada.

La energía producida, E, se calcula con

$$P\ t = E\ (J = sw) \quad (4)$$

donde P es la potencia y t es el tiempo.

7.1.3 Estructura de la programación

En este apartado explicaremos el funcionamiento de las funciones simulación, instante_t_ y de los selectores y mostraremos la importancia que tienen en el programa resultante.

Para el sistema se evalúa en cada instante t de la lista de tiempo la función instante_t_. Esta función es única para cada tipo de central y agrupa distintos métodos específicos para cada una de ellas. Estos métodos reciben un nombre cuyo formato general es selector barra baja más la variable que calcula en el instante t.

Hay bastantes funciones denominadas como sistema_ seguido por el nombre del tipo de simulación. La variedad se explica por la mayor eficiencia en tiempo computacional para el uso de una función que es específica para un tipo de simulación. Un esquema general de la estructura se muestra en la Figura 9.

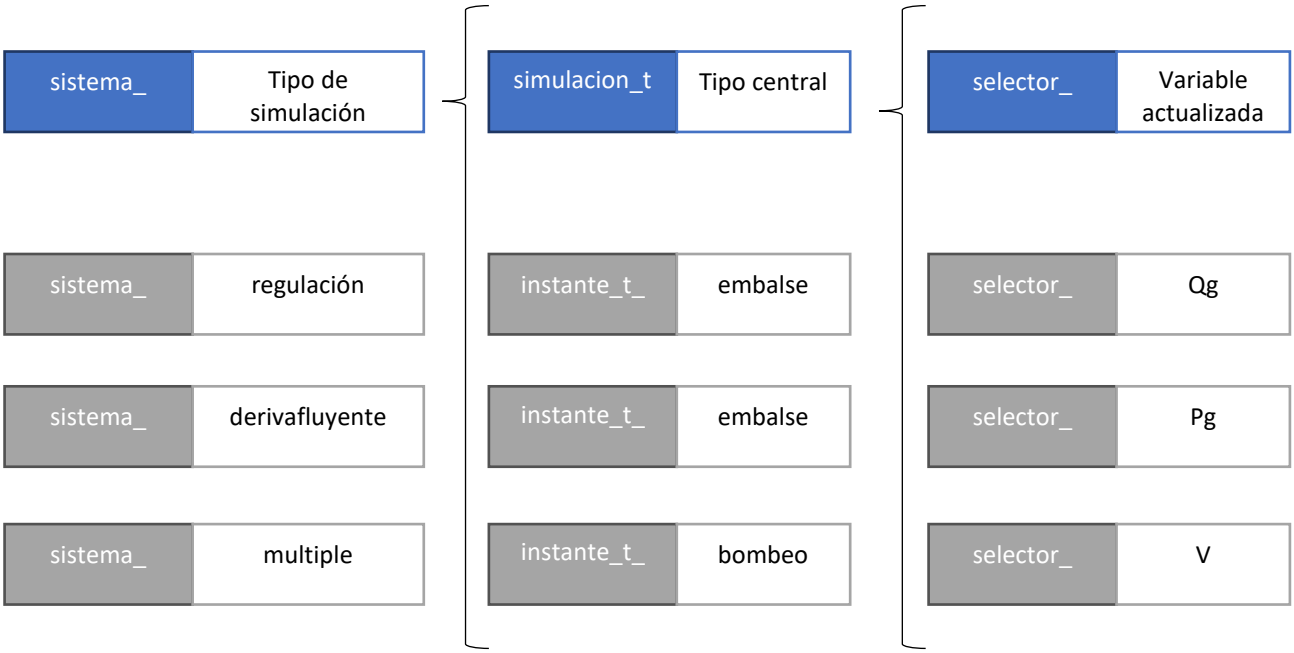


Figura 9: Estructura de la simulación

7.2 CENTRALES FLUYENTES Y DE DERIVACIÓN

La potencia que genera una central fluyente o de derivación no es controlable y depende del caudal que está pasando por el río en ese momento. El caudal del río en el que se ponen estas centrales ha de ser lo menos arbitrario posible, por lo que en la simulación no incluiremos variaciones diarias ni semanales de caudal, considerando un modelo sinusoidal estacionario de periodicidad de un año. Además, comprobaremos que la potencia que genera el modelo se asemeja a la de un sistema real.

7.2.1 Estructura

La estructura de las centrales fluyentes y de derivación es muy similar. La principal diferencia está en que la variable que controla la parte del caudal que pasa por el generador (P_Qg) (en tantos por uno) será igual a 1 en las centrales fluyentes y positiva y próxima a cero en las centrales de derivación.

Primero se obtiene el caudal de entrada (que entrará en esta central entre t y t+1), para calcular posteriormente el caudal del generador (Qg), el que sale por el río (QsaleRio) y el que sale en total contando las necesidades, (QsaleTotal), que en estas centrales será el mismo que el caudal de entrada.

```
def instante_t_derivacionfluyente(t, datos, presa, necesidad, g=9.8, den=998, periodo=3600):
    '''Versión 1
    Calcula las magnitudes características de la central para un sistema en el tiempo t.

    Esta función invoca a un conjunto de métodos de la clase Datos, que se ejecutan
    para cada instante de tiempo t del total para simular la central de derivación o
    la central fluyente.

    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -periodo: segundos entre cada instante de tiempo t

    SALIDA: objeto datos actualizado en el instante t
    '''
    datos.selector_QentraTotal(t)
    datos.Qg[t] = (datos.QentraTotal[t] - necesidad.Qsale_m3s[t])*presa.P_Qg
    datos.QsaleRio[t] = datos.QentraTotal[t] - necesidad.Qsale_m3s[t]
    datos.QsaleTotal[t] = datos.QentraTotal[t]

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)

    return datos
```

Figura 10: Función instante_t_derivacionfluyente

Además, para estas centrales se ha considerado que la altura del río (H) es constante durante todo el año.

7.2.2 Ejemplo central fluyente

A continuación, se presentan y comentan los resultados proporcionados por el modelo que hemos construido.

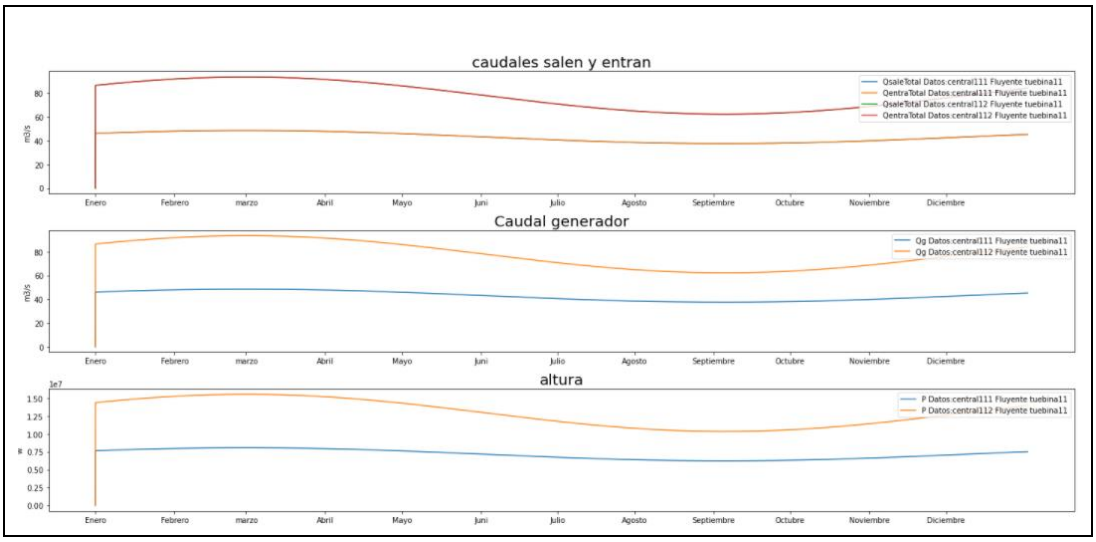


Figura 11: Ejemplo de dos centrales fluyentes individuales.

En la Figura 11 se puede observar el comportamiento de varias variables correspondientes a dos centrales fluyentes, denominadas 111 y 112, con distinto caudal. La central 111 está alimentada por el caudal del río Duero y la central 112 por el del río Pisuerga.

Nótese que la potencia generada se encuentra rondando los 10 MW. El valor de 10 MW es la potencia que delimita las grandes centrales hidroeléctricas (que generan mayor potencia que 10 MW) y las pequeñas centrales hidroeléctricas (que generan menor potencia que 10 MW) (43).

7.2.3 Ejemplo central derivación

En este apartado se presenta la simulación para dos casos de centrales de derivación, denominadas 211 y 212. La derivación elegida ha sido del 50% y los ríos seleccionados han sido el Pisuerga para la central 211 y el Duero para la 212.

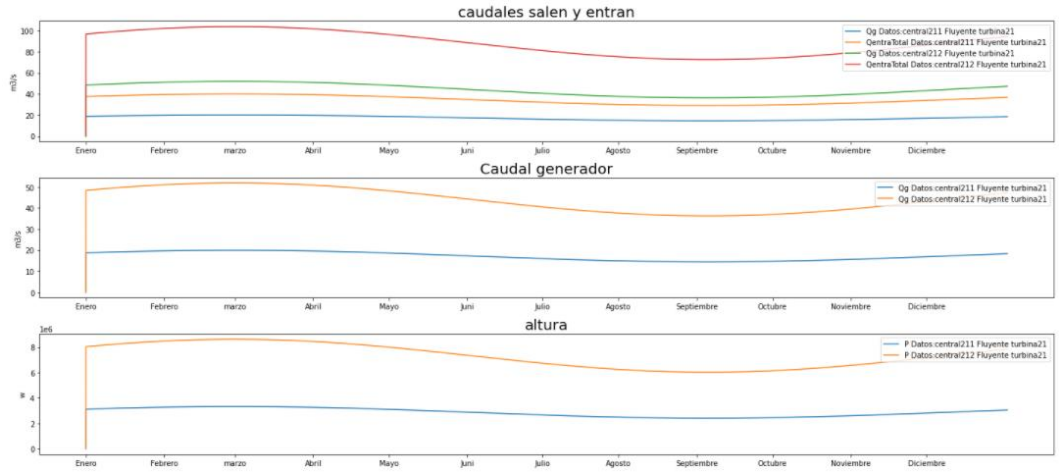


Figura 12: Ejemplo de dos centrales de derivación individuales.

Nótese que la potencia que generan estas centrales las clasifica en el grupo de las pequeñas centrales hidroeléctricas.

7.3 CENTRALES DE REGULACIÓN Y DE BOMBEO

La potencia que generan las centrales de regulación y de bombeo sí es controlable y depende del caudal que pase por la bomba y la turbina. El caudal del río en el que se instalan estas centrales puede no ser estable, es más, en muchos casos las presas se usan para garantizar el abastecimiento a la población (apartados 7.4 y 7.5.2) cuando estos caudales son reducidos. En este apartado se presentarán las funciones que modelan las bombas y las centrales de regulación, indicando sus diferencias, y se introducirán las clases Aliviadero, Bomba y Turbina, necesarias para simular este tipo de centrales.

En el caso de las centrales de regulación y bombeo hay que tener presentes dos variables importantes que las centrales fluyentes y de derivación, diseñadas en el apartado anterior, o no poseen o se toman como un valor constante:

El selector de volumen (V) y el selector de altura (H) manejan los datos obtenidos por los selectores del tiempo t anterior.

7.3.1 Estructura

Las centrales de regulación (Figura 13) y de bombeo (Figura 14) poseen la misma estructura con la excepción de la adición del selector_Qb (función que modifica el caudal de la bomba) y el selector_Pb (función que calcula el consumo de potencia de la bomba). En la Tabla 1 se puede observar, de una manera abreviada, la estructura.

Tabla 1: Selectores y variables modificadas en centrales de derivación y de bombeo

Selectores	**descripción**
Datos de inicio del periodo	
V	Volumen de la central
H	Altura de la central
Caudales	
Q_entraTotal	Caudal total que entra desde t hasta t+1
Q_saleRio	Caudal que sale por el río desde t hasta t+1
Q_saleTotal	Caudal total que sale desde t hasta t+1
Qg	Caudal que sale por el generador desde t hasta t+1
Qb (solo en centrales reversibles)	Caudal que entra por bombeo desde t hasta t+1
Qaliviadero	Caudal que sale por el aliviadero desde t hasta t+1
Qdesvordamiento	Caudal que se absorbe cuando se desborda desde t hasta t+1
Potencias consumida y generada en el periodo	
Pg_ms	Potencia generada durante t a t+1
Pb (solo en centrales reversibles)	Potencia consumida durante t a t+1

```
def instante_t_regulacion(datos, t, presa, necesidad, g=9.8, den=998, periodo=3600):
    '''Versión 2
    Calcula las magnitudes características de la central para un sistema en el tiempo t.

    Esta función invoca a un conjunto de métodos de la clase Datos, que se ejecutan
    para cada instante de tiempo t del total para simular la central de regulación.

    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -periodo: segundos entre cada instante de tiempo t

    SALIDA: objeto datos actualizado en el instante t
        '''
    datos.selector_V(t, periodo=3600)
    datos.selector_H(t, presa=presa)

    datos.selector_QentraTotal(t)
    datos.selector_Qg(t, necesidad, presa=presa, g=g, den=den, periodo=periodo)
    datos.selector_Qaliviadero(t, presa=presa,necesidades=necesidad, g=g, den=den)
    datos.selector_Qdesbordamiento(t, presa=presa, periodo=periodo)
    datos.selector_QsaleTotal(t, necesidad, presa)
    datos.selector_QsaleRio(t, presa=presa)

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)

    return datos
```

Figura 13: Función instante_t_regulación

Al principio, se calculan el volumen (V) y la altura (H) a la que corresponde el valor que tendrían estos parámetros de la central al inicio del periodo t. Posteriormente, se calcula el caudal de bombeo en el caso de la central reversible y el caudal de entrada, este orden es necesario para que en el caudal de entrada del momento t figure el caudal de la bomba y este caudal que haya entrado sea visible en la simulación t+1. Posteriormente, y siguiendo la misma lógica, se calculan distintos caudales de salida dependientes de esta central y el caudal de salida total y el caudal de salida que sale por el río.

```
def instante_t_bombeoS(datos, t, necesidad, presa, g=9.8, den=998, periodo=3600):
    '''Versión 2
    Calcula las magnitudes características de la central para un sistema en el tiempo t.

    Esta función invoca a un conjunto de métodos de la clase Datos, que se ejecutan
    para cada instante de tiempo t del total para simular la central de bombeo.

    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -necesidad: objeto de la clase Necesidades
        -presa: objeto de la clase Presa
        -g: gravedad
        -den: densidad
        -periodo: segundos entre cada instante de tiempo t

    SALIDA: objeto datos actualizado en el instante t
        '''
    datos.selector_V(t, periodo=3600)
    datos.selector_H(t, presa=presa)

    datos.selector_Qb(t, necesidad, presa=presa,listDatos=None, g=g, den=den, periodo=periodo)
    datos.selector_QentraTotal(t)
    datos.selector_Qg(t, necesidad, presa=presa, g=g, den=den, periodo=periodo)
    datos.selector_Qaliviadero(t, presa=presa,necesidades=necesidad, g=g, den=den)
    datos.selector_Qdesbordamiento(t, presa=presa, periodo=periodo)
    datos.selector_QsaleTotal(t, necesidad, presa)
    datos.selector_QsaleRio(t, presa=presa)

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)
    datos.selector_Pb(t, presa=presa, g=9.8, den=998)

    return datos
```

Figura 14: Función instante_t_bombeoS

Posteriormente con los caudales Qg y Qb se calcula la potencia generada por el generador (Pg_ms) y la potencia consumida por la bomba (Pg).

7.3.2 Aliviadero y absorción

En este apartado vamos a detallar las ecuaciones en las que se basa el programa para los cálculos del caudal del aliviadero y de la absorción. Es decir, las dos fugas complementarias de la central en caso de que la altura del embalse sea superior a uno de los límites prefijados.

Comenzaremos con el cálculo del caudal desalojado por el **aliviadero.** Para ello nos apoyaremos en el teorema de Torricelli que es un caso particular del teorema de Bernoulli (42) para un volumen con una abertura en la parte inferior. En concreto, el Teorema de Torricelli[4] establece que la energía potencial de la superficie se convierte en la energía cinética del chorro de salida tal como se muestra en la ecuación 5

$$V_t = \sqrt{2 \cdot g \left(h + \frac{v_0^2}{2 \cdot g} \right)} \qquad 5.$$

[4] (42) página 182

donde g es la gravedad, h la diferencia de altura desde el nivel del agua hasta el aliviadero, v_0 la velocidad inicial y V_t es la velocidad del caudal de salida.

En el límite cuando la velocidad inicial v_0 tiende a 0 la expresión anterior se reduce a:

$$V_t(v_0 \sim 0) = \sqrt{2 \cdot g \cdot h} \qquad 6.$$

Y el caudal descargado será $Q = V_t \cdot Sc$

Después, para mejorar la simulación, se pueden introducir los coeficientes C_v (coeficiente de velocidad) y C_d (coeficiente de descarga). Añadiendo estos coeficientes la ecuación resultante es:

$$Q_{aliviadero} = (ScC_d)(C_v\sqrt{2 \cdot g \cdot h}) \qquad 7.$$

Para la programación, recordemos que estamos considerando una versión discretizada de nuestros sistemas, en lo que al tiempo se refiere. En el marco de esta aproximación discreta se han planteado dos casos posibles:

El **primer caso** corresponde a la situación en la que la altura de inicio del periodo está por encima de la altura donde se encuentra el aliviadero. Así, para calcular el volumen de salida del aliviadero tenemos:

$$V_{ali} = \begin{cases} Q_{ali}(H(t))T \; si \; Q_{ali}(H(t))T > V_{ex}\;(t) y \; V_{ex}\;(t) > 0 \\ V_{ex} \; si \; Q_{ali}(H(t))T > V_{ex}\;(t) y \; V_{ex}(t) > 0 \\ Q_{ali}(H(t))T\,\dfrac{x_{ali}(t)}{T} \; si \; y \; V_{ex}(t) < 0 \end{cases} \qquad 8.$$

donde Q_{ali} es el caudal del aliviadero calculado con la ecuación 7, T es el periodo en segundos, que en nuestro caso es 3600, V_{ex} es el volumen en exceso que hay al final del periodo haciendo los cálculos sin el aliviadero y x_{ali} es la porción de tiempo antes de que la altura del embalse se sitúe por debajo de la altura del aliviadero, tomando el caudal del aliviadero constante para cada lapso de tiempo t:

$$H_f(t) = H(t) \mp \frac{Q_{entra}(t)T - Q_{ali}(t)T}{base} \qquad 9.$$

donde H_f es la altura al final del periodo, Q_{entra} es la diferencia del caudal que entra con el que sale, sin tener en cuenta el caudal de aliviadero o de la absorción y base es la base de la central.

El valor H_f se emplea para el cálculo de m:

$$m(t) = \frac{H_f(t) - H(t)}{T} \quad 10.$$

que es el valor de la pendiente de la siguiente recta:

$$h = H(t) + m(t)\, x \quad 11.$$

Despejando, calculamos x para el valor de h correspondiente a la altura del aliviadero:

$$x_{ali}(t) = H_{ali} - H(t)/m(t) \quad 12.$$

El **segundo caso** se refiere a la situación en la que la altura de inicio del periodo está por debajo de la altura donde se encuentra el aliviadero, pero la altura final está por encima (calculo hecho sin contar con el caudal del aliviadero). El volumen de agua que sale por el aliviadero por cada lapso de tiempo t está dado por:

$$V_{ali}(t) = \begin{cases} V_{ex}\ (t) si\ Q_{Hali} > Q_{entra} \\ Q_{Hali} T \dfrac{T - x_{ali}}{T}\ si\ y\ V_{ex}(t) < 0 \end{cases} \quad 13.$$

donde Q_{Halies} es el caudal del aliviadero calculado con la ecuación 7, para la altura del aliviadero.

Para este caso, los cálculos para obtener x_{ali} son similares a los del caso anterior. De hecho, se puede utilizar la misma ecuación 12 pero ahora, para determinar la pendiente de la recta m, el valor de H_f se computa sin incluir el caudal del aliviadero, a diferencia de lo que ocurría en la ecuación 9:

$$H_f(t) = H(t) + \frac{Q_{entra}(t) T}{base} \quad 14.$$

El caudal de **absorción** se va a tomar como un parámetro constante, como podría ser el caudal que una ciudad es capaz de desalojar a través del sistema de recogida de lluvias para evacuar el exceso de precipitaciones.

Para un primer modelo se ha asumido que el caudal del aliviadero calculado al principio del periodo permanece constante. Esto produce bastantes oscilaciones pues no contempla que en un instante de dicho periodo la altura del embalse puede ser superior a donde se encuentra esta abertura, así que, posteriormente, se añadirá la condición de que en ese caso el sistema deberá calcular el volumen del aliviadero al final del periodo y si la suma de los caudales de salida es superior a los caudales de entrada la altura permanecerá a la altura en la que se encuentra el aliviadero.

7.3.3 Turbina y el caudal del generador

En este apartado vamos a emplear una función senoidal simple, cuya obtención se explica posteriormente en el apartado 7.4.1, para representar el caudal de entrada de los ríos.

```
class Turbina:
    def __init__(self, Nombre='t_prueba1', r=0.85, Q_max=None,Q_min=None, Qg_funcion_propia=0):
        '''VERSION2 (modulo version 15 )
        Inicializar los datos de la turbina

        ATRIBUTOS:
            -Nombre: nombre de la turbina
            -r: rendimiento de la turbina
            -Q_max: caudal máximo al que puede funcionar la turbina
            -Q_min: caudal mínimo al que puede funcionar la turbina
            -Qg_funcion_propia: funcion para calcular que pasa por la turbina'''
        self.Nombre = Nombre
        self.Q_max = Q_max
        self.Q_min = Q_min
        self.r = r
        self.Qg_funcion_propia=Qg_funcion_propia
    #presa.turbina.Qg_funcion_propia(self,t, necesidad, presa,g=9.8, den=998, periodo=3600)

    def __str__(self):
        '''VERSION 1
        Devuelve informacion del objeto en cadena string'''
        string = self.Nombre+' r'+str(self.r)
        return string
```

Figura 15: Clase Turbina

Para el cálculo del caudal del generador se ha planteado un algoritmo sencillo para una estimación aproximada. No obstante, el programa soporta el uso de otros algoritmos para, por ejemplo, lograr una mayor precisión. Este trabajo no trata de crear el mejor algoritmo para seleccionar el caudal preferible para la turbina por lo que no es el valor más importante para modelar el sistema. Lo que sí ha de ser importante a la hora de realizar las simulaciones es que este parámetro sea directamente proporcional al caudal que sale de la central y directamente proporcional a la potencia generada (ecuación 3). Por lo tanto, si se modifica el caudal del generador se modifica el caudal de salida y la potencia.

Vamos ahora a empezar con un caudal de generador constante para ver los efectos que tiene este en nuestra central (expresión 15)

$$Q_g = Q_K \qquad 15.$$

donde Q_g es el caudal del generador, y Q_k es un caudal constante prefijado.

En la Figura 16 podemos observar el comportamiento del sistema que corresponde a un generador cuyo caudal está descrito por la expresión 15. Los caudales de entrada de las distintas centrales están proporcionados por los ríos Bernardos, Támega y Carrión, con los datos que se pueden encontrar en el Anexo 3.1 Caudales ríos en España.

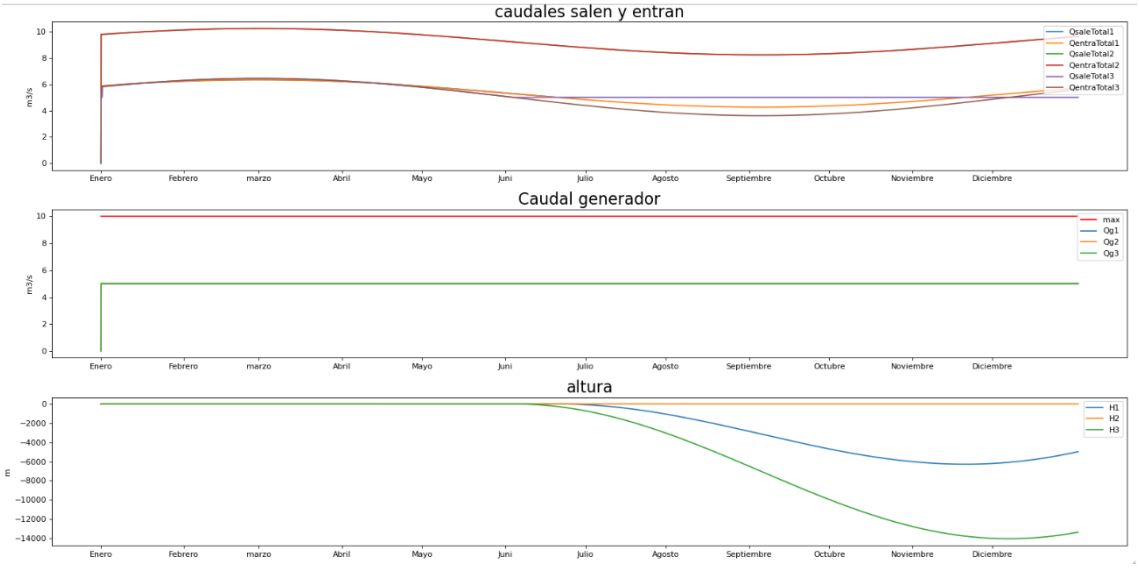


Figura 16: Primer modelo de turbina

Para mejorar este sistema y evitar que la altura de la central se haga negativa (hecho que es imposible) se empleará un caudal del generador descrito por la función 16, que se corresponde con la ecuación 15 pero con una altura mínima por debajo de la cual el caudal de salida es nulo. Esto servirá para evitar que la altura de salida se haga negativa.

$$Q_g(H)=\begin{cases} Q_K \; si \; H > H_{min} \\ 0 \; si \; H < H_{min} \end{cases} \qquad 16.$$

donde Q_g Es el caudal del generador, H es la altura actual, H_{min} es la altura mínima y Q_k es un caudal constante prefijado.

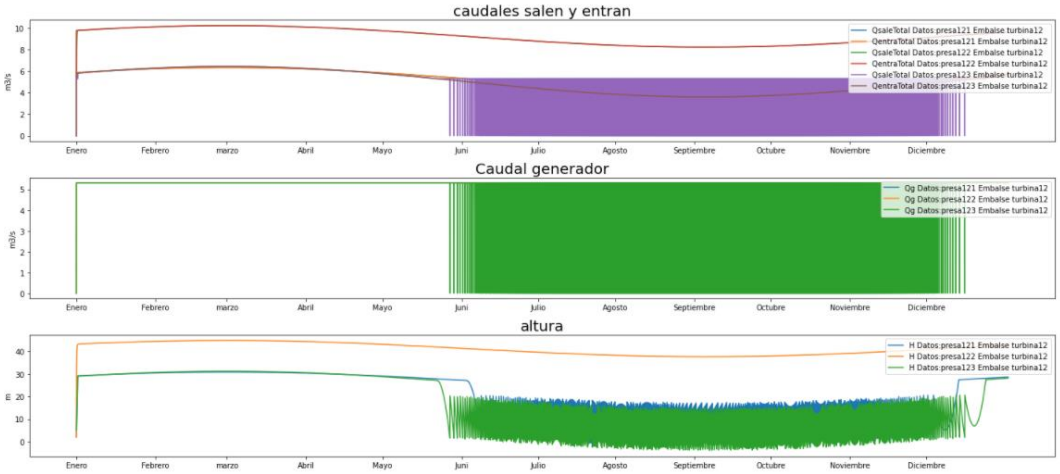


Figura 17: Segundo modelo de turbina

Podemos observar en la Figura 17 que se mantiene en rango estacionario en cuanto a la altura se refiere. El caudal del generador es una función constante a trozos: cuando el caudal del generador supera una altura mínima su valor es Q, siendo nulo el valor del

caudal en caso contrario. Hay que tener en cuenta que las centrales reales, al igual que ocurre en esta simulación, no siempre están en funcionamiento.

Ahora para buscar un equilibrio sin que la altura del embalse alcance el mínimo, vamos a añadir un escalón con el fin de simular la situación en la que una central real tuviera más de un grupo de turbinas y cada grupo estuviera orientado a un rango de alturas. El sistema que se va a considerar es el que corresponde a la función 17 y se muestra en la Figura 18.

$$Q_g(H) = \begin{cases} Q_2 \; Si \; H > H_1 \\ Q_1 \; Si \; H_1 > H > H_{min} \\ 0 \; Si \; H < H_{min} \end{cases} \qquad 17.$$

$$siendo \; Q_2 > Q_1 > 0$$

$$y \; siendo \; H_1 > H_{min} > 0$$

donde Q_g es el caudal del generador, H es la altura actual, H_{min} es la altura mínima, H_1 es una altura intermedia y Q_1 y Q_2 son caudales constantes prefijados.

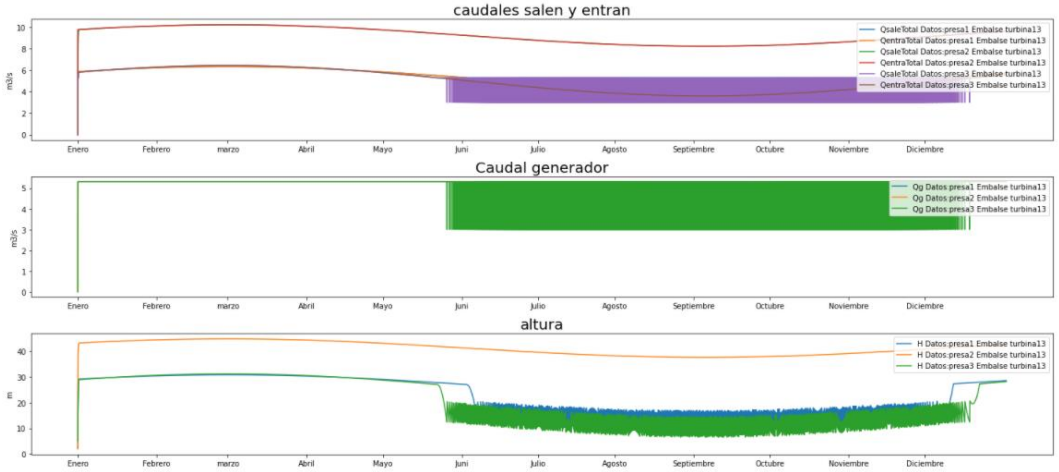


Figura 18: Tercer modelo de turbina

Como vemos en la simulación representada en la Figura 18, el caudal que pasa por la turbina oscila entre 3 m^3/s y 5.1 m^3/s (que serían Q_1 y Q_2) dependiendo de la altura de la central.

7.3.4 Bomba

La estructura y funcionamiento de la clase Bomba se asemeja mucho a la de la clase Turbina, con la diferencia de que en este tipo de central se puede modelar la relación entre el caudal de bombeo y la potencia consumida. En la Figura 19 se puede apreciar su constructor.

```
class Bomba:
    def __init__(self, Nombre='bomba1', r=0.85, Qb_funcion_propia=0,Pb_funcion_propia=0):
        '''VERSION1(modulo version 14 )
        Inicializa los datos de la bomba

        ATRIBUTOS:
            -Nombre: nombre de la bomba
            -r: rendimiento de la bomba
            -Qb_funcion_propia: funcion para calcular el caudal que pasa por la bomba
            -Pb_funcion_propia: funcion para calcular la potencia consumida por la bomba
           '''
        self.Nombre = Nombre
        self.r = r
        self.Qb_funcion_propia=Qb_funcion_propia
        self.Pb_funcion_propia=Pb_funcion_propia
    # presa.bomba.Qb_funcion_propia(self,t, presa, necesidad, listDatos, g=9.8, den=998, periodo=3600)
    # presa.bomba.Pb_funcion_propia(self,t, presa, g=9.8, den=998, periodo=3600)

    def __str__(self):
        '''VERSION 1
        Devuelve informacion del objeto en cadena string'''
        string = self.Nombre+' r'+str(self.r)
        return string
```

Figura 19: Clase Bomba

Tiene atributos importantes desde el punto de vista de la simulación, como es el Nombre, el rendimiento r y las funciones en las que se introduce el algoritmo para el cálculo del caudal de la bomba (Qb) y la potencia consumida (Pb).

Ahora vamos a observar el comportamiento de la central al producirse el bombeo con un algoritmo similar al empleado para la turbina en el apartado 7.3.3 (con un caudal de bombeo cuando la altura sea inferior a la altura mínima) y en los apartados de consumos (apartado7.5.2) mostraremos otros tipos de algoritmos y su efecto sobre la potencia.

En la Figura 20 se va a comparar el comportamiento de la altura de una central de bombeo y de una central de regulación con los mismos datos. Se realizará utilizando como entrada el río Pisuerga al 40% de su caudal normal y una altura mínima de 15. El río se ha modelado con un polinomio trigonométrico calculado por interpolación, tal como se expresa en el apartado 7.4.2.

Se puede observar en la Figura 20 que, con la central de bombeo, la altura en los meses con menos caudal es superior que en una central de regulación.

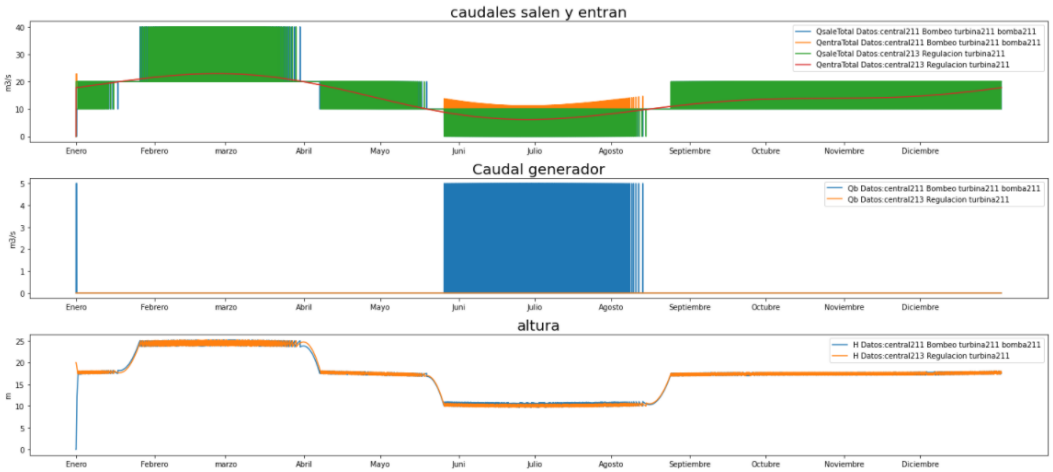


Figura 20: Comparación de una central de regulación con una de bombeo

A la hora de elegir un sistema tendríamos que pensar si se está dispuesto a que la central absorba potencia para aumentar la altura de la central o si, realmente, en este caso no nos importan tanto las reservas de agua como la potencia producida.

7.4 MODELADO DE VARIABLES PERIÓDICAS

En este apartado se presentará un breve resumen de los dos principales métodos utilizados para modelar algunas entradas del sistema. En concreto, estas entradas serán el caudal de los ríos, la potencia consumida y las necesidades de agua.

7.4.1 Modelado sinusoidal simple

Para modelar la demanda hay que tener en cuenta que la dinámica del sistema se aproximará discretizando mediante saltos de tempo de 1h. El modelo más simple para simular una demanda periódica usando funciones trigonométricas viene dado por una expresión del siguiente tipo:

$$\frac{maximo - minimo}{2} cos\left(-\frac{(2\pi)}{T} p_{maximo} + \frac{(2\pi)}{T} t\right) + media \quad (18)$$

Que, adaptando la notación, equivale a:

$$f(t) = A\, cos(\, wt - \varphi) + media \quad (19)$$

En la expresión anterior w es la velocidad angular, dato que podemos calcular a través del periodo:

$$w = \frac{2\pi}{T} \quad (20)$$

A continuación se indica una forma de obtener el **ángulo inicial**. Este ángulo determinará en qué momento t se alcanza el valor máximo. Recordemos que la función coseno alcanza su valor máximo y mínimo de acuerdo con las siguientes expresiones:

$$cos(\theta) = 1 \;\; si \;\; \theta = 2\pi n \;\; / \;\; n \in \mathbb{Z} \quad (21)$$

$$cos(\theta) = -1 \;\; si \;\; \theta = \pi(2n-1) \;\; / \;\; n \in \mathbb{Z} \quad (22)$$

Por lo tanto, para seleccionar el ángulo inicial se multiplicará el valor de t en el que se alcanza el máximo por la velocidad angular, lo que nos proporciona la siguiente fórmula:

$$\varphi = \frac{(2\pi)}{T} t_{máximo} \quad 23$$

donde T es el periodo, $\frac{(2\pi)}{T}$ la velocidad angular y $t_{máximo}$ el instante t donde encontramos un máximo.

Para el **cálculo de la amplitud** solo tendremos que restar el valor máximo y el punto mínimo para tener la diferencia y dividir este valor entre 2 para que la mitad de esta diferencia se dé cuando el coseno sea positivo y la otra mitad cuando el coseno sea negativo:

$$A = \frac{P_{máximo} - P_{mínimo}}{2} \quad 24$$

donde $P_{máximo}$ es el valor máximo de la función que estamos aproximando, $P_{mínimo}$ es su valor mínimo y A es la amplitud.

Para calcular la **media** solo tendremos que sumar el valor máximo y el mínimo y dividirlo por dos:

$$m = \frac{P_{máximo} + P_{mínimo}}{2} \quad 25$$

Dependiendo del conjunto de datos, se pueden usar diferentes estrategias para aproximar los valores máximo y mínimo y la media. Por ejemplo, se puede utilizar la media o la mediana para estimar el valor medio y para los máximos y mínimos se puede emplear la varianza o los valores máximos y mínimos, entre otros.

Finalizamos esta sección con un comentario sobre el periodo: los datos que se utilizan en esta memoria permiten identificar distintos intervalos temporales que comportan algún tipo de periodicidad. El año representa un intervalo de periodicidad natural pero también se pueden observar comportamientos periódicos semanales y diarios. La siguiente tabla recoge el valor de los períodos que acabamos de comentar.

Tabla 2: Periodo anual, semanal y diario en horas

	anual	Semanal	día
Periodo (T) horas	24h/diax365dia	24h/día x7dia	24h

7.4.2 Interpolación polinomial

La aproximación de las entradas del sistema mediante el modelado sinusoidal simple tiene el problema de proporcionar un ajuste que, en muchas ocasiones, termina siendo muy pobre. En este apartado vamos a mostrar algunos tipos de interpolaciones que

podríamos usar para mejorar el ajuste. Vamos a comenzar por la **interpolación polinomial de Lagrange** [5]para una función

$$f(x) = y \qquad 26.$$

en los nodos conocidos x_1, x_2, …, x_K donde la función toma los valores y_1, y_2,…, y_K (ecuación 26).

El polinomio que interpola a la función 26 en los nodos especificados viene dado por la siguiente expresión:

$$p_L(x) = \sum_{k=1}^{K} y_k \, l_k(x) \qquad 27.$$

donde l_k es lo que se llama polinomio elemental de Lagrange asociado al nodo k-ésimo. Los polinomios l_k se pueden construir de la siguiente forma:

$$l_k(x) = \prod_{i=1; i\neq k}^{K} \frac{x - x_i}{x_k - x_i} \qquad 28.$$

Como las abscisas de nuestro sistema tienen carácter temporal, la x se denota mediante la variable t y, por lo tanto, en nuestro caso los datos son de la forma {t, y}.

Con este método de interpolación tenemos un modelo del sistema que es no periódico. Si el sistema tuviese cierta periodicidad, la interpolación de Lagrange fallaría para valores de x fuera del intervalo tomado (suponemos aquí que el intervalo es de longitud igual a un período). Con el fin de reproducir la periodicidad (diaria, semanal, anual) de nuestros sistemas emplearemos polinomios interpoladores trigonométricos como los que se describen a continuación.

Para efectuar **la interpolación trigonométrica**[6] se va a coger un conjunto de datos {x,y} correspondientes a la función que se quiere aproximar. Un polinomio trigonométrico general de grado K tiene la forma de la ecuación 29 y consta de 2K+1 coeficientes indeterminados.

$$T(x) = a_0 + \sum_{k=1}^{K} a_k \cos(kx) + \sum_{k=1}^{K} b_k \sin(kx) \qquad 29.$$

$$a_0, \dots, a_K; b_0, \dots, b_K$$
$$0 \leq x_0, \dots, x_{K2+1} \leq 2\pi$$

El problema de interpolación consiste en encontrar los coeficientes para la ecuación *29* que satisfagan las condiciones de interpolación dadas por los datos $\{x_k, y_k\}$, siendo k un numero entero que va desde 1 hasta K.

Primero vamos a describir **la interpolación trigonométrica impar** para un sistema con K puntos, donde K es un número impar

El polinomio trigonométrico interpolador está dado por

[5] (53) apartado 18.2
[6] (54) **,** (55) **capítulo 3 y** (56) **capítulo 10**

$$T_{impar}(x) = \sum_{k=1}^{K} y_k * S_{impar_k}(x) \quad 30.$$

donde S_{impark} es:

$$S_{impar_k}(x) = \prod_{n=1, n\neq k}^{K} \frac{sin\frac{1}{2}(x - x_n)}{sin\frac{1}{2}(x_k - x_n)} \quad 31.$$

Para adaptar la expresión 31 a nuestros sistemas, efectuamos el siguiente cambio de variable:

$$x = t\frac{2\pi}{T} \quad 32.$$

donde T es el periodo y t el tiempo.

Sustituyendo en la ecuación 31 las x por la expresión 32 tenemos:

$$S_{impar_k}(t) = \prod_{n=1, n\neq k}^{K} \frac{sin\frac{1}{2}\left((t - t_n)\frac{2\pi}{T}\right)}{sin\frac{1}{2}\left((t_k - t_n)\frac{2\pi}{T}\right)} \quad 33.$$

Para un problema de i**nterpolación trigonométrica par,** el número K de nodos será par. Se puede demostrar que los polinomios trigonométricos S_{park} incluyen en el caso par otro parámetro de entrada α_k cuyos valores se pueden elegir de diversas formas como, por ejemplo, cogiendo la mayor frecuencia de todo el periodo.

El polinomio trigonométrico interpolador tiene en el caso par la misma estructura que la vista para el caso impar:

$$T_{par}(t) = \sum_{k=1}^{K} y_k * S_{par_k}(t) \quad 34.$$

Lo que será diferente es el término $S_{park,,}$ al que se añadirá un término dependiente del parámetro $\alpha_{k,}$ que se comentó anteriormente:

$$S_{par_k}(x) = \frac{sin\frac{1}{2}(x - \alpha_k)}{sin\frac{1}{2}(x_k - \alpha_k)} \prod_{n=1, n\neq k}^{K} \frac{sin\frac{1}{2}(x - x_n)}{sin\frac{1}{2}(x_k - x_n)} \quad 35.$$

En la expresión 35 se tendrán que sustituir las x por los valores del tiempo proporcionados por la expresión 32 y así obtenemos la ecuación que nosotros usaremos

$$S_{par_k}(t) = \frac{sin\frac{1}{2}\left(t\frac{2\pi}{T} - \alpha_k\right)}{sin\frac{1}{2}\left(t_k\frac{2\pi}{T} - \alpha_k\right)} \prod_{n=1, n\neq k}^{K} \frac{sin\frac{1}{2}\left((t-t_n)\frac{2\pi}{T}\right)}{sin\frac{1}{2}\left((t_k-t_n)\frac{2\pi}{T}\right)} \quad 36.$$

En este trabajo se utilizará, exclusivamente, el polinomio interpolador trigonométrico.

7.5 SUMINISTRO

7.5.1 Lluvias

Las lluvias las podemos describir usando datos de la precipitación total en un mes y los días en los que ha llovido en cada mes para cada capital de provincia en España. Al considerar las capitales de provincia en España tendremos acceso a multitud de climas y a las diversas situaciones climatológicas que sufren a lo largo del año. Estos datos son tomados por (44).

Para modelar las lluvias del sistema también debemos tener en cuenta el área en la que se va a absorber el agua de la lluvia, así como cuándo y cuánto va a llover. Para el área de recolección vamos a hacer una estimación. Hay que tener presente que los datos que tenemos sobre el caudal de los ríos incluyen también las lluvias en sus aguas arriba, por lo tanto, la simulación de la lluvia se usará para ver el efecto de las lluvias en la zona que se considere.

7.5.1.1 Programación:

Para obtener los días en los que va a llover se necesita una lista con los días que tiene cada mes y una lista con los días que llueve en cada mes. Tantas veces como días llueve en ese mes se introduce en la lista de salida un día de forma aleatoria (de ese mes) que representa el día que lloverá. (Figura 21).

```
def dias_lluvia(DiasDeLluvia, Dias_Mes=Dias_Mes):
    """VERSION 2.

    Los días del mes en el que se van a dar las precipitaciones
    ENTRADA:
        - DiasDeLluvia: lista con 12 elementos ( uno para cada mes ) con los
        días de lluvia que tiene cada mes
        - Dias_Mes: días que tiene cada mes
    SALIDA:
        Lista de los días en los que llueve a lo largo de un año
    """
    dialluvia = []
    dias = 0
    for i in np.arange(len(Dias_Mes)):
        dias = dias+Dias_Mes[i]
        for j in range(DiasDeLluvia[i]):
            dia_nuevo = random.randrange(1, Dias_Mes[i], 1)+dias-Dias_Mes[i]
            while dia_nuevo in dialluvia:
                dia_nuevo = random.randrange(1, Dias_Mes[i],1)+dias-Dias_Mes[i]
            dialluvia = dialluvia + [dia_nuevo]
    return dialluvia
```

Figura 21: Función dias_lluvia

De forma simultánea, vamos a obtener las precipitaciones de cada mes de modo que los datos de precipitaciones del mes se dividan de forma aleatoria entre los días del mes en los que va a llover.

```
def preci_dia(DiasDeLLuvia, precipitacion, minimo_precipitación=1,
              Dias_Mes=Dias_Mes):
    """VERSION 2.

    Lista de precipitaciones por cada día que llueve
    ENTRADA:
        - DiasDeLLuvia: nº de días en los que llueve de cada mes
        - precipitacion: precipitaciones de cada mes
        - Dias_Mes: días que tiene cada mes
    SALIDA:
        Lista precipitación/día mm/m2
    """
    LMP = precipitacion.copy()
    PD = []
    for i in np.arange(len(Dias_Mes)):
        random_numbers = np.random.uniform(low=0, high=2, size=DiasDeLLuvia[i])
        Preci =(random_numbers/sum(np.round(random_numbers,1))*LMP[i]).tolist()
        PD = PD + Preci

    return PD
```

Figura 22: Función preci_dia

El valor de retorno de la función que encontramos en la Figura 22 es una lista del mismo tamaño que la devuelta por la función de la Figura 21 con los valores que corresponden a las precipitaciones del día indicado en la anterior lista y con el mismo índice (Figura 23).

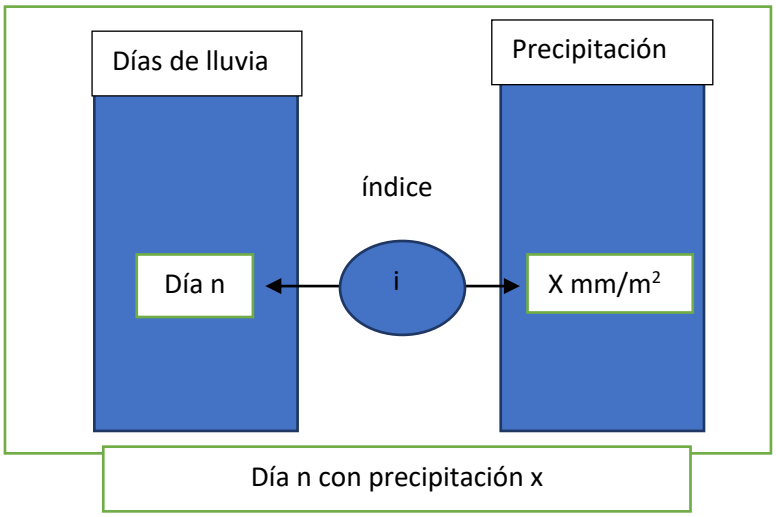


Figura 23: Esquema de la estructura de las dos listas que generan preci_dia y dias_lluvia

Por último, aunando los datos obtenidos mediante las anteriores funciones, los introducimos en la tercera función que devuelve una lista con tamaño de las horas que

tiene un año en las que en las horas de los días que se suponen que llueve aparezcan las precipitaciones en mm/m2. Esto lo haremos suponiendo que las precipitaciones en el día permanecen constantes y repartiendo las precipitaciones totales por cada hora del día (Figura 24).

```
def preci_hora(Preci_dia, dias_deLluvia, horas_año=8760):
    """VERSION 1.

    Lista anual con precipitaciones de cada dia (haya o no)
    ENTRADA:
        Requiere 2 listas : Preci_dia y dias_deLluvia del mismo tamaño
        - Preci_dia: la precipitación de cada día
        - dias_deLluvia: los días que llueve en un año
        - horas_año: horas que tiene un año
    SALIDA:
        Lista precipitación-hora mm/m2
    """
    PH = np.zeros(horas_año, dtype=float)
    unos = np.ones(24, float)
    for i in np.arange(len(dias_deLluvia)):
        hora = dias_deLluvia[i]*24
        PH[hora:(hora+24)] = PH[hora:(hora+24)] + unos*Preci_dia[i]/24

    return PH
```

Figura 24: Función preci_hora

Después de obtener la cantidad de agua que cae por hora, tenemos que calcular el volumen total de agua que afecta a nuestro sistema (Figura 25). Esto se hará en nuestro caso cambiando las unidades a las del SI y calculando esas precipitaciones para toda el área donde el caudal desemboque en la central.

```
def volumen_preci_total(precipitacion_hora, area_abastecida,
                        unidades_preci='mm', unidades_area='m2'):
    """VERSION 1.

    Volumen de agua de precipitaciones que han caído por cada hora
    ENTRADAS:
        - precipitacion_hora: lista de precipitaciones por hora
        - area_abastecida: área que recolecta el agua de lluvia
        - unidades_preci: unidades de las precipitaciones para pasarlo a metros
        - unidades_area: unidades de precipitacion_hora del area_abastecida
    SALIDA:
        Volumen de agua total recogido por el sistema en m3
    """
    PH = precipitacion_hora.copy()
    if unidades_preci == 'mm':
        PH = PH/(10**3)
    if unidades_area == 'km2':
        area_abastecida = area_abastecida*10**6

    return PH*area_abastecida
```

Figura 25: Función volumen_preci_total

7.5.1.1.1 Otras funciones útiles

Las funciones de este apartado tratan de simplificar el código para la creación de las listas con las lluvias y hacen uso de las funciones mencionadas anteriormente en esta misma subsección (7.5.1).

```
def precipitacion_hora(Dias, lluvia):
    """VERSION 1.

    Crea una lista del tamaño de un año con las precipitaciones que han caído
    en cada instante de tiempo siendo cada instante de tiempo igual a una hora
    con datos como entrada los días de lluvia y las precipitaciones totales de
    cada mes
    ENTRADA(2 listas de igual tamaño):
        - dias: días del mes en los que llueve
        - lluvia: cantidad de precipitaciones que ha caído en el mes
    SALIDA:
        Lista de tamaño las horas que tiene un año con las precipitaciones que
        han caído en cada hora mm/m2
    """
    DLluvia = dias_lluvia(Dias)
    PDia = preci_dia(Dias, lluvia)
    return preci_hora(PDia, DLluvia)
```

Figura 26: Función precipitación_hora

Las dos funciones, precipitación_hora (Figura 26) y precipitación_hora_ciudad (Figura 27), nos devuelven la misma salida, la lista de la lluvia que cae en cada instante de tiempo t. La función precipitación_hora tiene como entrada la lista con los días que llueve en cada mes y la lista de cuánto llueve cada mes. La otra función (precipitación_hora_ciudad) tiene como entrada un diccionario con la clave dada por el nombre del lugar de la lluvia y dos listas dentro de tamaño de los meses del año, la primera con la precipitación total de cada mes y la segunda con los días que llueve en un mes

```
def precipitacion_hora_ciudad(diccionario, ciudad):
    """VERSIÓN 1.

    Crea una lista del tamaño de un año con las precipitaciones que han caído
    en cada instante de tiempo siendo cada instante de tiempo igual a una hora
    con datos como entrada el diccionario con las ciudades como claves y los
    datos de las precipitaciones (días que llueven y precipitaciones totales)
    y la ciudad ( la clave) del diccionario
    ENTRADA:
        - diccionario: diccionario en donde se encuentran los datos de días
        que llueven por mes
            y precipitación total mensual
        - ciudad: clave con el que acceder a los datos de días que llueve en
        un mes y precipitación total mensual
    SALIDA:
        Lista de tamaño las horas que tiene un año con las precipitaciones que
        han caído en cada hora mm/m2
    """
    lluvia = DatosPluviales[ciudad][0]
    Dias = DatosPluviales[ciudad][1]

    DLluvia = dias_lluvia(Dias)
    PDia = preci_dia(Dias, lluvia)
    return preci_hora(PDia, DLluvia)
```

Figura 27: Función precipitación_hora_ciudad

7.5.1.2 Representación de las lluvias

Con los datos del Anexo 4. Precipitaciones, se van a emplear las funciones descritas en el apartado anterior y en el Anexo 6.1 Modulo de Herramientas Modelado para crear la lista de los datos que corresponden a la lluvia de Valladolid (Figura *28*).

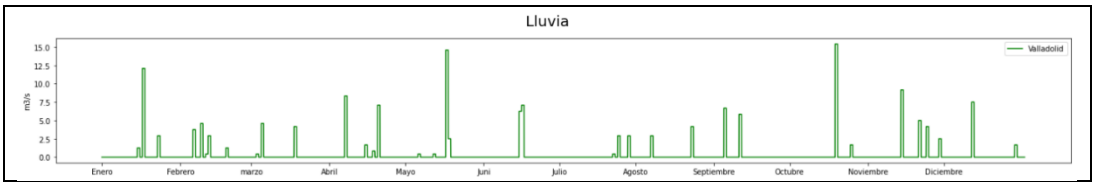


Figura 28: Lluvia en Valladolid

Se puede observar cómo en los meses de verano la lluvia es inferior, y menos frecuente que en invierno, como corresponde a los dantos del anexo.

7.5.1.3 Ejemplo simulación individual

En este apartado vamos a analizar cómo repercuten las lluvias en las centrales. Para la simulación solo se emplearán las centrales de tipo embalse, pues son las que pueden almacenar el agua de las lluvias torrenciales.

La selección de la zona que abarcará las lluvias que contribuyen al caudal de este embalse viene marcada por el área de la presa y por el área urbana. Las aguas infiltradas en el terreno pueden acabar evaporándose, en acuíferos o en el río a través del terreno y esta suele ser la zona más baja de la cuenca fluvial, excepto en caso de lluvias en una ciudad en la que gran parte desemboca en el río más cercano (en el caso de que la ciudad cuente con red separativa[7]). No obstante, en estas primeras simulaciones se supone que este caudal de lluvia desemboca en aguas debajo de la central, como si estuviera diseñado para evitar un desbordamiento o la central estuviera en una zona próxima, pero a mayor altitud que la ciudad.

Estos primeros ejemplos se realizarán con la misma área de precipitación que el tamaño del embalse. En particular, esta simulación se realizará con las precipitaciones de Valladolid, descritas en el apartado anterior.

[7] La red separativa existe en ciudades que poseen dos redes para la recolección de agua: una para las aguas negras y otra para las aguas pluviales.

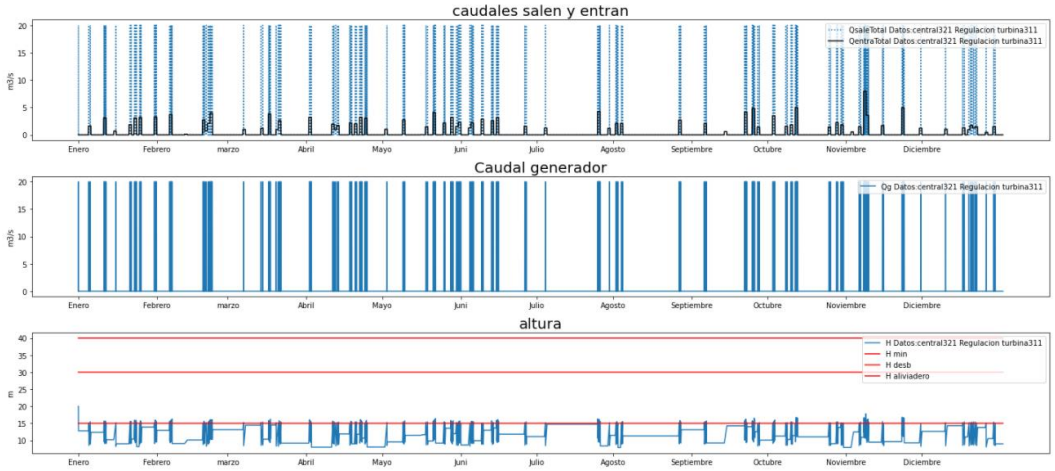


Figura 29: Lluvia en una central de regulación durante un año en Valladolid

En la Figura 29 se muestra el comportamiento de una central de regulación cuya única entrada es la lluvia. Como se puede ver en las gráficas, el caudal del generador se pone en marcha cuando la altura supera la altura mínima. Dado que este tiene que permanecer constante como mínimo durante una hora, cuando la lluvia cesa, la altura del agua de la central se queda por debajo de la altura mínima.

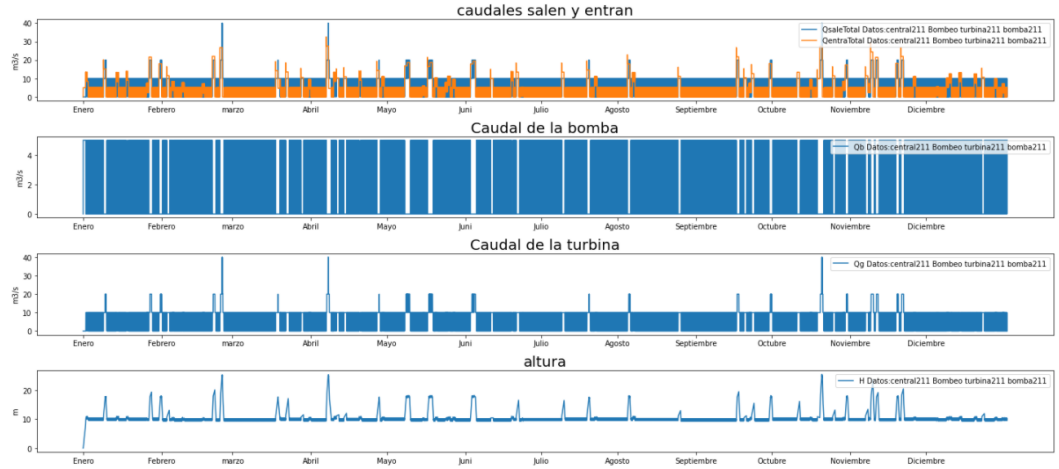


Figura 30: Lluvia en una central reversible durante un año en Valladolid

En la Figura 30 se puede observar el caso de una central reversible o de bombeo. A diferencia de lo que ocurre en el ejemplo anterior, la altura se ajusta más a la altura mínima. Esto es porque la central de bombeo y de regulación se compensan para que la altura se quede en torno a ese rango. En la Figura 31 se puede ver este comportamiento del embalse, antes, durante y después de una lluvia, con más claridad.

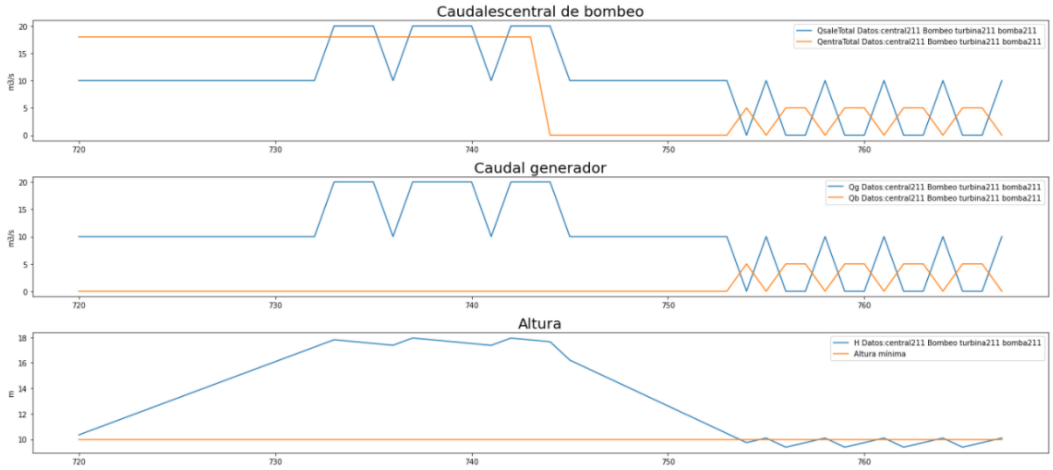


Figura 31: Lluvia en una central reversible durante dos días (31 de marzo y 1 de febrero) en Valladolid

Una mejora posible sería que la altura mínima se extendiera a un rango en el que no funcionara ni la turbina ni la bomba.

7.5.2 Ríos

Para modelar los ríos nos vamos a limitar a considerar solo ríos españoles debido a que existen datos muy fiables sobre sus caudales. Además, la gran cantidad de ríos existentes en España y la gran variedad de climas de nuestro país hace que la muestra considerada incluya ejemplos relevantes de prácticamente todos los tipos de cursos fluviales.

Los datos con los que contaremos son el caudal medio anual, el caudal máximo y el caudal mínimo (Anexo 3.1 Caudales ríos en España). Por otra parte, vamos a recoger datos de distintas centrales en el mundo, con sus respectivas producciones energéticas, y los compararemos con las simulaciones realizadas.

La siguiente subsección presenta simulaciones que ilustran la utilización de los datos que acabamos de presentar.

7.5.2.1 Ejemplo: río Pisuerga

En la Figura 32 tenemos al río Pisuerga con los datos del máximo y el mínimo, con los que se obtiene la variación anual. Para el sistema descrito mediante el modelo senoidal se tomará el valor medio de los valores máximos y mínimo. Esto es debido a que la media real, o aritmética, no es tan céntrica y en este modelo pretendemos que coincidan los máximos y mínimos para que los valores del caudal que tiene el río a lo largo del periodo estén en el intervalo de valores que en realidad tiene.

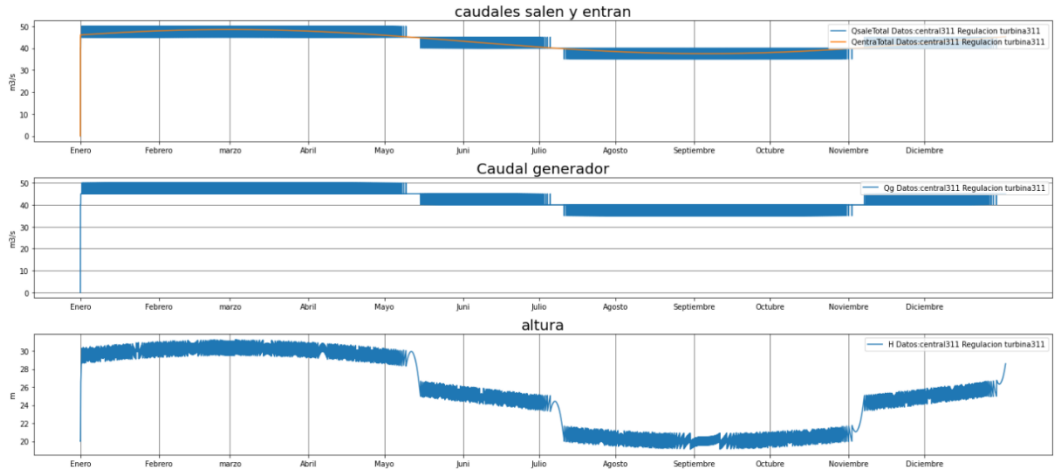


Figura 32: Simulación con datos del río Pisuerga con un sistema senoidal

En la Figura 33 se presenta la simulación proporcionada por el modelo con el polinomio trigonométrico impar.

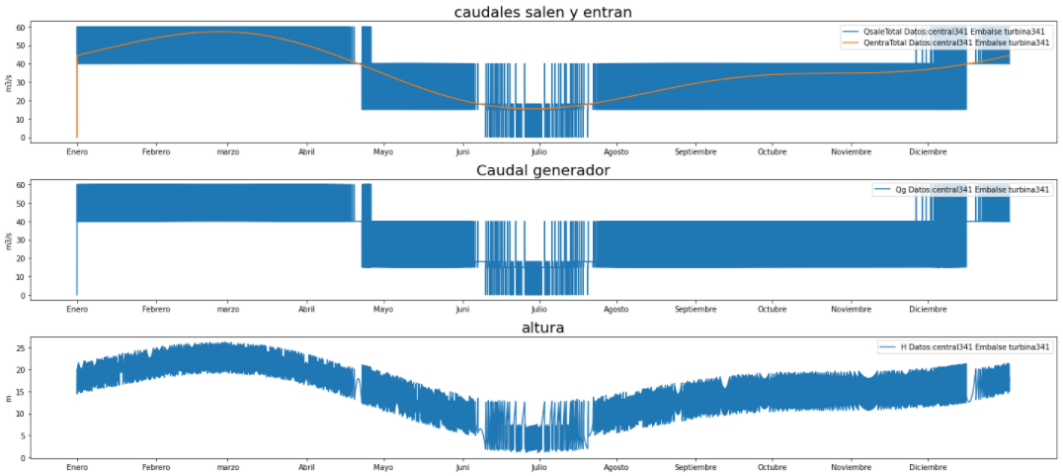


Figura 33: Central individual de regulación con interpolación trigonométrica impar del caudal de entrada del río Pisuerga

Al tener la turbina los mismos rangos, pero tener mayor diferencia en la segunda central, la variación de los caudales de generador es apreciablemente mayor.

En la Figura 34 se puede apreciar la diferencia de caudales de ambos modelos y en la Figura 35 se aprecia cómo el sistema se mantiene estable en torno a unos rangos para la entrada senoidal (central 311) y para la entrada de caudal con la interpolación trigonométrica (central 341).

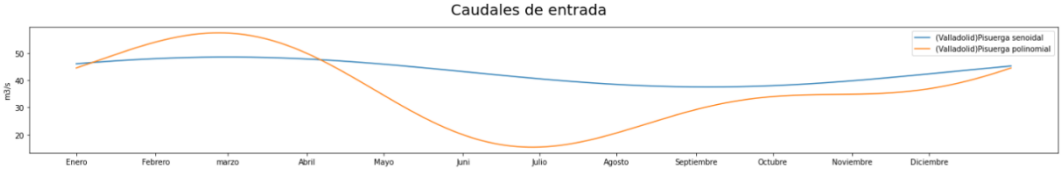


Figura 34: Comparación del modelo del río Pisuerga simple con el modelo complejo

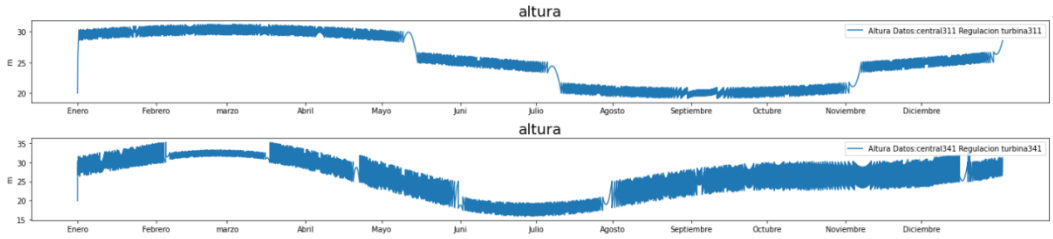


Figura 35: Comparación de la simulación del modelo caudal del río Pisuerga con el polinomio interpolador trigonométrico y el modelo simple

7.6 CONSUMOS

7.6.1 Consumo hídrico

Para calcular el consumo hídrico lo que vamos a hacer es recoger datos de distintas zonas con distintas densidades de población, información que podemos obtener del Instituto Nacional de Estadística (45). También recopilaremos datos sobre la demanda por individuo en España.

Para realizar las simulaciones se va a emplear una demanda de agua constante dado que no tenemos información de su fluctuación a lo largo del día para distintos rangos de caudales. Después se observará el comportamiento del sistema.

7.6.1.1 Consumo en distintas ciudades

En este apartado vamos a representar el consumo de agua en tres municipios españoles (Figura 36) Bercero, Toro y Valladolid. Tres localidades que representan las necesidades de un pueblo, una ciudad pequeña y una ciudad. La población respectivamente es de 191, 8532 y 297225 habitantes y se considerará un consumo medio por habitante de 136 l/día (45).

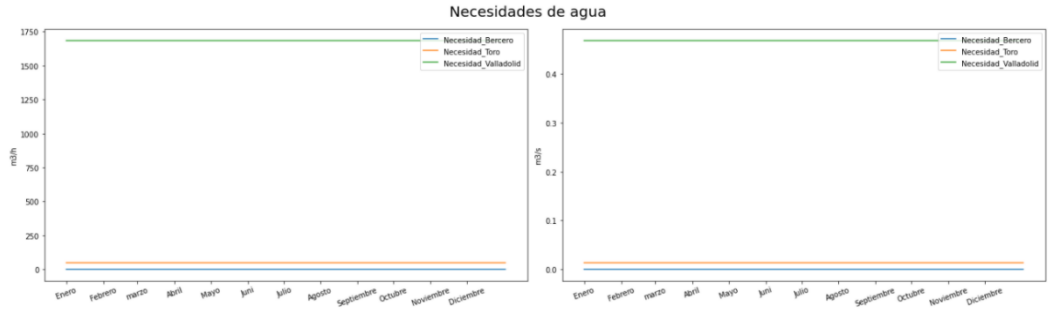


Figura 36: Consumo de agua en los municipios de Bercero, Toro y Valladolid

7.6.1.2 Simulación con necesidades de agua con ríos caudalosos

El primer ejemplo se realiza con el río Pisuerga con una central de regulación (Figura 37) y una central de bombeo (Figura 38) y una turbina y bomba dependiente de la altura de la central.

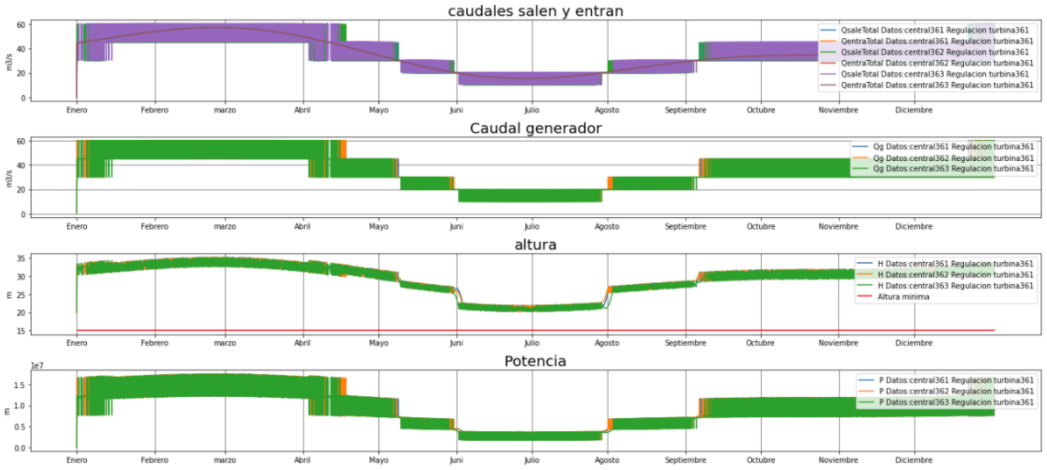


Figura 37: Representación de necesidades con central de regulación en el río Pisuerga

Se puede observar comparando la Figura 37 y la Figura 38 que, al no bajar la altura de la central por debajo de la altura mínima, el bombeo no se activa y el comportamiento con las tres necesidades en ambos casos es idéntica. Para los sistemas de centrales de regulación, las centrales que responden a las necesidades de Bercero, Toro y Valladolid son, respectivamente, las centrales 361, 362, y 363. Para las centrales de bombeo, las centrales que responden a las necesidades de Bercero, Toro y Valladolid son, respectivamente, las centrales 231, 232 y 233.

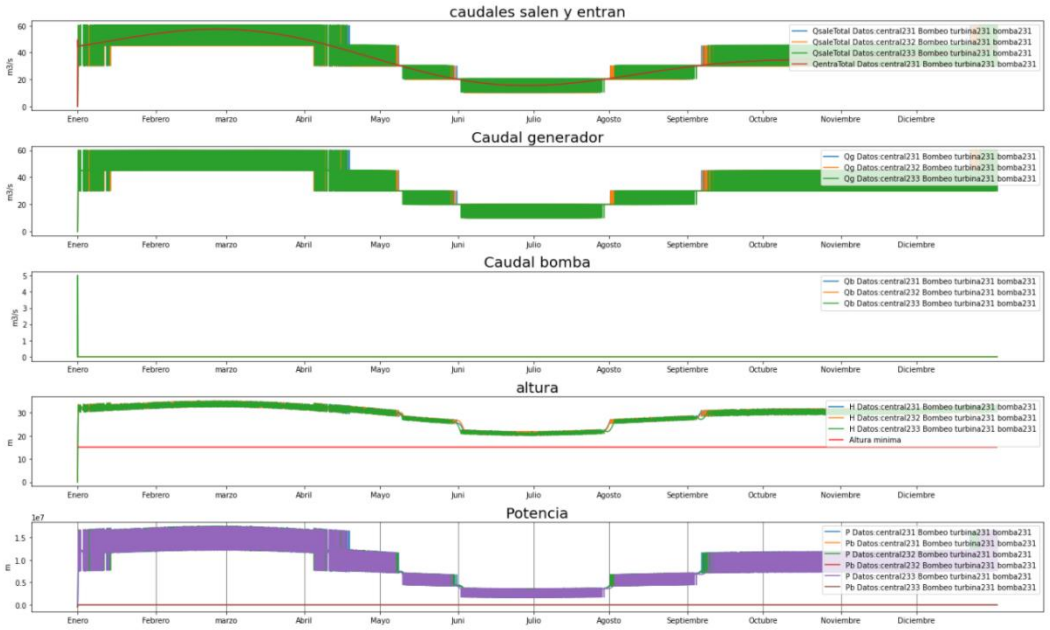


Figura 38: Representación de necesidades con central de bombeo en el río Pisuerga

7.6.1.3 Simulación con necesidades de agua con ríos pequeños

Ahora, con un río menos caudaloso (río Támega), podemos observar el comportamiento del sistema en una central de regulación (Figura 39) y una central hidroeléctrica de bombeo (Figura 40).

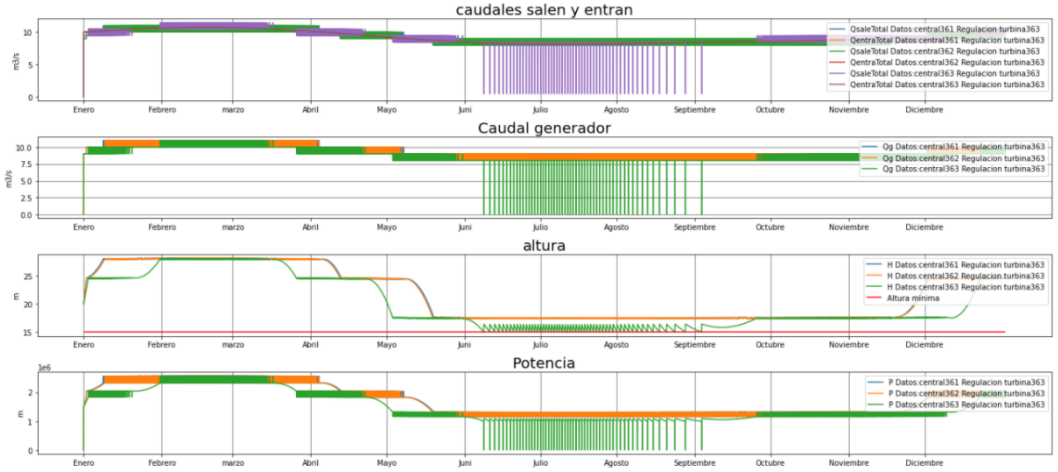


Figura 39: Representación de necesidades con central de regulación en el río Támega

En este caso, podemos observar que en la representación de la central de bombeo para las necesidades de agua de Valladolid (central 233 de la Figura 40) el comportamiento de la central es distinto que en el caso de una central de regulación (central 263 de la Figura 39). Se puede apreciar cómo la altura permanece entre las alturas fijadas para el cambio de caudal del generador.

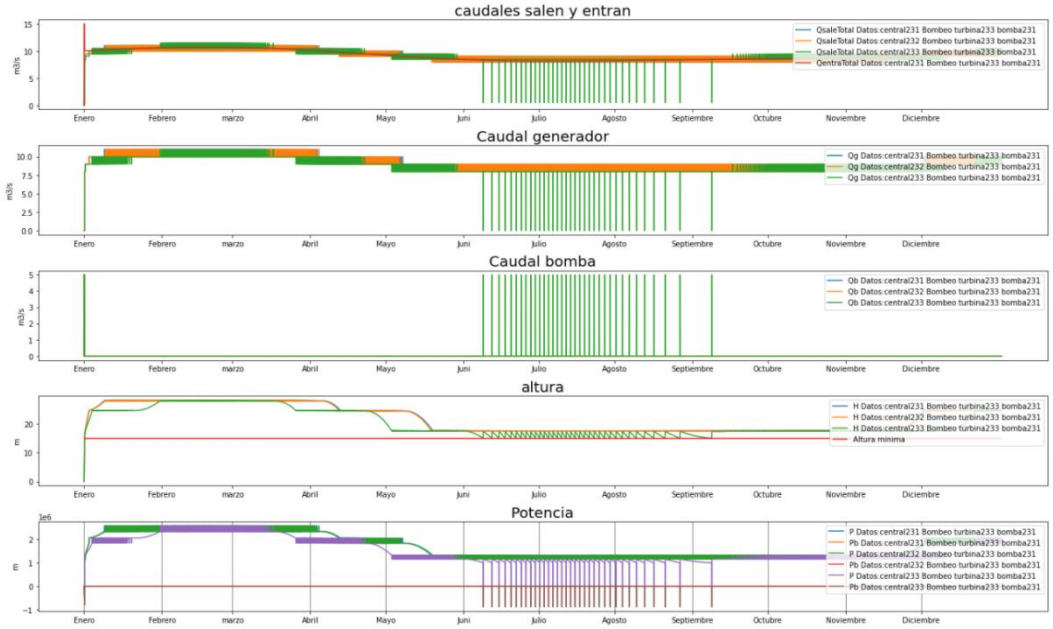


Figura 40: Representación de necesidades con central de bombeo en el río Támega

7.6.2 Potencia

En este Apartado se va a modelar la demanda de potencia diaria, semanal y anual. Posteriormente se simulará la demanda diaria. No del año entero, pues al tener el algoritmo del cálculo del caudal de la bomba una ecuación no lineal el tiempo de cómputo es excesivo.

7.6.2.1 Modelado de la demanda de potencia

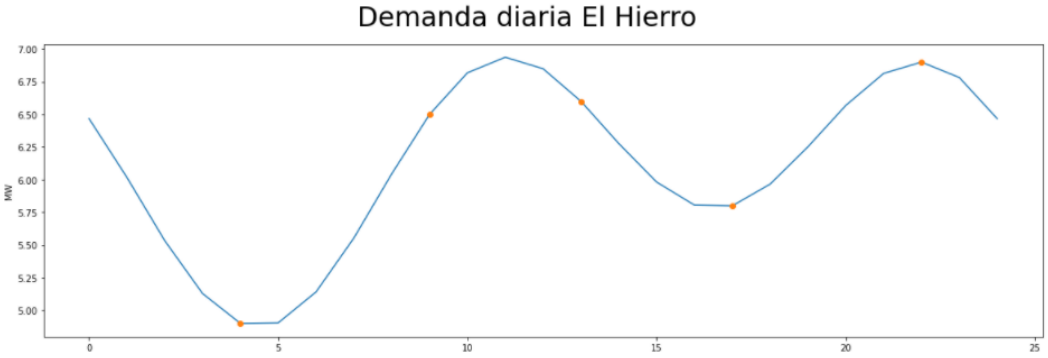


Figura 41: Modelo polinomio interpolador trigonométrico de la demanda diaria de la isla de El Hierro

Los datos para la realización del modelo se han obtenido de (46)

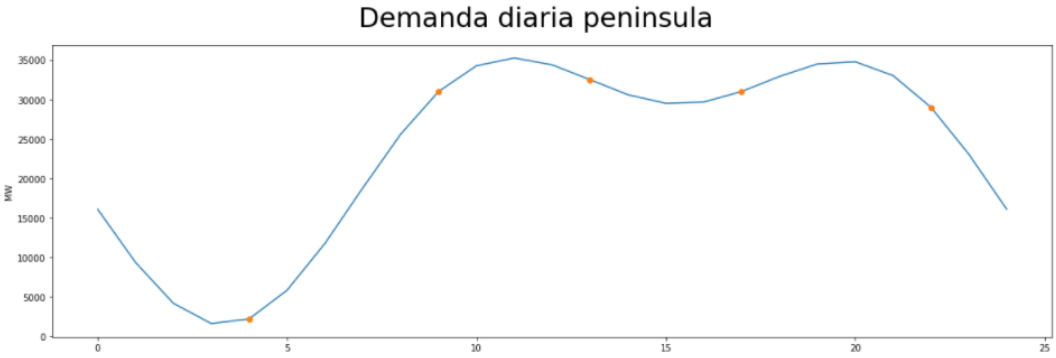


Figura 42: Modelo polinomio interpolador trigonométrico de la demanda diaria de la España peninsular

Los datos para la realización del modelo se han obtenido de (46)

Como podemos ver en la Figura 41 y en la Figura 42, de las 12 de la noche a las 6 de la mañana el uso de potencia es menor mientras que a partir de las 7 de la mañana y hasta las 22 de la noche es mayor el consumo. Se puede observar cómo hay cierta relación entre el horario de sueño de la población general y la potencia demandada.

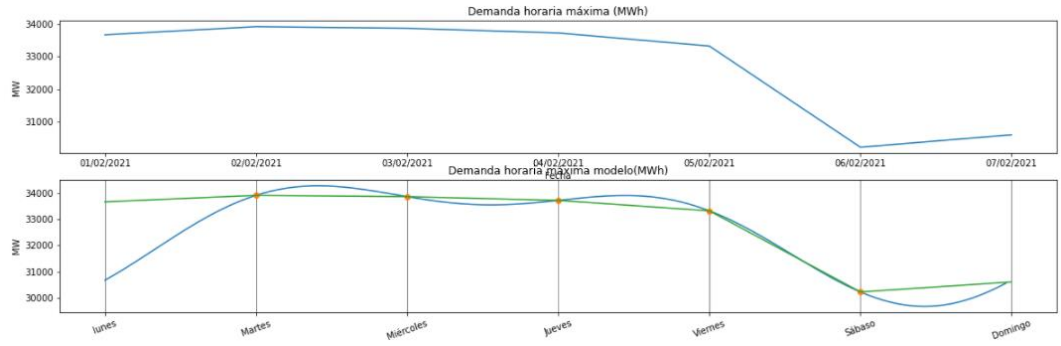


Figura 43: Modelo polinomio interpolador trigonométrico de la demanda semanal de la España peninsular

Los datos para la realización del modelo se han obtenido de (46)

En la Figura 43 se puede observar la demanda semanal. En la gráfica superior se muestran los datos obtenidos de (46) y en la gráfica inferior el resultado proporcionado por el modelo. Se observa que la potencia demandada es inferior en el fin de semana.

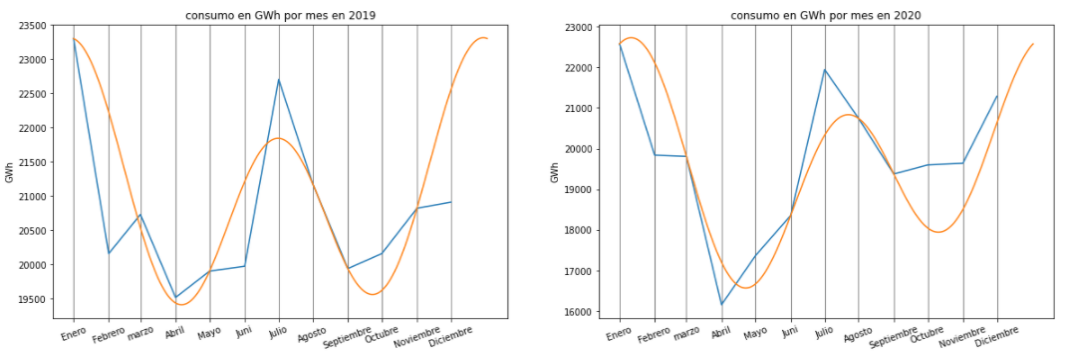


Figura 44: Modelo polinomio interpolador trigonométrico de la demanda anual de la España peninsular

Los datos para la realización del modelo se han obtenido de (46)

En la Figura 44 se muestra la demanda mensual, a la izquierda la de 2020 y a la de la derecha la de 2021. Como cabe esperar, se observa que en las estaciones con las temperaturas extremas la demanda es superior.

7.6.2.2 Simulación

Para la simulación solo se usarán los datos de consumo diario de la Isla del Hierro en una semana, pues el tiempo de cómputo para un intervalo temporal más grande es superior al admisible. Por esta razón, se elegirá una semana y, aun cuando el comportamiento en cada día es diferente, este período nos proporcionará información relevante sobre la variación.

A continuación, vamos a presentar un ejemplo de un caso en el que sería muy útil el empleo de una central de bombeo. Para la central de bombeo supondremos que cuando la demanda es superior a la demanda media lo que funciona es la turbina y cuando la demanda es inferior a la demanda diaria lo que funciona es la bomba, no pudiendo funcionar ambas (la turbina y la bomba) simultáneamente. Suponemos una fuente constante de energía igual a la demanda media y que la central hidroeléctrica de bombeo se emplea para generar y compensar esta potencia generada (a través del empleo de la turbina o la bomba). En la Figura 45 se muestra la simulación.

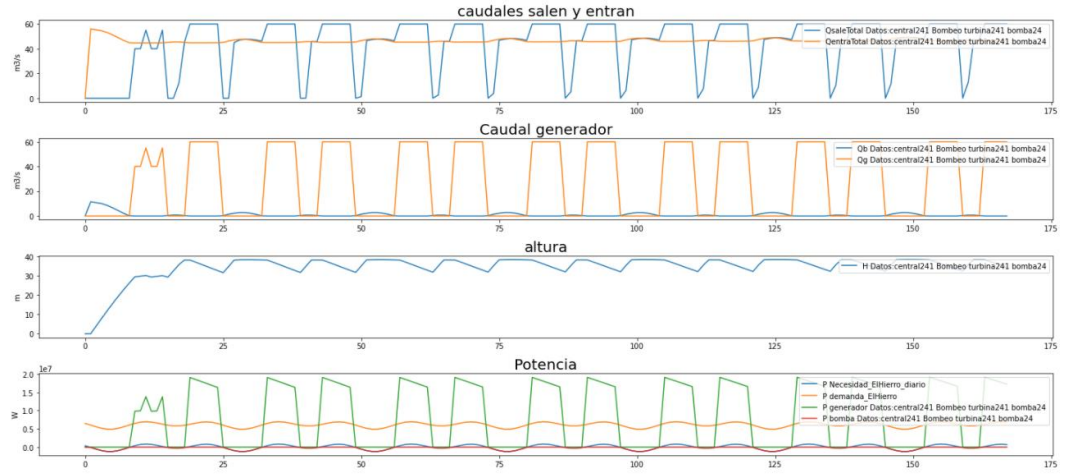


Figura 45: Simulación de central de bombeo con el control de la bomba por las necesidades de potencia

El problema en este sistema es que la altura en la que permanece durante la mayor parte del tiempo es la misma a la que se encuentra el aliviadero. Esa es la causa por la que hay salida de caudal aun siendo el turno de la bomba y el motivo por el que este sistema no podría estar controlado solamente por las necesidades de potencia, sino que también habría que controlar el nivel de las aguas para evitar desbordamientos.

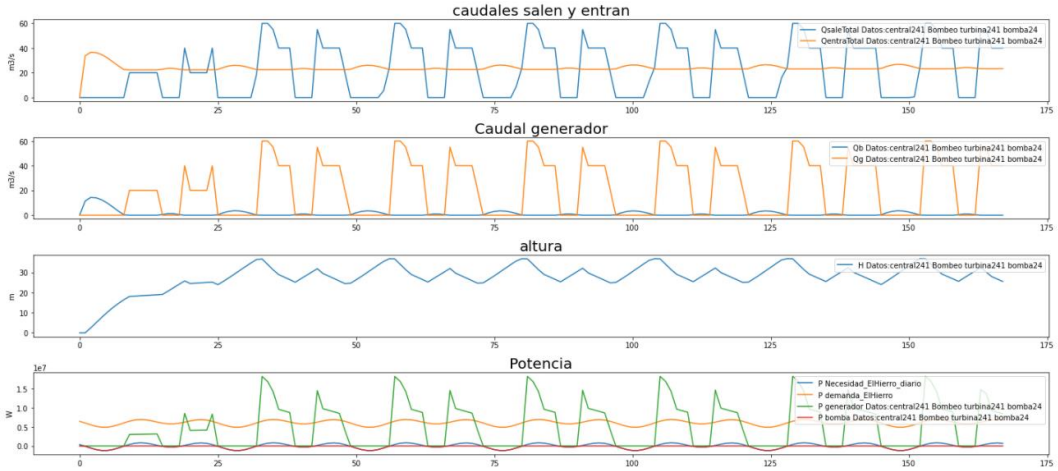


Figura 46: Simulación de central de bombeo con el control de la bomba por las necesidades de potencia con la mitad del caudal del río Pisuerga

Para una segunda simulación (Figura 46) se emplea un caudal de entrada que será la mitad del utilizado en el ejemplo anterior. Como se puede observar, el caudal de salida se corresponde en su totalidad con el caudal que pasa por la turbina. Otra conclusión a la que se puede llegar es que se sigue satisfaciendo la demanda

Capítulo 8. SISTEMA DE MÚLTIPLES CENTRALES

En este capítulo vamos a hablar de los distintos elementos que componen la interconexión de las centrales. Entre estos elementos están los que constituyen la interconexión directa (de las turbinas), las interconexiones entre distintas centrales, la inversa (las bombas de las centrales reversibles) y los retardos.

Los notebooks con las simulaciones de este apartado se pueden encontrar en el Anexo 7.2 y en el Anexo 7.3.

8.1 INTERCONEXIÓN DIRECTA (DEL CAUDAL DEL GENERADOR)

En este apartado se explicará la estructura de la programación para la interconexión del caudal de salida de las centrales. Para ello hay que plantearse: ¿todo el volumen de salida llega a la central? La respuesta es negativa. El volumen de agua que va a las necesidades no irá a la próxima central, ni las involucradas en el bombeo, tema que se abordará en el apartado 8.3. Así pues, se requieren dos variables, QsaleTotal y QsaleRio, la primera es el caudal que en cada periodo t sale de la central y la segunda variable es el caudal que saldrá por la cuenca natural del río.

```
def interconexion__QentraCentral(listDatos, listPresa):
    '''VERSIÓN 1
    Realiza las interconexiones directa, procedente de los caudal de salida de otras centrales

    ENTRADAS:
        -listDatos: lista de objetos de la clase Presa_Datos(datos en el sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA: realizar las interconexiones caudal de salida - caudal de entrada
    '''
    for n in range(len(listDatos)):
        listDatos[n].NumPresaEntrada = []
        for m in range(len(listDatos)):
            if listPresa[n].Nombre == listPresa[m].Presa_de_salida:
                listDatos[n].NumPresaEntrada += [m]

    return listDatos
```

Figura 47: Función interconexion_QentraCentral

Para los cálculos de este tipo de centrales lo que se ha hecho ha sido crear dos funciones denominadas interconexión_QentraCentral (Figura 47) y actualizar_QentraCentral (Figura 48), de forma que la primera crea el enlace a través de la central de salida que se ha especificado en la inicialización del objeto y la otra va actualizando la variable QentraCentral en cada instante de tiempo t. Dentro de la última función se recorre la lista listaDatos (lista de objetos de la clase Presa_Datos) para actualizar los valores de dicha variable en todos los objetos de la lista.

```
def actualizar_QentraCentral(t, listDatos, listPresa):
    """Versión  1.

    Actualiza los caudales de entrada procedente de otras centrales en el
    instante t
    ENTRADAS:
        -t: instante de tiempo de simulación
        -listDatos: lista de objetos de la clase Presa_Datos (datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA:
        -Lista de objeto datos
    """
    for n in range(len(listDatos)):
        for m in listDatos[n].NumPresaEntrada:
            retardo = t-listPresa[m].retardo_salida
            listDatos[n].Qe_central[t] += listDatos[m].QsaleRio[retardo]

    return listDatos
```

Figura 48: Función actualizar_QentraCentral

En la Figura 49 se puede ver cómo se utilizan estas dos funciones, antes del bucle temporal, para realizar estas interconexiones, y en el bucle temporal, para actualizar los valores.

```
def sistema_multiderivacionfluyente(t_total, listPresa, necesidad, g=9.8,
                                    den=998, intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con múltiples centrales fluyentes y
    derivación
    ENTRADAS:
        -t_total: lista con cada índice de la simulación desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """
    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos + [Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos, listPresa= listPresa)
    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos = actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] = instante_t_derivacionfluyente(t, datos=listDatos[i], presa=listPresa[i], necesidad=necesidad[i]
                                                g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            listDatos = actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] = instante_t_derivacionfluyente(t=t, datos=listDatos[i], presa=listPresa[i], necesidad=necesidad,
                                                        g=g, den=den, intervalo=intervalo)
    return listDatos
```

Figura 49: Función sistema_multiderivacionfluyente

8.1.1 Ejemplo con centrales de regulación

La primera simulación que realizaremos corresponde a la central de regulación que cumple con el esquema de la Figura 50.

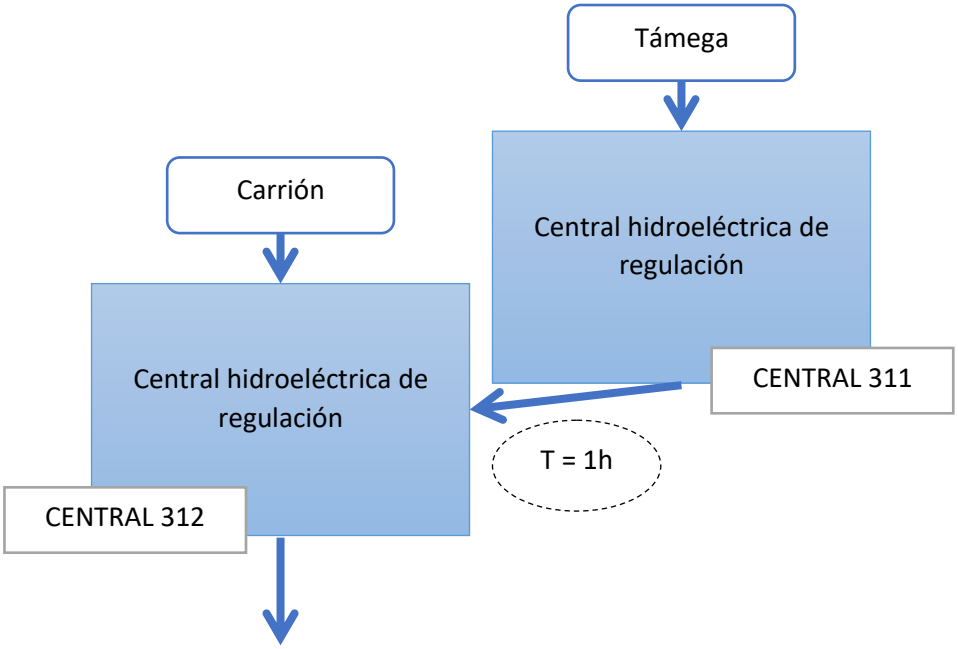


Figura 50: Esquema de interconexión entre centrales de regulación

Los caudales de entrada son respectivamente los del rio Támega para la central 311 y Carrión para la central 312, además del caudal que salga de la central 311. Además, se ha elegido una base de 10.000 m^2 y una altura máxima de 30 m y mínima de 10 m. La turbina de estas centrales es donde el caudal del generador depende de la altura actual de la central.

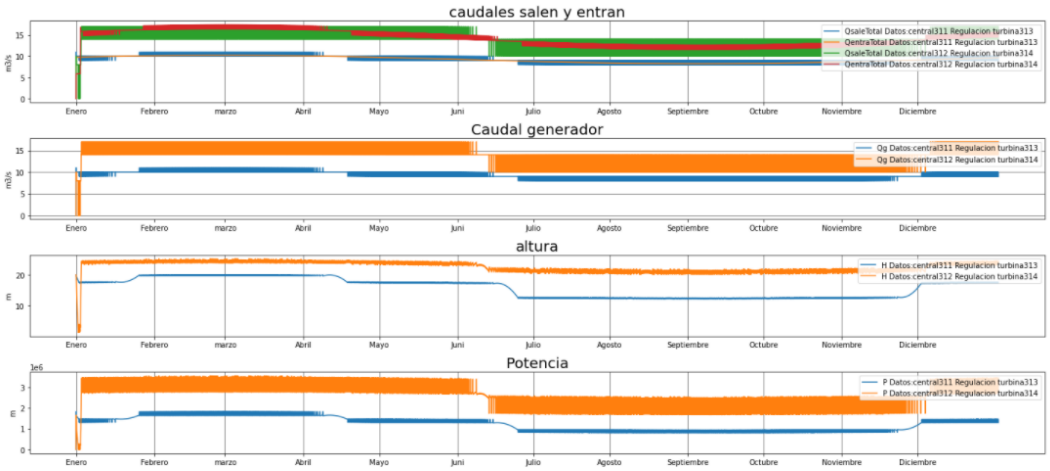


Figura 51: Representación de la simulación entre dos centrales de regulación

El comportamiento del sistema se muestra en la Figura 51. Se puede observar cómo el caudal que entra en la central 312 viene afectado por el caudal que sale por la central 311. La potencia generada por la central también se ve correlacionada con el caudal del

generador (así coinciden la oscilación de la potencia con las del caudal) y la altura (al ser directamente proporcional a esta, cuando la altura disminuye, la potencia generada también disminuye).

8.1.2 Ejemplo con centrales fluyentes

Ahora se va a realizar una interconexión entre dos centrales fluyentes, con el río Támega y el río Carrión como caudales de entrada en cada una. Se puede ver el esquema de las interconexiones de esta simulación en la Figura 52.

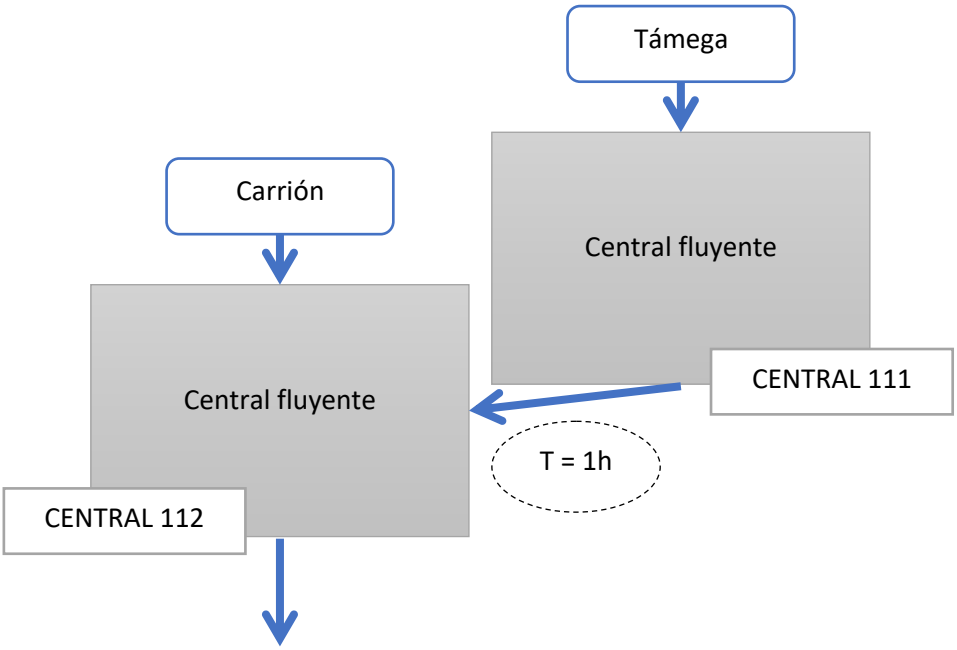


Figura 52: Esquema de interconexión entre dos centrales fluyentes

En la representación de los resultados de este sistema (Figura 53) se puede observar que la central 112 está compuesta por el caudal del río de entrada y por el caudal de salida de la central 111.

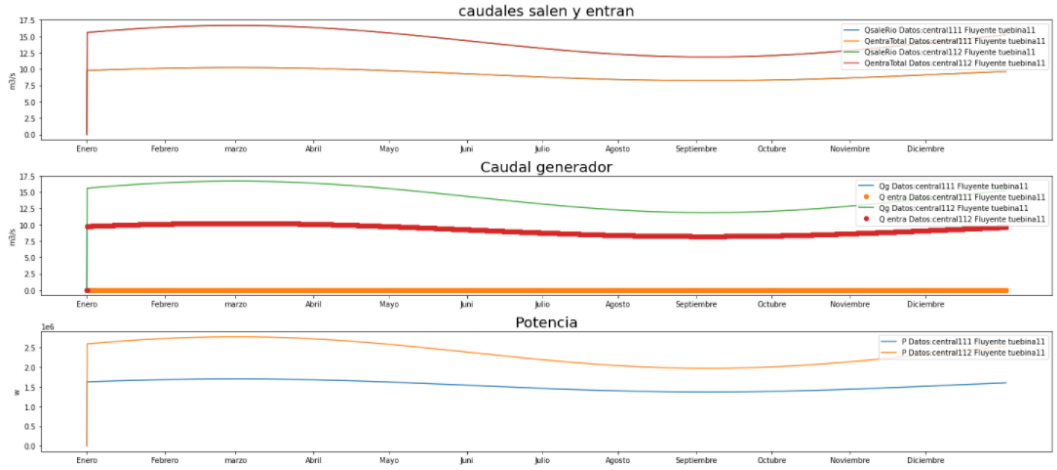


Figura 53: Representación de la simulación entre dos centrales fluyentes

8.1.3 Ejemplo con central de derivación

En la siguiente simulación podemos observar la interconexión entre dos centrales de derivación. El esquema de esta simulación se muestra en la Figura 54. Estas dos centrales tienen como entrada el río Támega y el rio Carrión respectivamente. Además, el caudal de salida de la central 211 (la alimentada por el río Támega) también es un caudal de entrada de la central 212.

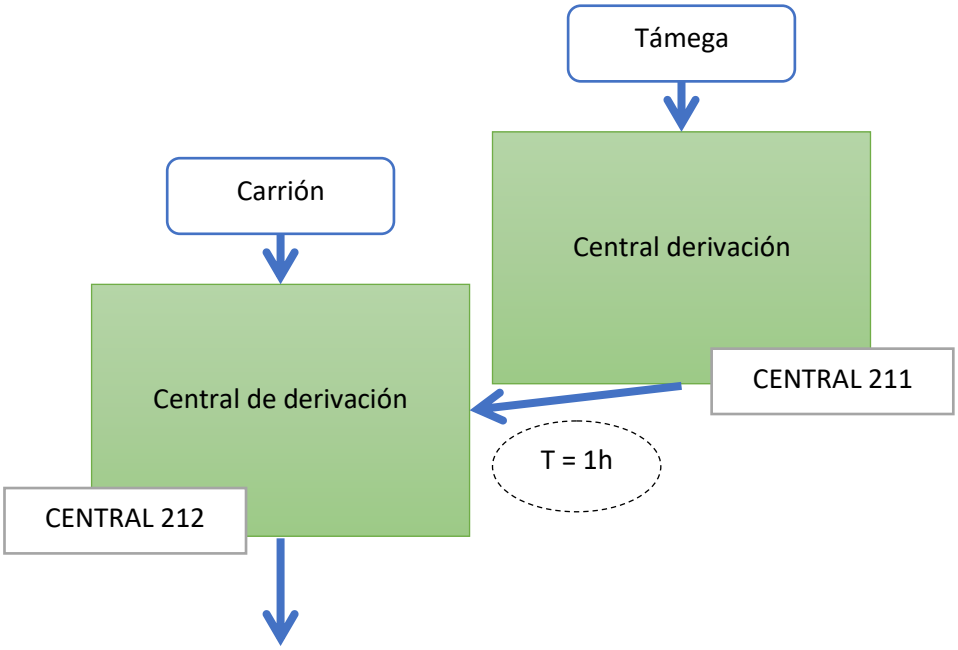


Figura 54: Esquema de interconexión entre dos centrales de derivación

Si se observan las gráficas de la Figura 55, se puede ver cómo la central 212 se ve afectada por la entrada de la central 211.

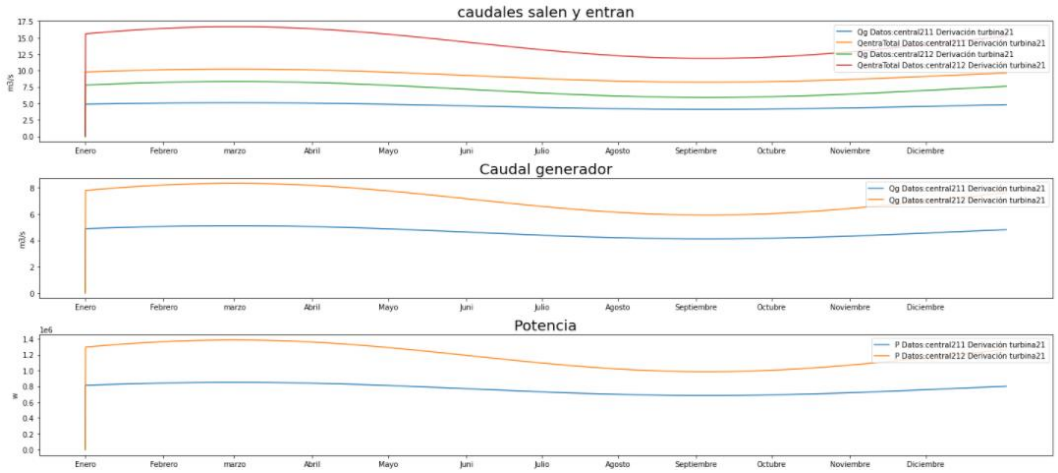


Figura 55: Representación de la simulación entre centrales de derivación

8.2 INTERCONEXIÓN MIXTA

En este apartado se van a interconectar distintos tipos de centrales. Para ello emplearemos las dos funciones necesarias para la interconexión, que ya fueron utilizadas en el apartado anterior, y una sentencia condicional que nos permita acceder a la función instante_t_ que le corresponda a cada tipo de central.

En la Figura 56 podemos ver cómo el atributo Tipo_central que tenía antes la clase Presa ahora es empleado para elegir qué tipo de función se debe ejecutar. También se observan las funciones involucradas en las interconexiones entre distintas centrales.

```
def sistema_multicentral(t_total, listPresa, necesidad, g=9.8, den=998,
                         intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con múltiples centrales
    ENTRADAS:
        -t_total: lista con cada índice de la simulación desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """

    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos + [Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos, listPresa= listPresa)

    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos = actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                if (listPresa[i].Tipo_central[0] == 'F' or listPresa[i].Tipo_central[0] =='D'):
                    listDatos[i] = instante_t_derivacionfluyente(t, listDatos[i], presa=listPresa[i], necesidad=necesidad[i])

                else:
                    listDatos[i] = instante_t_regulacion(listDatos[i], t, necesidad=necesidad[i],
                                                         presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            listDatos = actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                if listPresa[i].Tipo_central[0] == 'F' or listPresa[i].Tipo_central[0] =='D':
                    listDatos[i] = instante_t_derivacionfluyente(t, listDatos[i], presa=listPresa[i], necesidad=necesidad)

                else:
                    listDatos[i] = instante_t_regulacion(listDatos[i], t, necesidad=necesidad,
                                                         presa=listPresa[i], g=g, den=den, intervalo=intervalo)

    return listDatos
```

Figura 56: Función sistema_multicentral

8.2.1 Ejemplo con interconexión de regulación y derivación

Se va a simular ahora la interconexión entre una central de regulación y una central fluyente siguiendo el esquema de la Figura 57. Las entradas serán el río Támega y el río Carrión, respectivamente.

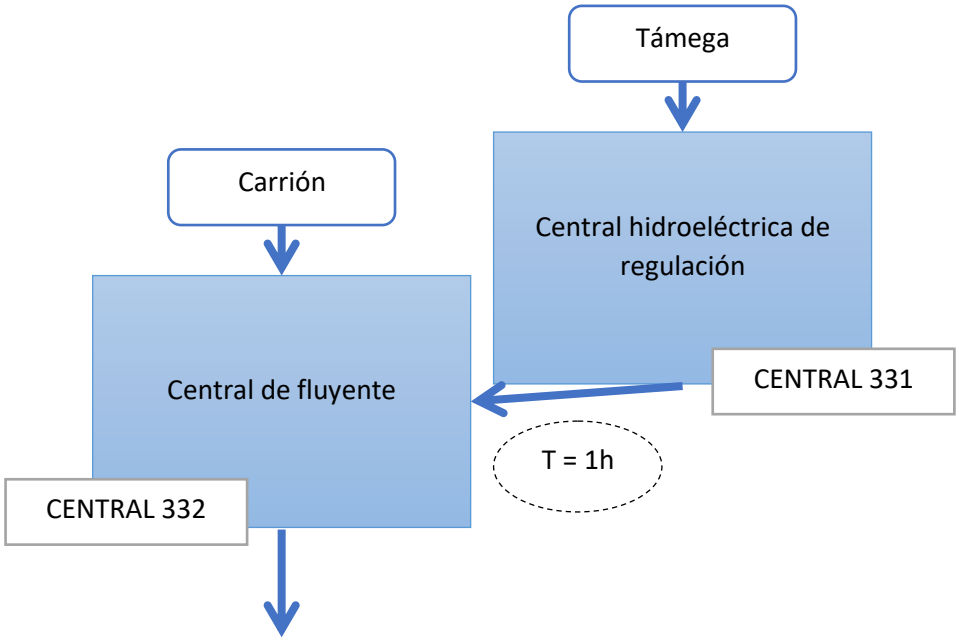


Figura 57: Esquema de la interconexión entre una central de regulación y una central de derivación

En la Figura 58 se representan los resultados. Se puede ver cómo la entrada de caudal de la central fluyente se ve afectada por la central de regulación.

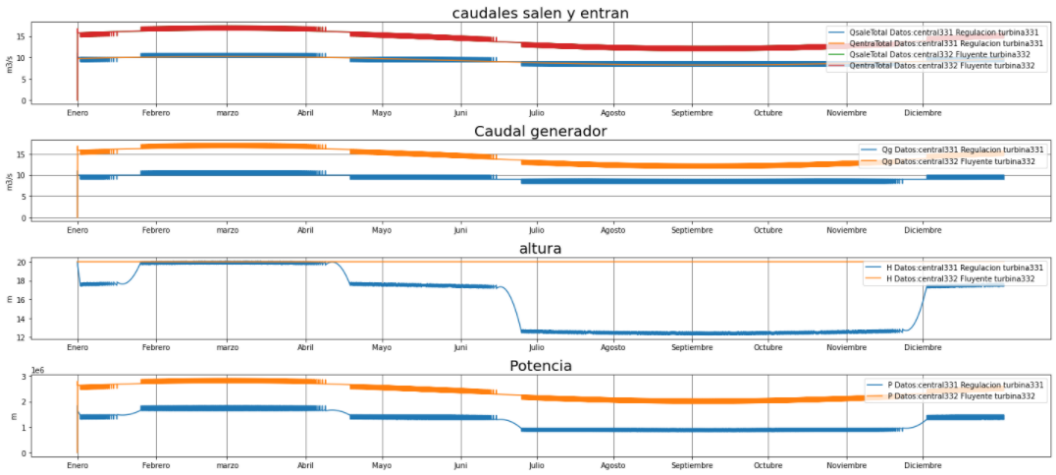


Figura 58: Representación de la simulación entre una central de regulación y una central de derivación

8.3 INTERCONEXIÓN INVERSA (BOMBEO)

En esta sección el modelo va a incorporar la comunicación entre centrales a través del bombeo de agua. En las centrales de bombeo se usan las funciones para la interconexión directa (por turbina), empleada en el modelado de las centrales de regulación, y otras dos funciones similares, una para realizar la interconexión por bombeo y otra para indicar que el caudal entrante por bombeo de la central reversible (Qm) ha de estar reflejado en el caudal de salida (Qs_central) de la central de la que lo obtiene.

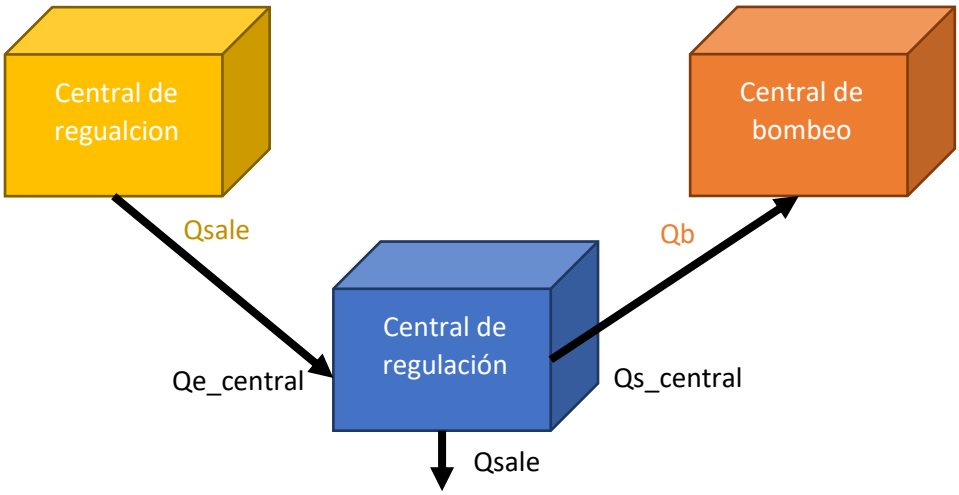


Figura 59: Esquema de las listas del objeto Presa_datos en las interconexiones

```
def interconexion__QsaleCentral(listDatos, listPresa):
    '''VERSIÓN 1
    Realiza las interconexiones de bombeo

    ENTRADAS:
        -listDatos: lista de objetos de la clase Presa_Datos(datos en el sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA: realizar las interconexiones por bombeo
    '''
    for n in range(len(listDatos)):
        listDatos[n].NumPresaBombeo = []
        for m in range(len(listDatos)):
            if listPresa[n].Nombre == listPresa[m].Presa_de_absorcion:
                listDatos[n].NumPresaEntrada += [m]

    return listDatos
```

Figura 60: Función interconexion_QsaleCentral

Las funciones para la interconexión por bombeo funcionan del mismo modo que en la interconexión directa salvo la diferencia de que en el mismo periodo que llega el caudal a la central de bombeo (Qb) ese caudal ha de salir de la central de la que lo obtiene (Figura 61). Por lo tanto, la estructura del programa es distinta a la de la interconexión mixta directa (en la que no hay centrales de bombeo) dado que, en este caso, en cada periodo hay que ejecutar primero los selectores del caudal de bombeo (selector_Qb) para posteriormente actualizar los caudales de salida de las centrales (Qs_central), con el fin de que esté reflejado el caudal que sale de la central por el bombeo de una central superior en la simulación al mismo tiempo t.

```
def actualizar_QsaleCentral(t,listDatos, listPresa):
    '''VERSIÓN 1
    Actualiza los caudales de salida procedente del bombeo de otrasa centrales en el instante t

    ENTRADAS:
        -t: instante de tiempo de simulación
        -listDatos: lista de objetos de la clase Presa_Datos(datos en el sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA: objeto datos actualizado en el instante t
    '''
    for n in range(len(listDatos)):
        for m in listDatos[n].NumPresaEntrada:
            listDatos[n].Qs_central[t]+=listDatos[m].Qb[t]
    return listDatos
```

Figura 61: Función actualizar_QsaleCentral

Para poder ejecutar el selector_Qb en las simulaciones del sistema múltiple, la función que simula las bombas está dividida en instante_t_bombeoM1 e instante_t_bombeoM2. La función bombeoM1 actualiza V, H y Qb y se ejecuta antes que las demás funciones instante_t_. Posteriormente, como se hacía en los apartados 0 y 8.2 se ejecutan las funciones instante_t_ de todas las centrales del sistema (con instante_t_bombeoM2 para las de bombeo).

8.3.1 Ejemplo de interconexión de central de bombeo y de regulación

Ahora vamos a simular una central de regulación con bombeo, para ello consideraremos caudales modelados mediante sinusoidales simples. Los caudales de entrada serán, respectivamente, el río Pisuerga – 10m^3/s y el río Duero al 50% para la central de bombeo 311 y la central de regulación 312 (Figura 62). Estos valores se han elegido para analizar el efecto de la escasez de agua.

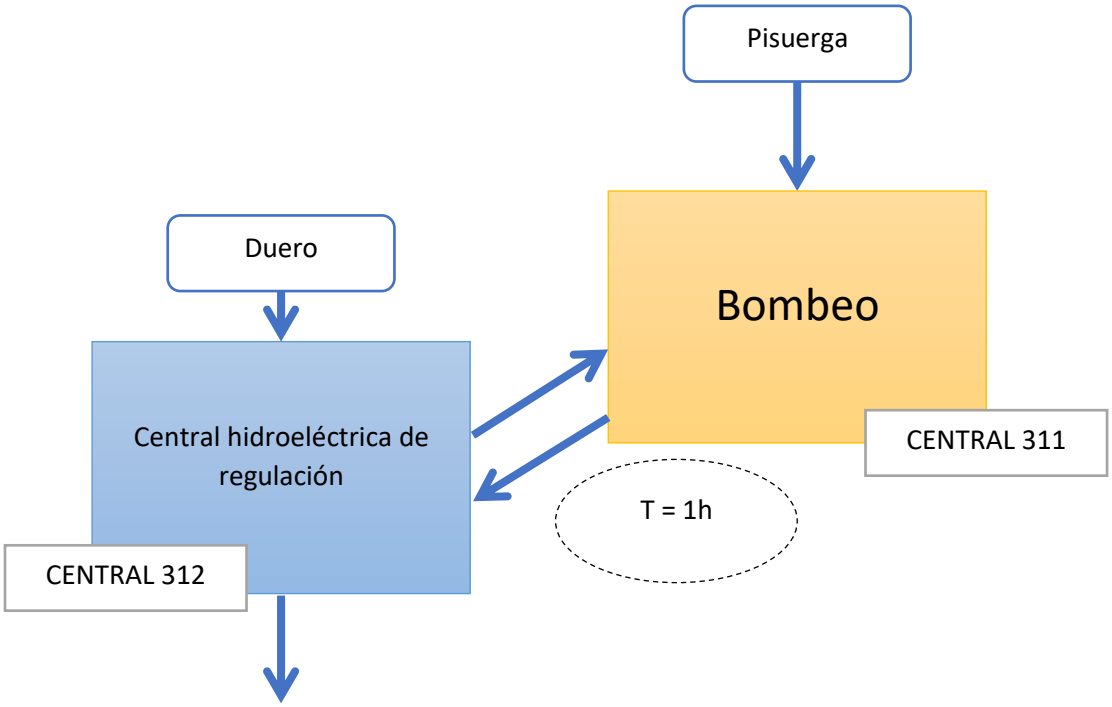


Figura 62: Esquema de interconexión entre central de regulación y reversible o de bombeo

Como se puede apreciar en la Figura 63, cuando la altura de la central de bombeo baja de la altura mínima, la bomba se pone en funcionamiento. Esto provoca un aumento del caudal de entrada de la central 311 y un aumento de la caudal de salida de la central 312.

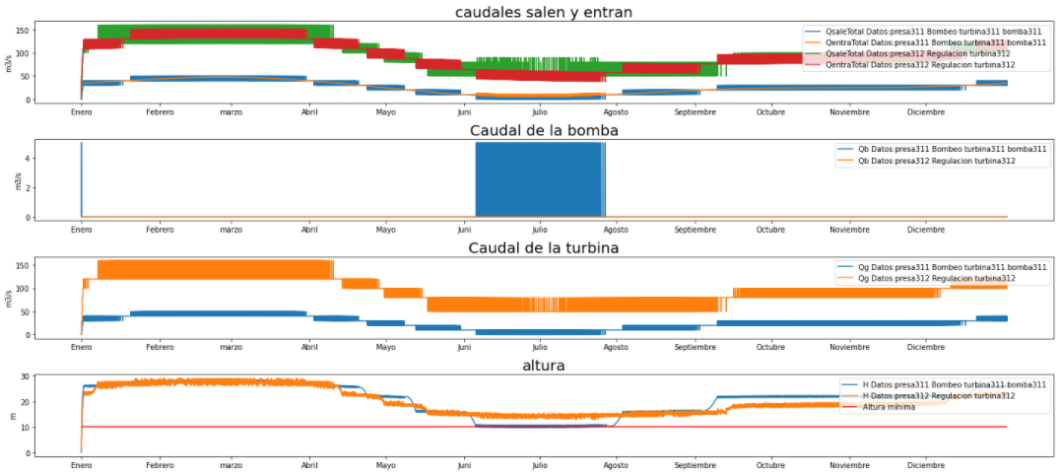


Figura 63: Representación de la simulación entre una central de regulación y una central reversible o de bombeo

8.4 RETARDOS

En este apartado vamos a hablar de la implementación del retardo entre el caudal de salida de una central (QsaleRio) y el caudal de entrada de la central de llegada (Qe_central). El retardo es el tiempo que tarda el caudal del río en llegar a la siguiente

central (de forma directa). El atributo de retardo (retardo_salida) se puede observar en la Figura 47.

8.4.1 Ejemplo: central de regulación

Vamos ahora a observar el comportamiento del sistema de dos centrales de regulación para un retardo de 40h (Figura 64).

Nótese que la altura de la central se encuentra entre su máximo (30m) y su mínimo (2m).

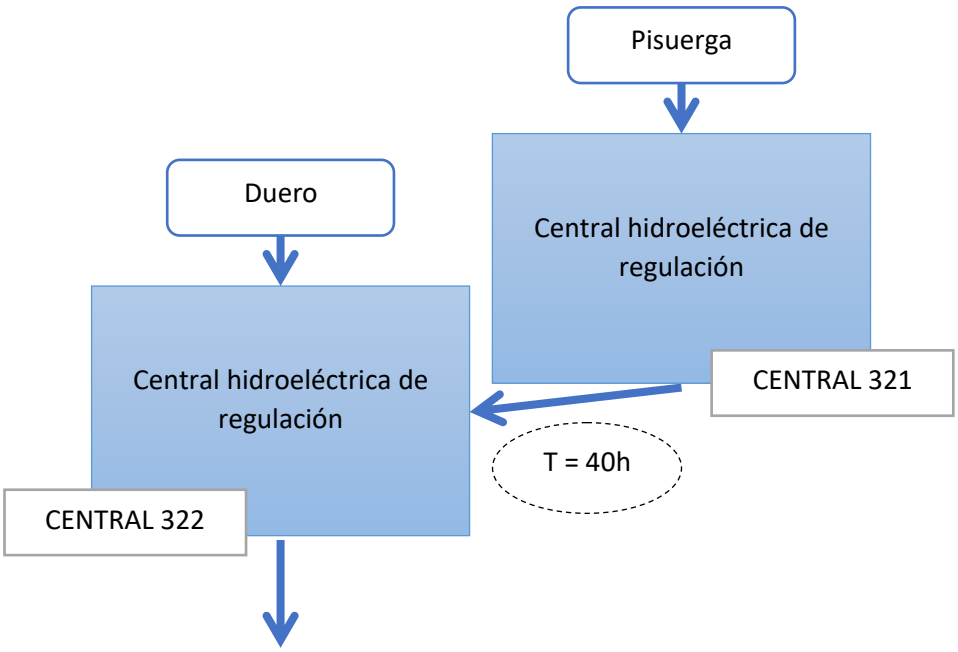


Figura 64: Esquema de interconexión de dos centrales de regulación con retardo

Como podemos observar en la simulación del sistema con retardo (Figura 65), el sistema se comporta de manera similar al sistema con el retardo normal de 1h de la Figura 51 pero con cierta demora en la llegada del caudal. Al principio de la simulación, cuando aún no ha pasado ese tiempo de demora, hay un tramo en el que aún no ha llegado nada al río receptor.

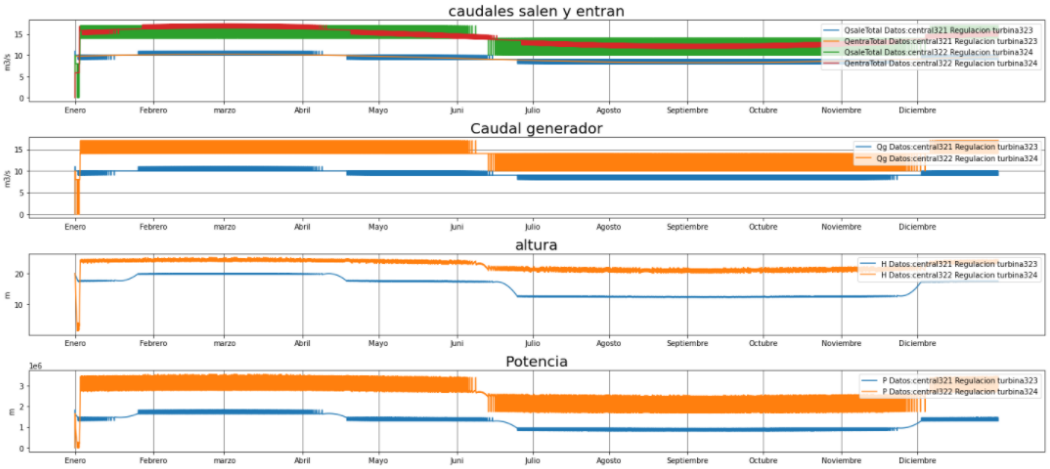


Figura 65: Representación de la simulación entre dos centrales de regulación con retardo

8.4.2 Ejemplo: dos centrales fluyentes

A continuación, se presentará la simulación con retardo de dos centrales fluyentes (Figura 66). La central 121 tiene como entrada de caudal el río Pisuerga y la 122 tiene como entrada el río Duero.

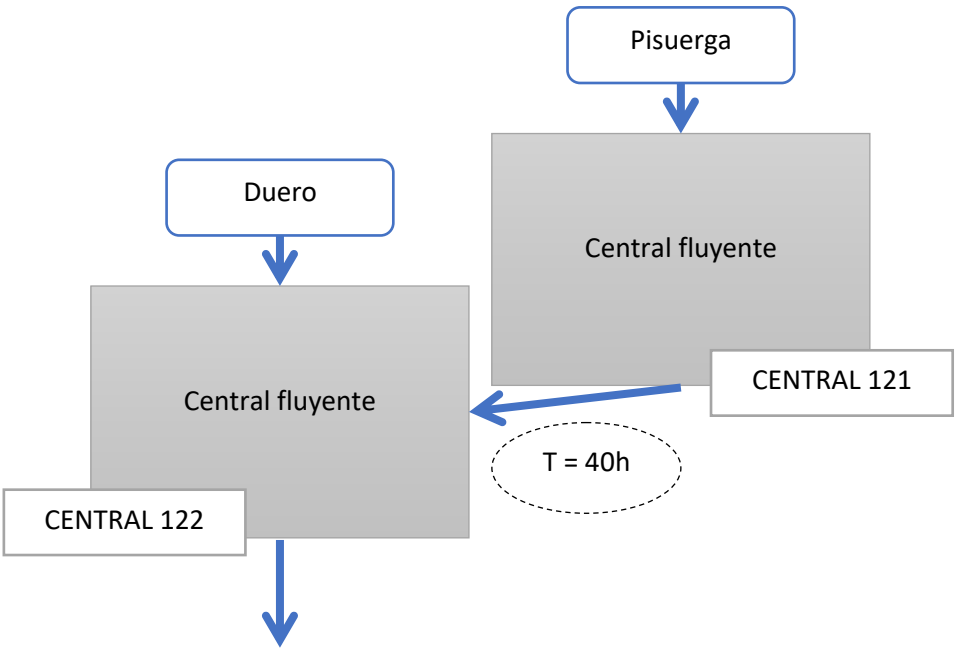


Figura 66: Esquema de interconexión de dos centrales fluyentes con retardo

Como se puede observar en la Figura 67, el caudal de salida de la central 121 llega con cierto retardo a la central 122. Al principio de la simulación hay un lapso de tiempo, de igual magnitud al retardo, en el que la central receptora aún no ha recibido el caudal procedente de la central 121.

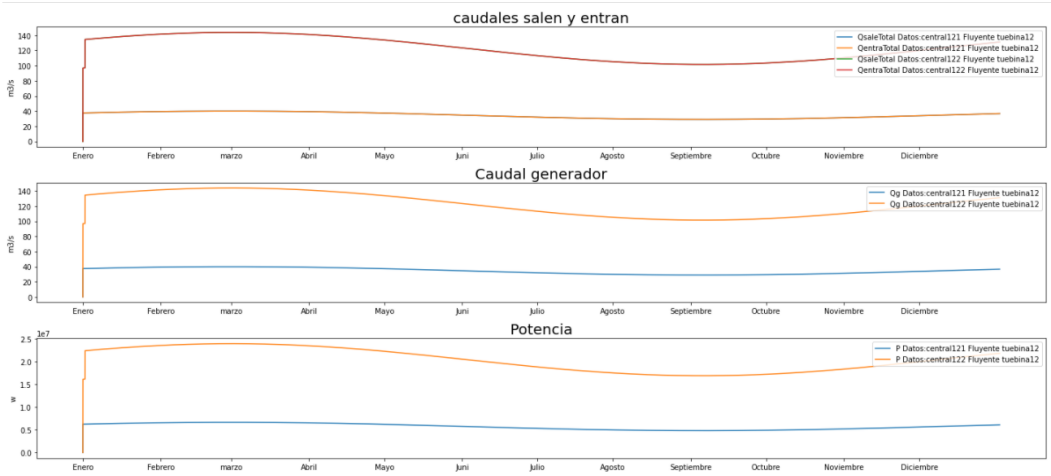


Figura 67: Representación de la simulación entre dos centrales fluyentes con retardo

8.4.3 Ejemplo central derivación

Por último, se realizará la simulación con retardo de dos centrales de derivación (Figura 68). La central 121 tiene como entrada de caudal el río Pisuerga y la 122 tiene como entrada el río Duero.

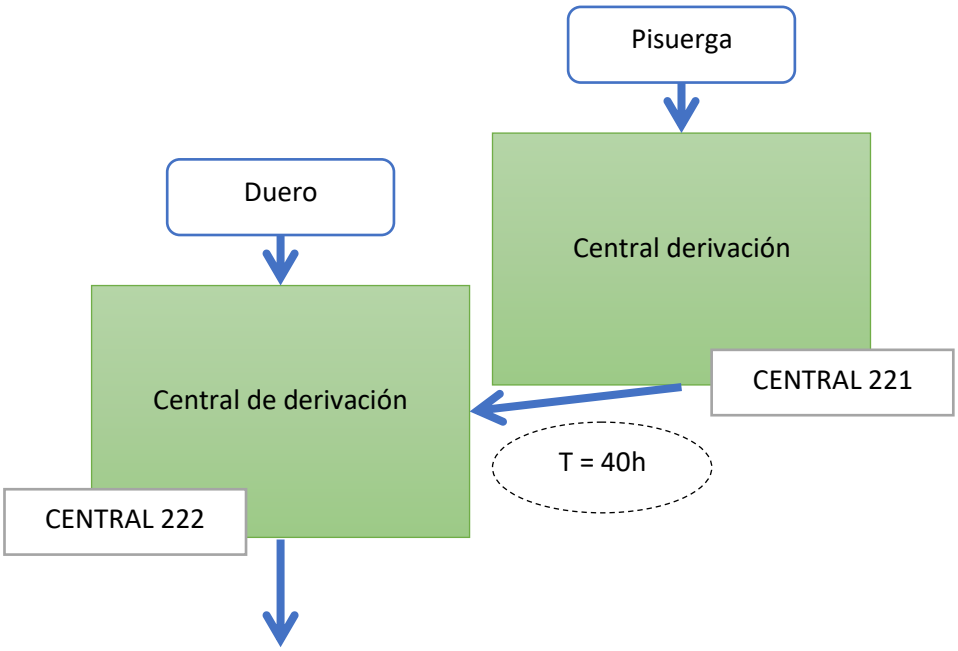


Figura 68: Esquema de interconexión de dos centrales de derivación con retardo

Como se puede observar en la Figura 69 el caudal de salida de la central 221 llega con retardo al caudal de entrada de la central 222, al igual que en los ejemplos anteriores.

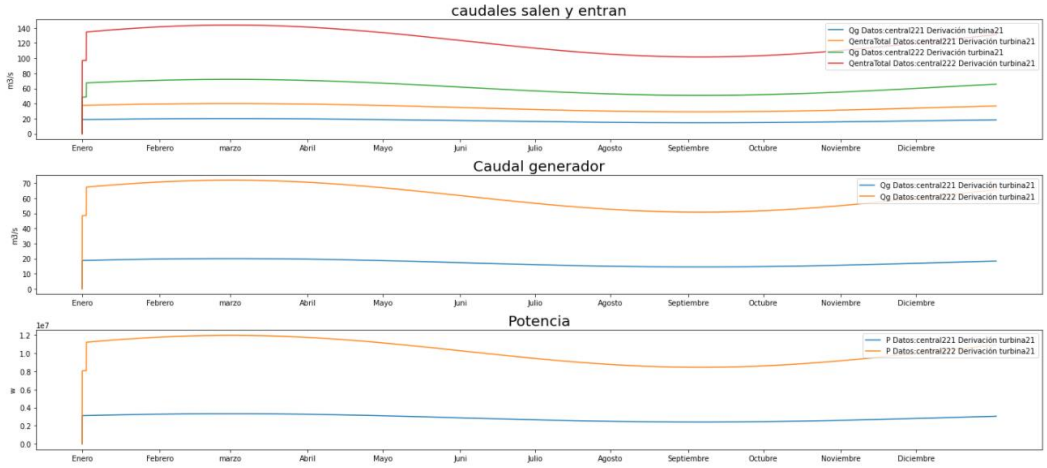


Figura 69: Representación de la simulación entre dos centrales de derivación con retardo

POSIBLES CAMPOS DE MEJORA

A continuación, se indican algunas ideas que apuntan posibles líneas de trabajo que podrían servir, en el futuro, para mejorar los resultados obtenidos en esta Memoria:

- Incorporar estrategias que posibiliten la manipulación del sistema fluyente para controlar su altura.
- Incluir en el modelo las variaciones de las necesidades hídricas y de potencia de la población debidas a los cambios estacionales de temperatura a lo largo del año.
- Añadir otras fuentes de energía renovables que sirvan como complemento a las centrales hidroeléctricas y analizar en qué condiciones las presas se pueden desacoplar del sistema de generación de energía eléctrica reservándolas para combatir los periodos de sequía.
- Incorporar técnicas de interpolación que permitan analizar intervalos temporales desprovistos de periodicidad en la descripción de los ríos y las demandas.
- Adoptar un enfoque continuo, frente a la discretización temporal empleada en esta Memoria, utilizando ecuaciones diferenciales para modelar tanto las centrales individuales como los sistemas de presas. El enfoque continuo abre la posibilidad de emplear técnicas numéricas diseñadas para EDOs. En particular, este enfoque permitiría construir modelos más precisos para describir el caudal del aliviadero y la absorción.
- Considerar esquemas de interconexión de tipo más general entre centrales, definidos en términos de grafos.
- Mejorar la programación de los notebooks, incorporando elementos gráficos interactivos (widgets), que faciliten su manejo y mejoren la experiencia del usuario.
- Reorganizar y dotar de más estructura a los módulos de Python desarrollados en este trabajo. En particular, se podría descomponer el módulo herramientas_presa en dos submódulos, uno de computación y otro de representación.
- Emplear un sistema de control de versiones, por ejemplo git, para mejorar y facilitar el desarrollo del software.

BIBLIOGRAFÍA

1. Carpio Ibañez, José. *Modelo de simulación de la explotación de un sistema hidrotérmico de generación eléctrica.* Universidad Politécnica de Madrid. Madrid : s.n., 1988. Tesis doctoral.

2. Pérez Diaz, Juan Ignacio. *Modelos de explotación a corto plazo de centrales hidroeléctricas.* Universidad Politécnica de Madrid. Madrid : s.n., 2008. Tesis doctoral.

3. Aquae. fundacionacuae.com. *fundación aquae.* [En línea] https://www.fundacionaquae.org/agua-cambio-climatico-efectos/.

4. Iberdola. Medio ambiente: iberdrola. *iberdrola.* [En línea] https://www.iberdrola.com/medio-ambiente/impacto-del-cambio-climatico.

5. El Tiempo, Meteored. Aparecen enormes agujeros en los campos de Turquía: ¿puede pasar aquí? [En línea] https://www.tiempo.com/noticias/actualidad/aparecen-enormes-agujeros-dolinas-konya-tuquia-2021-puede-pasar-en-espana.html.

6. Informativos Telecinco. ¿Qué está provocando estos agujeros capaces de tragarse un autobús en Turquía? *eltiempohoy.* 30 de Abril de 2021.

7. MARTÍN-ARROYO, JAVIER. elpais.com. *El Pais.* [En línea] 28 de Abril de 2021. https://elpais.com/clima-y-medio-ambiente/2021-04-28/cinco-anos-despues-del-plan-para-proteger-el-acuifero-de-donana-solo-se-aplica-el-17-de-las-medidas.html.

8. Instituto geológico y minero de España. igme.es. *Instituto geológico y minero de España.* [En línea] https://www.igme.es/zonas_humedas/donana/medio_fisico/hidrogeologia.htm.

9. La sequía y la sobreexplotación de los acuíferos subterráneos empeoran la situación de las Tablas de Daimiel. *consumer.es/medio-ambiente/.* [En línea] 10 de Septiembre de 2009. https://www.consumer.es/medio-ambiente/la-sequia-y-la-sobreexplotacion-de-los-acuiferos-subterraneos-empeoran-la-situacion-de-las-tablas-de-daimiel.html.

10. Ministerio de Transición Ecológia y Reto Demográfico. Ministerio para la Transición Ecológia y Reto Demográfico. [En línea] Gobierno de España. https://www.miteco.gob.es/es/ministerio/default.aspx.

11. Real Academia Española. Definiciones: RAE. *RAE.* [En línea] https://dle.rae.es/.

12. IPCC. ipcc.ch/. [En línea] © 2022 Intergovernmental Panel on Climate Change (IPCC). https://www.ipcc.ch/.

13. Naciones Unidas. un.org. *Naciones unidas. Accion por el clima.* [En línea] https://www.un.org/es/climatechange.

14. Union Europea. Climate action, acuerdo de Paris: ec.europa.eu. *ec.europa.eu.* [En línea] https://ec.europa.eu/clima/policies/international/negotiations/paris_es).

15. *Energías renovables, energías duraderas.* Ruiz de Elvira Serra, Antonio . 20008, Física y sociedad , págs. 14-17.

16. Iberdrola. conocenos/energetica-del-futuro/energias-renovables: Iberdrola. *iberdrola.com.* [En línea] https://www.iberdrola.com/conocenos/energetica-del-futuro/energias-renovables.

17. CEMAER 2019. Energía no renovable: CEMAER. *Centro de Estudios en Medio Ambiente y Energías Renovables.* [En línea] Centro de estudion en medio ambiente y energías renovables . https://www.cemaer.org/energia-no-renovable/.

18. Enérgya-VM. Energias renovables y no renovables: Enérgya-VM. *Enérgya-VM.* [En línea] https://www.energyavm.es/energias-renovables-y-no-renovables/.

19. BBVA. Energía, Energía fosil: bbva.com. *bbva.com.* [En línea] https://www.bbva.com/es/sostenibilidad/que-es-el-combustible-fosil-la-energia-que-se-obtiene-de-la-materia-organica/.

20. Amestoy Alonso, José. *El Planeta Tierra en peligro: Calentamiento global, cambio climático, soluciones.* Alicante : Editorial Club Universitario.

21. Zúñiga López, Ignacion y Crespo del Arco, Emilia. *Meteorología y climatología.* s.l. : Universidad Nacional de Educación a Distancia, 2015.

22. IPCC. *Cambio climático 2014.* IPCC. Informe de síntesis. Grupo Intergubernamental de expertos sobre el cambio climático.

23. K. Pachauri, Rajendra, Meyer, Leo y The Core Writing Team. *Climate Change 2014.* © 2022 Intergovernmental Panel on Climate Change (IPCC). 2014. Synthesis Report.

24. Martín León, Francisco. 'Partenón' subterráneo: sistema que protege a Tokio de inundaciones. *Meteored Tiempo.com.* 25 de Octubre de 2020.

25. Ortiz, Diego Arguedas. La enorme catedral subterránea que protege a Tokio de las inundaciones. *BBC.* 26 de Enero de 2019.

26. © 2021 ONU-Habitat México. La Ciudad Esponja: ONU-Habitat. *ONU-Habitat.* [En línea] 22 de Marzo de 2018. https://onuhabitat.org.mx/index.php/la-ciudad-esponja.

27. La ciudad china de Hebi se convierte en una "ciudad esponja". *ABC.* 9 de Diciembre de 2018.

28. nexiarenovables. nexiarenovables.com. [En línea] http://nexiarenovables.com/centrales-hidroelectricas-en-espana/.

29. Hispagua Sistema Español de Información sobre el Agua. Las grandes presas en españa: Hispagua. *Hispagua Sistema Español de información sobre el agua.* [En línea] Sistema Español de información sobre el agua. https://hispagua.cedex.es/sites/default/files/suplementos/presas/presas.htm.

30. Agencia Extremeña de la Energía. Los cuatro países de mayor consumo eléctrico del mundo priorizan energías renovables. [En línea] http://www.agenex.net/es/cursos/44-noticias/noticias-de-otros-medios/340-los-cuatro-paises-de-mayor-consumo-electrico-del-mundo-priorizan-energias-renovables.

31. Structuralia. Structuralia. *www.structuralia.com.* [En línea] 17 de Abril de 2018. https://blog.structuralia.com/tipologia-de-presas-ii-presas-de-materiales-sueltos#:~:text=Tipos%20de%20presas%20de%20materiales%20sueltos%20El%20hecho,los%20cuales%20se%20explican%20en%20detalle%20a%20continuaci%C3%B3n%3A.

32. —. Structuralia: Tipos de presas II: presas de materiales sueltos. *structuralia.com.* [En línea] 17 de Abril de 2018. https://blog.structuralia.com/tipologia-de-presas-ii-presas-de-materiales-sueltos#:~:text=Tipos%20de%20presas%20de%20materiales%20sueltos%20El%20hecho,los%20cuales%20se%20explican%20en%20detalle%20a%20continuaci%C3%B3n%3A.

33. enelgreenpower. enelgreenpower. *enelgreenpower.com.* [En línea] https://www.enelgreenpower.com/es/learning-hub/energias-renovables/energia-hidroelectrica/central-hidroelectrica.

34. Python Software Foundation. About: Python. *python.org.* [En línea] https://www.python.org/.

35. The NumPy community. Numpy. *numpy.org.* [En línea] https://numpy.org/doc/stable/index.html#.

36. matplotlib. pagina inicial: matplotlib . *matplotlib.org.* [En línea] https://matplotlib.org/ .

37. Pandas. pandas.pydata.org. *Pagina principal de pandas.pydata.org.* [En línea] https://pandas.pydata.org/.

38. Jupyter kernels: github. *Jupyter kernels.* [En línea] https://github.com/jupyter/jupyter/wiki/jupyter-kernels.

39. Jupyter Project . Acerca de nosotros: jupyter. *jupyter.org.* [En línea] https://jupyter.org/about.

40. Project Jupyter. jupyterlab.readthedocs.io. *JupyterLab Documentation.* [En línea] https://jupyterlab.readthedocs.io.

41. © 2022 Anaconda Inc. Anaconda. [En línea] https://www.anaconda.com/about-us.

42. White, Frank M. *Mecanica de fluidos.* Madrid : Mc Graw Hill, 2004.

43. Castro, Adrián. *Minicentrales hidroeléctricas.* Madrid : Instituto para la Diversificación y Ahorro de la Energía, 2006.

44. Sevillano, J. https://javiersevillano.es. [En línea] https://javiersevillano.es/PrecipitacionMediaAnual.htm#.

45. Instituto Nacional de Estadística. INE. *INE.* [En línea] https://www.ine.es/index.htm.

46. Red eléctrica Española. Red Eléctrica Española. [En línea] https://www.ree.es/es.

47. Irigoyen Pérez, Alberto y García Piña, Javier. seprem.es. [En línea] 5 de Abril de 2021. https://www.seprem.es/boletin/caudales.pdf.

48. Martín, María. Blog Las tres presas mas grandes del mundo: Escuela de ingeniería y medioambiente. *Escuela de ingeniería y medioambiente.* [En línea] http://eimaformacion.com/las-tres-presas-mas-altas-de-espana/.

49. Ministerio de transición ecológica y el reto demográfico. Presas en funcionde la altura: sig.mapama.gob. *sig.mapama.gob.* [En línea] https://sig.mapama.gob.es/WebServices/clientews/snczi/default.aspx?origen=8&nombre=PRESA_ESTADISTICA_2&claves=&valores=.

50. —. Presas segun su tipología: sig.mapama.gob. *sig.mapama.gob.* [En línea] https://sig.mapama.gob.es/WebServices/clientews/snczi/default.aspx?origen=8&nombre=PRESA_ESTADISTICA_3&claves=&valores=.

51. —. Evolucion del numero de presas: sig.mapama.gob.es. *sig.mapama.gob.es.* [En línea] https://sig.mapama.gob.es/WebServices/clientews/snczi/default.aspx?origen=8&nombre=PRESA_ESTADISTICA_4&claves=&valores=.

52. blog: iagua. *iagua.* [En línea] https://www.iagua.es/blogs/spancold/%C2%BFcuantas-presas-hay-hoy-en-espana.

53. C. Chapra, Steven y P. Canale, Raymond . *Métodos numéricos para ingenieros.* Quinta. s.l. : MC Graw-Hill, 2006.

54. JOHNSON, DON H. , T. HEIDEMAN, MICHAEL y C. SIDNEY BURRUS. Gauss and the History of the Fast Fourier Transform. [En línea] https://web.archive.org/web/20170914220621/https://pdfs.semanticscholar.org/1790/fe007bc1ab161a1ea814748b42e3acbdc958.pdf.

55. Atkinson, Kendall E. An Introduction to Numerical Analysis (2nd edition). New York : John Wiley & Sons, 1988.

56. Zygmund, A. *Trigonometric Series, Volume II.* s.l. : Cambridge University Press, 1988.

ANEXO 1. POBLACIÓN EN DISTINTAS CAPITALES DE PROVINCIA EN ESPAÑA

Datos procedentes de: (45)

	Total
	2020
Total, capitales de provincia	15.174.588
02003 Albacete	174.336
03014 Alicante/Alacant	337.482
04013 Almería	201.322
05019 Ávila	58.369
06015 Badajoz	150.984
08019 Barcelona	1.664.182
48020 Bilbao	350.184
09059 Burgos	176.418
10037 Cáceres	96.255
11012 Cádiz	115.439
12040 Castelló de la Plana	174.264
13034 Ciudad Real	75.504
14021 Córdoba	326.039
15030 Coruña, A	247.604
16078 Cuenca	54.621
20069 Donostia/San Sebastián	188.240
17079 Girona	103.369
18087 Granada	233.648
19130 Guadalajara	87.484
21041 Huelva	143.837
22125 Huesca	53.956
23050 Jaén	112.757
24089 León	124.028
25120 Lleida	140.403
26089 Logroño	152.485
27028 Lugo	98.519
28079 Madrid	3.334.730
29067 Málaga	578.460
30030 Murcia	459.403
32054 Ourense	105.643
33044 Oviedo	219.910
34120 Palencia	78.144
07040 Palma	422.587
35016 Palmas de Gran Canaria, Las	381.223
31201 Pamplona/Iruña	203.944
36038 Pontevedra	83.260
37274 Salamanca	144.825
38038 Santa Cruz de Tenerife	209.194

39075 Santander	173.375
40194 Segovia	52.057
41091 Sevilla	691.395
42173 Soria	39.821
43148 Tarragona	136.496
44216 Teruel	36.240
45168 Toledo	85.811
46250 València	800.215
47186 Valladolid00	0
01059 Vitoria-Gasteiz	253.996
49275 Zamora	60.988
50297 Zaragoza	681.877

ANEXO 2. PRESAS AFLUENTES DEL DUERO

Confederación Hidrográfica del Duero a día 5 de julio de 2021

Hora de los datos: **8:00 AM**. Los datos son provisionales y están sujetos a revisión.

		Volumen embalsado				Datos semanales
SISTEMA **Embalse**	**Capacidad** (hm3)	**Actual** (hm3)	**Actual** (%)	**Año ant.** (hm3)	**Media 10 años ant.** (hm3)	**Variación de Vol.** (hm3)
LEÓN (SISTEMA ESLA Y ÓRBIGO)						
Villameca	20,0	18,1	90,7	16,1	15,4	-0,2
Barrios de Luna	308,0	267,3	86,8	251,8	234,4	-15,4
Porma	317,0	260,8	82,3	251,9	236,6	-11,0
Riaño	651,0	508,4	78,1	510,6	490,7	-18,7
Total	1.296,0	1.054,7	81,4	1.030,5	977,0	-45,4
PALENCIA (SISTEMA CARRIÓN)						
Camporredondo	70,0	43,4	62,0	53,2	44,3	-1,0
Compuerto	95,0	66,3	69,8	75,2	70,2	-2,8
Total	165,0	109,7	66,5	128,4	114,4	-3,8
PALENCIA (SISTEMA PISUERGA)						
Cervera - Ruesga	10,0	10,0	100,0	10,3	9,6	0,0
La Requejada	65,0	35,9	55,2	47,1	43,6	-0,7
Aguilar de Campoo	247,0	155,8	63,1	224,4	159,6	-4,5
Total	322,0	202,0	62,7	281,9	212,9	-5,2
BURGOS (SISTEMA ARLANZA)						
Arlanzón	22,0	20,8	94,4	19,5	19,0	+0,2
Uzquiza	75,0	53,1	70,8	64,3	58,8	-0,7
Total	97,0	73,9	76,2	83,8	77,9	-0,5
SORIA (SISTEMA ALTO DUERO)						
Cuerda del Pozo	248,7	192,4	77,4	220,7	194,9	-3,4
SEGOVIA (SISTEMA RIAZA-DURATÓN Y CEGA-ERESMA-ADAJA)						
Linares del Arroyo	54,4	44,0	80,8	44,7	39,4	-1,2
El Pontón Alto	7,4	7,3	99,0	7,3	7,2	0,0
Total	61,8	51,3	83,0	52,0	46,6	-1,2
ÁVILA (SISTEMA CEGA-ERESMA-ADAJA)						
Castro de las Cogotas	59,0	51,1	86,6	53,5	42,9	-1,8
SALAMANCA (SISTEMA TORMES)						
Santa Teresa	496,0	392,5	79,1	415,4	390,5	-11,4
SALAMANCA (SISTEMA ÁGUEDA)						
Irueña	110,0	49,0	44,5	62,0	64,3	-2,0
Águeda	22,0	20,2	91,7	21,2	14,1	-1,4
Total	132,0	69,1	52,4	83,2	78,5	-3,4
TOTAL	**2.877,5**	**2.196,8**	**76,3**	**2.349,3**	**2.135,6**	**-76,0**
% DEL TOTAL			**76,3**	**81,6**	**74,2**	

ANEXO 3.1 CAUDALES RÍOS EN ESPAÑA

Este documento será usado para tener parámetros orientativos para la simulación del sistema.

Los datos vienen del archivo: (47)

Cuenca	Ríos	Estaciones	Caudal m3/s 2020	Caudal m3/s 2021	max m3/s	min m3/s
cantabrico Oriental	Bidasoa	1106 Endarlaza	15,83	11,98	14,8	10,03
cantabrico Oriental	oria	1080 Oria en Andoain	-	-		
cantabrico Oriental	Ibaizabal	1163 Lemoa	2,28	1,96	3,35	1,22
cantábrico Occidental	Asón	1196 Coterillo	8,65	4,63	7,72	-
cantábrico Occidental	Besaya	1237 Torrelavega	11,95	3,23	3,86	2,6
cantábrico Occidental	Deva	1269 Panes	-	-	-	-
cantábrico Occidental	Cares	1276 Mier	14,38	15,01	18,58	11,83
cantábrico Occidental	Sella	1295 Cangas de Onís	9,84	10,09	18,42	4,16
cantábrico Occidental	Narcea	1359 Requejo	31,54	23,08	76,92	12,87
cantábrico Occidental	Eo	1427 San Tirso	9,55	7,55	8,12	6,75
miño-sil	Sil	1768 Emb. de San Esteban	140,19	119,57	177,51	70,34
miño-sil	Miño	1629 Emb. de Los Peares	144,97	74,59	158,32	28,08
miño-sil	Miño	1641 Emb. de Frieira	348,81	247,04	350,02	169,34
miño-sil	Limia	1808 Emb. de Las Conchas	18,84	10,85	28,61	2,21
miño-sil	Ladra	1619 Begonte	19,92	12,74	14,81	11,85
miño-sil	Miño	1622 Lugo	56,29	35,41	39,32	33,13
miño-sil	Cúa	1724 Quilós	10,94	8,45	8,99	7,66
miño-sil	Miño	1631 Velle	305,1	215,14	310,03	120,6
miño-sil	Tea	1645 Puenteareas	13,06	6,66	7,83	5,69
miño-sil	Jares	1791 Prada	8,5	6,33	10,83	0,5
miño-sil	Salas	1807 Salas	0,72	3,77	11,73	0,55
miño-sil	Sil	1719 Ponferrada	3,56	3,91	4,02	3,84
miño-sil	Cabe	1765 Monforte	4,87	3,7	4,18	3,39
Galicia Costa	Masma	431 Masma en Mondoñedo	3,38	1,74	1,95	1,43
Galicia Costa	Landro	438 Landro en Viveiro	6,77	-	-	-
Galicia Costa	Mera	443 Mera en Ortigueira	2,8	2,25	2,46	2
Galicia Costa	Mandeo	464 Mandeo en Irixoa	6,28	4,36	6,84	2,79
Galicia Costa	Anllons	485 Anllons en Ponteceso	9,22	8,9	12,3	7,75
Galicia Costa	Tambre	520 Tambre en Oroso	15,48	12,93	14,07	11,76
Galicia Costa	Deza	552 Deza en Silleda	14,33	12,55	19,35	11,08
Galicia Costa	Ulla	554 Ulla en Teo	56,55	38,43	47,5	33,98

Galicia Costa	Lérez	578 Lérez en Pontevedra	14,73	13,08	15,11	12
Duero	Duero	Aranda	29,43	15,76	29,91	4,65
Duero	Duero	Toro	88,19	74,45	109,18	46,58
Duero	Esla	Ricobayo (salida embalse)	23,1	8,51	-	-
Duero	Duero	Saucelle	66,98	258,61	802,11	1,48
Duero	Órbigo	Cebrones	12,52	18,15	23,31	12,44
Duero	Eria	Morla	2,26	3,35	3,76	2,39
Duero	Esla	Castropepe	26,99	28,33	30,59	25,82
Duero	Carrión	Palencia	4,46	5,03	6,89	4,06
Duero	Pisuerga	Herrera de Pisuerga	4,45	4,81	8,18	3,45
Duero	Pisuerga	Valladolid	34,49	42,43	54,24	32,58
Duero	Arlanza	Quintana del Puente	21,63	19,81	21,25	18,15
Duero	Tormes	Salamanca	13,94	11,06	12,19	9,92
Duero	Eresma	Bernardos	7,01	5,3	6,39	4,31
Duero	Adaja	Ávila	2,96	2,29	2,66	1,8
Duero	Támega	Rabal	12,5	9,23	10,63	8,63
Duero	Bernesga	León	6,58	6,67	8,07	5,31
Duero	Porma	Secos de Porma	8,57	6,76	7,83	6,04
Duero	Tormes	Tormes en Hoyos del Espino	1,06	1,24	1,44	1,11
Duero	Besandino	Besande	0,55	0,47	0,52	0,4
Tajo	Tajo	Alcántara	30,28	153	-	-
Tajo	Lozoya	Atazar	1,82	5,11	-	-
Tajo	Tajo	Azután	29,71	55	-	-
Tajo	Tajo	Bolarque	26,43	37,32	-	-
Tajo	Árrago	Borbollón	1,81	2	-	-
Tajo	Guadiela	Buendía	14,08	9,01	-	-
Tajo	Tajo	Castrejón	25,08	34,58	-	-
Tajo	Alberche	Cazalegas	4,48	10,52	-	-
Tajo	Tajo	Cedillo	41,45	160,29	-	-
Tajo	Tajo	Entrepeñas	16,43	20,46	-	-
Tajo	Alagón	Gabriel y Galán	1,01	25,43	-	-
Tajo	Alberche	Picadas	8,56	7,85	-	-
Tajo	Tiétar	Rosarito	0,57	5,21	-	-
Tajo	Alberche	San Juan	10,81	7,08	-	-
Tajo	Tajo	Torrejón - Tajo	36,26	127,57	-	-
Tajo	Tiétar	Torrejón - Tiétar	1,8	30,86	-	-
Tajo	Tajo	Valdecañas	26,23	106	-	-
Tajo	Alagón	Valdeobispo	3,49	33,63	-	-
Tajo	Lozoya	Pinilla (Entrada)	2,12	3,64	-	-
Guadiana	Guadiana	La Cubeta	1,98	2,19	-	-
Guadiana	Guadiana	Badajoz			-	-
Guadiana	Guadiana	Cijara (Entrada)	0,08	5,06	-	-
Guadalquivir	Guadalquivir	Pedro Marín	2,04	3	16,14	1,7

Guadalquivir	Guadalquivir	Mengíbar	5,67	9,56	26,03	1,78
Guadalquivir	Guadalquivir	El Carpio	5,89	8,3	19,78	4,56
Guadalquivir	Guadalquivir	Alcalá del Río	13,27	11,68	42,57	2
Guadalquivir	Genil	Loja	5,26	3,55	4,2	2,93
Guadalquivir	Genil	Écija	2,32	14,3	30,59	0,89
Cuenca Mediterránea Andaluza	Guadiaro	San Pablo Buceite	16,19	6,11	6,81	5,5
Cuenca Mediterránea Andaluza	Genal	Jubrique	2,06	0,57	0,65	0,47
Cuenca Mediterránea Andaluza	Grande	Las Millanas	2,88	0,98	1,03	0,88
Cuenca Mediterránea Andaluza	Guadalhorce	Bobadilla	3,05	0,29	0,51	0,13
Cuenca Mediterránea Andaluza	Verde de Almuñécar	Cázulas	0,15	0	0	0
Segura	Segura	Almadenes	10,49	17,01	-	-
Segura	Segura	Segura en Rojales	27,8	1,72	-	-
Júcar	Júcar	Salida embalse el Naranjero	12,6	21,71	-	-
Júcar	Turia	Zagra	17,5	8,53	-	-
Júcar	Mijares	Villarreal	13,5	5,61	-	-
Júcar	Mijares	Azud de Santa Quiteria	-	1,7	-	-
Júcar	Turia	Azud del Repartiment	-	0	-	-
Júcar	Júcar	Azud de La Marquesa	-	0,61	-	-
Ebro	Ebro	9001 Ebro en Miranda	21,77	25,56	31,93	17,49
Ebro	Ebro	9280 Ebro en Logroño	69,93	57,3	82,15	22,47
Ebro	Ebro	9002 Ebro en Castejón	180,45	76,15	88,75	61,72
Ebro	Ebro	9011 Ebro en Zaragoza	237,36	86,65	122,72	74,9
Ebro	Ebro	9027 Ebro en Tortosa	524,86	156,21	214,53	120,14
Ebro	Zadorra	9074 Zadorra en Arce	8,86	6,82	7,99	5,96
Ebro	Ega	9003 Ega en Andosilla	13,88	8,61	13,92	7,27
Ebro	Arga	9004 Arga en Funes	21,96	11,75	17,42	7,09
Ebro	Aragón	9005 Aragón en Caparroso	45,59	17,19	35,16	3,24
Ebro	Arba	9260 Arba en Tauste	18,59	6,73	10,7	4,6
Ebro	Gállego	9089 Gállego en Zaragoza	20,33	2,71	4,33	2,13
Ebro	Cinca	9017 Cinca en Fraga	94,14	21,1	23,95	19,05
Ebro	Tirón	9281 Tirón en Haro	11,65	8	8,05	7,66
Ebro	Najerilla	9038 Najerilla-Torremontalbo	8,12	7,23	7,23	7,23
Ebro	Jalón	9087 Jalón en Grisén	19,81	0,46	1,09	0,19
Ebro	Martín	9014 Martín en Híjar	4,29	0,49	0,59	0,43
Ebro	Guadalope	9099 Guadalope en Caspe	48,99	0,77	1,2	0,68
Ebro	Valira	9022 Valira en Seo de Urgell	8,97	10,21	17,05	4,44
Ebro	Ara	9040 Ara en Boltaña	16,67	16,16	21,65	10,6
Ebro	Veral	9080 Veral en Zuriza	0,71	0,85	1,03	0,64

Ebro	Muga	Castelló D' Empúries	1,29	0,71	0,86	0,45
Cuencas Internas de Cataluña	Fluvià	Esponellá	4,6	3,08	5,23	1,18
Cuencas Internas de Cataluña	Ter	Masies de Roda	10,96	6,97	10,87	3,1
Cuencas Internas de Cataluña	Ter	Gerona (Ter)	9,93	3,55	4,46	2,44
Cuencas Internas de Cataluña	Ter	Torroella de Montgrí	31,32	4,39	6,58	2,95
Cuencas Internas de Cataluña	Tordera	Fogars de la Selva (Can Simó)	4,51	0,98	1,14	0,85
Cuencas Internas de Cataluña	Besòs	Santa Coloma de Gramenet	3,18	2,07	2,85	1,45
Cuencas Internas de Cataluña	Cardener	Súria	2,82	1,76	2,58	0,65
Cuencas Internas de Cataluña	Llobregat	Castellbell i El Vilar	11,78	8,66	9,7	5,73
Cuencas Internas de Cataluña	Llobregat	Sant Joan Despí	6,94	4,59	6,79	3,03
Cuencas Internas de Cataluña	Francolí	Tarragona	1,9	0,42	1,06	0

ANEXO 3.2 DATOS GUÍA ALGUNAS PRESAS DE ESPAÑA

Datos procedentes de: (48)

La almendra (salamanca)

Tipo de presa:	Bóveda
Cota coronación (m):	732
Altura desde cimientos (m):	202
Longitud de coronación (m):	567
Cota cimentación (m):	530
Cota del cauce en la presa (m):	547
Volumen del cuerpo presa (1000 m^3):	2,188

Canales (granada)

Tipo de presa:	Materiales sueltos núcleo arcilla
Cota coronación (m):	966
Altura desde cimientos (m):	156
Longitud de coronación (m):	340
Cota cimentación (m):	808
Cota del cauce en la presa (m):	818
Volumen del cuerpo presa (1000 m^3):	7.248

Canelles(Huesca)

Tipo de presa:	Bóveda
Cota coronación (m):	508
Altura desde cimientos (m):	151
Longitud de coronación (m):	210
Cota cimentación (m):	357
Cota del cauce en la presa (m):	370
Volumen del cuerpo presa (1000 m^3):	332,994

Anexo 3.3 DATOS DE PRESAS ESPAÑA

Datos procedentes de: (49)

Presas en Función de la Altura

Criterio	Presas	Porcentaje
Menores de 15 m.	550	35,76 %
Entre 15 y 30 m.	465	30,23 %
Entre 30 y 60 m.	332	21,59 %
Entre 60 y 100 m.	148	9,62 %
Mayores de 100 m.	43	2,80 %

Datos procedentes de: (50)

Presas según su Tipología

Tipología	Número de Presas	Porcentaje
Arco Gravedad	48	3,12 %
Bóveda	56	3,64 %
Bóvedas Múltiples	1	0,07 %
Contrafuertes	31	2,02 %
Gravedad	814	52,93 %
Gravedad Y Contrafuertes	6	0,39 %
Gravedad Y Mampostería	17	1,11 %
Gravedad Y Materiales Sueltos Homogénea	17	1,11 %
Gravedad Y Materiales Sueltos Pantalla	4	0,26 %
Gravedad Y Materiales Sueltos P Asfáltica	3	0,20 %
Hormigón Armado	4	0,26 %
Hormigón Compactado	22	1,43 %
Mampostería	23	1,50 %
Materiales Sueltos Homogénea	268	17,43 %
Materiales Sueltos Núcleo Arcilla	134	8,71 %
Materiales Sueltos P Asfáltica	21	1,37 %
Materiales Sueltos P Hormigón	37	2,41 %
Materiales Sueltos P Lámina	13	0,85 %
Materiales Sueltos Y Mampostería	5	0,33 %
Materiales Sueltos Zonificada	7	0,46 %
Presa Móvil	1	0,07 %
	6	0,39 %

Datos procedentes de: (51)

Evolución del Número de Presas

Decenio	Presas	Porcentaje
Antes De 1900	55	3,58 %
1900 - 1909	14	0,91 %
1910 - 1919	52	3,38 %
1920 - 1929	41	2,67 %
1930 - 1939	48	3,12 %
1940 - 1949	60	3,90 %
1950 - 1959	161	10,47 %
1960 - 1969	212	13,78 %
1970 - 1979	187	12,16 %
1980 - 1989	164	10,66 %
1990 - 1999	186	12,09 %
Posteriores Al 2000	45	2,93 %

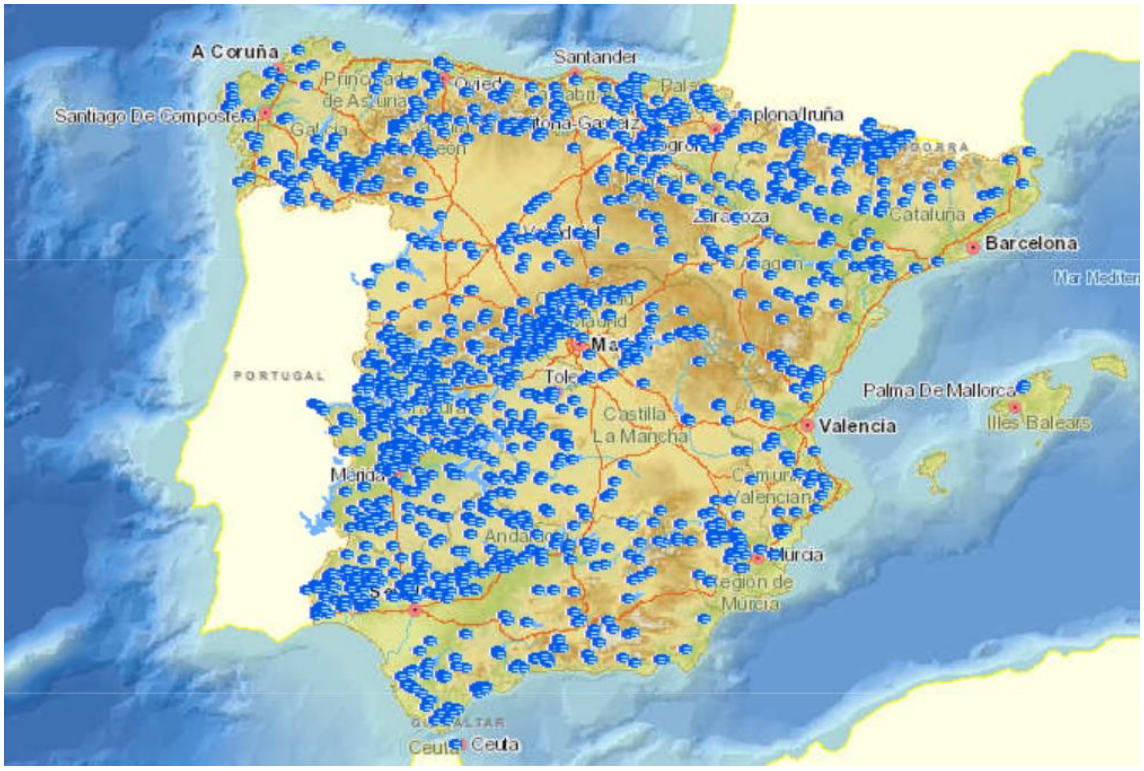
ANEXO 3.4 MAPA PRESAS Y LAGOS EN ESPAÑA:

Imágenes obtenidas del Ministerio de Tecnología y Comercio. Para más información, consultar el siguiente enlace (10)

https://www.miteco.gob.es/es/agua/temas/seguridad-de-presas-y-embalses/

https://www.mapa.gob.es/es/ministerio/servicios/publicaciones/II.E.%20Agua_tcm30-84065.pdd

LOCALIZACIÓN DE LAS ESTACIONES DEL PROGRAMA DE CONTROL DE VIGILANCIA – LAGOS Y EMBALSES 2011


A Coruña
Santander
Santiago De Compostela
Pamplona/Iruña
Zaragoza
Cataluña
Barcelona
Mar Mediterráneo
PORTUGAL
Toledo
Palma De Mallorca
Illes Balears
Castilla La Mancha
Valencia
Murcia
Región de Murcia
Ceuta


Canarias
Santa Cruz De Tenerife
Las Palmas De Gran Canaria
Canarias

Zonas con peligro de inundación por lluvia:

ANEXO 4. PRECIPITACIONES

Datos procedentes de: (44)

Precipitación media (mm) anual y mensual en capitales de provincia

	ANUAL	MEDIA MESES	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV	DIC
ALBACETE (AEROPUERTO)	367	31	21	24	28	48	48	36	12	14	32	42	34	28
ALICANTE	336	28	22	26	26	30	33	17	6	8	47	52	42	26
ALMERÍA (AEROPUERTO)	196	16	23	21	15	20	14	10	1	1	12	28	28	23
ÁVILA	400	33	32	22	23	42	50	37	16	19	29	40	43	44
BADAJOZ (TALAVERA LA REAL)	463	39	52	43	33	52	40	18	4	5	23	56	64	73
BARCELONA (AEROPUERTO)	640	53	41	29	42	49	59	42	20	61	85	91	58	51
BILBAO (AEROPUERTO)	1195	100	126	97	94	124	90	64	62	82	74	121	141	116
BURGOS (AEROPUERTO)	555	46	46	42	31	65	69	46	30	27	36	50	56	57
CÁCERES	523	44	58	43	35	49	48	23	7	8	26	59	80	87
CÁDIZ (JEREZ AEROPUERTO)	598	50	89	60	42	54	37	13	2	6	22	67	86	109
CASTELLÓN	442	37	35	26	29	38	37	20	12	29	62	71	41	46
CEUTA (MONTE HACHO)	586	49	87	87	59	56	28	13	1	1	11	61	76	108
CIUDAD REAL	396	33	36	34	28	44	43	29	9	7	22	47	42	55
CÓRDOBA (AEROPUERTO)	536	45	64	53	40	61	34	17	3	3	24	62	85	89
CORUÑA	1008	84	128	102	79	85	80	42	30	35	68	110	114	135
CUENCA	507	42	45	41	32	56	60	44	15	17	37	53	49	58

GERONA (AEROPUERTO)	724	60	65	44	53	67	80	66	30	48	68	83	70	63
GRANADA (AEROPUERTO)	357	30	41	38	30	38	28	17	4	3	16	42	48	53
GUADALAJARA (MOLINA DE ARAGÓN)	500	42	31	31	31	54	74	51	29	29	44	46	39	41
HUELVA	490	41	73	43	36	46	30	9	3	4	21	56	74	95
HUESCA (AEROPUERTO)	535	45	39	32	34	53	62	47	20	38	54	54	50	51
JAÉN	493	41	55	50	44	54	43	18	2	9	26	55	62	75
LAS PALMAS (AEROPUERTO)	134	11	18	24	14	7	2	0	0	0	10	13	18	27
LEÓN (AEROPUERTO)	556	46	58	46	29	50	58	39	28	24	39	56	58	70
LÉRIDA	369	31	26	14	27	37	49	34	12	21	39	39	28	28
LOGROÑO (AEROPUERTO)	399	33	27	23	26	44	48	47	31	23	24	31	36	37
LUGO (AEROPUERTO)	1084	90	122	108	86	94	93	52	34	34	77	115	122	146
MADRID	436	36	37	35	26	47	52	25	15	10	28	49	56	56
MÁLAGA (AEROPUERTO)	524	44	81	55	49	41	25	12	2	6	16	56	95	88
MALLORCA	427	36	43	34	26	43	30	11	5	17	39	68	58	45
MELILLA	370	31	58	58	47	38	27	10	1	3	10	29	44	47
MURCIA (ALCANTARILLA AEROPUERTO)	301	25	25	28	30	27	32	20	5	10	27	44	32	21
ORENSE (AEROPUERTO)	817	68	90	81	54	70	67	39	19	23	57	97	93	124
OVIEDO	973	81	85	85	82	109	94	53	52	55	64	98	101	96
PAMPLONA (AEROPUERTO)	721	60	63	52	52	77	74	47	40	43	43	74	80	75
PONTEVEDRA (AEROPUERTO)	1691	141	204	190	126	140	129	66	44	47	108	185	198	254
SALAMANCA (AEROPU	382	32	31	27	22	39	48	34	16	11	32	39	42	42

ERTO)														
SAN SEBASTIÁN (AEROPUERTO)	1738	145	168	150	144	168	138	96	98	112	138	174	186	167
SANTANDER (AEROPUERTO)	1246	104	123	104	105	125	89	62	52	72	85	135	146	117
SEGOVIA	464	39	38	34	30	47	60	38	21	21	30	46	48	50
SEVILLA (AEROPUERTO)	534	45	65	54	38	57	34	13	2	6	23	62	84	95
SORIA	502	42	39	38	28	53	61	46	34	30	31	45	45	51
TARRAGONA (REUS AEROPUERTO)	504	42	38	23	35	40	60	38	15	51	77	65	49	40
TENERIFE (AEROPUERTO)	557	46	98	69	65	54	22	12	6	5	20	48	70	87
TERUEL	373	31	17	14	19	36	56	43	30	40	36	42	22	20
TOLEDO	357	30	28	28	25	41	44	28	12	9	22	38	40	44
VALENCIA	454	38	36	32	35	37	34	23	9	19	51	74	51	52
VALLADOLID	435	36	40	32	23	44	47	33	16	18	31	42	51	56
VITORIA (AEROPUERTO)	779	65	76	65	61	86	70	51	43	45	42	74	89	80
ZAMORA	363	30	34	28	18	36	42	30	15	13	22	38	42	44
ZARAGOZA (AEROPUERTO)	318	27	22	20	20	35	44	31	18	17	27	30	30	23

días de Lluvia(número medio mensual/anual de días con Precipitación superior o igual a 1 mm)

	ANUAL	ENE	FEB	MAR	ABR	MAY	JUN	JUL	AGO	SEP	OCT	NOV	DIC
ALBACETE (AEROPUERTO)	53	4	4	5	6	7	4	1	2	3	5	5	5
ALICANTE	37	4	3	4	4	4	2	1	1	3	4	4	4
ALMERÍA (AEROPUERTO)	26	3	3	3	3	2	1	0	0	1	3	3	3
ÁVILA	66	6	5	4	8	9	5	2	2	4	6	6	7
BADAJOZ (TALAVERA LA REAL)	61	7	6	5	7	6	3	1	1	3	7	7	8
BARCELONA (AEROPUERTO)	55	5	4	5	5	5	4	2	4	5	6	5	5
BILBAO	128	13	11	11	13	12	8	7	8	9	11	12	12

(AEROPUERTO)													
BURGOS (AEROPUERTO)	85	8	8	6	9	10	6	4	4	5	8	8	9
CÁCERES	64	7	7	5	7	6	3	1	1	3	7	7	9
CÁDIZ (JEREZ AEROPUERTO)	54	7	7	5	6	4	2	0	0	2	6	7	9
CASTELLÓN	45	4	3	4	5	5	3	2	3	4	5	4	4
CEUTA (MONTE HACHO)	79	10	11	9	8	5	3	1	1	3	8	9	11
CIUDAD REAL	62	6	6	5	8	7	4	1	1	3	6	6	8
CÓRDOBA (AEROPUERTO)	56	7	6	5	8	5	2	1	1	2	6	6	8
CORUÑA	131	14	14	12	13	11	7	5	6	8	12	14	15
CUENCA	73	7	6	6	8	8	5	2	3	5	8	7	8
GERONA (AEROPUERTO)	67	5	5	6	7	7	6	4	5	6	6	5	5
GRANADA (AEROPUERTO)	52	6	6	5	7	4	2	0	1	2	5	6	7
GUADALAJARA (MOLINA DE ARAGÓN)	78	6	6	6	9	10	7	4	4	5	7	6	8
HUELVA	50	7	6	5	6	4	1	0	0	2	5	6	8
HUESCA (AEROPUERTO)	62	6	5	4	6	8	5	3	4	4	6	6	6
JAÉN	56	6	6	5	7	6	2	0	1	3	6	7	8
LAS PALMAS (AEROPUERTO)	21	3	3	3	1	0	0	0	0	1	2	3	4
LEÓN (AEROPUERTO)	78	8	7	6	8	9	6	4	3	4	8	7	9
LÉRIDA	46	4	3	4	5	6	4	2	3	4	4	4	4
LOGROÑO (AEROPUERTO)	67	6	5	5	7	8	5	4	4	4	6	6	6
LUGO (AEROPUERTO)	131	14	13	12	13	13	7	5	5	8	13	14	14
MADRID	63	6	6	5	7	8	4	2	2	3	6	6	7
MÁLAGA (AEROPUERTO)	43	6	5	4	5	3	2	0	0	2	4	5	6
MALLORCA	52	5	5	4	6	4	2	1	1	4	7	6	6
MELILLA	44	6	6	5	5	3	1	0	1	2	4	5	5
MURCIA	35	3	3	3	4	4	2	1	1	2	4	4	4

(ALCANTARILLA AEROPUERTO)													
ORENSE (AEROPUERTO)	97	10	10	8	11	10	5	3	3	6	10	10	11
OVIEDO	122	10	11	11	13	12	8	8	8	8	11	11	11
PAMPLONA (AEROPUERTO)	95	9	9	8	10	10	6	5	5	6	8	9	10
PONTEVEDRA (AEROPUERTO)	133	15	13	12	13	13	8	5	5	8	13	14	15
SALAMANCA (AEROPUERTO)	66	6	6	5	7	8	5	3	2	4	7	7	7
SAN SEBASTIÁN (AEROPUERTO)	140	13	12	12	14	12	10	9	10	10	12	13	12
SANTANDER (AEROPUERTO)	128	13	12	12	13	11	8	7	7	9	12	13	12
SEGOVIA	76	7	7	6	8	10	5	3	3	4	7	8	8
SEVILLA (AEROPUERTO)	52	6	6	5	7	4	2	0	0	2	6	6	8
SORIA	80	7	7	6	8	10	6	4	4	5	7	7	8
TARRAGONA (REUS AEROPUERTO)	51	4	3	4	6	6	4	2	4	5	5	4	4
TENERIFE (AEROPUERTO)	66	8	7	8	6	4	3	1	1	3	7	8	9
TERUEL	59	4	3	4	7	8	6	3	4	5	6	4	5
TOLEDO	56	6	5	4	7	7	3	2	2	3	6	6	6
VALENCIA	44	4	3	4	5	5	3	1	2	4	5	4	5
VALLADOLID	71	7	6	5	8	9	5	3	3	4	7	6	8
VITORIA (AEROPUERTO)	103	10	10	9	12	10	6	5	5	6	9	10	11
ZAMORA	64	6	6	4	7	8	4	2	2	4	6	7	7
ZARAGOZA (AEROPUERTO)	50	4	4	4	5	6	4	3	2	3	5	5	5

ANEXO 5. DEMANDA DE POTENCIA

Datos procedentes de: (46)

Fecha	Demanda horaria máxima (MWh)	Demanda diaria (GWh)
01/02/2021	33662,968	690,431618
02/02/2021	33912,178	707,869698
03/02/2021	33862,5705	708,674577
04/02/2021	33719,0679	706,453021
05/02/2021	33319,9779	706,543497
06/02/2021	30225,152	639,684334
07/02/2021	30605,828	602,71842
08/02/2021	35157,213	721,435104
09/02/2021	36148,848	753,022419
10/02/2021	34686,246	731,59349
11/02/2021	34229,918	721,828643
12/02/2021	33587,5983	708,658764
13/02/2021	29761,342	629,790133
14/02/2021	29594,8615	594,301282
15/02/2021	34449,807	706,694989
16/02/2021	34176,974	715,331484
17/02/2021	34594,803	715,949311
18/02/2021	34367,098	718,850821
19/02/2021	33980,425	713,544875
20/02/2021	29792,4185	634,774739
21/02/2021	30191,766	605,511998
22/02/2021	34383,905	707,054609
23/02/2021	34253,706	714,389011
24/02/2021	34143,4455	710,879827
25/02/2021	34015,887	708,471367
26/02/2021	33420,869	703,37944
27/02/2021	29826,092	626,718383
28/02/2021	29427,8685	587,882467
01/03/2021	34204,777	696,081078
02/03/2021	34469,599	713,550057
03/03/2021	34060,322	708,222376
04/03/2021	33819,727	704,050329
05/03/2021	33402,225	699,874496

06/03/2021	29696,459	624,44615
07/03/2021	29661,435	589,76227
08/03/2021	35085,729	705,782568
09/03/2021	35026,344	721,386999
10/03/2021	34871,103	713,089884
11/03/2021	33869,915	702,519225
12/03/2021	32821,969	692,617783
13/03/2021	29226,1545	612,479752
14/03/2021	29043,639	574,178824
15/03/2021	33762,494	682,97972
16/03/2021	33976,695	695,594315
17/03/2021	33472,7	693,089459
18/03/2021	33827,218	704,513024
19/03/2021	31733,388	667,735823
20/03/2021	30769,349	633,480151
21/03/2021	30259,8789	594,476381
22/03/2021	34450,887	695,543436
23/03/2021	33979,525	699,999615
24/03/2021	33567,591	691,828704
25/03/2021	33159,574	686,299127
26/03/2021	32174,207	679,852919
27/03/2021	28796,853	611,501425
28/03/2021	28223,143	538,263528
29/03/2021	31336,1467	653,369963
30/03/2021	31002,633	657,088145
31/03/2021	30150,3435	640,896793
01/04/2021	27150,216	568,859958
02/04/2021	25824,955	524,699155
03/04/2021	26768,721	543,584269
04/04/2021	26192,953	518,352854
05/04/2021	28622,703	568,43257
06/04/2021	31433,459	642,679223
07/04/2021	32067,729	668,739331
08/04/2021	31963,318	673,509619
09/04/2021	32557,555	675,156725
10/04/2021	28975,232	605,081663
11/04/2021	28232,522	560,438001
12/04/2021	31712,094	655,451197
13/04/2021	32531,91	679,856956
14/04/2021	32319,919	678,838607
15/04/2021	32438,961	683,850078
16/04/2021	31878,9568	675,642336
17/04/2021	28718,339	606,200226
18/04/2021	28236,343	559,264917

19/04/2021	31847,11	660,680511
20/04/2021	31748,213	672,558658
21/04/2021	31784,93	673,894246
22/04/2021	32082,415	674,722469
23/04/2021	30599,9754	647,973049
24/04/2021	27761,4579	585,444217
25/04/2021	27492,556	551,64345
26/04/2021	31568,493	652,510384
27/04/2021	31790,111	665,91143
28/04/2021	31265,979	661,338497
29/04/2021	31354,215	660,835086
30/04/2021	30983,867	651,942752

ANEXO 6.1 MODULO DE HERRAMIENTAS MODELADO

```
import numpy as np
import random
Dias_Mes = [31, 28, 31, 30, 31, 30, 30, 31, 30, 31, 30, 31]
# ------------------------------------------------[
UNIDADES ]


def transformacion_metros3(valor, unidades):
    """VERSION 1.

    Transformar valores a m3
    10**6m3=1Hm
    10**3M3=1Dm3...
    ENTRADAS:
     - valor: unidad o vector de valores
     - unidades: unidades en las que se encuentra
    """
    if unidades == 'Hm3':
        return valor*(10**6)
    elif unidades == 'Dm3':
        return valor*(10**3)
    elif unidades == 'L':
        return valor/(10**3)
    elif unidades == 'mm3':
        return valor/(10**9)


def transformacion_Litros(valor, unidades):
    """VERSION 1.

    Transformar valores a L
    10**6m3=1Hm
    10**3M3=1Dm3...
    ENTRADAS:
     - valor: unidad o vector de valores
     - unidades: unidades en las que se encuentra
    """
    if unidades == 'Hm3':
        return valor*(10**9)
    elif unidades == 'Dm3':
        return valor*(10**6)
    elif unidades == 'm3':
        return valor*(10**3)
    elif unidades == 'L':
        return valor
    elif unidades == 'mm3':
        return valor/(10**6)
```

```
def transformacion_horas(valor, unidades):
    """VERSION 1.

    Transformar valores a horas
    ENTRADAS:
     - valor: unidad o vector de valores
     - unidades: unidades en las que se encuentra
    """
    if unidades == 'ano':
        return valor/(365*24)
    elif unidades == 'dia':
        return valor/(24)
    elif unidades == 'min':
        return valor * 60
    elif unidades == 's':
        return valor * 60 * 60


def transformacion_m3dia(valor, uni_t='ano', uni_m='Hm3'):
    """VERSION 1.

    Transforma las unidades a m3/día
    ENTRADAS:
     - valor: unidad o vector de valores
     - uni_t: unidades en las que se encuentra de tiempo
     - uni_m: unidades en las que se encuentra de volumen
    """
    return
transformacion_horas(transformacion_metros3(valor, uni_m),
uni_t)


def transformacion_Ldia(valor, uni_t='ano', uni_m='Hm3'):
    """VERSION 1.

    ENTRADAS:
     - valor: unidad o vector de valores
     - uni_t: unidades en las que se encuentra de tiempo
     - uni_m: unidades en las que se encuentra de volumen
    """
    return transformacion_horas(transformacion_Litros(valor,
uni_m), uni_t)


def w_s2Kw_h(valor):
    """VERSION 1.

    Transforma de W/s a Kw/h
    ENTRADAS:
     - valor: unidad o vector de valores
     - uni_t: unidades en las que se encuentra de tiempo
```

```
     - uni_m: unidades en las que se encuentra de volumen
    """
    return valor/10**3*60*60

# Tratamos de generalizar las ecuaciones que se utilizan
para los modelados


def generar_grafica_anual(t, media, variacion_anual=0,
max_an=4344,
                          variacion_semanal=0, max_sem=0,
variacion_dia=0,
                          max_dia=0):
    """VERSION 1.

    Genera grafica anual con divisiones por hora
    ENTRADAS:
        - t: lista del tiempo
        - media: media anual
        - variacion_anual: diferencia entre el punto máximo
y el punto mínimo
        a lo largo de un año
        - max_an: hora del máximo anual
        - variacion_semanal: diferencia entre el punto
máximo y el punto
        mínimo a lo largo de una semana
        - max_sem: hora del máximo semanal
        - variacion_dia: diferencia entre el punto máximo y
el punto mínimo a
        lo largo de un día
        - max_dia: hora del máximo diario
    SALIDA:
    Lista del mismo tamaño que t con los datos que simulan
un parámetro por
    cada instante t
    """
    var_an = variacion_anual/2
    var_dia = variacion_dia/2
    var_sem = variacion_semanal/2
    RESULTADO = (np.cos(-
2*np.pi/24*max_dia+2*np.pi/24*t)*var_dia
                 + np.cos(-
2*np.pi/(24*7)*max_sem+2*np.pi/(24*7)*t)*var_sem
                 + np.cos(-
np.pi/(24*375)*max_an+np.pi/(24*375)*t)*var_an
                 + media)
    return RESULTADO

# ------------------------------------------------[ DATOS ]

# n hm3/año
# Afluyentes del Duero
```

```
DatosCaudales={'temega': 394.2, 'Tuela': 134.1, 'Aliste':
133.8, 'Tera': 821.8, 'Duerna': 94.9, 'Eria': 198.6, 'Luna':
557.6, 'Omanas': 348.4, 'Orbigo': 1576.1,
'Tuerto': 334.5, 'Bernega': 619.2, 'Cea': 282.3, 'Curueno':
295.25, 'Esla': 5265.8, 'Porma': 803.7, 'Sequillo': 79,
'Torio': 275.5, 'Valdzeraduey': 185.6,
'Carrion': 579.5, 'Cueza': 38.6, 'Ucieza': 50.2,
'Valdeginate': 53.6, 'Esgueva': 69.8, 'Odra': 68.9,
'Pisuerga': 256.3, 'Valdavia': 162.7, 'Arlanza': 936,
'Arlanzon': 379.2,
'Rituerto': 82.1, 'Ucero': 174.7, 'Riaza': 142.5, 'Duraton':
160.7, 'Adaja': 412.5, 'Arevalillo': 27.76, 'Cega': 232.1,
'Eresma': 256.2, 'Piron': 74.4, 'Voltoya': 57.2,
'Bajoz': 26, 'Gureña': 63.7, 'Trabancosf': 77.1,
'Zapardiel': 44.6, 'Almar': 1223.5, 'Alhandiga': 35.68,
'Aravalle': 145, 'Corneja': 51.9, 'Tormes': 1272.1,
'Valmuza': 50,
'Agadon': 50.4, 'Agueda': 608.7, 'Camaces': 23.1, 'Huebra':
258.7, 'Yeltes': 88.8}

# Datos de lluvias en distintas ciudades Españolas
# Ciudad :  [cuanto ha llovido en total en el mes(mm/m2)] [
días de lluvias en el mes (días)]
DatosPluviales={'Albacete':
[[21,24,28,48,48,36,12,14,32,42,34,28],[4,4,5,6,7,4,1,2,3,5,
5,5]],
    'Alicante':
[[22,26,26,30,30,17,6,8,47,52,42,26],[4,3,4,4,4,2,1,1,3,4,4,
4]],
    'Almeria':
[[23,21,15,20,20,10,1,1,12,28,28,15],[3,3,3,3,2,1,0,0,1,3,3,
3]],
    'Avila':
[[32,22,23,42,42,37,16,19,29,40,43,23],[6,5,4,8,9,5,2,2,4,6,
6,7]],
    'Badajoz':
[[52,43,33,52,52,18,4,5,23,56,64,33],[7,6,5,7,6,6,1,1,5,7,7,
8]],
    'Barcelona' :
[[41,29,42,49,49,42,20,61,85,91,58,42],[5,4,5,5,5,4,2,2,5,5,
5,5]],
    'Bilbao':
[[126,97,94,124,124,64,62,82,74,121,141,94],[13,11,11,13,12,
11,7,7,11,13,13,12]],
    'Burgos':
[[46,42,31,65,65,46,30,27,36,50,56,31],[8,8,6,9,10,8,4,4,6,8
,8,9]],
    'Caceres':
[[58,43,35,49,49,23,7,8,26,59,80,35],[7,7,5,7,6,7,1,1,5,7,7,
9]],
```

```
    'Cadiz' :
[[89,60,42,54,54,13,2,6,22,67,86,42],[7,7,5,6,4,7,0,0,5,7,7,
9]],
    'Castellon':
[[35,26,29,38,38,20,12,29,62,71,41,29],[4,3,4,5,5,3,2,2,4,4,
4,4]],
    'Ceuta':
[[87,87,59,56,56,13,1,1,11,61,76,59],[10,11,9,8,5,11,1,1,9,1
0,10,11]],
    'Ciudad Real':
[[36,34,28,44,44,29,9,7,22,47,42,28],[6,6,5,8,7,6,1,1,5,6,6,
8]],
    'Cordoba':
[[64,53,40,61,61,17,3,3,24,62,85,40],[7,6,5,8,5,6,1,1,5,7,7,
8]],
    'Coruna':
[[128,102,79,85,85,42,30,35,68,110,114,79],[14,14,12,13,11,1
4,5,5,12,14,14,15]],
    'Cuenca':
[[45,41,32,56,56,44,15,17,37,53,49,32],[7,6,6,8,8,6,2,2,6,7,
7,8]],
    'Gerona':
[[65,44,53,67,67,66,30,48,68,83,70,53],[5,5,6,7,7,5,4,4,6,5,
5,5]],
    'Granada':
[[41,38,30,38,38,17,4,3,16,42,48,30],[6,6,5,7,4,6,0,0,5,6,6,
7]],
    'Guadalajara':
[[31,31,31,54,54,51,29,29,44,46,39,31],[6,6,6,9,10,6,4,4,6,6
,6,8]],
    'Huelva':
[[73,43,36,46,46,9,3,4,21,56,74,36],[7,6,5,6,4,6,0,0,5,7,7,8
]],
    'Huesca':
[[39,32,34,53,53,47,20,38,54,54,50,34],[6,5,4,6,8,5,3,3,4,6,
6,6]],
    'Jaen':
[[55,50,44,54,54,18,2,9,26,55,62,44],[6,6,5,7,6,6,0,0,5,6,6,
8]],
    'Las Palmas':
[[18,24,14,7,7,0,0,0,10,13,18,14],[3,3,3,1,0,3,0,0,3,3,3,4]]
,
    'Leon':
[[58,46,29,50,50,39,28,24,39,56,58,29],[8,7,6,8,9,7,4,4,6,8,
8,9]],
    'Lerida':
[[26,14,27,37,37,34,12,21,39,39,28,27],[4,3,4,5,6,3,2,2,4,4,
4,4]],
    'Logroño':
[[27,23,26,44,44,47,31,23,24,31,36,26],[6,5,5,7,8,5,4,4,5,6,
6,6]],
```

```
    'Lugo':
[[122,108,86,94,94,52,34,34,77,115,122,86],[14,13,12,13,13,1
3,5,5,12,14,14,14]],
    'Madrid':
[[37,35,26,47,47,25,15,10,28,49,56,26],[6,6,5,7,8,6,2,2,5,6,
6,7]],
    'Malaga':
[[81,55,49,41,41,12,2,6,16,56,95,49],[6,5,4,5,3,5,0,0,4,6,6,
6]],
    'Mallorca':
[[43,34,26,43,43,11,5,17,39,68,58,26],[5,5,4,6,4,5,1,1,4,5,5
,6]],
    'Melilla':
[[58,58,47,38,38,10,1,3,10,29,44,47],[6,6,5,5,3,6,0,0,5,6,6,
5]],
    'Murcia':
[[25,28,30,27,27,20,5,10,27,44,32,30],[3,3,3,4,4,3,1,1,3,3,3
,4]],
    'Orense':
[[90,81,54,70,70,39,19,23,57,97,93,54],[10,10,8,11,10,10,3,3
,8,10,10,11]],
    'Oviedo':
[[85,85,82,109,109,53,52,55,64,98,101,82],[10,11,11,13,12,11
,8,8,11,10,10,11]],
    'Pamplona':
[[63,52,52,77,77,47,40,43,43,74,80,52],[9,9,8,10,10,9,5,5,8,
9,9,10]],
    'Pontevedra':
[[204,190,126,140,140,66,44,47,108,185,198,126],[15,13,12,13
,13,13,5,5,12,15,15,15]],
    'Salamanca':
[[31,27,22,39,39,34,16,11,32,39,42,22],[6,6,5,7,8,6,3,3,5,6,
6,7]],
    'San Sebastian':
[[168,150,144,168,168,96,98,112,138,174,186,144],[13,12,12,1
4,12,12,9,9,12,13,13,12]],
    'Santander':
[[123,104,105,125,125,62,52,72,85,135,146,105],[13,12,12,13,
11,12,7,7,12,13,13,12]],
    'Segovia':
[[38,34,30,47,47,38,21,21,30,46,48,30],[7,7,6,8,10,7,3,3,6,7
,7,8]],
    'Sevilla':
[[65,54,38,57,57,13,2,6,23,62,84,38],[6,6,5,7,4,6,0,0,5,6,6,
8]],
    'Soria':
[[39,38,28,53,53,46,34,30,31,45,45,28],[7,7,6,8,10,7,4,4,6,7
,7,8]],
    'Tarragona' :
[[38,23,35,40,40,38,15,51,77,65,49,35],[4,3,4,6,6,3,2,2,4,4,
4,4]],
```

```
    'Tenerife' :
[[98,69,65,54,54,12,6,5,20,48,70,65],[8,7,8,6,4,7,1,1,8,8,8,
9]],
    'Teruel':
[[17,14,19,36,36,43,30,40,36,42,22,19],[4,3,4,7,8,3,3,3,4,4,
4,5]],
    'Toledo':
[[28,28,25,41,41,28,12,9,22,38,40,25],[6,5,4,7,7,5,2,2,4,6,6
,6]],
    'Valencia':
[[36,32,35,37,37,23,9,19,51,74,51,35],[4,3,4,5,5,3,1,1,4,4,4
,5]],
    'Valladolid':
[[40,32,23,44,44,33,16,18,31,42,51,23],[7,6,5,8,9,6,3,3,5,7,
7,8]],
    'Vitoria':
[[76,65,61,86,86,51,43,45,42,74,89,61],[10,10,9,12,10,10,5,5
,9,10,10,11]],
    'Zamora':
[[34,28,18,36,36,30,15,13,22,38,42,18],[6,6,4,7,8,6,2,2,4,6,
6,7]],
    'Zaragoza (Aeropuerto)':
[[22,20,20,35,35,31,18,17,27,30,30,20],[4,4,4,5,6,4,3,3,4,4,
4,5]]}

#-------------------------------------------------[ MODELADO
LLUVIA]

# dias_lluvia <- funciones
# DiasLluvia <- clases
# nombres más libertad


def dias_lluvia(DiasDeLluvia, Dias_Mes=Dias_Mes):
    """VERSION 2.

    Los días del mes en el que se van a dar las
precipitaciones
    ENTRADA:
        - DiasDeLluvia: lista con 12 elementos ( uno para
cada mes ) con los
        días de lluvia que tiene cada mes
        - Dias_Mes: días que tiene cada mes
    SALIDA:
        Lista de los días en los que llueve a lo largo de un
año
    """
    dialluvia = []
    dias = 0
    for i in np.arange(len(Dias_Mes)):
        dias = dias+Dias_Mes[i]
        for j in range(DiasDeLluvia[i]):
```

```
            dia_nuevo = random.randrange(1, Dias_Mes[i],
1)+dias-Dias_Mes[i]
            while dia_nuevo in dialluvia:
                dia_nuevo = random.randrange(1,
Dias_Mes[i],1)+dias-Dias_Mes[i]
            dialluvia = dialluvia + [dia_nuevo]
    return dialluvia


def preci_dia(DiasDeLLuvia, precipitacion,
minimo_precipitación=1,
              Dias_Mes=Dias_Mes):
    """VERSION 2.

    Lista de precipitaciones por cada día que llueve
    ENTRADA:
        - DiasDeLLuvia: n° de días en los que llueve de cada
mes
        - precipitacion: precipitaciones de cada mes
        - Dias_Mes: días que tiene cada mes
    SALIDA:
        Lista precipitación/día mm/m2
    """
    LMP = precipitacion.copy()
    PD = []
    for i in np.arange(len(Dias_Mes)):
        random_numbers = np.random.uniform(low=0, high=2,
size=DiasDeLLuvia[i])
        Preci
=(random_numbers/sum(np.round(random_numbers,1))*LMP[i]).tol
ist()
        PD = PD + Preci

    return PD


def preci_hora(Preci_dia, dias_deLluvia, horas_año=8760):
    """VERSION 1.

    Lista anual con precipitaciones de cada dia (haya o no)
    ENTRADA:
        Requiere 2 listas : Preci_dia y dias_deLluvia del
mismo tamaño
        - Preci_dia: la precipitación de cada día
        - dias_deLluvia: los días que llueve en un año
        - horas_año: horas que tiene un año
    SALIDA:
        Lista precipitación-hora mm/m2
    """
    PH = np.zeros(horas_año, dtype=float)
    unos = np.ones(24, float)
    for i in np.arange(len(dias_deLluvia)):
        hora = dias_deLluvia[i]*24
```

```
        PH[hora:(hora+24)] = PH[hora:(hora+24)] +
unos*Preci_dia[i]/24

    return PH


def volumen_preci_total(precipitacion_hora, area_abastecida,
                        unidades_preci='mm',
unidades_area='m2'):
    """VERSION 1.

    Volumen de agua de precipitaciones que han caído por
cada hora
    ENTRADAS:
        - precipitacion_hora: lista de precipitaciones por
hora
        - area_abastecida: área que recolecta el agua de
lluvia
        - unidades_preci: unidades de las precipitaciones
para pasarlo a metros
        - unidades_area: unidades de precipitacion_hora del
area_abastecida
    SALIDA:
        Volumen de agua total recogido por el sistema en m3
    """
    PH = precipitacion_hora.copy()
    if unidades_preci == 'mm':
        PH = PH/(10**3)
    if unidades_area == 'km2':
        area_abastecida = area_abastecida*10**6

    return PH*area_abastecida


def precipitacion_hora(Dias, lluvia):
    """VERSION 1.

    Crea una lista del tamaño de un año con las
precipitaciones que han caído
    en cada instante de tiempo siendo cada instante de
tiempo igual a una hora
    con datos como entrada los días de lluvia y las
precipitaciones totales de
    cada mes
    ENTRADA(2 listas de igual tamaño):
        - dias: días del mes en los que llueve
        - lluvia: cantidad de precipitaciones que ha caído
en el mes
    SALIDA:
        Lista de tamaño las horas que tiene un año con las
precipitaciones que
        han caído en cada hora mm/m2
    """
```

```
    DLluvia = dias_lluvia(Dias)
    PDia = preci_dia(Dias, lluvia)
    return preci_hora(PDia, DLluvia)


def precipitacion_hora_ciudad(diccionario, ciudad):
    """VERSIÓN 1.

    Crea una lista del tamaño de un año con las
precipitaciones que han caído
    en cada instante de tiempo siendo cada instante de
tiempo igual a una hora
    con datos como entrada el diccionario con las ciudades
como claves y los
    datos de las precipitaciones (días que llueven y
precipitaciones totales)
    y la ciudad (la clave) del diccionario
    ENTRADA:
        - diccionario: diccionario en donde se encuentran
los datos de días
        que llueven por mes
            y precipitación total mensual
        - ciudad: clave con la que acceder a los datos de
días que llueve en
        un mes y precipitación total mensual
    SALIDA:
        Lista de tamaño las horas que tiene un año con las
precipitaciones que
        han caído en cada hora mm/m2
    """
    lluvia = DatosPluviales[ciudad][0]
    Dias = DatosPluviales[ciudad][1]

    DLluvia = dias_lluvia(Dias)
    PDia = preci_dia(Dias, lluvia)
    return preci_hora(PDia, DLluvia)

# ------------------------------------------------[
MODELADO DEMANDA DE AGUA]


def consumo_hidrico_habitantes(habitantes, promedio=133,
periodo_horas=24):
    """VERSION 1.

    Calcula el consumo de agua en una población de x
habitantes a la hora con
    el promedio de consumo por habitante/día
    ENTRADAS:
        - habitantes: habitantes de la población
        - promedio: promedio de consumo de agua por persona
en el periodo que
        sea (m3)
```

```
        - periodo_horas: horas entre las que hay que dividir
el promedio
    SALIDA:
    Consumo hídrico total
    """
    return (habitantes*promedio) / periodo_horas


def consumo_hidrico (t, consumo, variacion_anual=0,
max_an=4344,
                     variacion_dia=0, max_dia=0):
    """VERSION 1.

    Crea una lista con el consumo hídrico por cada hora del
tamaño igual a
    la lista t
    ENTRADAS:
        - t: lista del tiempo
        - media: media anual
        - variacion_anual: diferencia entre el punto máximo
y el punto mínimo
        a lo largo de un año
        - max_an: hora del máximo anual
        - variacion_dia: diferencia entre el punto máximo y
el punto mínimo a
        lo largo de un dia
        - max_dia: hora del máximo diario
    SALIDA:
    Lista del mismo tamaño que t con los datos que simulan
un parámetro por
    cada instante t
    """
    var_an = variacion_anual/2
    var_dia = variacion_dia/2

    DIA = np.cos(-2*np.pi/24*max_dia + 2*np.pi/24*t)*var_dia
    ANO = np.cos(-
np.pi*2/(24*375)*max_an+np.pi*2/(24*375)*t)*var_an

    return DIA + ANO + consumo

# ----------------------------------------------[ MODELADO
POTENCIA NECESARIA]


def generar_Potencia_anual(t, media, variacion_doble_an=0,
max_doble_an=5*24,
                           variacion_anual=0, max_an=4344,
variacion_semanal=0,
                           max_sem=74, variacion_dia=0,
max_dia=20, dias=365):
    """VERSION 1.
```

```
    Calcula la demanda de potencia de cada hora con tantas
horas como tamaño
    tiene la lista t
    ENTRADAS:
        - t: lista del tiempo
        - media: media anual
        - variacion_doble_an: diferencia entre el punto
máximo y el punto
        mínimo a lo largo de un año
        - max_doble_an: una de las horas en las que se dan
máximos
        - variacion_anual: diferencia entre el punto máximo
y el punto mínimo
        a lo largo de un año
        - max_an: hora del máximo anual
        - variacion_semanal: diferencia entre el punto
máximo y el punto
        mínimo a lo largo de una semana
        - max_sem: hora del máximo semanal
        - variacion_dia: diferencia entre el punto máximo y
el punto mínimo a
        lo largo de un día
        - max_dia: hora del máximo diario
    """
    var_an = variacion_anual/2
    var_doble_an = variacion_doble_an/2
    var_dia = variacion_dia/2
    var_sem = variacion_semanal/2

    DIA = np.cos(-2*np.pi/24*max_dia + 2*np.pi/24*t)*var_dia
    SEMANA = np.cos(-2*np.pi/(24*7)*max_sem +
2*np.pi/(24*7)*t)*var_sem
    ANO = np.cos(-
np.pi*2/(24*dias)*max_an+np.pi*2/(24*dias)*t)*var_an
    DOBLE_ANO = np.cos(-np.pi*4/(24*dias)*max_doble_an +
np.pi*4/(24*dias)*t)*var_doble_an

    return DIA + SEMANA + ANO + DOBLE_ANO + media

# ---------------------------------------------[ MODELADO
POTENCIA NECESARIA]


def caudal_rio(t, media, variacion_anual=0, max_an=4344,
variacion_semanal=0,
               max_sem=0, variacion_dia=0, max_dia=0,):
    """VERSION 1.

    Calcula el caudal del rio en cada instante de t.
    Los datos han de venir en m3/h
    ENTRADAS:
        - t: lista del tiempo
        - media: media anual
```

```
        - variacion_anual: diferencia entre el punto máximo
y el punto mínimo a
        lo largo de un año
        - max_an: hora del máximo anual
        - variacion_semanal: diferencia entre el punto
máximo y el punto
        mínimo a lo largo de una semana
        - max_sem: hora del máximo semanal
        - variacion_dia: diferencia entre el punto máximo y
el punto mínimo a
        lo largo de un día
        - max_dia: hora del máximo diario
    """
    var_an = variacion_anual/2
    var_dia = variacion_dia/2
    var_sem = variacion_semanal/2

    DIA = np.cos(-2*np.pi/24*max_dia + 2*np.pi/24*t)*var_dia
    SEMANA = np.cos(-2*np.pi/(24*7)*max_sem +
2*np.pi/(24*7)*t)*var_sem
    ANO = np.cos(-
np.pi*2/(24*375)*max_an+np.pi*2/(24*375)*t)*var_an

    return DIA + SEMANA + ANO + media

# ------------------------------------------------[Polinomio
trigonométrico]


def lista_mes_hora(lista_meses):
    """VERSION 1.

    Obtiene una lista con el tiempo desde el día 1 desde las
0 de la
    madrugada hasta la primera hora de cada mes que se
encuentra en la
    lista de entrada

    ENTRADAS:
        - lista_meses: Lista cuyos valores se encuentran
desde el 0 al
        11 y que se corresponden cronológicamente con cada
més del año

    """
    horaMes =
[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,272*
24,303*24,333*24]
    lista_final = np.linspace(0, 0, len(lista_meses))
    i = 0
    for n in lista_meses:
        lista_final[i] = horaMes[n]
        i += 1
```

```
    return lista_final


def lista_semana_hora(lista_semana):
    """VERSION 1.

    Obtiene una lista con el tiempo desde el el lunes desde
las 0 de
    la madrugada hasta la primera hora de cada día que se
encuentra en
    la lista de entrada

    ENTRADAS:
        - lista_semana: Lista cuyos valores se encuentran
desde el 0
        al 6 y que se corresponden cronológicamente con cada
día de la
        semana

    """
    horaSemana = [0, 24, 48, 72, 96, 120, 144]
    lista_final = np.linspace(0, 0, len(lista_semana))
    i = 0
    for n in lista_semana:
        lista_final[i] = horaSemana[n]
        i += 1
    return lista_final


def poly_odd(time_t, list_t, list_A, T=365*24):
    """VERSION 1.

    Calcula el polinomio interpolador trigonométrico para
todos los
    instantes en time_t

    ENTRADAS:
        - time_t: lista de instantes t
        - lista_t: lista de tiempos (x)
        - lista_A: lista de la amplitud
        - T: Periodo
    """
    def poly_odd_ind (t, list_t, list_A, K):
        """VERSION 1.
        Calcula el polinomio interpolador trigonométrico
instante de t[REVISAR]

        ENTRADAS:
            - t: tiempo t
            - lista_t: lista de tiempos (x)
            - lista_A: lista de la amplitud
            - k: frecuencia
        """
```

```
        list_pant = []
        i = -1
        sum = 0
        for tk in list_t:
            i += 1
            mult = list_A[i]
            for tn in list_t:
                if tn != tk:
                    mult=mult*np.sin(1/2*(K*t-
K*tn))/np.sin(1/2*(tk-tn)*K)
            sum += mult
        return sum

    K = 2*np.pi/T
    i = 0
    sol = np.zeros(len(time_t))
    for t in time_t:
        sol[i] = poly_odd_ind(t, list_t, list_A, K)
        i += 1

    return sol
```

ANEXO 6.2 MÓDULO DE HERRAMIENTAS PRESAS

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

dias_año = 365
horas_dia = 24
horas_año = dias_año*horas_dia

tini = 0
tfin = horas_año
t_total = np.arange(tfin-tini)
# ------------------------------------------------
[CLASES]


class Aliviadero:
    """Versión  1."""

    def __init__(self, Area=0, Hentrada=10, Hsalida=5,
Coeficiente_Descarga=1E-4):
        """Versión  1.

        ATRIBUTOS
            -Area: area de entrada al aliviadero
            -Hentrada: altura de abertura de la presa al
aliviadero
            -Hsalida: altura en la que se vierte el líquido
            -Coeficiente_Descarga: Cp * Cd
                -Cp:
                -Cd:
        """
        self.Area = Area
        self.Hentrada = Hentrada
        self.Hsalida = Hsalida
        self.Coeficiente_Descarga = Coeficiente_Descarga

    def calcula_Q(self,z,g=9.8):
        """Versión  1.

        Calcula el caudal de salida del aliviadero en m3/s
        ENTRADAS:
            -z: altura de la central
            -g: gravedad
        """
        aliv = self.Coeficiente_Descarga *
np.sqrt(2*g*np.abs(z-self.Hentrada))*self.Area
        return np.heaviside(z - self.Hentrada,  0 ) * aliv
```

```
class Presa:
    """Versión  3."""

    def __init__(self, t_total, turbina=None,
Nombre='presa1',
                 Tipo_central='Embalse', Q_entrada=0,
Presa_salida='',
                 Presa_de_absorcion=[], Presa_salida_t=1,
P_Qg=0.9,
                 H_min=2, H_max=10, H_des=100, Hi=0,
base=2000,
                 base_absorcion=3000, base_desbordado=3000,
Hs=0,
                 Qdesbordado=0, aliviadero=Aliviadero()):
        """Inicializar los datos (Versión  2).

        ATRIBUTOS:
            -t_total: Tiempo de la simulación
            -Nombre: Nombre único de la presa
            -Tipo_central: que tipo de central es:
                -Presa
                -Fluyente
            -Q_entrada: lista con el caudal de entrada por
cada instante. del
            mismo tamaño de t_total
            -Presa_entrada: listas con las presas de entrada
            -Presa_salida: lista con las presas de salidas
            -Presa_salida_t: tiempo en llegar a la presa de
salida
            -P_Qg: Potencia generada en w/s
            -H_min: Altura mínima
            -H_max: Altura máxima
            -H_des: Altura de desbordamiento
            -Hi: Altura inicial del sistema
            -base: base de la presa
            -base_absorcion: base para los cálculos de
lluvias
            -base_desbordado:base para los cálculos en la
situación de
            desbordamiento
            -Hs: Altura de salida de la turbina
            -D_aliviadero: de versiones anteriores empleado
para el aliviadero
            -Qdesbordado: caudal máximo de absorción en caso
de desbordamiento
            -aliviadero: objeto de la clase aliviadero()->
datos para controlar
            el funcionamiento de este
        """
        self.Nombre = Nombre
        if Tipo_central[0] == 'd' or Tipo_central[0] == 'D':
```

```
            self.Tipo_central = 'Derivación'
        elif Tipo_central[0] == 'f' or Tipo_central[0] ==
'F':
            self.Tipo_central = 'Fluyente'
        elif Tipo_central[0] == 'b' or Tipo_central[0] ==
'B':
            self.Tipo_central = 'Bombeo'
        elif (Tipo_central[0] == 'e' or Tipo_central[0] ==
'E'
              or Tipo_central[0] == 'r' or Tipo_central[0]
== 'R'
              or Tipo_central[0] == 'p' or Tipo_central[0]
== 'P'):
            self.Tipo_central = 'Regulacion'
        else:
             assert False, 'No es valido el tipo de central'

        self.retardo_salida = Presa_salida_t
        self.turbina = turbina

        self.H_min = H_min
        self.H_max = H_max
        self.H_des = H_des
        self.Hi = Hi
        self.base = base
        self.base_absorcion = base_absorcion
        self.base_desbordado = base_desbordado
        self.Hs = Hs
        self.Qdesbordado = Qdesbordado
        self.aliviadero = aliviadero
        # Presas flullentes
        self.P_Qg = P_Qg
        # Presas de conexión
        self.Presa_entrada = []
        self.Presa_de_salida = Presa_salida
        self.Presa_de_absorcion = Presa_de_absorcion
        # Caudal de entrada
        if type(Q_entrada) == int:
            self.Q_entrada = np.linspace(0, 0, len(t_total))
            self.Qentra_m3s = np.linspace(0, 0,
len(t_total))
        else:
            self.Q_entrada = Q_entrada.copy()
            self.Qentra_m3s = Q_entrada.copy()/3600

        self.t_total = t_total
        self.t_len = len(t_total)

    def desbordado_Funcion(self,H):
        return self.Qdesbordado*np.heaviside(H - self.H_des,
0 )

    def indicar_Qentra(self, caudal, t_total=0):
```

```
        """Versión  1.

        Modifica el caudal de entrada introduciendo los
caudales en m3/h
        ENTRADAS:
            -Caudal: caudal de entrada nuevo
            -t_total: tiempo de la simulación
        """
        if type(caudal) == int:
            self.Q_entrada = np.linspace(caudal, caudal,
len(t_total))
            self.Qentra_m3s = np.linspace(caudal/3600,
caudal/3600,
                                          len(t_total))
        else:
            self.t_total
            self.Q_entrada = caudal.copy()
            self.Qentra_m3s = caudal.copy()/3600

    def indicar_Qentra_m3s(self, caudal, t_total=0):
        """Versión  1.

        Modifica el caudál de entrada introduciendo los
caudales en m3/s
        ENTRADAS:
            -Caudal: caudal de entrada nuevo
            -t_total: tiempo de la simulación
        """
        if type(caudal) == int:
            self.Q_entrada = np.linspace(caudal*3600,
caudal*3600,
                                         len(t_total))
            self.Qentra_m3s = np.linspace(caudal, caudal,
len(t_total))
        else:
            self.Q_entrada = caudal.copy()*3600
            self.Qentra_m3s = caudal.copy()

    def __str__(self):
        """Versión  1."""
        return self.Nombre


class Presa_Datos:
    """Versión 1.

    Clase para los calculos del sistema y el almacenamiento
de los resultados
    """

    def __init__(self, presa, t_total, Nombre='Dato'):
        """Versión  4.
```

```
        Inicializar los datos
        ATRIBUTOS:
            - presa: objeto de presa
            - turbina: objeto de turbina
            - bomba: objeto de bomba
            - Hi: altura inicial
            - Vi: volumen inicial
            - t_total: lista con cada índice de la
simulación desde el
            instante inicial hasta el final
            - Nombre: nombre de los datos
            - Tipo_central: tipo de la central
            - Nombre_turbina: nombre de la turbina
            - Qg: caudal del generador
            - QsaleTotal:
            - V: volumen de la central en cada instante de
tiempo
            - H: altura de la central en cada instante de
tiempo
            - Pg_ms: potencia generada en la central por
cada instante de
            tiempo
        """
        self.t_len = len(t_total)

        if self.t_len > 0:
            # Presa vs central_flullente
            self.Tipo_central = presa.Tipo_central
            try:
                self.Nombre_turbina = presa.turbina.Nombre
                self.Nombre = 'Datos:'+presa.Nombre+'
'+presa.Tipo_central+' '+presa.turbina.Nombre
            except:
                raise Exception("Error en la inicialización
del objeto presa_Datos; Falta una turbina")

            try:
                self.V = np.linspace(0, 0, self.t_len)
                self.H = np.linspace(0, 0, self.t_len)
                self.Pg_ms = np.linspace(0, 0, self.t_len)
                self.NumPresaEntrada = []

                # Caudales de entrada y salida
                self.Qg = np.linspace(0, 0, self.t_len)
                self.QsaleTotal = np.linspace(0, 0,
self.t_len)
                self.QsaleRio = np.linspace(0, 0,
self.t_len)
                self.QentraTotal = np.linspace(0, 0,
self.t_len)

                # Caudales de ríos
                self.Qe = presa.Q_entrada.copy()
```

```
                self.Qe_m3s = presa.Qentra_m3s.copy()

                # Interconexión
                self.Qe_central = np.linspace(0, 0,
self.t_len)  # Qg

                # Centrales de bombeo
                self.Qs_central = np.linspace(0, 0,
self.t_len)  # Qb
                self.Qb = np.linspace(0, 0, self.t_len)
                self.Pb = np.linspace(0, 0, self.t_len)

                # Caudales salida de emergencia
                self.Valiviadero = np.linspace(0, 0,
self.t_len)
                self.Qdesbordamiento = np.linspace(0, 0,
self.t_len)

            except:
                raise Exception("Error en la inicialización
del objeto presa_Datos; atributos comunes")

            if presa.Tipo_central[0] == 'F' or
presa.Tipo_central[0] =='D' :
                self.H = np.linspace(presa.Hi, presa.Hi,
self.t_len)
                self.V = None
                self.Valiviadero = None
            elif presa.Tipo_central[0] == 'B' or
presa.Tipo_central[0] =='b' :

                try:
                    self.Nombre +=' '+presa.bomba.Nombre
                except:
                    raise Exception("Error en la
inicialización del objeto presa_Datos, No hay bomba")
            else:
                self.V[0] = presa.base * presa.Hi
                self.H[0] = presa.Hi

        else:
            raise Exception("Error en la inicialización del
objeto presa_Datos")

    def selector_Qdesbordamiento(self, t, presa, necesidad,
intervalo=3600):
        """Versión  1.

        Calcula el caudal que sale por el desbordamiento
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa
            -necesidad: datos de necesidades
```

```
            -intervalo: segundos de cada iteración
        """
        Qsale1 = self.QentraTotal[t] -(self.Qg[t] +
self.Valiviadero[t]/intervalo + necesidad.Qsale_m3s[t] +
self.Qs_central[t])
        if self.H[t] > presa.H_des:

            Q_necesario = presa.base_desbordado*(self.H[t]-
presa.H_des)/intervalo
            Q_desb = Q_necesario - (Qsale1)
            if Q_desb < presa.Qdesbordado and Q_desb > 0:
                self.Qdesbordamiento[t] = Q_desb

            elif Q_desb > presa.Qdesbordado  :
                self.Qdesbordamiento[t] = presa.Qdesbordado
            else:
                self.Qdesbordamiento[t] = 0

    def selector_Qg(self, t, necesidad, presa, g=9.8,
den=998, intervalo=3600):
        """Versión  2.

        Calcula el caudal del generador
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa y datos de la turbina
            -necesidad: datos de necesidades
            -g: gravedad
            -den: densidad
            -intervalo: segundos de cada iteración
        """
        presa.turbina.Qg_funcion_propia(self,t, necesidad,
presa,g=g, den=den, intervalo=intervalo)

    def selector_Qb(self, t, necesidad, presa, listDatos,
g=9.8, den=998, intervalo=3600):
        """Versión  2.

        Calcula el caudal de la bomba
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa y datos de la turbina
            -necesidad: datos de necesidades
            -g: gravedad
            -den: densidad
            -intervalo: segundos de cada iteración
        """
        presa.bomba.Qb_funcion_propia(self,t=t, presa =
presa, necesidad=necesidad, listDatos=listDatos,g=g,
den=den, intervalo=intervalo)

    def selector_QentraTotal(self, t):
        """Versión  1.
```

```
        Calcula el caudal de entrada en cada instante de
tiempo
        ENTRADAS:
            -t: instante de tiempo de simulación
        """
        self.QentraTotal[t] = self.Qe_m3s[t] +
self.Qe_central[t] +  self.Qb[t]

    def selector_QsaleTotal(self, t, necesidad, presa,
intervalo=3600):
        """Versión  2.

        Calcula el caudal total que sale en cada instante de
tiempo por la presa
        ENTRADAS:
        -t: instante de tiempo de simulación
        -presa: datos de la presa
        -necesidad: datos de necesidades
        """
        self.QsaleTotal[t] = self.Qg[t] +
self.Valiviadero[t]/intervalo + self.Qdesbordamiento[t]  +
necesidad.Qsale_m3s[t] + self.Qs_central[t]

    def selector_QsaleRio(self, t, presa, intervalo=3600):
        """Versión  2.

        Calcula el caudal que sale en cada instante de
tiempo por la presa por
        el río
        ENTRADAS:
        -t: instante de tiempo de simulación
        -presa: datos de la presa
        """
        self.QsaleRio[t] = self.Qg[t] +
self.Valiviadero[t]/intervalo + self.Qdesbordamiento[t]

    def selector_Pg_ms(self, t, presa, g=9.8, den=998):
        """Versión  2.

        Calcula la potencia generada en cada instante de
tiempo
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa
                -turb: datos de la turbina
            -g: gravedad
            -den: densidad
        """
        self.Pg_ms[t] =
self.Qg[t]*g*den*self.H[t]*presa.turbina.r  # m2Kg/s3
```

```
    def selector_Pb(self, t, presa, necesidad, g=9.8,
den=998, intervalo=3600):
        """Versión  1.

        Calcula la potencia consumida en cada instante de
tiempo por la bomba
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa
                -bomba: datos de la turbina
            -g: gravedad
            -den: densidad
            -intervalo: segundos de cada intervalo
        """
        presa.bomba.Pb_funcion_propia(self, t=t,
presa=presa,
                                      necesidad=necesidad,
g=g, den=den,
                                      intervalo=intervalo)

    def selector_H(self, t, presa):
        """Versión  1.

        Calcula la altura en cada instante de tiempo para
una presa
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa
        """
        H_des = presa.H_des
        base = presa.base
        base_desbordado = presa.base_desbordado
        H = self.V[t] / base - presa.Hs
        if H > H_des:
            H = H_des+(self.V[t]-base*H_des) /
base_desbordado
            self.H[t] = H

        else:
            self.H[t] = H

    def selector_V(self, t, intervalo=3600):
        """Versión  2.

        Calcula el volumen de la presa en cada instante de
tiempo
        ENTRADAS:
            -t: instante de tiempo de simulación
            -intervalo: segundos de cada iteración
        SALIDA:
            modifica V en la posición t. el valor
corresponde al volumen que
            había al comienzo de esta iteración
```

```
        """
        self.V[t] = self.V[t-1] + self.QentraTotal[t-
1]*intervalo - self.QsaleTotal[t-1]*intervalo



    def selector_Valiviadero(self, t, presa, necesidad,
g=9.8, den=998,
                             intervalo=3600):
        """Versión  6.

        Calcula el caudal del aliviadero en cada instante de
tiempo
        ENTRADAS:
            -t: instante de tiempo de simulación
            -presa: datos de la presa
            -necesidad: datos de necesidades
            -g: gravedad
            -den: densidad
            -intervalo: segundos de cada iteración
        """
        Qentra = self.QentraTotal[t] - self.Qg[t] -
necesidad.Qsale_m3s[t] + self.Qs_central[t]

        if (self.H[t] > presa.aliviadero.Hentrada) :
            V1 =
presa.aliviadero.calcula_Q(self.H[t],g=9.8)*intervalo
            H = (self.H[t]-presa.aliviadero.Hentrada)
            V2 = H*presa.base + (Qentra)*intervalo

            if (V2 < V1 and V2 > 0):
                self.Valiviadero[t] = np.heaviside(V2,
0)*V2

            elif (V1 > 0 and V2 > 0):
                self.Valiviadero[t] = np.heaviside(V1,
0)*V1

            else:
                HF = self.H[t] - (V1 +
Qentra*intervalo)/presa.base

                m = (HF - self.H[t])/intervalo
                x = (presa.aliviadero.Hentrada -self.H[t])/m

                if x < intervalo and t > 0:
                    V3 = V1*x/intervalo
                    self.Valiviadero[t] = np.heaviside(V3,
0)*V3
                else:
                    self.Valiviadero[t] = 0
```

```
        elif self.H[t]+Qentra*intervalo/presa.base >
presa.aliviadero.Hentrada:
            V1 =
presa.aliviadero.calcula_Q(presa.aliviadero.Hentrada,g=9.8)*
intervalo
            Q1 =
presa.aliviadero.calcula_Q(presa.aliviadero.Hentrada,g=9.8)*
intervalo

            if (Qentra - Q1< 0):
                V4 = self.V[t] + Qentra*intervalo -
presa.aliviadero.Hentrada*presa.base
                self.Valiviadero[t] =np.heaviside(V4,  0
)*V4

            else:
                HF = self.H[t] -
(Qentra*intervalo)/presa.base
                m = (HF - self.H[t])/intervalo
                x = (presa.aliviadero.Hentrada -self.H[t])/m

                if x < intervalo and t > 0:
                    V3 = V1*x/intervalo
                    self.Valiviadero[t] = np.heaviside(V3,
0)*V3
                else:
                    self.Valiviadero[t] = 0
        else:
            self.Valiviadero[t] = 0


    def __str__(self):
        """Versión  1.

        Devuelve información del objeto en cadena string
        """
        string = 'datos de la presa:'+self.Nombre+' con
turbina:'
        +self.Nombre_turbina
        return string


class Turbina:
    """Versión 1.

    Clase que posee los datos de la turbina
    """

    def __init__(self, Nombre='t_prueba1', r=0.85,
Q_max=None, Q_min=None,
                 Qg_funcion_propia=0, alpha=0.9):
        """Versión 2.
```

```
        Inicializar los datos de la turbina
        ATRIBUTOS:
            -Nombre: nombre de la turbina
            -r: rendimiento de la turbina
            -Q_max: caudal máximo al que puede funcionar la
turbina
            -Q_min: caudal mínimo al que puede funcionar la
turbina
            -Qg_funcion_propia: función para calcular que
pasa por la turbina
            -alpha: parámetro
        """
        self.Nombre = Nombre
        self.Q_max = Q_max
        self.Q_min = Q_min
        self.r = r
        self.Qg_funcion_propia = Qg_funcion_propia
        self.alpha = alpha


    def __str__(self):
        """Versión  1.

        Devuelve información del objeto en cadena string
        """
        string = self.Nombre+' r'+str(self.r)
        return string


class Bomba:
    """Versión 1.

    Clase que posee los datos de la bomba
    """

    def __init__(self, Nombre='bomba1', r=0.85,
Qb_funcion_propia=0,
                 Pb_funcion_propia=0, alpha=0.9, area=3):
        """Versión 1.

        Inicializa los datos de la bomba
        ATRIBUTOS:
            -Nombre: nombre de la bomba
            -r: rendimiento de la bomba
            -Qb_funcion_propia: función para calcular el
caudal que pasa por la
            bomba
            -Pb_funcion_propia: función para calcular la
potencia consumida por
            la bomba
        """
        self.Nombre = Nombre
        self.r = r
```

```
        self.Qb_funcion_propia=Qb_funcion_propia
        self.Pb_funcion_propia=Pb_funcion_propia
        self.alpha=alpha
        self.area=area
    # presa.bomba.Qb_funcion_propia(self,t, presa,
necesidad, listDatos, g=9.8, den=998, intervalo=3600)
    # presa.bomba.Pb_funcion_propia(self,t, presa, g=9.8,
den=998, intervalo=3600)

    def __str__(self):
        """Versión  1.

        Devuelve información del objeto en cadena string
        """
        string = self.Nombre+' r'+str(self.r)
        return string


class Necesidades:
    """Versión 1.

    Clase que posee las necesidades de agua y de potencia en
cada instante de
    tiempo
    """

    def __init__(self, t_total, Nombre='Necesidades1',
Potencia_necesaria=0,
                 Caudal_necesario=0):
        """Versión  1.

        ATRIBUTOS:
            - t_total:
            - Nombre: Nombre descriptivo de las necesidades
            - Potencia_necesaria: potencia demandada
            - Caudal_necesario: caudal demandado
        """
        self.Nombre = Nombre
        t_len =len(t_total)
        if type(Caudal_necesario) == int:
            self.Caudal_necesario = np.linspace(0, 0, t_len)
            self.Qsale_m3s = np.linspace(0, 0, t_len)
        else:
            self.Caudal_necesario = Caudal_necesario.copy()
            self.Qsale_m3s = Caudal_necesario.copy()/3600

        if type(Potencia_necesaria) == int:
            self.Potencia_necesaria = np.linspace(0, 0,
t_len)  # MW
            self.Pgn_ms = np.linspace(0, 0, t_len) # W
        else:
            self.Potencia_necesaria =
Potencia_necesaria.copy()
```

```
            self.Pgn_ms = Potencia_necesaria.copy()*1000000
# (MW a W)

    def division_necesidad(self, t_total, listCentrales,
lista_division):
        """Versión  1.

        Divide los datos del objeto necesidades y los mete
en nuevos objetos
        de la misma clase del mismo tamaño que tiene
listaCentrales
        ENTRADAS:
            -t_total:
            -listCentrales: lista con todas las centrales
            -lista_division: lista de tamaño 2* lista de
necesidades con una
            fila para la demanda de potencia y la segunda
para la demanda de
            caudal
        SALIDAS:
            Lista de objetos de la clase necesidades con la
division de esta
        """
        listNecesidades = []
        for i in range(len(listCentrales)):
            pot =
self.Potencia_necesaria*lista_division[0][i]
            caudal =
self.Caudal_necesario*lista_division[1][i]
            necesidad_x = Necesidades(t_total,
Potencia_necesaria=pot,

Caudal_necesario=caudal)
            listNecesidades = listNecesidades +
[necesidad_x]
        return listNecesidades

# ------------------------------------------------
[Iteradores]


def instante_t_regulacion(datos, t, presa, necesidad, g=9.8,
den=998,
                          intervalo=3600):
    """Versión  2.

    Calcula las magnitudes características de la central
para un sistema en el
    tiempo t.

    Esta función invoca a un conjunto de métodos de la clase
Datos, que se
```

```
    ejecutan para cada instante de tiempo t del total para
simular la central
    de regulación.
    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t

    SALIDA:
        -Objeto datos actualizado en el instante t
    """
    datos.selector_V(t, intervalo=3600)
    datos.selector_H(t, presa=presa)

    datos.selector_QentraTotal(t)
    datos.selector_Qg(t, necesidad, presa=presa, g=g,
den=den, intervalo=intervalo)
    datos.selector_Valiviadero(t, presa=presa,
necesidad=necesidad, g=g,
                               den=den)
    datos.selector_Qdesbordamiento(t, presa=presa,
necesidad=necesidad, intervalo=intervalo)
    datos.selector_QsaleTotal(t, necesidad, presa)
    datos.selector_QsaleRio(t, presa=presa)

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)

    return datos


def instante_t_derivacionfluyente(t, datos, presa,
necesidad, g=9.8, den=998,
                                  intervalo=3600):
    """Versión  1.

    Calcula las magnitudes características de la central
para un sistema en el
    tiempo t.

    Esta función invoca a un conjunto de métodos de la clase
Datos, que se
    ejecutan para cada instante de tiempo t del total para
simular la central
    de derivación o la central fluyente.
    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
```

```
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t

    SALIDA:
        -Objeto datos actualizado en el instante t
    """
    datos.selector_QentraTotal(t)
    datos.Qg[t] = (datos.QentraTotal[t] -
necesidad.Qsale_m3s[t])*presa.P_Qg
    datos.QsaleRio[t] = datos.QentraTotal[t] -
necesidad.Qsale_m3s[t]
    datos.QsaleTotal[t] = datos.QentraTotal[t]

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)

    return datos


def instante_t_bombeoS(datos, t, necesidad, presa, g=9.8,
den=998,
                       intervalo=3600):
    """Versión  2.

    Calcula las magnitudes características de la central
para un sistema en el
    tiempo t.

    Esta función invoca a un conjunto de métodos de la clase
Datos, que se
    ejecutan para cada instante de tiempo t del total para
simular la central
    de bombeo.
    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -necesidad: objeto de la clase Necesidades
        -presa: objeto de la clase Presa
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t

    SALIDA:
        -Objeto datos actualizado en el instante t
    """
    datos.selector_V(t, intervalo=3600)
    datos.selector_H(t, presa=presa)

    datos.selector_Qb(t, necesidad, presa=presa,
listDatos=None, g=g, den=den,
                      intervalo=intervalo)
    datos.selector_QentraTotal(t)
```

```
    datos.selector_Qg(t, necesidad, presa=presa, g=g,
den=den, intervalo=intervalo)

    datos.selector_Valiviadero(t, presa=presa,
necesidad=necesidad, g=g,
                               den=den)
    datos.selector_Qdesbordamiento(t, presa=presa,
necesidad=necesidad, intervalo=intervalo)
    datos.selector_QsaleTotal(t, necesidad, presa)
    datos.selector_QsaleRio(t, presa=presa)

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)
    datos.selector_Pb(t, presa=presa, necesidad=necesidad,
g=9.8, den=998)

    return datos


def instante_t_bombeoM1(datos, t, necesidad, presa,
listDatos=None, g=9.8,
                        den=998, intervalo=3600):
    """Versión  2.

    ENTRADAS:
        -datos: objeto de la clase Presa_Datos
        -t: instante de tiempo de simulación
        -necesidad: objeto de la clase Necesidades
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -listDatos: lista de todas las centrales en sistema
múltiple
    SALIDA:
        -Objeto datos actualizado en el instante t
    """
    datos.selector_V(t, intervalo=3600)
    datos.selector_H(t, presa=presa)
    datos.selector_Qb(t, necesidad, presa=presa,
listDatos=listDatos, g=g,
                      den=den, intervalo=intervalo)
    return datos


def instante_t_bombeoM2(datos, t, necesidad=None,
presa=None, listDatos=None,
                        g=9.8, den=998, intervalo=3600):
    """Versión  1.

    ENTRADAS:
        -t: instante de tiempo de simulación
        -presa: datos de la presa
        -necesidad: datos de necesidades
        -listDatos: lista de datos en el sistema múltiple
    SALIDA:
```

```
        -Objeto datos actualizado en el instante t
    """
    datos.selector_QentraTotal(t)
    datos.selector_Qg(t, necesidad, presa=presa, g=g,
den=den, intervalo=intervalo)
    datos.selector_Valiviadero(t, presa=presa,
necesidad=necesidad, g=g,
                               den=den)
    datos.selector_Qdesbordamiento(t, presa=presa,
necesidad=necesidad, intervalo=intervalo)
    datos.selector_QsaleTotal(t, necesidad, presa)
    datos.selector_QsaleRio(t, presa=presa)

    datos.selector_Pg_ms(t, presa=presa, g=9.8, den=998)
    datos.selector_Pb(t, presa=presa, necesidad=necesidad,
g=9.8, den=998)

    return datos


# ----------------------------------------------------
[Relaciones]

def actualizar_QentraCentral(t, listDatos, listPresa):
    """Versión  1.

    Actualiza los caudales de entrada procedente de otras
centrales en el
    instante t
    ENTRADAS:
        -t: instante de tiempo de simulación
        -listDatos: lista de objetos de la clase Presa_Datos
(datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA:
        -Lista de objeto datos
    """
    for n in range(len(listDatos)):
        for m in listDatos[n].NumPresaEntrada:
            retardo = t-listPresa[m].retardo_salida
            listDatos[n].Qe_central[t] +=
listDatos[m].QsaleRio[retardo]

    return listDatos


def interconexion__QentraCentral(listDatos, listPresa):
    """Versión  1.

    Realiza las interconexiones directas, procedentes de los
caudales de
```

```
    salida de otras centrales
    ENTRADAS:
        -listDatos: lista de objetos de la clase
Presa_Datos(datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa
    SALIDA:
        -Lista de objeto datos
    """
    for n in range(len(listDatos)):
        listDatos[n].NumPresaEntrada = []
        for m in range(len(listDatos)):
            if listPresa[n].Nombre ==
listPresa[m].Presa_de_salida:
                listDatos[n].NumPresaEntrada += [m]

    return listDatos


def actualizar_QsaleCentral(t, listDatos, listPresa):
    """Versión  1.

    Actualiza los caudales de entrada procedente de otras
centrales en el
    instante t
    ENTRADAS:
        -t: instante de tiempo de simulación
        -listDatos: lista de objetos de la clase Presa_Datos
(datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa
    SALIDA:
        -Lista de objeto datos
    """
    for n in range(len(listDatos)):
        for m in listDatos[n].NumPresaEntrada:
            listDatos[n].Qs_central[t] += listDatos[m].Qb[t]

    return listDatos


def interconexion__QsaleCentral(listDatos, listPresa):
    """Versión  1.

    Realiza las interconexiones de bombeo
    ENTRADAS:
        -listDatos: lista de objetos de la clase
Presa_Datos(datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa

    SALIDA:
        -Lista de objeto datos
```

```
    """
    for n in range(len(listDatos)):
        listDatos[n].NumPresaBombeo = []
        for m in range(len(listDatos)):
            if listPresa[n].Nombre ==
listPresa[m].Presa_de_absorcion:
                listDatos[n].NumPresaEntrada += [m]

    return listDatos


def multipleNL__QsaleCentral(t,listDatos, listPresa,
necesidad, g=9.8, den=998,
                             intervalo=3600):
    """Versión  1.

    Actualiza los caudales de salida procedente del bombeo
de otras centrales
    en el instante t
    ENTRADAS:
        -t: instante de tiempo de simulación
        -listDatos: lista de objetos de la clase
Presa_Datos(datos en el sistema múltiple)
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t

    SALIDA: Lista de objeto datos actualizado en el instante
t
    """
    for n in range(len(listDatos)):
        if listDatos[n].Tipo_central[0]=='B' or
listDatos[n].Tipo_central[0]=='b':
            listDatos[n] =
instante_t_bombeoM1(listDatos[n],t, necesidad,
presa=listPresa[n],listDatos=listDatos, g=g, den=den,
intervalo=intervalo)

    return listDatos

def multipleNO__QsaleCentral(t,listDatos, listPresa,
necesidad, g=9.8, den=998,
                             intervalo=3600):
    """Versión  1.

    Actualiza los caudales de salida procedente del bombeo
de otras centrales
    en el instante t
    ENTRADAS:
        -t: instante de tiempo de simulación
```

```
        -listDatos: lista de objetos de la clase
Presa_Datos(datos en el
        sistema múltiple)
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Lista de objeto datos actualizado en el instante
t
    """
    for n in range(len(listDatos)):
        if listDatos[n].Tipo_central[0] == 'B' or
listDatos[n].Tipo_central[0]=='b':
            listDatos[n] = instante_t_bombeoM1(listDatos[n],
t, listDatos=listDatos, presa=listPresa[n],
necesidad=necesidad, g=g, den=den, intervalo=intervalo)
    return listDatos


# ---------------------------------------------------
[Simulación]


def sistema_fluyente(t_total, presa, necesidad, g=9.8,
den=998, intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con una sola central
fluyente
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades

        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t

    SALIDA: Resultado de la simulación
        -objeto de la clase Presa_Datos
    """
    datos = Presa_Datos(presa=presa, t_total=t_total)
    for t in t_total[1:]:
        instante_t_derivacionfluyente(t, datos, presa,
necesidad=necesidad,
                                      g=g, den=den,
intervalo=intervalo)
    return datos
```

```
def sistema_derivacionfluyente(t_total, presa, necesidad,
g=9.8, den=998,
                               intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con una sola central
fluyente
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades

        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -objeto de la clase Presa_Datos
    """
    datos = Presa_Datos(presa=presa, t_total=t_total)
    for t in t_total[1:]:
        instante_t_derivacionfluyente(t, datos, presa,
necesidad=necesidad,
                                      g=g, den=den,
intervalo=intervalo)
    return datos


def sistema_embalse(t_total, presa, necesidad, g=9.8,
den=998, intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con una sola presa
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -objeto de la clase Presa_Datos
    """
    datos = Presa_Datos(presa=presa, t_total=t_total)
    for t in t_total[1:]:
        datos = instante_t_regulacion(datos, t,
necesidad=necesidad,
                                      presa=presa, g=g,
den=den,
                                      intervalo=intervalo)
```

```
    return datos  # en SI m3 m


def sistema_regulacion(t_total, presa, necesidad, g=9.8,
den=998,
                       intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con una sola presa
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -objeto de la clase Presa_Datos
        """

    datos = Presa_Datos(presa=presa, t_total=t_total)
    for t in t_total[1:]:
        datos = instante_t_regulacion(datos, t,
necesidad=necesidad,
                                      presa=presa, g=g,
den=den,
                                      intervalo=intervalo)
    return datos  # en SI m3 m


def sistema_bombeo(t_total, presa, necesidad, g=9.8,
den=998, intervalo=3600):
    """Versión  1
    Hace la simulación de un sistema con una sola presa con
bombeo

    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante inicial hasta el final
        -presa: objeto de la clase Presa
        -necesidad: objeto de la clase Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -objeto de la clase Presa_Datos
    """

    datos = Presa_Datos(presa=presa, t_total=t_total)
    for t in t_total[1:]:
```

```
        datos = instante_t_bombeoS(datos, t,
necesidad=necesidad, presa=presa,
                                   g=g, den=den,
intervalo=intervalo)
    return datos  # en SI m3 m


def sistema_multipresa(t_total, listPresa, necesidad, g=9.8,
den=998,
                       intervalo=3600):
    """Versión  4.

    Hace la simulación de un sistema con múltiples centrales
de tipo regulacion
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase
Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """
    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos +
[Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos,
listPresa= listPresa)

    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] =
instante_t_regulacion(listDatos[i], t,
necesidad=necesidad[i],

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
```

```
                listDatos[i] =
instante_t_regulacion(listDatos[i], t, necesidad=necesidad,

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    return listDatos


def sistema_multiregulacion(t_total, listPresa, necesidad,
g=9.8, den=998,
                            intervalo=3600):
    """Versión  4.

    Hace la simulación de un sistema con múltiples centrales
de tipo regulacion
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase
Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """
    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos +
[Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos,
listPresa= listPresa)

    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] =
instante_t_regulacion(listDatos[i], t,
necesidad=necesidad[i],

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)
```

```
            for i in range(len(listDatos)):

                listDatos[i] =
instante_t_regulacion(listDatos[i], t, necesidad=necesidad,

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    return listDatos

def sistema_multiderivacionfluyente(t_total, listPresa,
necesidad, g=9.8,
                                    den=998,
intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con múltiples centrales
fluyentes y
    derivación
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase
Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """
    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos +
[Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos,
listPresa= listPresa)
    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] =
instante_t_derivacionfluyente(t, datos=listDatos[i],
presa=listPresa[i], necesidad=necesidad[i],
                                                    g=g,
den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
```

```
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                listDatos[i] =
instante_t_derivacionfluyente(t=t, datos=listDatos[i],
presa=listPresa[i], necesidad=necesidad,

g=g, den=den, intervalo=intervalo)
    return listDatos


def sistema_multicentral(t_total, listPresa, necesidad,
g=9.8, den=998,
                         intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con múltiples centrales
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase
Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """

    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos +
[Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos,
listPresa= listPresa)

    if str(type(necesidad)) == "<class 'list'>":
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                if (listPresa[i].Tipo_central[0] == 'F' or
listPresa[i].Tipo_central[0] =='D'):
                    listDatos[i] =
instante_t_derivacionfluyente(t, listDatos[i],
presa=listPresa[i], necesidad=necesidad[i])

                else:
```

```
                    listDatos[i] =
instante_t_regulacion(listDatos[i], t,
necesidad=necesidad[i],

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                if listPresa[i].Tipo_central[0] == 'F' or
listPresa[i].Tipo_central[0] =='D':
                    listDatos[i] =
instante_t_derivacionfluyente(t, listDatos[i],
presa=listPresa[i], necesidad=necesidad)

                else:
                    listDatos[i] =
instante_t_regulacion(listDatos[i], t, necesidad=necesidad,

presa=listPresa[i], g=g, den=den, intervalo=intervalo)

    return listDatos


def sistema_multiple(t_total, listPresa, necesidad, g=9.8,
den=998,
                     intervalo=3600):
    """Versión  3.

    Hace la simulación de un sistema con múltiples centrales
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -listPresa: lista de objetos de la clase Presa
        -necesidad: objeto o listas de objetos de la clase
Necesidades
        -g: gravedad
        -den: densidad
        -intervalo: segundos entre cada instante de tiempo t
    SALIDA: Resultado de la simulación
        -Lista de objetos de la clase Presa_Datos
    """
    listDatos = []
    for i in range(len(listPresa)):
        listDatos = listDatos +
[Presa_Datos(presa=listPresa[i], t_total=t_total)]

    listDatos = interconexion__QentraCentral(listDatos,
listPresa= listPresa)
```

```
    listDatos = interconexion__QsaleCentral(listDatos,
listPresa= listPresa)

    if str(type(necesidad)) == "<class 'list'>":

        for t in t_total[1:]:
            listDatos =
multipleNL__QsaleCentral(t,listDatos=listDatos,
listPresa=listPresa, necesidad=necesidad[i],
                                                g=g,
den=den, intervalo=intervalo)
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)
            listDatos =
actualizar_QsaleCentral(t,listDatos,listPresa= listPresa)
            for i in range(len(listDatos)):
                if (listPresa[i].Tipo_central[0] == 'F' or
listPresa[i].Tipo_central[0] =='D'):
                    listDatos[i] =
instante_t_derivacionfluyente(t, listDatos[i],
presa=listPresa[i], necesidad=necesidad[i],
                                                      g=g,
den=den, intervalo=intervalo)

                elif (listPresa[i].Tipo_central[0] == 'B' or
listPresa[i].Tipo_central[0] =='b'):
                    listDatos[i] =
instante_t_bombeoM2(listDatos[i],t,
presa=listPresa[i],listDatos=listDatos,

necesidad=necesidad[i], g=g, den=den, intervalo=intervalo)

                else:
                    listDatos[i] =
instante_t_regulacion(listDatos[i], t,
necesidad=necesidad[i],

presa=listPresa[i],g=g, den=den, intervalo=intervalo)

    else:
        for t in t_total[1:]:
            multipleNO__QsaleCentral(t,listDatos=listDatos,
listPresa=listPresa, necesidad=necesidad,
                                     g=g, den=den,
intervalo=intervalo)
            listDatos =
actualizar_QentraCentral(t,listDatos,listPresa= listPresa)
            listDatos =
actualizar_QsaleCentral(t,listDatos,listPresa= listPresa)

            for i in range(len(listDatos)):
                if listPresa[i].Tipo_central[0] == 'F' or
listPresa[i].Tipo_central[0] =='D':
```

```
                    listDatos[i] =
instante_t_derivacionfluyente(t, listDatos[i],
presa=listPresa[i], necesidad=necesidad,
                                                          g=g,
den=den, intervalo=intervalo)

                elif (listPresa[i].Tipo_central[0] == 'B' or
listPresa[i].Tipo_central[0] =='b'):
                    listDatos[i] =
instante_t_bombeoM2(listDatos[i],t,
presa=listPresa[i],listDatos=listDatos,

necesidad=necesidad, g=g, den=den, intervalo=intervalo)

                else:
                    listDatos[i] =
instante_t_regulacion(listDatos[i], t, necesidad=necesidad,

presa=listPresa[i], g=g, den=den, intervalo=intervalo)
    return listDatos


def division_necesidad(t_total, necesidad, listCentrales,
lista_division):
    """Versión  1.

    Divide las necesidades para que haya un objeto de
necesidades por central
    ENTRADAS:
        -t_total: lista con cada índice de la simulación
desde el instante
        inicial hasta el final
        -necesidad: objeto de la clase necesidades
        -listCentrales: lista con las centrales del sistema
        -lista_division: lista 2 por número de centrales del
sistema
            - primera fila demanda de potencia
            - segunda fila demanda de caudal
    SALIDA:
        - Lista de objetos de la clase necesidades
    """
    listNecesidades = []
    for i in range(len(listCentrales)):
        pot =
necesidad.Potencia_necesaria*lista_division[0][i]
        caudal =
necesidad.Caudal_necesario*lista_division[1][i]
        necesidad_x = Necesidades(t_total,
Potencia_necesaria=pot,
                                  Caudal_necesario=caudal)
        listNecesidades = listNecesidades + [necesidad_x]
    return listNecesidades
```

```
# ------------------------------------------------------------
------[REPRESENTAR]
meses = ['Enero', 'Febrero', 'marzo', 'Abril', 'Mayo',
'Juni', 'Julio',
         'Agosto', 'Septiembre', 'Octubre', 'Noviembre',
'Diciembre']

horaMes = [0, 31*24, 59*24, 90*24, 120*24, 151*24, 181*24,
211*24, 242*24,
           272*24, 303*24, 333*24]

WIDE = 8
HEIGHT = 2
LETTER_SIZE = 8


def representar_QentraQsale(ListaDatos, t_total=t_total,
horatik=horaMes,
                            nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lisita de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('caudales salen y entran',
fontsize=2*LETTER_SIZE)
    for dato in ListaDatos:
        string1 = 'QsaleTotal '+dato.Nombre
        string2 = 'QentraTotal '+dato.Nombre
        plt.plot(t_total, dato.QsaleTotal[t_ini:t_fin],
label=string1)
        plt.plot(t_total, dato.QentraTotal[t_ini:t_fin],
label=string2)
    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)
```

```
def representar_Qgenerador(ListaDatos, t_total=t_total,
horatik=horaMes,
                           nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('Caudal del generador',
fontsize=2*LETTER_SIZE)
    for dato in ListaDatos:
        string1 = 'Qg '+dato.Nombre
        plt.plot(t_total, dato.Qg[t_ini:t_fin],
label=string1)
    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_Pgenerador(ListaDatos, t_total=t_total,
horatik=horaMes,
                           nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('Potencia', fontsize=LETTER_SIZE*2)
```

```
    for dato in ListaDatos:
        string1 = 'P '+dato.Nombre
        plt.plot(t_total, dato.Pg_ms[t_ini:t_fin],
label=string1)
    plt.ylabel('w', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_Altura(ListaDatos, t_total=t_total,
horatik=horaMes,
                       nombretik=meses, LH={},
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa la altura
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
        -LH: Diccionario de líneas horizontales para la
gráfica. La clave ha de
        ser el nombre y el valor es la abscisa donde se
encuentra la
        linea (x=oo,y=K)
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('altura', fontsize=LETTER_SIZE*2)
    for dato in ListaDatos:
        string1 = 'H '+dato.Nombre
        plt.plot(t_total, dato.H[t_ini:t_fin],
label=string1)
    for clave in LH.keys():
        plt.hlines(LH[clave], xmin=0, xmax=len(t_total),
color='red',
                   label=clave)
    plt.ylabel('m', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_Qbomba(ListaDatos, t_total=t_total,
horatik=horaMes,
```

```
                        nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa el caudal de la bomba
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('Caudal de la bomba', fontsize=2*LETTER_SIZE)
    for dato in ListaDatos:
        if sum(dato.Qb) != 0:
            string1 = 'Qb '+dato.Nombre
            plt.plot(t_total, dato.Qb[t_ini:t_fin],
label=string1)
    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_Pgenerador_Pbomba(ListaDatos,
t_total=t_total, horatik=horaMes,
                                  nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa la potencia del generador y de la bomba
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('Potencia', fontsize=LETTER_SIZE*2)
    for dato in ListaDatos:
        string1 = 'Generador '+dato.Nombre
```

```
        plt.plot(t_total, dato.Pg_ms[t_ini:t_fin],
label=string1)

        if sum(dato.Pb) != 0:
            string1 = 'Bomba '+dato.Nombre
            plt.plot(t_total, dato.Pb[t_ini:t_fin]*(-1),
label=string1)
    plt.ylabel('w', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_Pbomba(ListaDatos, t_total=t_total,
horatik=horaMes,
                       nombretik=meses,
LETTER_SIZE=LETTER_SIZE):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    t_ini = t_total[0]
    t_fin = t_total[-1] + 1
    plt.title('Potencia', fontsize=LETTER_SIZE*2)
    for dato in ListaDatos:
        if sum(dato.Pb) != 0:
            string1 = 'Bomba '+dato.Nombre
            plt.plot(t_total, dato.Pb[t_ini:t_fin]*(-1),
label=string1)
    plt.ylabel('w', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horatik, nombretik, fontsize=LETTER_SIZE)
    plt.legend(loc=1, fontsize=LETTER_SIZE)


def representar_REVERSIBLE(ListaDatos, t_total=t_total,
horatik=horaMes,
                           nombretik=meses, LH_Altura={},
LETTER_SIZE=LETTER_SIZE,
                           WIDE=WIDE, HEIGHT=HEIGHT):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
```

```
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
        -LH_Altura: Diccionario de líneas horizontales para
la gráfica de la
        altura. La clave ha de ser el nombre y el valor es
la abscisa donde se
        encuentra la linea (x=oo,y=K)
    """
    plt.figure(figsize=(WIDE, HEIGHT*3))

    plt.subplot(511)
    representar_QentraQsale(ListaDatos, t_total=t_total,
horatik=horatik,
                            nombretik=nombretik)

    plt.subplot(512)
    representar_Qgenerador(ListaDatos, t_total=t_total,
horatik=horatik,
                           nombretik=nombretik)

    plt.subplot(513)
    representar_Qbomba(ListaDatos, t_total=t_total,
horatik=horatik,
                       nombretik=nombretik)

    plt.subplot(514)
    representar_Altura(ListaDatos, t_total=t_total,
horatik=horatik,
                       nombretik=nombretik, LH=LH_Altura)

    plt.subplot(515)
    representar_Pgenerador_Pbomba(ListaDatos,
t_total=t_total, horatik=horatik,
                                  nombretik=nombretik)

    plt.tight_layout()
    plt.show()


def representar_REGULACION(ListaDatos, t_total=t_total,
horatik=horaMes,
                           nombretik=meses, LH_Altura={},
WIDE=WIDE, HEIGHT=HEIGHT):
    """Version 1.
```

```
    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
        -LH_Altura:Diccionario de líneas horizontales para
la gráfica de la
        altura. La clave ha de ser el nombre y el valor es
la abscisa donde se
        encuentra la linea (x=oo,y=K)
    """
    plt.figure(figsize=(WIDE, HEIGHT*3))

    plt.subplot(411)
    representar_QentraQsale(ListaDatos, t_total=t_total,
horatik=horatik,
                            nombretik=nombretik)

    plt.subplot(412)
    representar_Qgenerador(ListaDatos, t_total=t_total,
horatik=horatik,
                           nombretik=nombretik)

    plt.subplot(413)
    representar_Altura(ListaDatos, t_total=t_total,
horatik=horatik,
                       nombretik=nombretik, LH=LH_Altura)

    plt.subplot(414)
    representar_Pgenerador(ListaDatos, t_total=t_total,
horatik=horatik,
                           nombretik=nombretik)

    plt.tight_layout()
    plt.show()


def representar_FLUYENTE_DERIVACION(ListaDatos,
t_total=t_total,
                                    horatik=horaMes,
nombretik=meses,
                                    WIDE=WIDE,
HEIGHT=HEIGHT):
    """Version 1.

    Representa el caudal de salida y el caudal de entrada
    ENTRADA:
```

```
        -ListaDatos: Lista de objetos de la clase
Presa_Datos
        -t_total: Lista de horas a representar,
        -horatik: Lista de horas que corresponde a la
posicion que se va a
        denominar como
        -nombretik: Lista de nombres que se va a poner en
vez de la hora de la
        lista anterior
    """
    plt.figure(figsize=(WIDE, HEIGHT*3))

    plt.subplot(311)
    representar_QentraQsale(ListaDatos, t_total=t_total,
horatik=horatik,
                            nombretik=nombretik)

    plt.subplot(312)
    representar_Qgenerador(ListaDatos, t_total=t_total,
horatik=horatik,
                           nombretik=nombretik)

    plt.subplot(313)
    representar_Pgenerador(ListaDatos, t_total=t_total,
horatik=horatik,
                           nombretik=nombretik)

    plt.tight_layout()
    plt.show()
```

ANEXO 7.1 NOTEBOOK 1

1 MODELO CENTRALES INDIVIDUALES

Simulaciones de centrales individuales de derivación, fluyentes y reversibles.

Estructura del notebook:

1. Simulación de centrales fluyentes individuales
    - Simulación sin demanda
    - Simulación con demanda
2. Simulación de centrales de derivación individuales
    - Simulación sin demanda
    - Simulación con demanda
3. Simulación de centrales de regulación individuales
    - Simulación sin demanda
    - Simulación sin demanda con lluvia
    - Simulación sin demanda con caudales polinomiales
    - Simulación con demanda de Agua
    - Simulación con demanda de potencia

1. Importaciones e inicializaciones

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib widget
```

Módulos para el modelo de las distintas entradas y demandas:

In [2]:

```
import lib.herramientas_modelado as hm
```

Módulos para las simulaciones:

In [3]:

```
import lib.herramientas_presas as hp
```

2. Datos para las simulaciones:

In [4]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia

tini=0
tfin=horas_año
t_total= np.arange(tfin-tini)
```

3. Datos para la representación:

In [5]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]

semana =
['Lunes','Martes','Miercoles','Jueves','Viernes','Savado','Domingo
']
horaSemana = [0,24,24*2,24*3,24*4,24*5,24*6]
```

Para graficar más fácilmente haremos unos ajustes:

In [6]:

```
n = 0
WIDE = 8
HEIGHT = 2
LETTER_SIZE = 8
```

4. Creamos el objeto de la clase necesidades vacío:

In [7]:

```
Necesidades0 = hp.Necesidades(t_total,Nombre='Necesidades0')
```

1. FLUYENTE

1.1 SIMULACIÓN SIMPLE

1. Inicializamos las centrales

In [8]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central111',Presa_salida='cen
tral12',P_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central112',Presa_salida='',P
_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [9]:

```
turbina11 = hp.Turbina(Nombre='tuebina11',r=0.85)
```

In [10]:

```
presa1.turbina=turbina11
presa2.turbina=turbina11
```

3. Creamos los caudales

In [11]:

```
# Caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# Caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)/2

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [12]:

```
# Caudal Támega
Q_med1 = 9.23
Q_A1=2

# Caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83


Q_rio3=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio4=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

4. Calcular

In [13]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

Datos1 = hp.sistema_fluyente(t_total, presa1, Necesidades0)
Datos2 = hp.sistema_fluyente(t_total, presa2, Necesidades0)
ListaDatos1=[Datos1,Datos2]
```

In [14]:

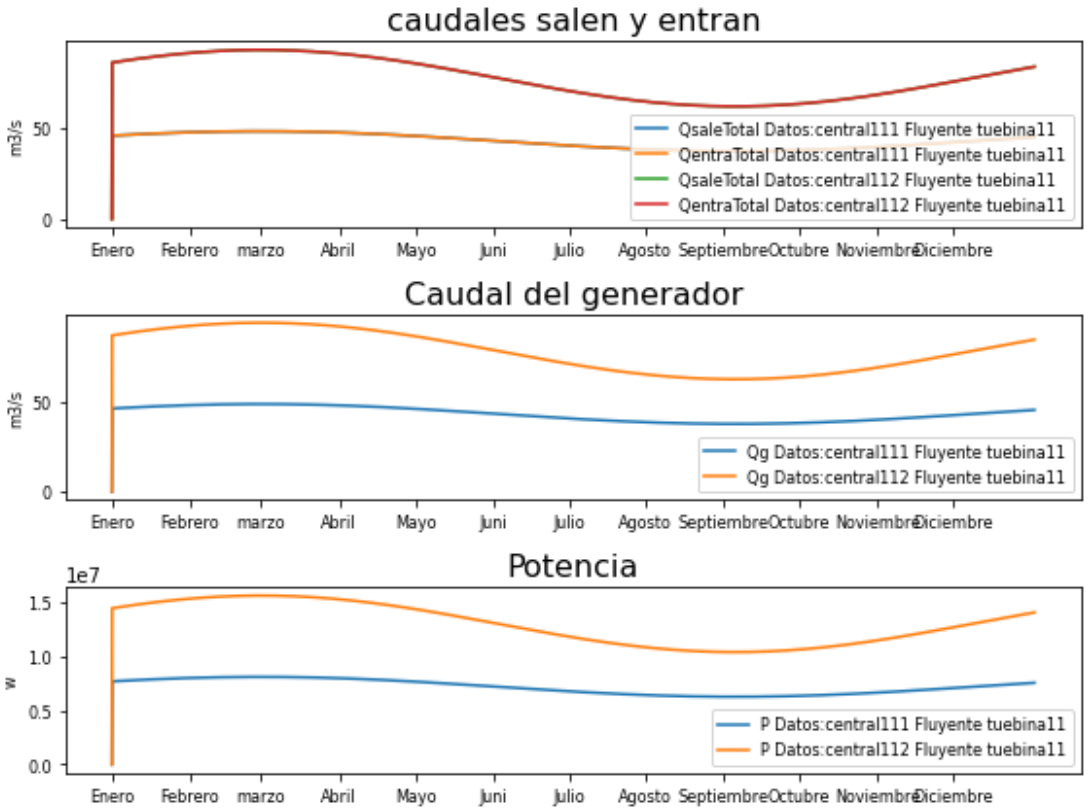
```
presa1.indicar_Qentra_m3s(Q_rio3)
```

```
presa2.indicar_Qentra_m3s(Q_rio4)

Datos1 = hp.sistema_fluyente(t_total, presa1, Necesidades0)
Datos2 = hp.sistema_fluyente(t_total, presa2, Necesidades0)
ListaDatos2=[Datos1,Datos2]
```

Gráfica de ríos grandes simple

In [15]:

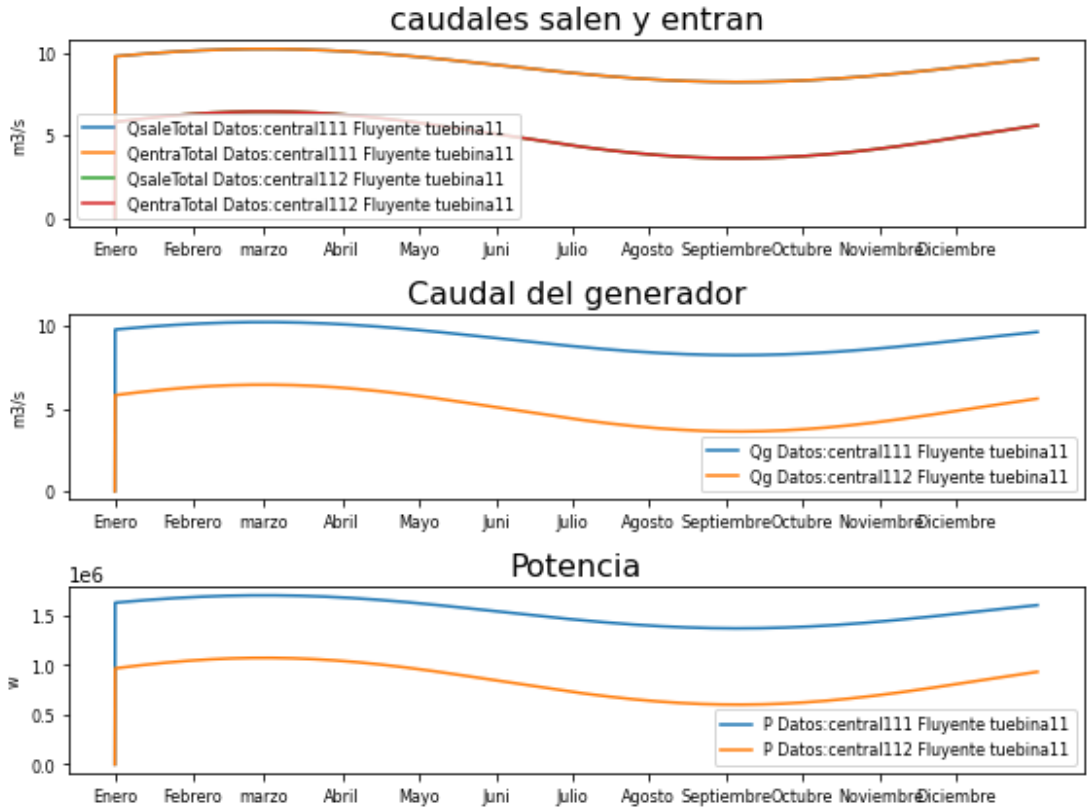
```
hp.representar_FLUYENTE_DERIVACION(ListaDatos1)
```



Gráfica de ríos pequeños simple

In [16]:

```
hp.representar_FLUYENTE_DERIVACION(ListaDatos2)
```



1.2 DEMANDA DE AGUA

1. Inicializamos las centrales

In [17]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central121',Presa_salida='',P
_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central122',Presa_salida='',P
_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [18]:

```
turbina12 = hp.Turbina(Nombre='tuebina12',r=0.85)
```

In [19]:

```
presa1.turbina=turbina12
presa2.turbina=turbina12
```

3. Creamos los caudales

In [20]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)/2

#mit=0.3

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [21]:

```
# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83


Q_rio2=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio3=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

5. Creamos las necesidades

Población 2021 Instituto Nacional de Estadística:

In [22]:

```
Poblacion_Bercero = 191
Población_Toro = 8532
población_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en España por persona

In [23]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [24]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [25]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [26]:

```
necesidad121=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad122=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad123=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
lista_Necesidades= [necesidad121,necesidad122,necesidad123]
```

5. Calcular

In [27]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

Datos1 = hp.sistema_fluyente(t_total, presa1, necesidad121)
Datos2 = hp.sistema_fluyente(t_total, presa1, necesidad122)
Datos3 = hp.sistema_fluyente(t_total, presa1, necesidad123)

ListaDatos1=[Datos1,Datos2,Datos3]
```

In [28]:

```
presa1.indicar_Qentra_m3s(Q_rio3)
presa2.indicar_Qentra_m3s(Q_rio4)
```

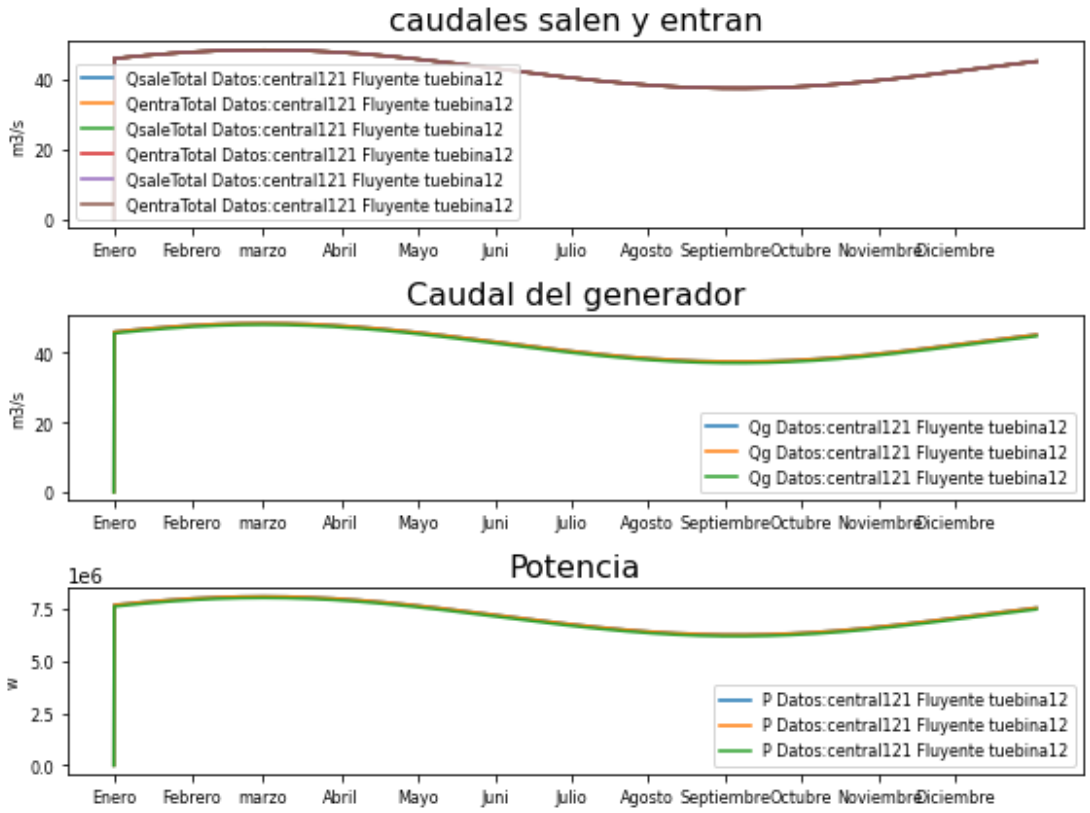
```
Datos1 = hp.sistema_fluyente(t_total, presa1, necesidad121)
Datos2 = hp.sistema_fluyente(t_total, presa1, necesidad122)
Datos3 = hp.sistema_fluyente(t_total, presa1, necesidad123)

ListaDatos2=[Datos1,Datos2,Datos3]
```

Mostrar

Gráfica de ríos grandes simple

In [29]:

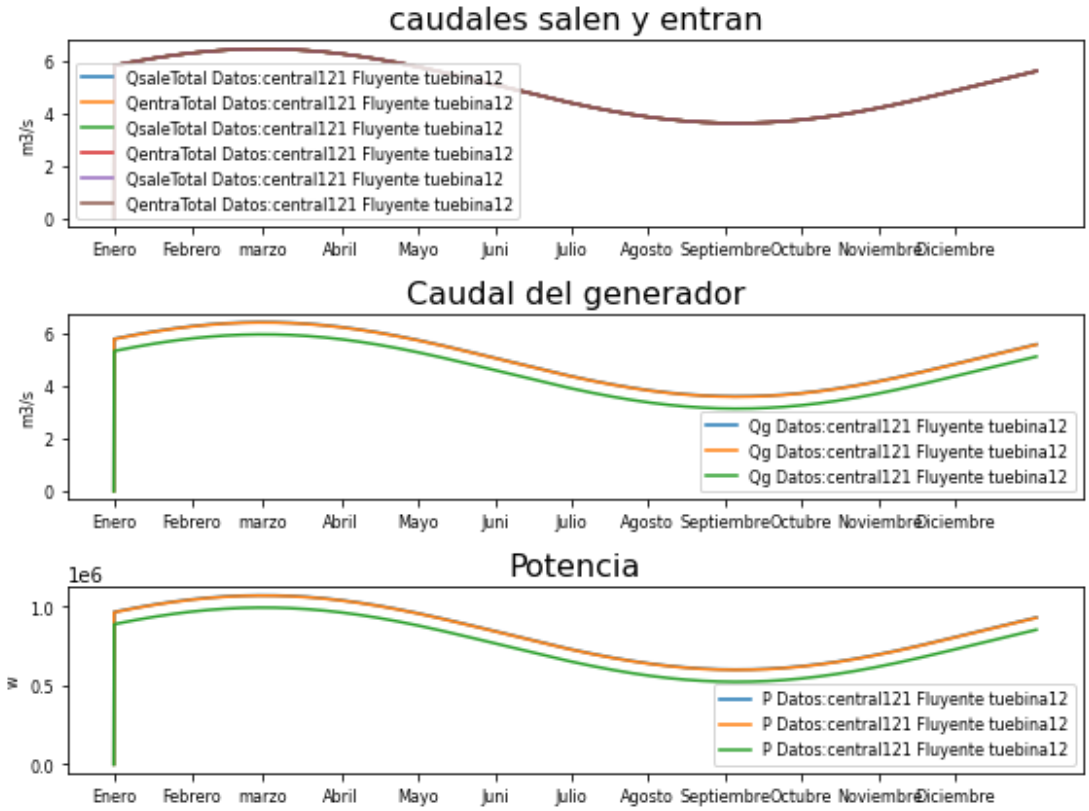
```
hp.representar_FLUYENTE_DERIVACION(ListaDatos1)
```



Gráfica de ríos pequeños simple

In [30]:

```
hp.representar_FLUYENTE_DERIVACION(ListaDatos2)
```



2. DERIVACIÓN

2.1 SIMULACIÓN SIMPLE

1. Inicializamos las centrales

In [31]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central211',Presa_salida='cen
tral12',P_Qg=0.5,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                    Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central212',Presa_salida='',P
_Qg=0.5,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                    Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [32]:

```
turbina21 = hp.Turbina(Nombre='turbina21',r=0.85)
```

In [33]:

```
presa1.turbina=turbina21
presa2.turbina=turbina21
```

3. Creamos los caudales

In [34]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)/2

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [35]:

```
# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83


Q_rio2=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio3=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [36]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
```

4. Calcular

In [37]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

datos1 = hp.sistema_derivacionfluyente(t_total, presa1,
Necesidades0)
```

```
datos2 = hp.sistema_derivacionfluyente(t_total, presa2,
Necesidades0)

ListaDatos1=[Datos1,Datos2]
```

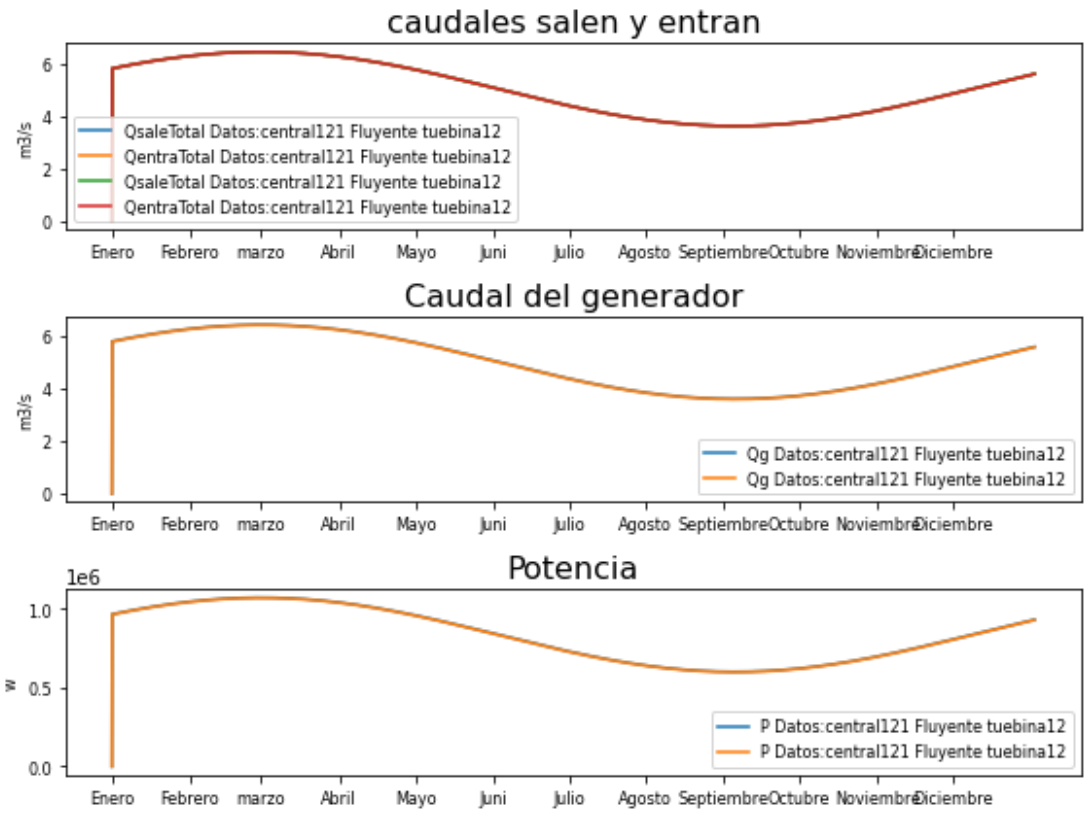
In [38]:

```
presa1.indicar_Qentra_m3s(Q_rio3)
presa2.indicar_Qentra_m3s(Q_rio4)

datos1 = hp.sistema_derivacionfluyente(t_total, presa1,
Necesidades0)
datos2 = hp.sistema_derivacionfluyente(t_total, presa2,
Necesidades0)

ListaDatos2=[Datos1,Datos2]
```

Gráfica de ríos grandes simple

In [39]:

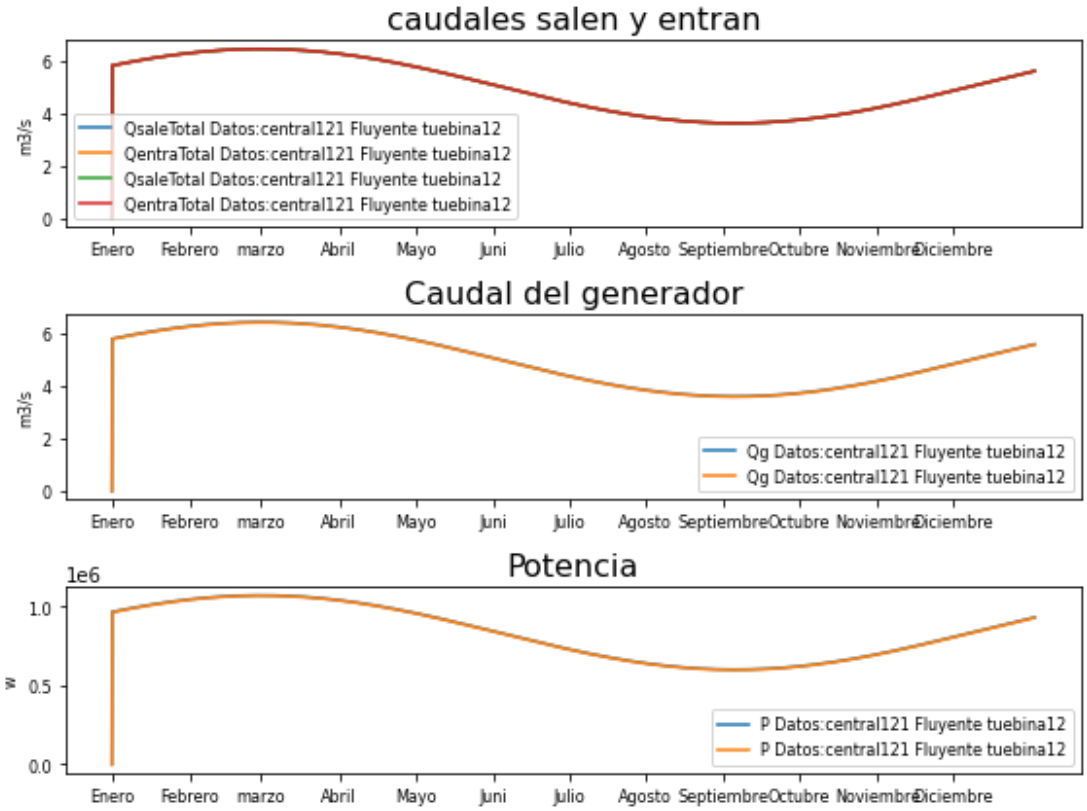
```
hp.representar_FLUYENTE_DERIVACION(ListaDatos1)
```



Gráfica de ríos pequeños simple

In [40]:

```
hp.representar_FLUYENTE_DERIVACION(ListaDatos2)
```



2.2 DEMANDA DE AGUA

1. Inicializamos las centrales

In [41]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central221',Presa_salida='cen
tral22',P_Qg=0.5,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                    Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central222',Presa_salida='',P
_Qg=0.5,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                    Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [42]:

```
turbina22 = hp.Turbina(Nombre='tuebina22',r=0.85)
```

In [43]:

```
presa1.turbina=turbina22
presa2.turbina=turbina22
```

3. Creamos los caudales

In [44]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)/2

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [45]:

```
# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83


Q_rio2=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio3=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

5. Creamos las necesidades

población 2021 Instituto Nacional de Estadística:

In [46]:

```
Poblacion_Bercero = 191
Poblacion_Toro = 8532
Poblacion_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en España por persona

In [47]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [48]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [49]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [50]:

```
necesidad221=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad222=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad223=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
lista_Necesidades= [necesidad221,necesidad222,necesidad223]
```

5. Calcular

In [51]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

Datos1 = hp.sistema_fluyente(t_total, presa1, necesidad221)
Datos2 = hp.sistema_fluyente(t_total, presa1, necesidad222)
Datos3 = hp.sistema_fluyente(t_total, presa1, necesidad223)

ListaDatos1=[Datos1,Datos2,Datos3]
```

In [52]:

```
presa1.indicar_Qentra_m3s(Q_rio3)
presa2.indicar_Qentra_m3s(Q_rio4)

Datos1 = hp.sistema_fluyente(t_total, presa1, necesidad221)
```

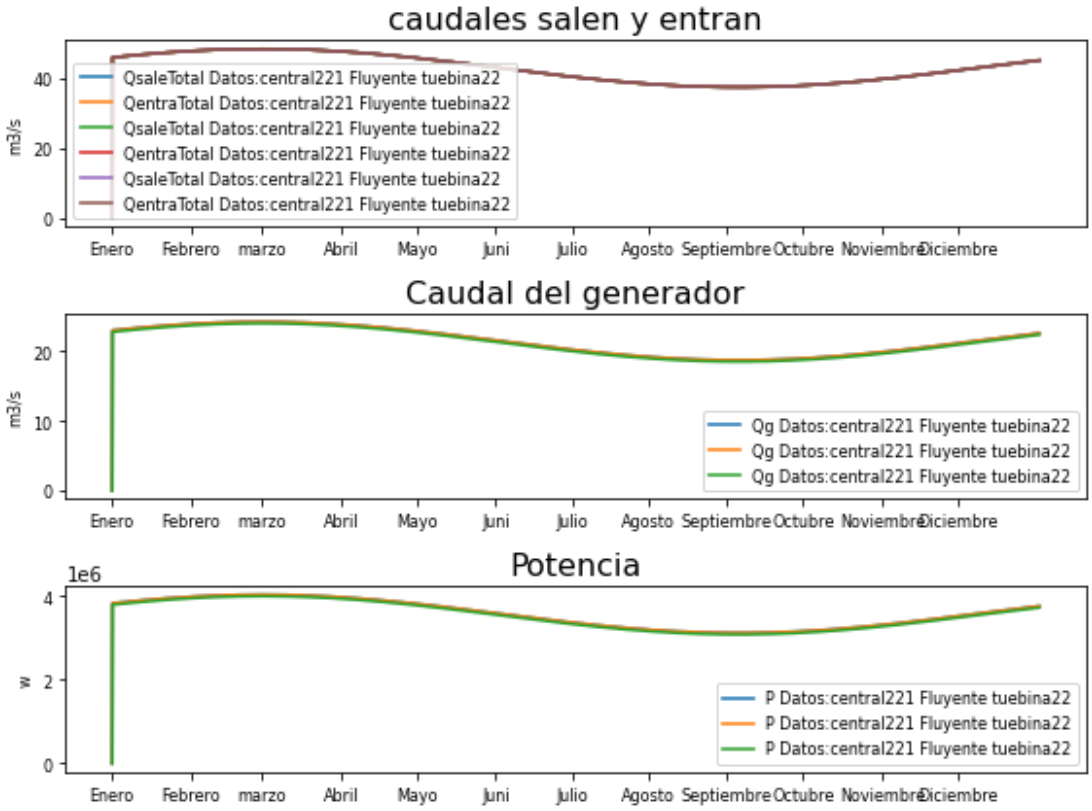
```
Datos2 = hp.sistema_fluyente(t_total, presa1, necesidad222)
Datos3 = hp.sistema_fluyente(t_total, presa1, necesidad223)

ListaDatos2=[Datos1,Datos2,Datos3]
```

Mostrar

Gráfica de ríos grandes simple

In [53]:

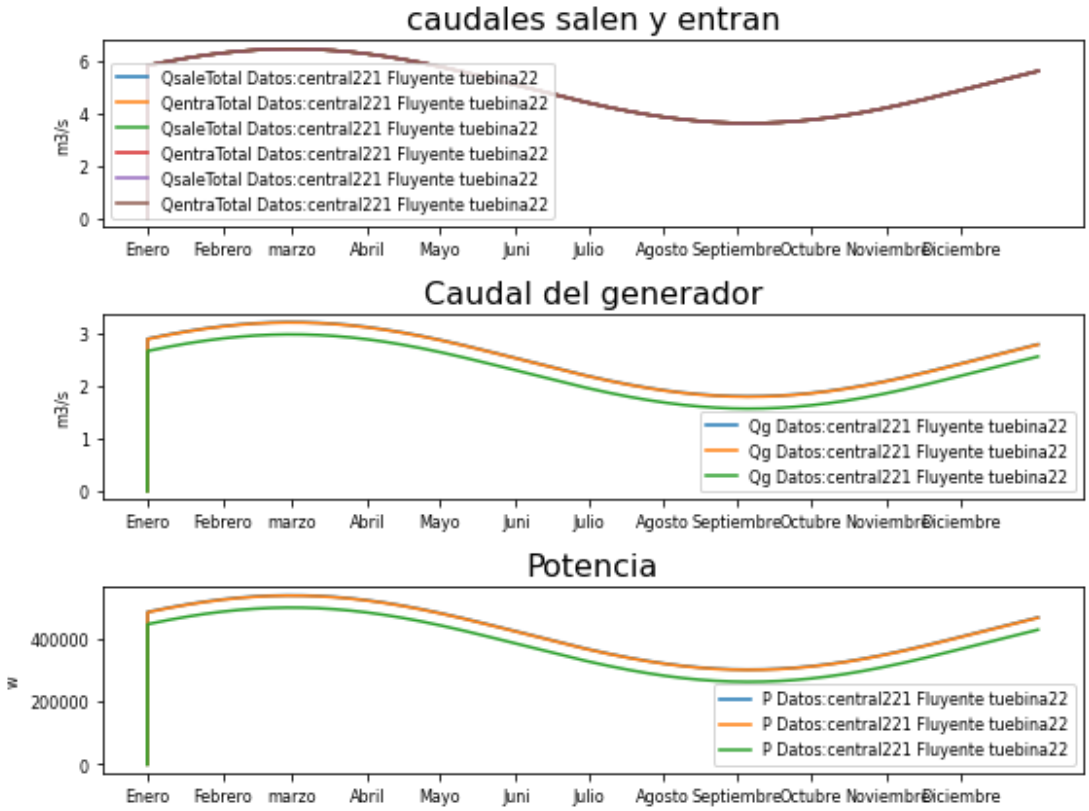
```
hp.representar_FLUYENTE_DERIVACION(ListaDatos1)
```



Gráfica de ríos pequeños simple

In [54]:

```
hp.representar_FLUYENTE_DERIVACION(ListaDatos2)
```



3. REGULACIÓN

3.1 SIMULACIÓN SIMPLE

1. Inicializamos las centrales

In [55]:

```
presa1 = hp.Presa(t_total,Q_entrada=0,Nombre='central311',P_Qg=1,

base=10000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=30,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central312',Presa_salida='',P
_Qg=1,

base=10000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=30,Area=20))
```

2. Inicializamos la turbina

In [56]:

```
def funcion_turbina_311(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):

    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.4:
        self.Qg[t] = 35
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 45
    else:
        self.Qg[t] = 50
```

In [57]:

```
turbina311 =
hp.Turbina(Nombre='turbina311',r=0.85,Qg_funcion_propia=funcion_tu
rbina_311)
```

In [58]:

```
def funcion_turbina_312(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.5:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.6:
        self.Qg[t] = 60
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 70
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 90
    else:
        self.Qg[t] = 100
```

In [59]:

```
turbina312 =
hp.Turbina(Nombre='turbina313',r=0.85,Qg_funcion_propia=funcion_tu
rbina_312)
```

In [60]:

```
def funcion_turbina_313(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [61]:

```
turbina313 =
hp.Turbina(Nombre='turbina313',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_313)
```

In [62]:

```
def funcion_turbina_314(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 5
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7
```

In [63]:

```
turbina314 =
hp.Turbina(Nombre='turbina314',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_314)
```

3. Creamos los caudales

In [64]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
```

```
Q_A2 =(109.18-46.58)/2

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [65]:

```
# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83


Q_rio3=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio4=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

4. Calcular

In [66]:

```
presa1.turbina=turbina311
presa2.turbina=turbina312

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)

ListaDatos311=[datos1,datos2]
```

In [67]:

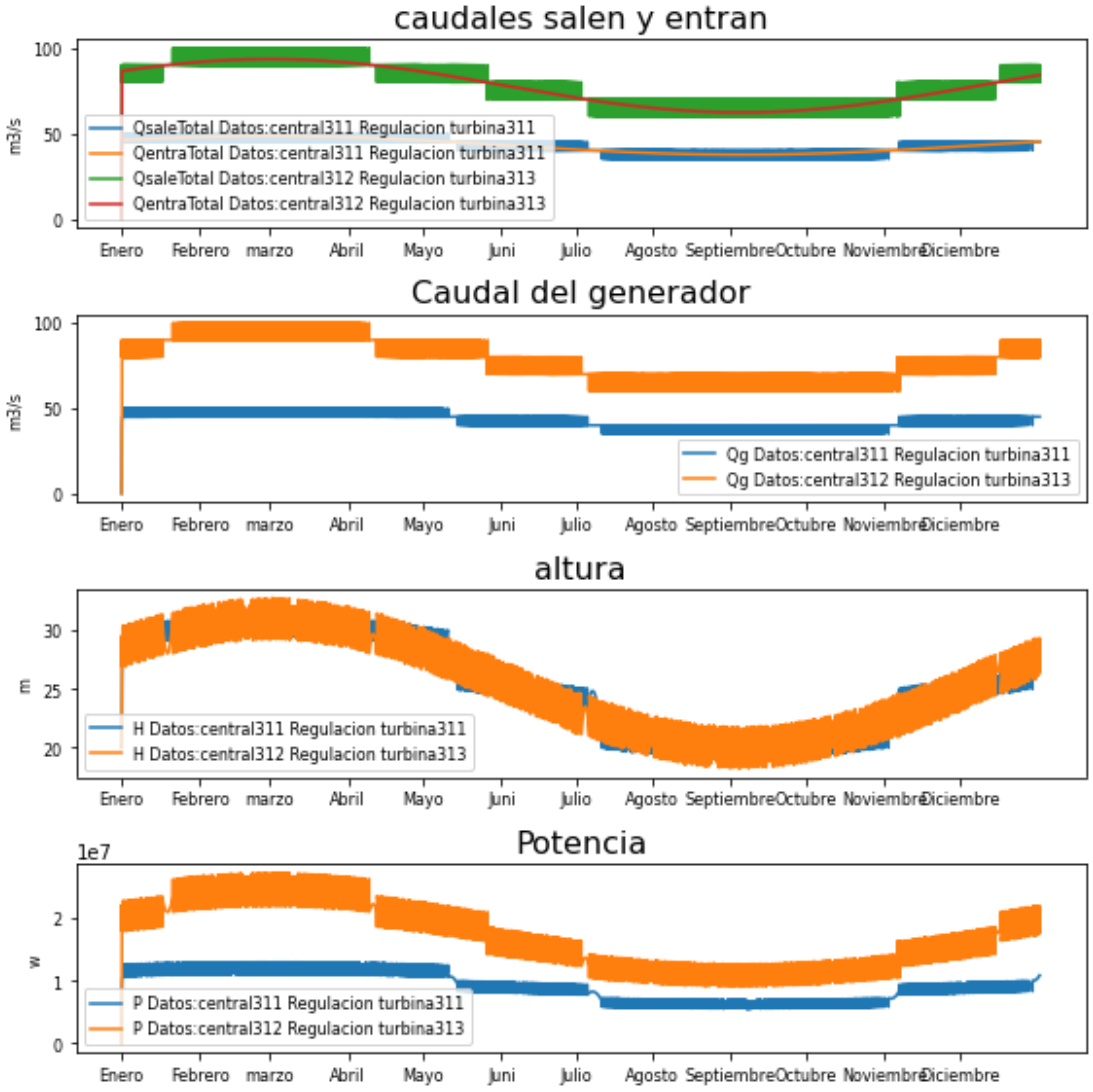
```
presa1.turbina=turbina313
presa2.turbina=turbina314

presa1.indicar_Qentra_m3s(Q_rio3)
presa2.indicar_Qentra_m3s(Q_rio4)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
```

```
ListaDatos312=[datos1,datos2]
```

Gráfica de ríos grandes simple

In [68]:

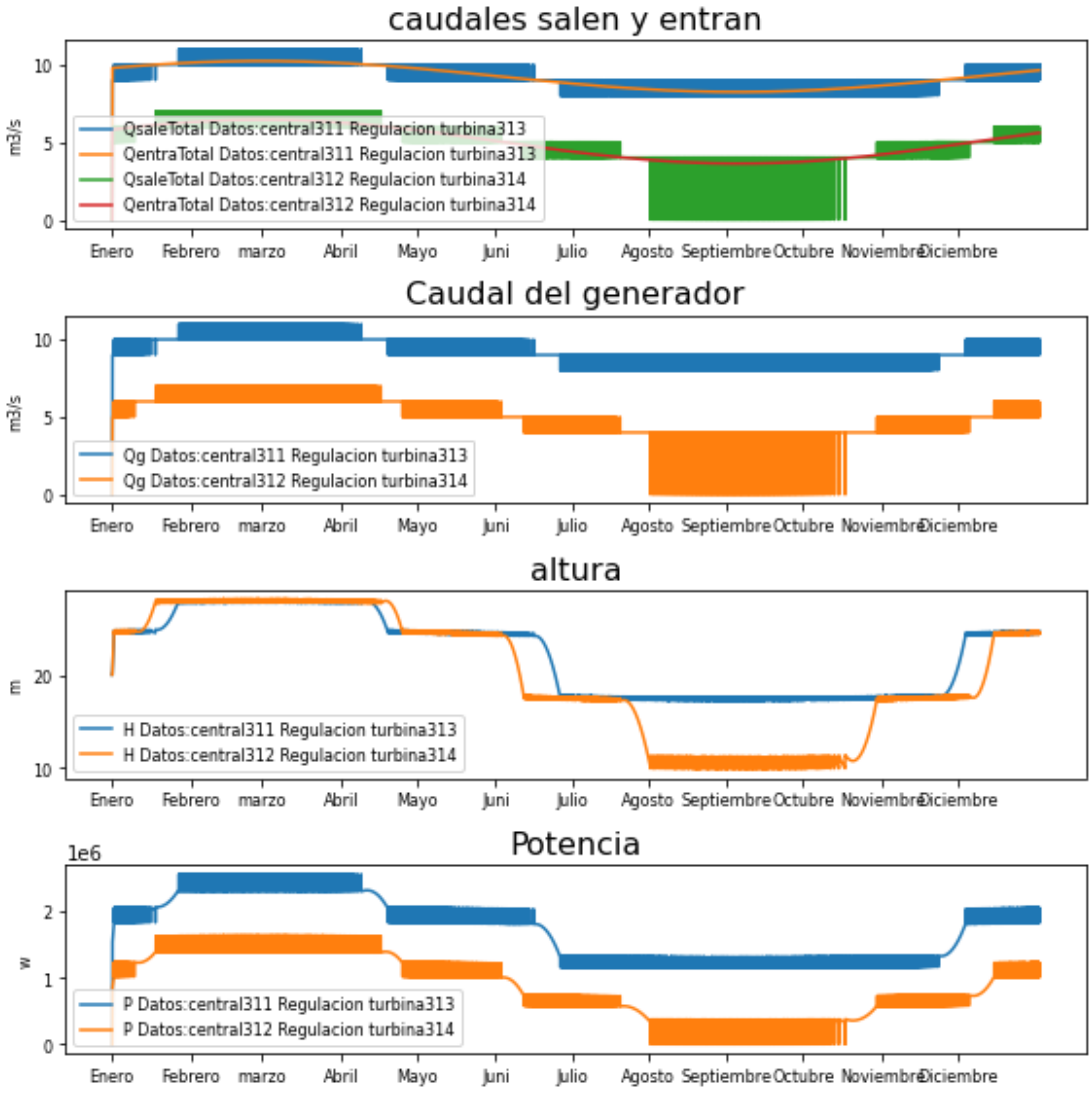
```
hp.representar_REGULACION(ListaDatos311)
```



Gráfica de ríos pequeños simple

In [69]:

```
hp.representar_REGULACION(ListaDatos312)
```



3.2 SIMULACIÓN LLUVIA

1. Inicializamos las centrales

In [70]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central321',Presa_salida='',P
_Qg=1,H_min=15,

base=10000,base_absorcion=50000,Hi=20,H_max=35,Tipo_central='Regul
ación',H_des=40,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central322',Presa_salida='',P
_Qg=1,H_min=15,

base=10000,base_absorcion=50000,Hi=20,H_max=35,Tipo_central='Regul
ación',H_des=40,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
presa3 =
hp.Presa(t_total,Q_entrada=0,Nombre='central323',Presa_salida='',P
_Qg=1,H_min=15,

base=10000,base_absorcion=50000,Hi=20,H_max=35,Tipo_central='Regul
ación',H_des=40,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
```

2. Inicializamos las turbinas

In [71]:

```
presa1.turbina=turbina311
presa2.turbina=turbina311
presa3.turbina=turbina311
```

3. Definición de las lluvias

In [72]:

```
Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Valladolid')
Q_lluvia1 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,

area_abastecida=presa1.base,#presa1.base_absorcion,

unidades_preci='mm',unidades_area='m2')

Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Zamora')
Q_lluvia2 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,
                                   area_abastecida=presa2.base,

unidades_preci='mm',unidades_area='m2')
```

In [73]:

```
Q_precipitaciones43=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Malaga')
```

```
Q_lluvia3 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones43 ,
                                   area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
Q_precipitaciones44=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Segovia')
Q_lluvia4 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones44 ,
                                   area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
Q_precipitaciones45=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'San Sebastian')
Q_lluvia5 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones45 ,
                                   area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

4. Calcular

In [74]:

```
presa1.indicar_Qentra_m3s(Q_lluvia1)
presa2.indicar_Qentra_m3s(Q_lluvia2)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)

ListaDatos1=[datos1,datos2]
```

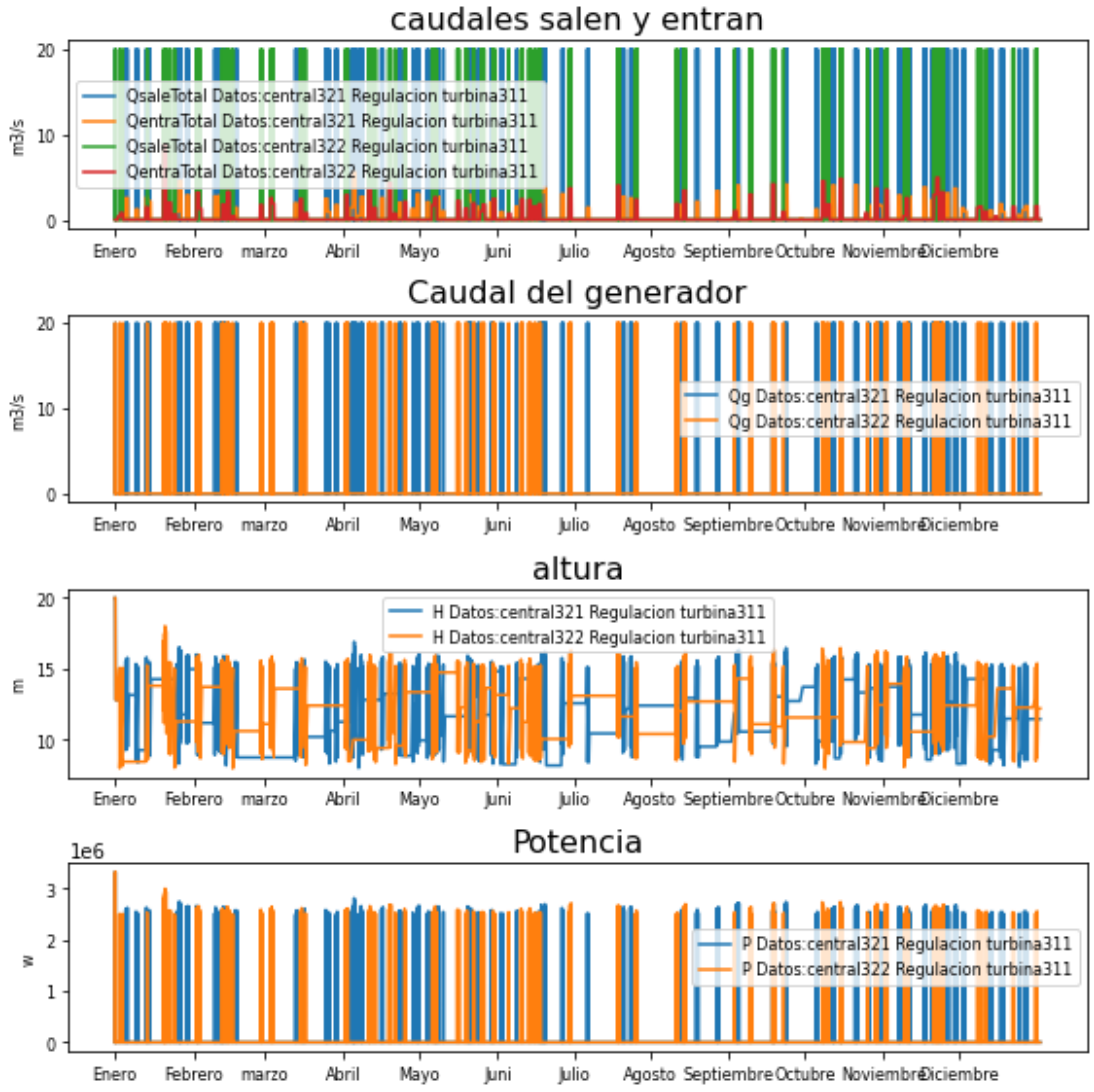
In [75]:

```
presa1.indicar_Qentra_m3s(Q_lluvia3)
presa2.indicar_Qentra_m3s(Q_lluvia4)
presa3.indicar_Qentra_m3s(Q_lluvia5)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
datos3 = hp.sistema_regulacion(t_total, presa=presa3,
necesidad=Necesidades0)

ListaDatos2=[datos1,datos2,datos3]
```

Gráfica para Valladolid y Zamora sin río

In [76]:

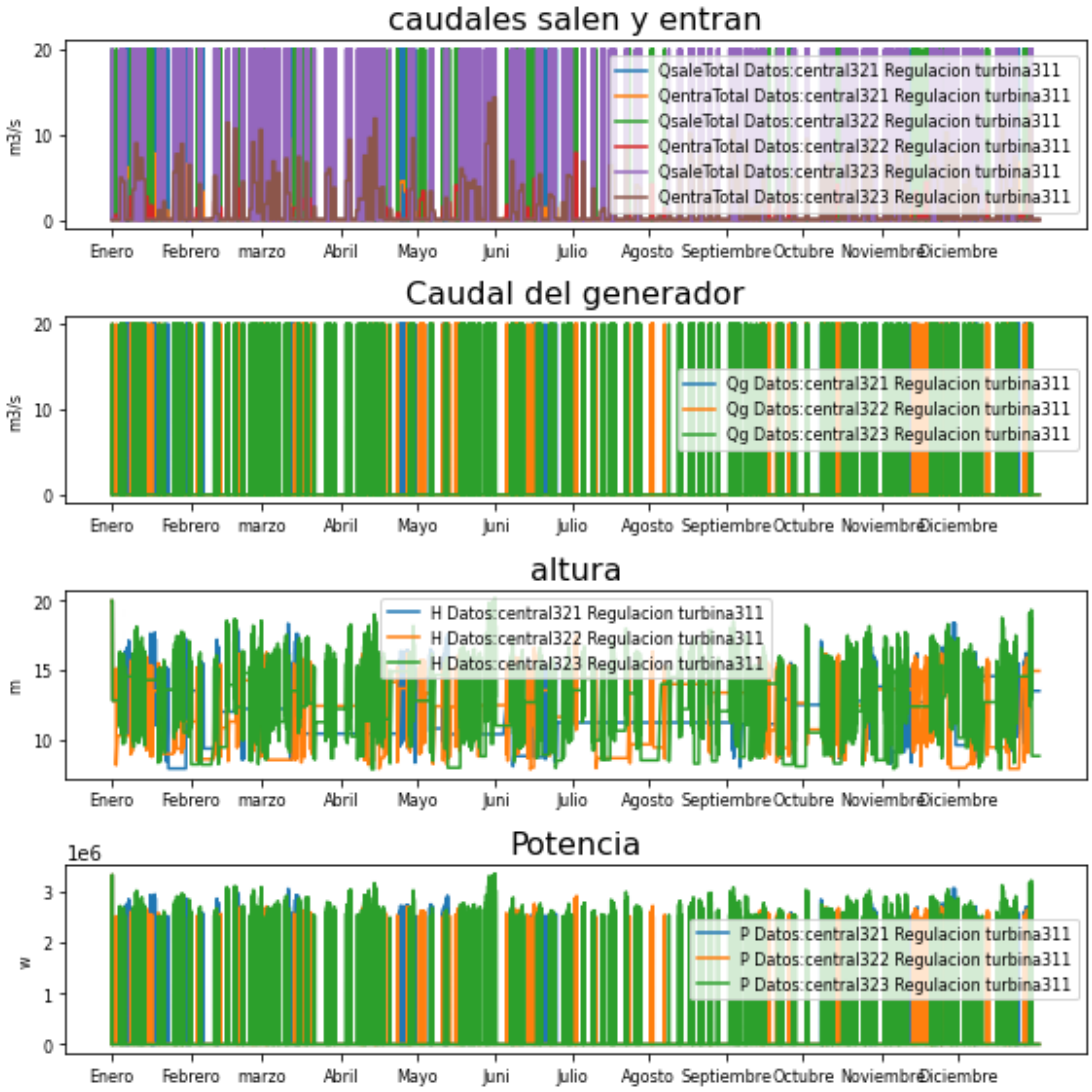
```
hp.representar_REGULACION(ListaDatos1)
```



Gráfica para Málaga, Segovia y San Sebastián sin río

In [77]:

```
hp.representar_REGULACION(ListaDatos2)
```



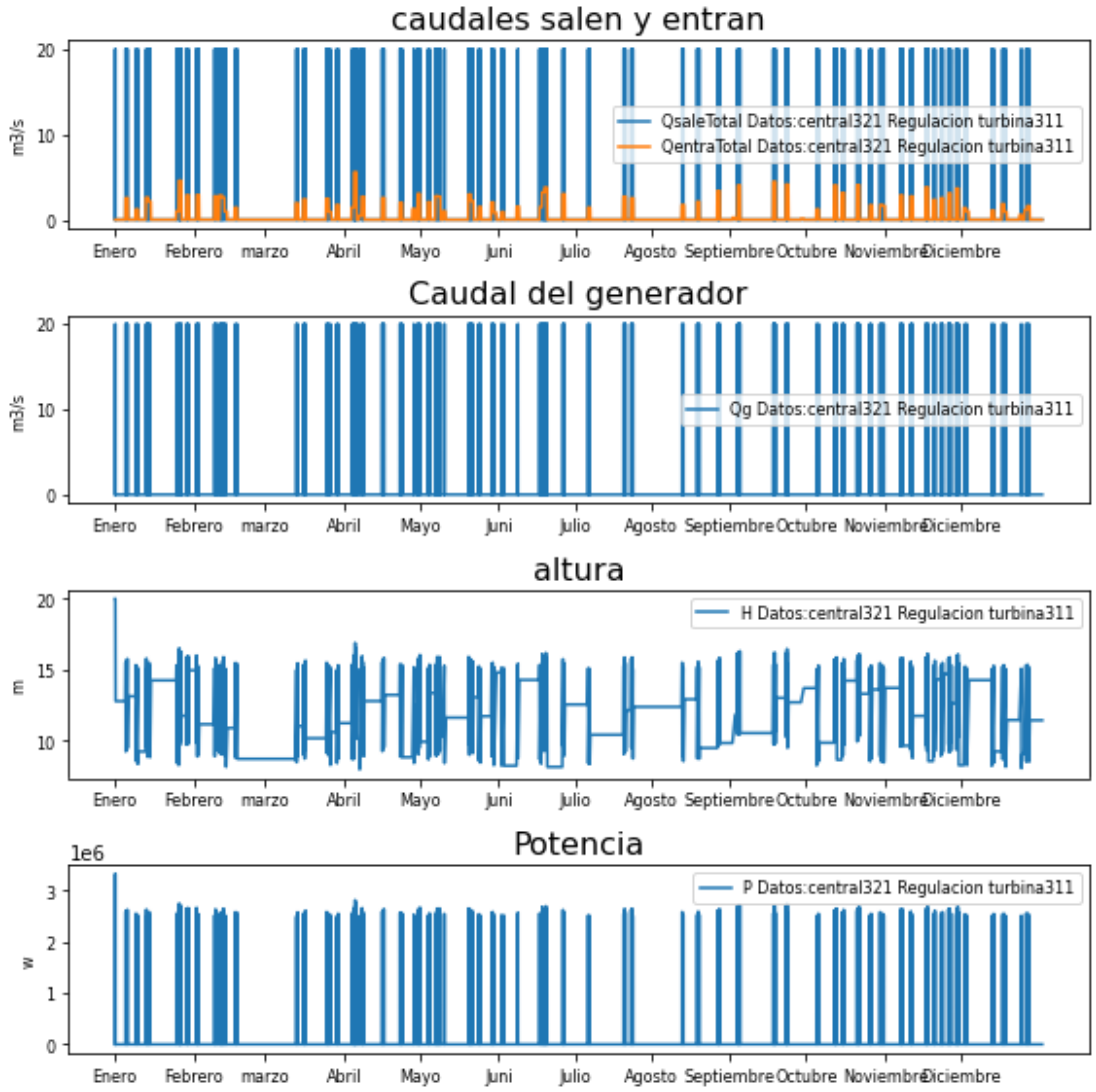
Gráfica precipitaciones Valladolid

In [78]:

```
ListaDatos32 = [ListaDatos1[0]]
```

In [79]:

```
hp.representar_REGULACION([ListaDatos1[0]])
```



Lluvia

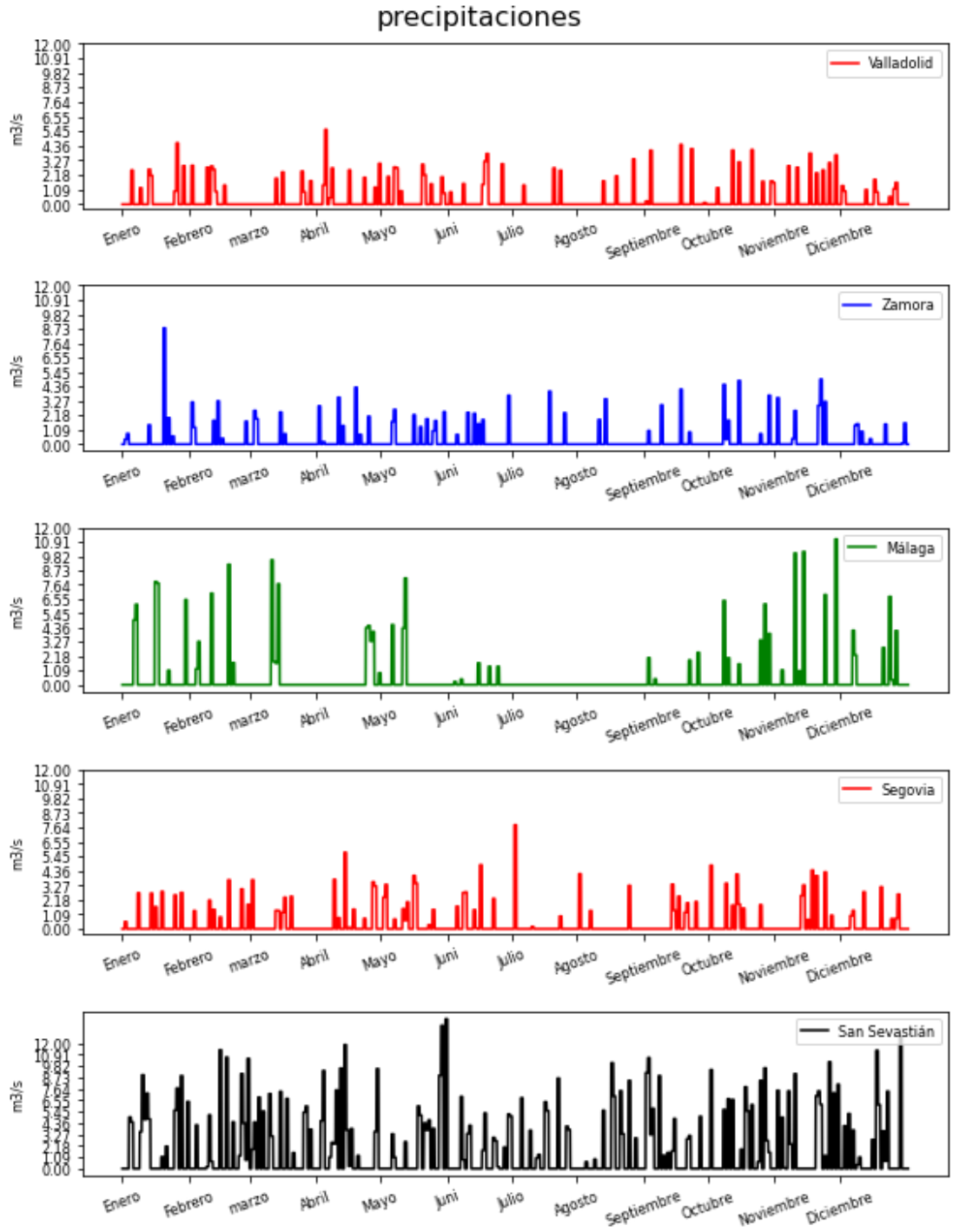
In [80]:

```
Nombres_Provincia = ['Valladolid','Zamora','Málaga','Segovia','San
Sevastián']
colores = ['r','b','g','r','k']
ListaDatos32 = ListaDatos1 + ListaDatos2
```

In [81]:

```
filas =  round(len(Nombres_Provincia))+1
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('precipitaciones',fontsize=LETTER_SIZE*2)
for i in range(len(Nombres_Provincia)):
    plt.subplot(filas,1,i+1)

    plt.plot(t_total,ListaDatos32[i].QentraTotal,colores[i],label
= Nombres_Provincia[i] )
```

```
    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(np.linspace(0,12,12),fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



3.3 SIMULACIÓN CAUDAL NORMAL

1. Inicializamos las centrales

In [82]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central331',Presa_salida='cen
tral332',P_Qg=1,

base=10000,base_absorcion=10000,Hi=20,H_max=25,Tipo_central='Regul
ación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Hentrada=30,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central332',Presa_salida='',P
_Qg=1,

base=10000,base_absorcion=10000,Hi=20,H_max=25,Tipo_central='Regul
ación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Hentrada=30,Area=20))
```

2. Inicializamos la turbina

In [83]:

```
def funcion_turbina_331(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 32
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 54
```

In [84]:

```
def funcion_turbina_332(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 46
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] =70
    else:
        self.Qg[t] =109
```

In [85]:

```
turbina331 =
hp.Turbina(Nombre='turbina331',r=0.85,Qg_funcion_propia=funcion_tu
rbina_331)
turbina332 =
hp.Turbina(Nombre='turbina332',r=0.85,Qg_funcion_propia=funcion_tu
rbina_332)
```

3. Creamos los caudales

In [86]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = (54+32)/2
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)/2

Q_rioS1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(
31+28)*24)
Q_rioS2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(
31+28)*24)
```

4. Calcular

In [87]:

```
presa1.turbina=turbina311
presa2.turbina=turbina312

presa1.indicar_Qentra_m3s(Q_rioS1)
presa2.indicar_Qentra_m3s(Q_rioS2)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)

ListaDatos33=[datos1,datos2]
ListaPresa33=[presa1,presa2]
```

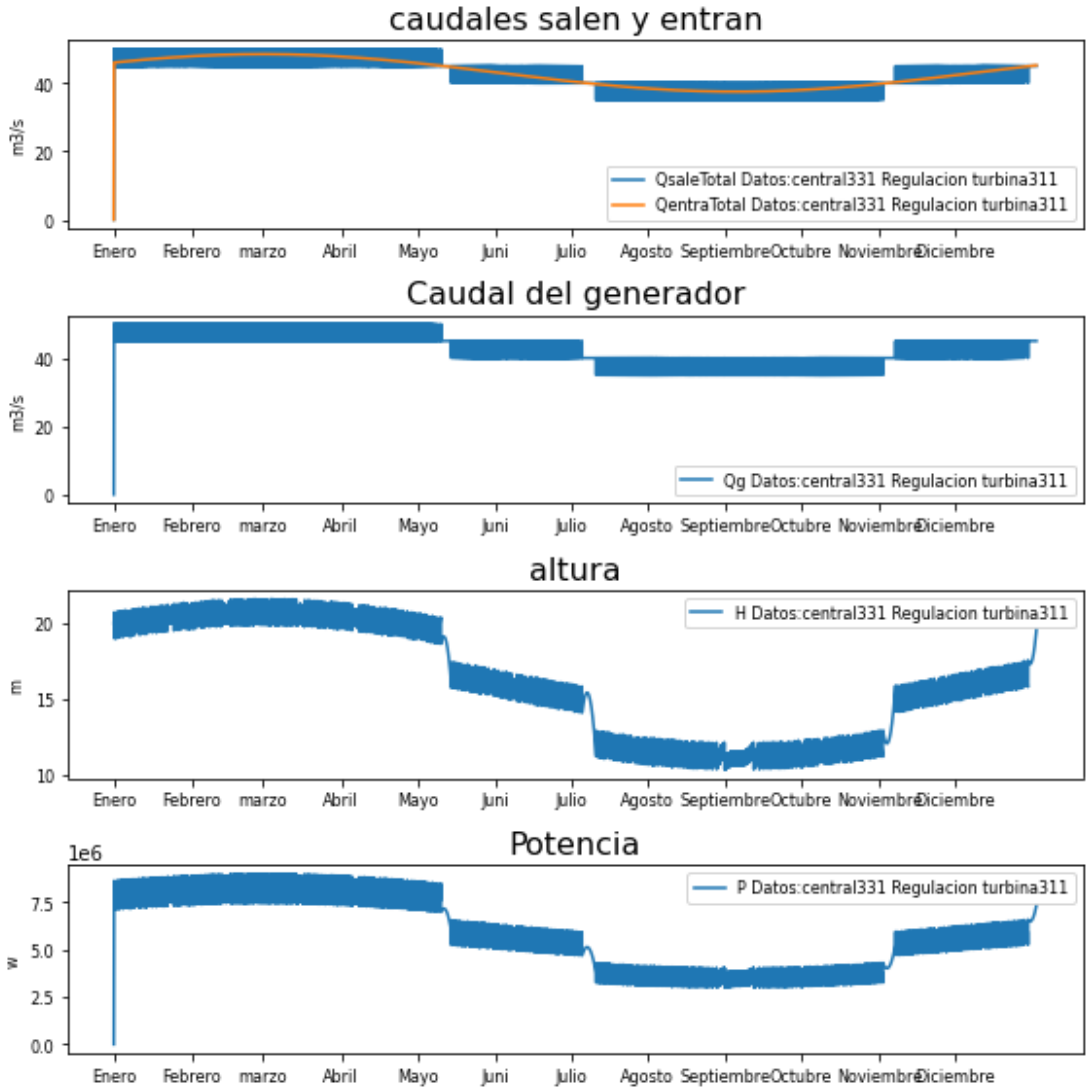
In [88]:

```
ListaDatos33=[datos1]
ListaPresa33=[presa1]
```

Gráfica de ríos grandes simple

In [89]:

```
hp.representar_REGULACION(ListaDatos33)
```



3.4 SIMULACIÓN CAUDAL POLINOMIO

1. Inicializamos las centrales

In [90]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central341',Presa_salida='cen
tral332',P_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central342',Presa_salida='',P
_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [91]:

```
def funcion_turbina_341(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 60

def funcion_turbina_342(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [92]:

```
turbina341 =
hp.Turbina(Nombre='turbina341',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_341)
turbina342 =
hp.Turbina(Nombre='turbina342',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_342)
```

3. Creamos los caudales

In [93]:

```
lista_meses=[0,1,2,6,10]
hora1=hm.lista_mes_hora(lista_meses)
lista_meses=[0,1,2,6,10]
hora2=hm.lista_mes_hora(lista_meses)
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [94]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

#mit=0.3


Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
Q_rioP2 = hm.poly_odd(t_total,hora,caudal2,horas_año)
```

4. Calcular

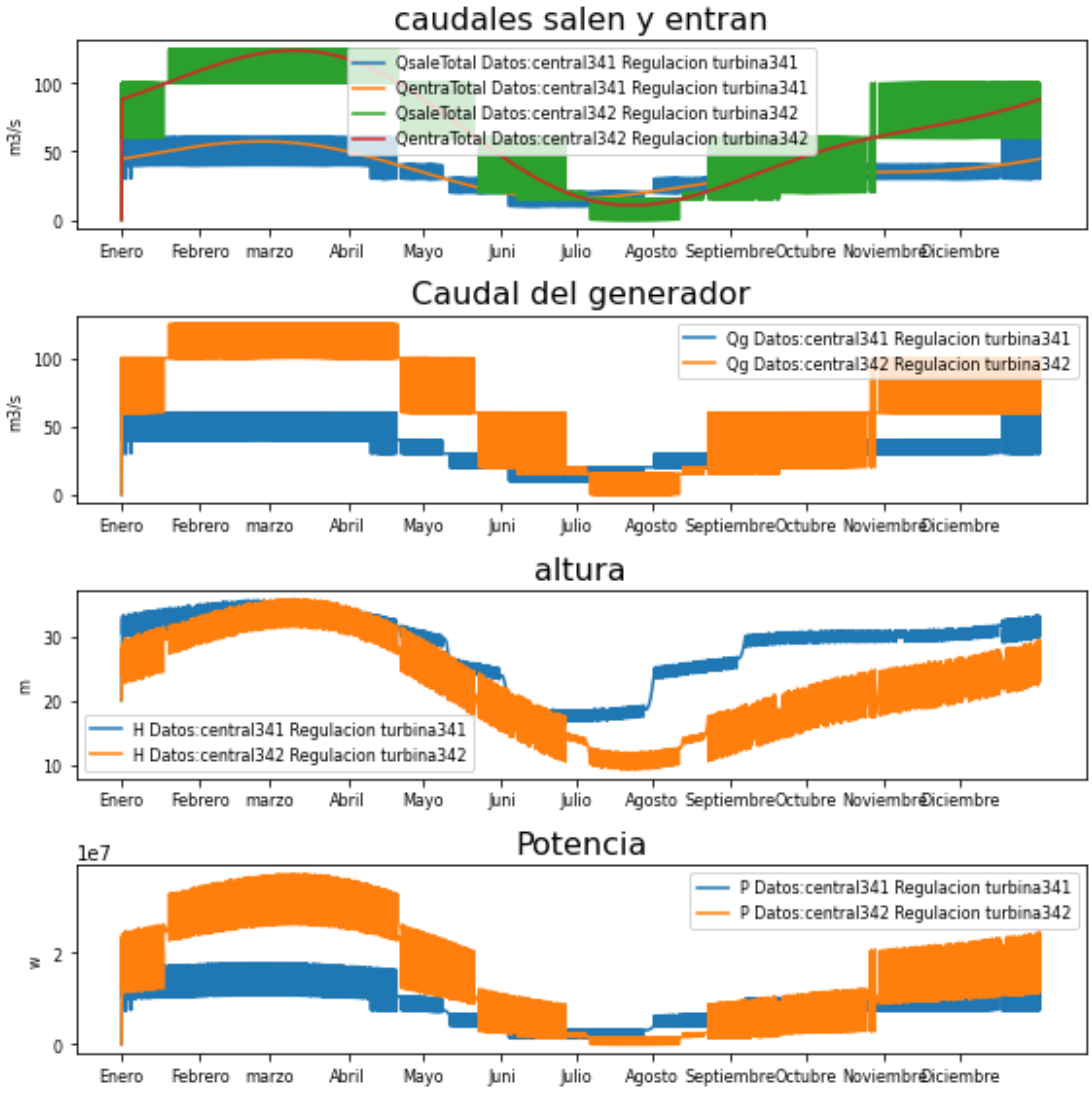
In [95]:

```
presa1.turbina=turbina341
presa2.turbina=turbina342

presa1.indicar_Qentra_m3s(Q_rioP1)
presa2.indicar_Qentra_m3s(Q_rioP2)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
ListaDatos34=[datos1,datos2]
ListaPresa34=[presa1,presa2]
```

Gráfica de ríos grandes polinomial

In [96]:

```
hp.representar_REGULACION(ListaDatos34)
```

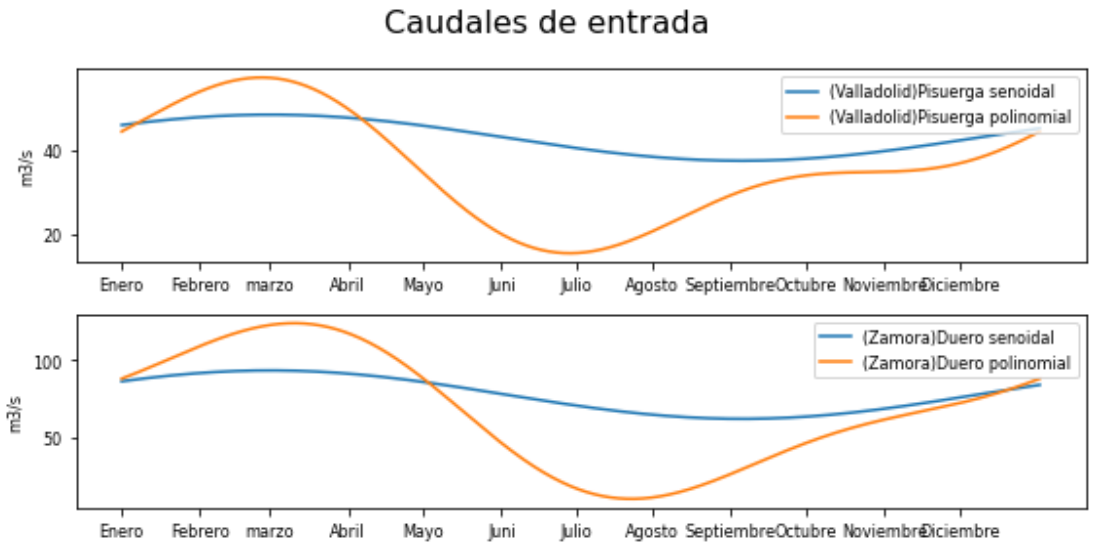


Comparación de Ríos

In [97]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.suptitle('Caudales de entrada',fontsize=2*LETTER_SIZE)
plt.subplot(211)

plt.plot(t_total,Q_rioS1,label = '(Valladolid)Pisuerga senoidal' )
plt.plot(t_total,Q_rioP1,label = '(Valladolid)Pisuerga polinomial'
)
#plt.plot(t_total,np.heaviside(Q_rio2-Q_rio1,0)*(Q_rio1),label =
'Zamora  -Duero-' )
```

```
plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.subplot(212)

plt.plot(t_total,Q_rioS2,label = '(Zamora)Duero senoidal' )
plt.plot(t_total,Q_rioP2,label = '(Zamora)Duero polinomial' )

plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



3.5 SIMULACIÓN CONJUNTA DE LLUVIA Y CAUDAL

1. Inicializamos las centrales

In [98]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central351',Presa_salida='cen
tral332',P_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,
```

```
Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central352',Presa_salida='',P
_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [99]:

```
def funcion_turbina_341(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 60

def funcion_turbina_342(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
```

```
        self.Qg[t] = 125
```

In [100]:

```
turbina351 =
hp.Turbina(Nombre='turbina351',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_341)
turbina352 =
hp.Turbina(Nombre='turbina352',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_342)
```

3. Creamos los caudales

In [101]:

```
lista_meses=[0,1,2,6,10]
hora1=hm.lista_mes_hora(lista_meses)
lista_meses=[0,1,2,6,10]
hora2=hm.lista_mes_hora(lista_meses)
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [102]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
Q_rioP2 = hm.poly_odd(t_total,hora,caudal2,horas_año)
```

In [103]:

```
Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Valladolid')
Q_lluvia1 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,

area_abastecida=presa1.base,#presa1.base_absorcion,

unidades_preci='mm',unidades_area='m2')

Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Zamora')
Q_lluvia2 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,
                                   area_abastecida=presa2.base,
```

```
unidades_preci='mm',unidades_area='m2')
```

4. Calcular

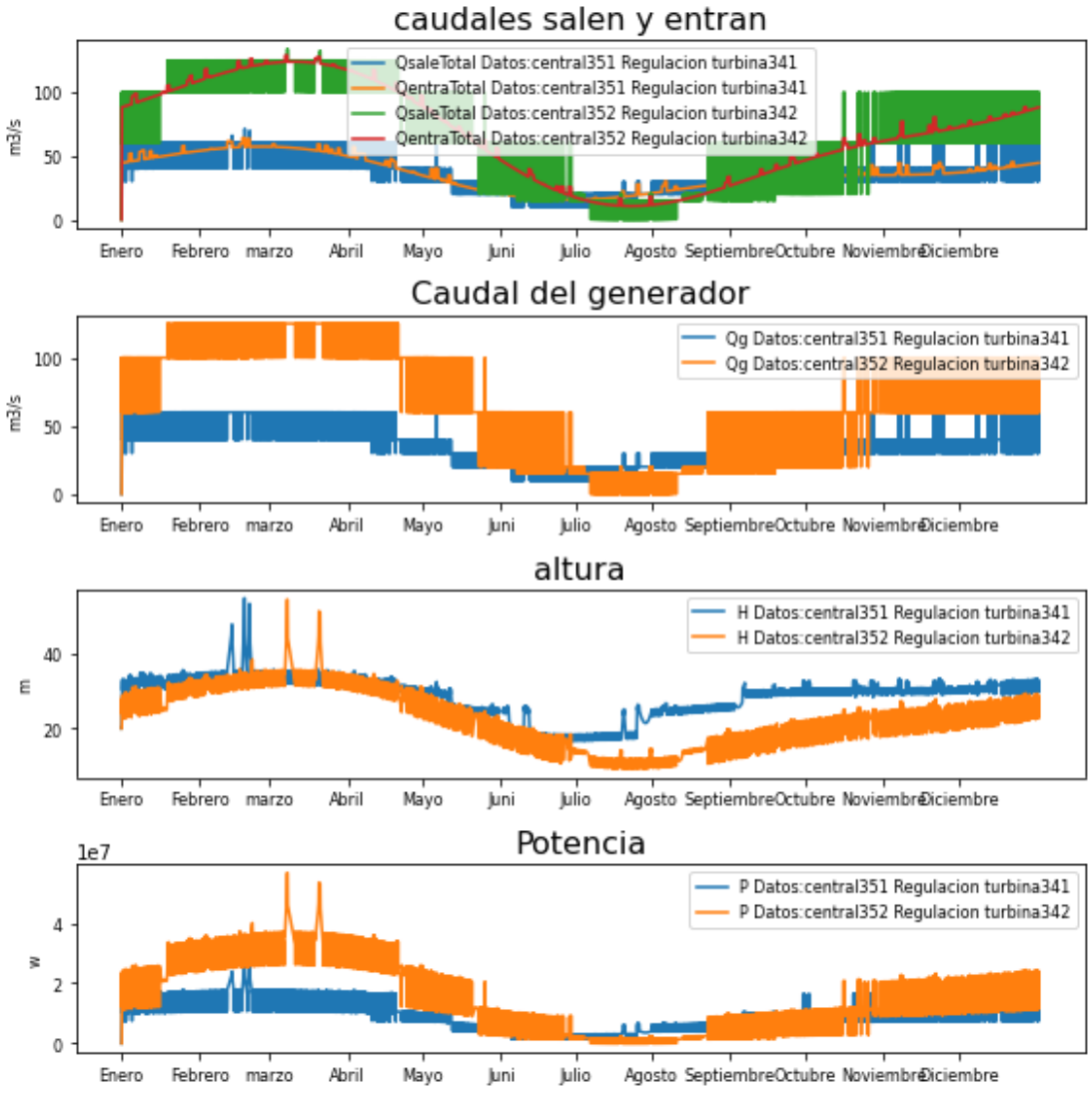
In [104]:

```
presa1.turbina=turbina341
presa2.turbina=turbina342

presa1.indicar_Qentra_m3s(Q_rioP1+Q_lluvia1)
presa2.indicar_Qentra_m3s(Q_rioP2+Q_lluvia2)

datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
ListaDatos35=[datos1,datos2]
```

Gráfica de ríos grandes simple

In [105]:

```
hp.representar_REGULACION(ListaDatos35)
```

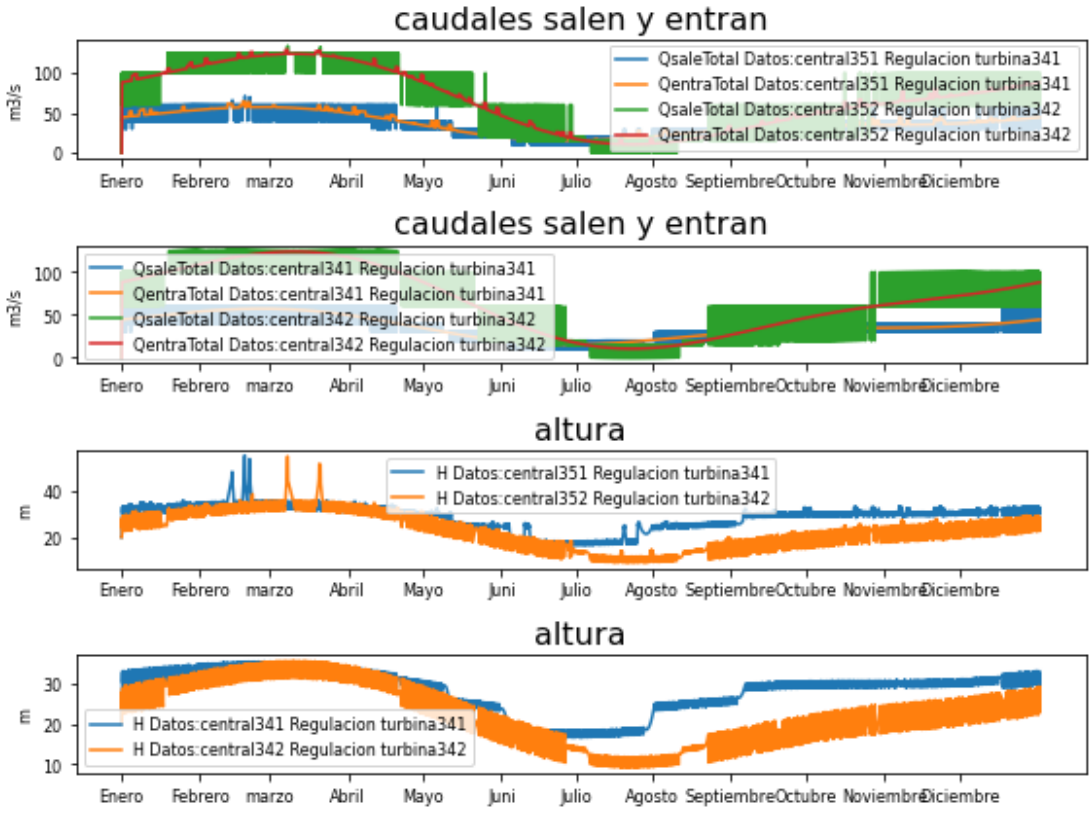


Comparación a modelo sin lluvia

In [106]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos35)


plt.subplot(412)
hp.representar_QentraQsale(ListaDatos34)


plt.subplot(413)
hp.representar_Altura(ListaDatos35)
```

```
plt.subplot(414)
hp.representar_Altura(ListaDatos34)

plt.tight_layout()
plt.show()
```



3.6 DEMANDA DE AGUA

1. Inicializamos las centrales

In [107]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central361',Presa_salida='',P
_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=15,Tipo_centr
al='Regulación',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central362',Presa_salida='',P
_Qg=1,H_min=15,
```

```
base=20000,base_absorcion=10000,Hi=20,H_max=25,Tipo_central='Regul
ación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Hentrada=30,Area=20))
```

2. Inicializamos la turbina

In [108]:

```
def funcion_turbina_361(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 45
    else:
        self.Qg[t] = 60

def funcion_turbina_362(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [109]:

```
turbina361 =
hp.Turbina(Nombre='turbina361',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_361)
```

```
turbina362 =
hp.Turbina(Nombre='turbina362',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_362)
```

In [110]:

```
def funcion_turbina_363(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [111]:

```
turbina363 =
hp.Turbina(Nombre='turbina363',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_363)
```

In [112]:

```
def funcion_turbina_364(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 5
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7
```

In [113]:

```
turbina364 =
hp.Turbina(Nombre='turbina364',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_364)
```

3. Creamos los caudales

In [114]:

```
lista_meses=[0,1,4,5,9]
```

```
hora=hm.lista_mes_hora(lista_meses)
```

In [115]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
#Q_rioP2=hm.poly_odd(t_total,hora,caudal2,horas_año)
```

In [116]:

```
lista_meses=[1,4,5,9,11]
hora2=hm.lista_mes_hora(lista_meses)
```

In [117]:

```
# caudalTámega
Q_med3 = 9.23

Q_max3 =10.63
Q_min3 =8.63

Q_A3=Q_max3-Q_min3
caudal3=[Q_max3,Q_med3,Q_min3*0.98 ,Q_min3*0.99,Q_med3  ]

Q_rioP3=hm.poly_odd(t_total,hora2,caudal3,horas_año)

# caudal Carrión
Q_med4 = 5.03
Q_max4 =6.89
Q_min4 =4.06
Q_A4=Q_max4-Q_min4
caudal4=[Q_max4,Q_med4,Q_min4*1.02,Q_min4*1.03,Q_med4 ]

Q_rioP4=hm.poly_odd(t_total,hora2,caudal4,horas_año)
```

4. Creamos las necesidades

Población 2021 Instituto Nacional de Estadística:

In [118]:

```
Poblacion_Bercero = 191
Poblacion_Toro = 8532
Poblacion_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en España por persona

In [119]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [120]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [121]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [122]:

```
necesidad361=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad362=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad363=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
lista_Necesidades= [necesidad361,necesidad362,necesidad363]
```

5. Calcular

In [123]:

```
presa1.turbina=turbina361
#presa2.turbina=turbina362

presa1.indicar_Qentra_m3s(Q_rioP1)
#presa2.indicar_Qentra_m3s(Q_rioP2)

presa1.Nombre='central361'
```

```
datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad361)
presa1.Nombre='central362'
datos2 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad362)
presa1.Nombre='central363'
datos3 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad363)

ListaDatos1=[datos1,datos2,datos3]
```

In [124]:

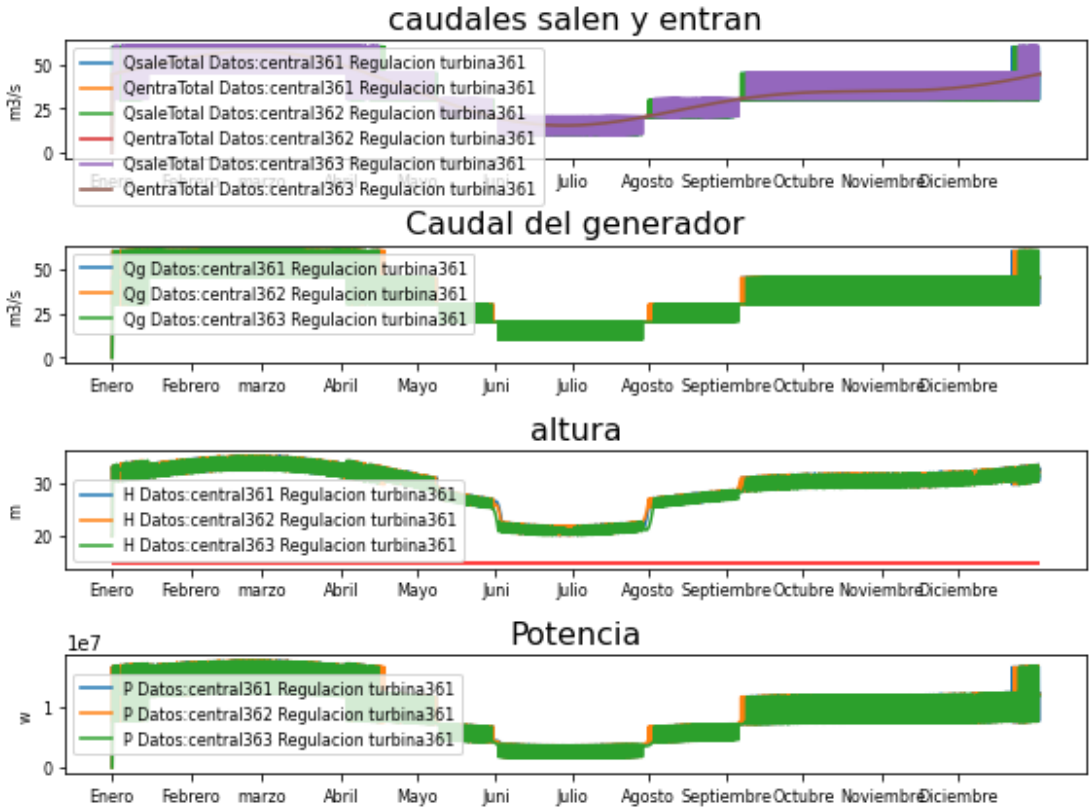
```
presa1.turbina=turbina363
#presa2.turbina=turbina364

presa1.indicar_Qentra_m3s(Q_rioP3)
#presa2.indicar_Qentra_m3s(Q_rioP4)

presa1.Nombre='central361'
datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad361)
presa1.Nombre='central362'
datos2 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad362)
presa1.Nombre='central363'
datos3 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=necesidad363)

ListaDatos2=[datos1,datos2,datos3]
```

Gráfica de ríos grandes simple

In [125]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos1)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos1)

plt.subplot(413)
hp.representar_Altura(ListaDatos1)
plt.hlines(presa1.H_min,xmin=0,xmax=len(t_total),color='red',label
= 'Altura minima')

plt.subplot(414)
hp.representar_Pgenerador(ListaDatos1)
```

```
plt.tight_layout()
plt.show()
```

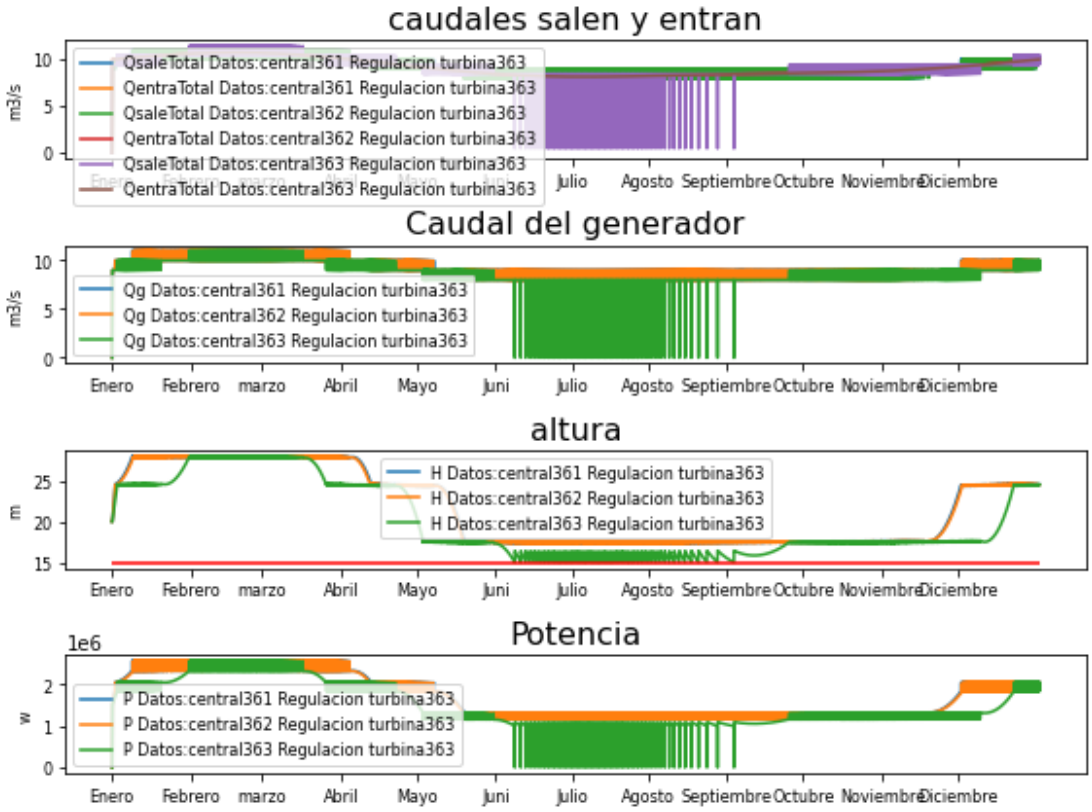


Gráfica de ríos pequeños simple

In [126]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos2)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos2)

plt.subplot(413)
hp.representar_Altura(ListaDatos2)
plt.hlines(presa1.H_min,xmin=0,xmax=len(t_total),color='red',label
= 'Altura minima')

plt.subplot(414)
hp.representar_Pgenerador(ListaDatos2)
```

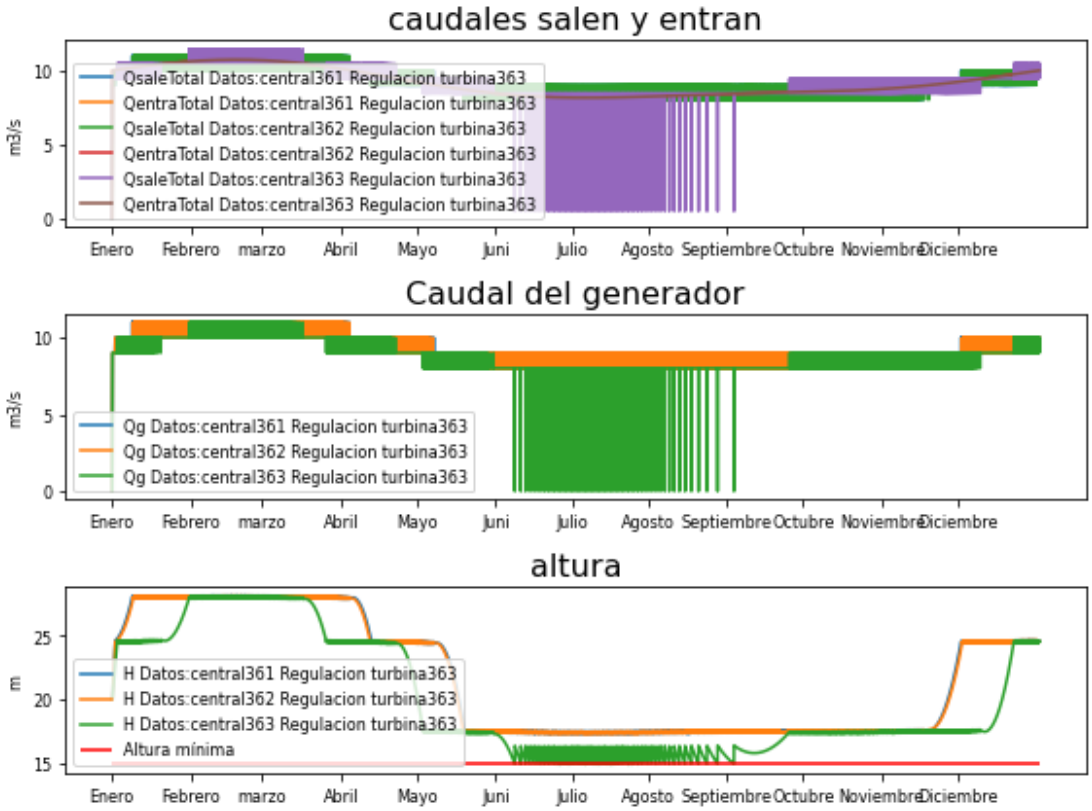
```
plt.tight_layout()
plt.show()
```



Caudales

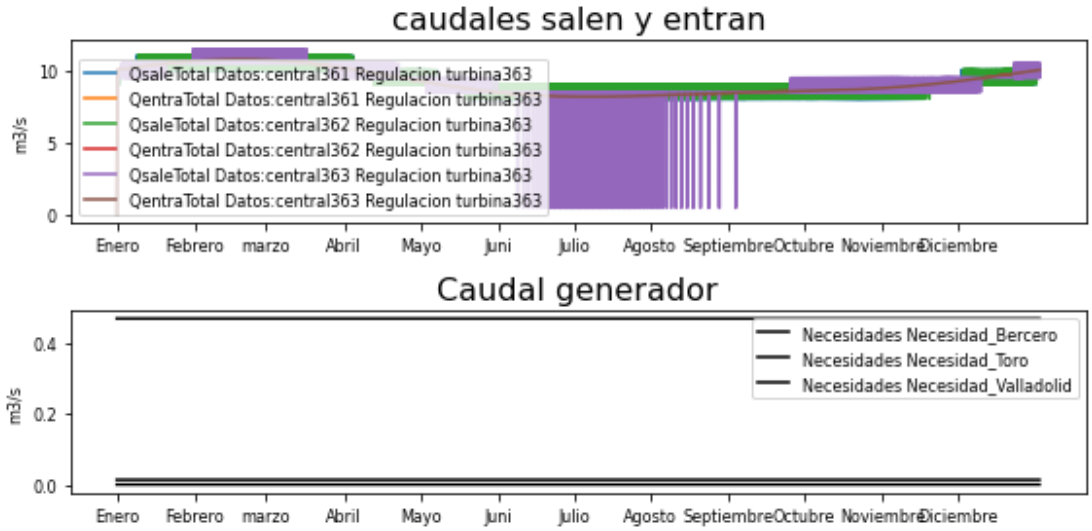
In [127]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_QentraQsale(ListaDatos2)


plt.subplot(312)
hp.representar_Qgenerador(ListaDatos2)


plt.subplot(313)
hp.representar_Altura(ListaDatos2,LH={'Altura
mínima':presa1.H_min})


plt.tight_layout()
plt.show()
```



In [128]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.subplot(2,1,1)
hp.representar_QentraQsale(ListaDatos2)

plt.subplot(2,1,2)
i=0
for dato in ListaDatos2:
    string1= 'Necesidades '+lista_Necesidades[i].Nombre
    plt.plot(t_total,lista_Necesidades[i].Qsale_m3s,'k',label =
string1 )
    i=i+1
plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)
plt.title('Caudal generador',fontsize=2*LETTER_SIZE)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



3.7 DEMANDA DE POTENCIA

In [129]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia

tini=0
tfin=7*horas_dia
t_total_semana= np.arange(tfin-tini)
```

In [130]:

```
from sympy.solvers import solve
from sympy import Symbol
```

In [131]:

```
from sympy.abc import x
```

1. Inicializamos las centrales

In [132]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total_semana,Q_entrada=0,Nombre='central371',Presa_sali
da='',P_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='regulacion',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=40))
```

```
presa2 =
hp.Presa(t_total_semana,Q_entrada=0,Nombre='central372',Presa_sali
da='',P_Qg=1,

base=20000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='regulacion',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=40))
```

2. Inicializamos la turbina

In [133]:

```
def funcion_turbina_371(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or necesidad.Pgn_ms[t]<0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 45
    else:
        self.Qg[t] = 60

def funcion_turbina_372(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min or necesidad.Pgn_ms[t]<0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [134]:

```
turbina371 =
hp.Turbina(Nombre='turbina371',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_371)
turbina372 =
hp.Turbina(Nombre='turbina372',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_372)
```

In [135]:

```
def funcion_turbina_373(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min or necesidad.Pgn_ms[t]<0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [136]:

```
turbina373 =
hp.Turbina(Nombre='turbina373',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_373)
```

In [137]:

```
def funcion_turbina_374(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min or necesidad.Pgn_ms[t]<0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 5
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7
```

In [138]:

```
turbina374 =
hp.Turbina(Nombre='turbina374',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_374)
```

3. Creamos los caudales de entrada

In [139]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [140]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total_semana,hora,caudal1,horas_año)
Q_rioP2 = hm.poly_odd(t_total_semana,hora,caudal2,horas_año)
```

4. Creamos la Necesidad

In [141]:

```
hora= [4,9,13,17,22]
potencia=[4.9,6.5,6.6,5.8,6.9]
```

In [142]:

```
t_sem = np.linspace(0, 24, num=25)
demanda_ElHierro = hm.poly_odd(t_total_semana,hora,potencia,24)
```

In [143]:

```
demanda_ElHierro_centrado =demanda_ElHierro-
np.mean(demanda_ElHierro)
```

In [144]:

```
necesidad371=
hp.Necesidades(t_total_semana,Nombre='Necesidad_ElHierro_diario',P
otencia_necesaria=demanda_ElHierro_centrado)
```

Demandas para distintas poblaciones

In [145]:

```
poblacion =11154  # Población de El Hierro

necesidad372=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P100',Potencia_nec
esaria=demanda_ElHierro_centrado/poblacion * 100)
necesidad373=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P1000',Potencia_ne
cesaria=demanda_ElHierro_centrado/poblacion * 1000)
necesidad374=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P10000',Potencia_n
ecesaria=demanda_ElHierro_centrado/poblacion * 10000)
```

```
necesidad375=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P50000',Potencia_n
ecesaria=demanda_ElHierro_centrado/poblacion * 50000)
necesidad376=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P100000',Potencia_
necesaria=demanda_ElHierro_centrado/poblacion * 100000)
```

Creamos los objetos necesidades

5. Calcular

In [146]:

```
presa1.turbina=turbina371
presa2.turbina=turbina372

presa1.indicar_Qentra_m3s(Q_rioP1[:len(t_total_semana)])
presa2.indicar_Qentra_m3s(Q_rioP2[:len(t_total_semana)])

datos1 = hp.sistema_regulacion(t_total_semana, presa=presa1,
necesidad=necesidad371)
datos2 = hp.sistema_regulacion(t_total_semana, presa=presa2,
necesidad=necesidad371)

ListaDatos1=[datos1]
```

In [147]:

```
presa1.turbina=turbina373
presa2.turbina=turbina374

presa1.indicar_Qentra_m3s(Q_rioP3[:len(t_total_semana)])
presa2.indicar_Qentra_m3s(Q_rioP4[:len(t_total_semana)])

datos1 = hp.sistema_regulacion(t_total_semana, presa=presa1,
necesidad=necesidad371)
datos2 = hp.sistema_regulacion(t_total_semana, presa=presa2,
necesidad=necesidad371)

ListaDatos2=[datos1,datos2]
```

In [148]:

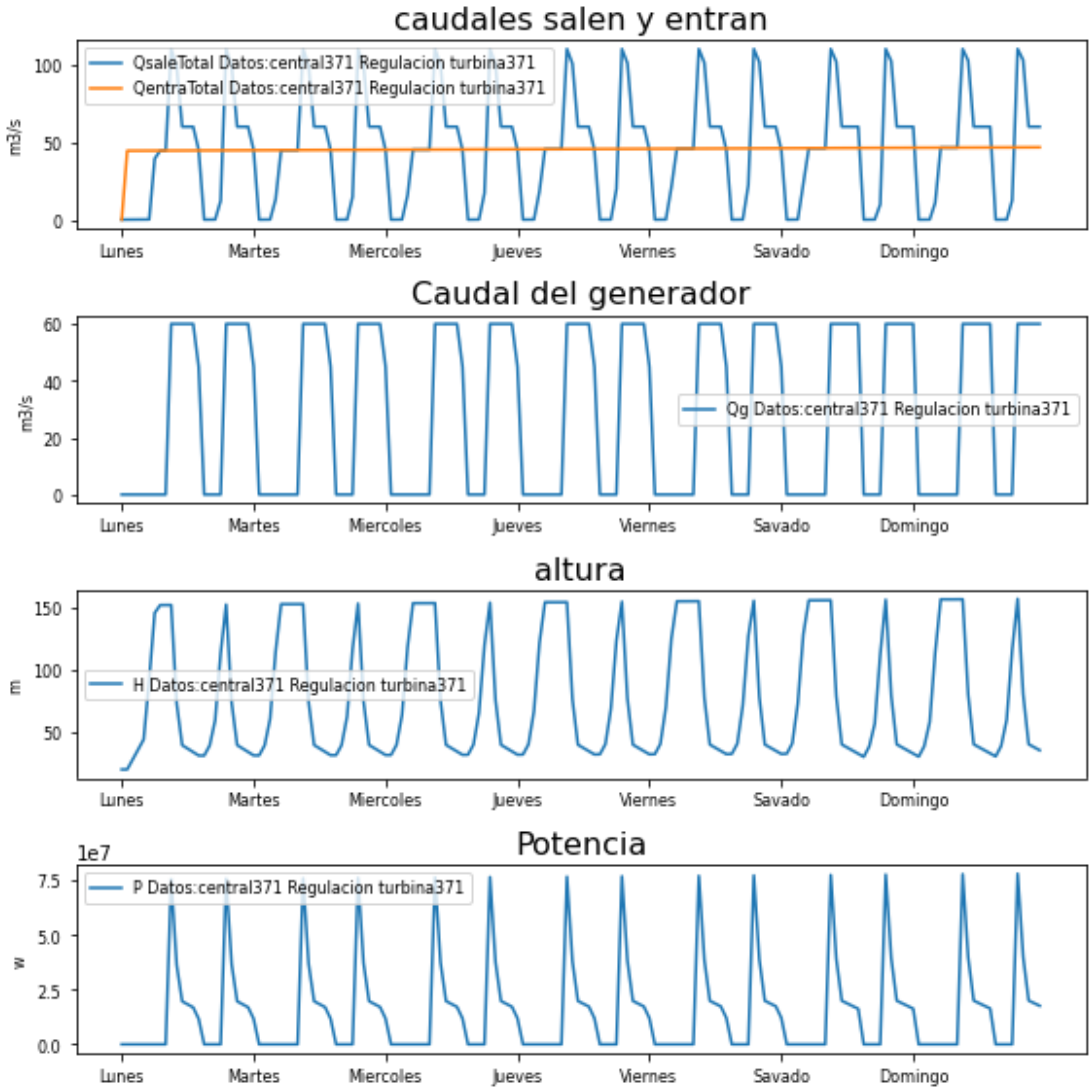
```
necesidad371=
hp.Necesidades(t_total_semana,Nombre='Necesidad_ElHierro_diario',P
otencia_necesaria=demanda_ElHierro_centrado)
```

5. Calcular

Gráfica de ríos grandes simple

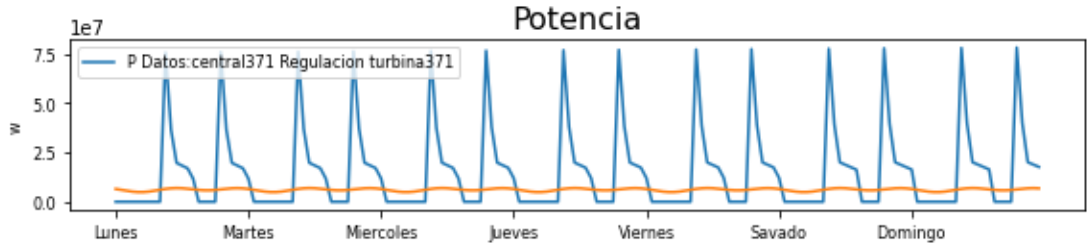
In [149]:

```
ListaDatos = ListaDatos1
hp.representar_REGULACION(ListaDatos, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
```



In [150]:

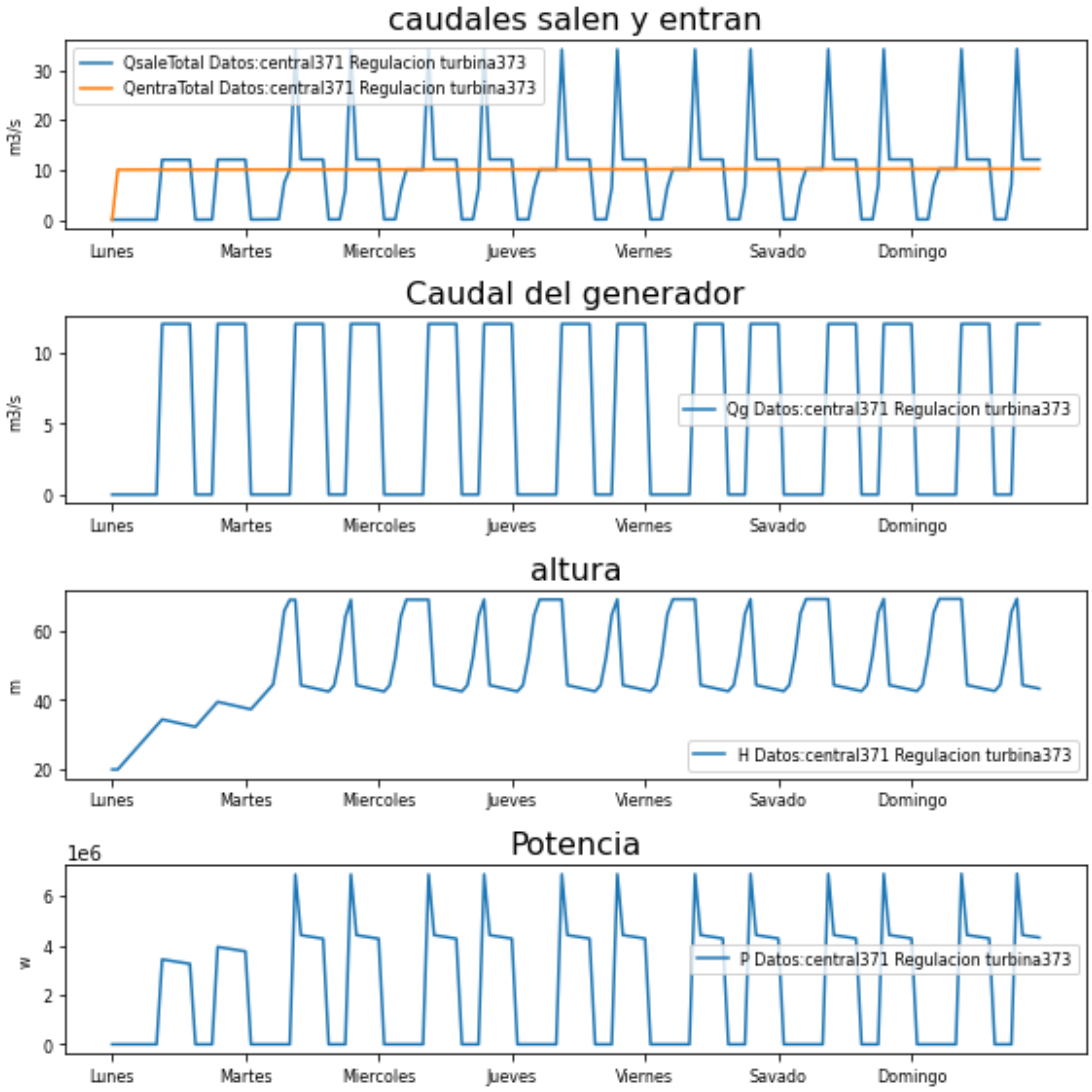
```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos1, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)

plt.tight_layout()
plt.show()
```



Gráfica de ríos pequeños simple

In [151]:

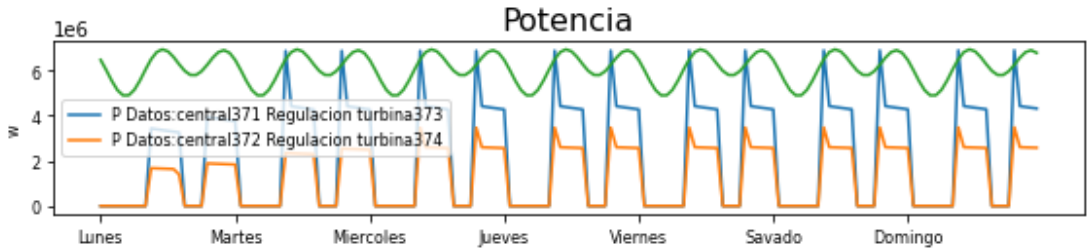
```
hp.representar_REGULACION([ListaDatos2[0]],
t_total=t_total_semana, horatik=horaSemana, nombretik=semana,
LH_Altura={})
```



In [152]:

```
plt.figure(figsize=(WIDE, HEIGHT))
```

```
plt.subplot(111)
hp.representar_Pgenerador(ListaDatos2, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)

plt.tight_layout()
plt.show()
```



ANEXO 7.2 NOTEBOOK 2

2 CENTRALES INTERCONECTADAS

Simulaciones de centrales de derivación, fluyentes y reversibles interconectadas.

Estructura del notebook:

1. Simulación de centrales fluyentes interconectadas
    - Interconexión normal
    - Interconexión con retardo
2. Simulación de centrales de derivación interconectadas
    - Interconexión normal
    - Interconexión con retardo
3. Simulación de centrales de regulación interconectadas
    - Interconexión normal
    - Interconexión con retardo
    - Interconexión con central fluyente

1. Importaciones e inicializaciones

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib widget
```

Módulos para el modelo de las distintas entradas y demandas:

In [2]:

```
import lib.herramientas_modelado as hm
```

Módulos para las simulaciones:

In [3]:

```
import lib.herramientas_presas as hp
```

2. Datos para las simulaciones:

In [4]:

```
dias_año = 365
horas_dia = 24
horas_año = dias_año*horas_dia

tini = 0
tfin = horas_año
t_total = np.arange(tfin-tini)
```

3. Datos para la representación:

In [5]:

```
meses =
['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Agosto',
'Septiembre','Octubre','Noviembre','Diciembre']
horaMes =
[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,272*24,303
*24,333*24]

semana =
['Lunes','Martes','Miercoles','Jueves','Viernes','Savado','Domingo
']
horaSemana = [0,24,24*2,24*3,24*4,24*5,24*6]
```

Para graficar más fácilmente haremos unos ajustes:

In [6]:

```
n = 0
WIDE = 8
HEIGHT = 2
LETTER_SIZE = 8
```

4. Parámetros estándar

In [7]:

```
# Objeto de la clase necesidades vacio
Necesidades0 = hp.Necesidades(t_total,Nombre='Necesidades0')
# Retardos
RETARDO = 40
```

1. FLUYENTE

1.1 INTERCONEXIONES

1. Inicializamos las centrales

In [8]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central111',Presa_salida='cen
tral112',P_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central112',Presa_salida='',P
_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [9]:

```
#Inicializamos las turbinas
turbina11 = hp.Turbina(Nombre='tuebina11',r=0.85)
```

In [10]:

```
presa1.turbina=turbina11
presa2.turbina=turbina11
```

3. Creamos los caudales

In [11]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 = (54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 = (109.18-46.58)/2

# caudalTámega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión

Q_med2 = 5.03
Q_A2 =2.83



#mit=0.3

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
```

4. Lista de centrales

In [12]:

```
listPresa11 = [presa1,presa2]
```

5. Calcular

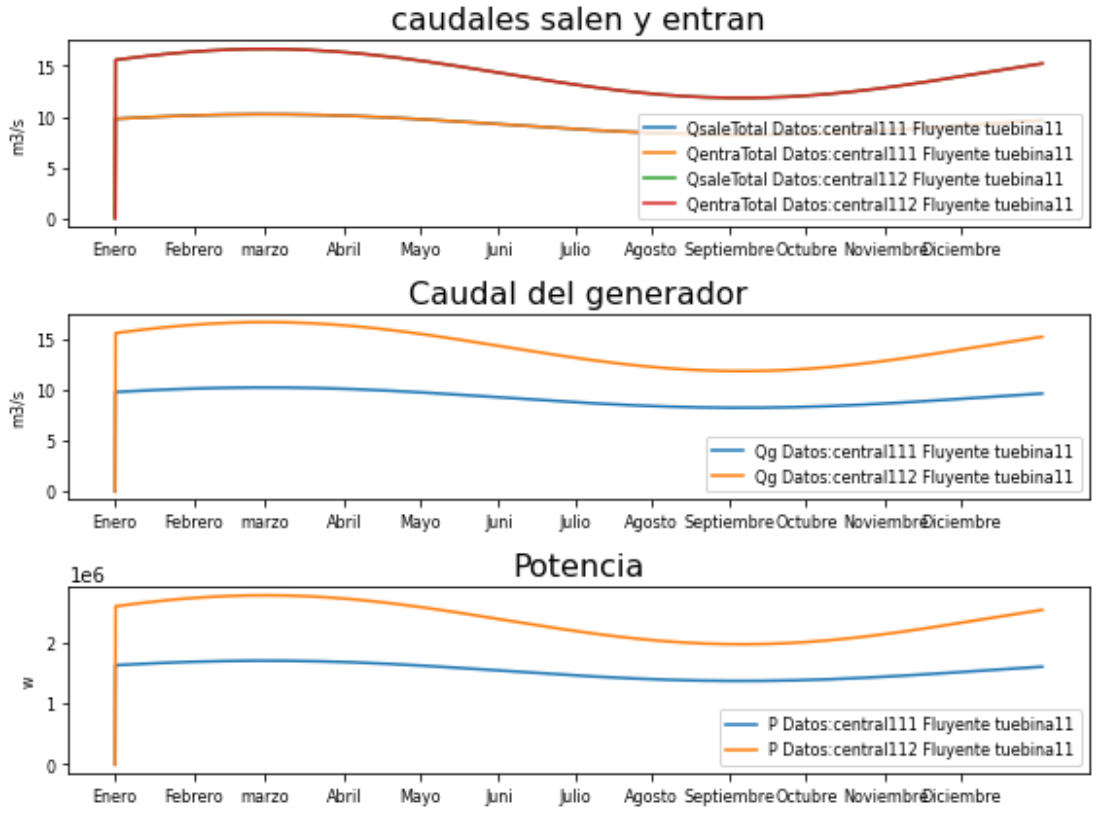
In [13]:

```
listaDatos11=hp.sistema_multiderivacionfluyente(t_total,
listPresa=listPresa11, necesidad=Necesidades0)
```

Gráficas

In [14]:

```
hp.representar_FLUYENTE_DERIVACION(listaDatos11)
```



1.2 RETARDO

1. Inicializamos las centrales

In [15]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central121',Presa_salida='cen
tral122',P_Qg=1,Presa_salida_t=RETARDO,
```

```
base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central122',Presa_salida='',P
_Qg=1,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyent
e',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [16]:

```
#Inicializamos las turbinas
turbina12 = hp.Turbina(Nombre='tuebina12',r=0.85)
```

In [17]:

```
presa1.turbina=turbina12
presa2.turbina=turbina12
```

3. Creamos los caudales

In [18]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 = (54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 = (109.18-46.58)/2

#mit=0.3

Q_rio1 =
hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(31+28)*2
4)
Q_rio2 =
hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(31+28)*2
4)

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
```

4. Lista de centrales

In [19]:

```
listPresa12 = [presa1,presa2]
```

5. Calcular

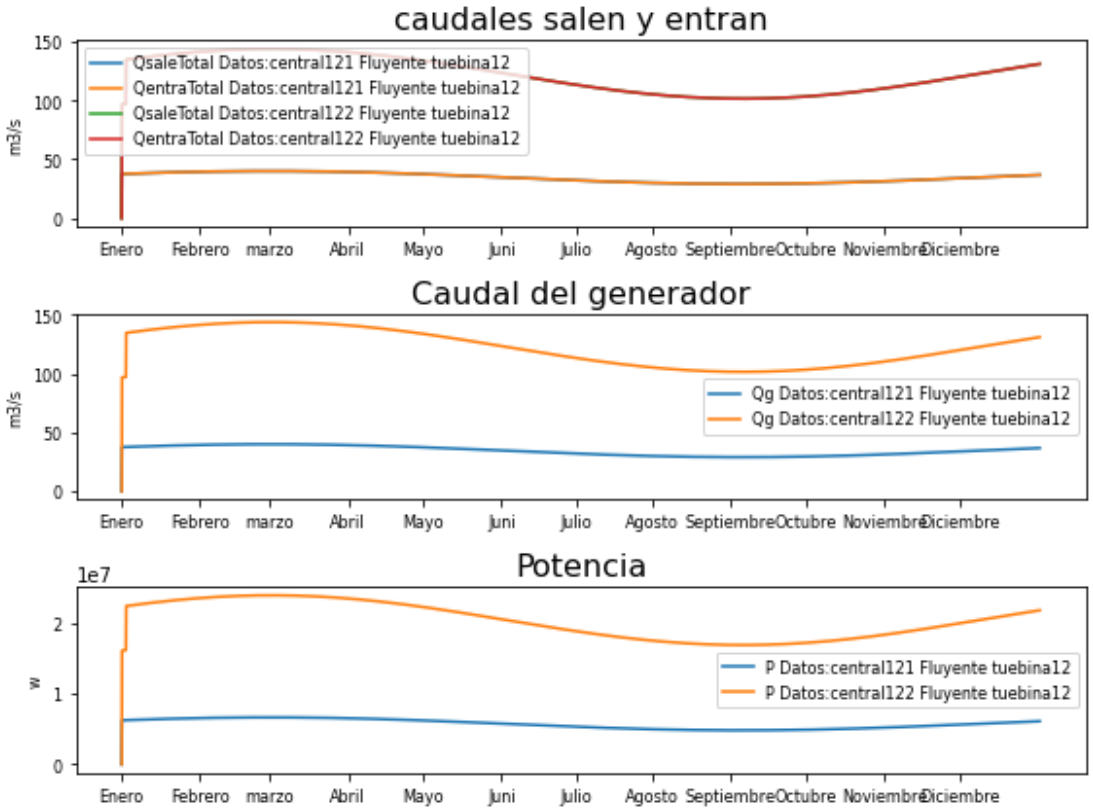
In [20]:

```
listaDatos12 = hp.sistema_multiderivacionfluyente(t_total,
listPresa=listPresa12, necesidad=Necesidades0)
```

Gráficas

In [21]:

```
hp.representar_FLUYENTE_DERIVACION(listaDatos12)
```



2. DERIVACIÓN

2.1 INTERCONEXIONES

In [22]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central211',Presa_salida='cen
tral212',P_Qg=0.5,

base=10000,base_absorcion=50000,Hi=20,H_max=25,Tipo_central='Deriv
acion',H_des=35,
                    Qdesbordado=10.0)
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central212',Presa_salida='',P
_Qg=0.5,

base=10000,base_absorcion=50000,Hi=20,H_max=25,Tipo_central='Deriv
acion',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [23]:

```
#Inicializamos las turbinas
turbina21 = hp.Turbina(Nombre='turbina21',r=0.85)
```

In [24]:

```
presa1.turbina = turbina21
presa2.turbina = turbina21
```

3. Creamos los caudales

In [25]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 = (54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 = (109.18-46.58)/2

# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión

Q_med2 = 5.03
Q_A2 =2.83


#mit=0.3

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
```

4. Lista de centrales

In [26]:

```
listPresa21 = [presa1,presa2]
```

5. Calcular

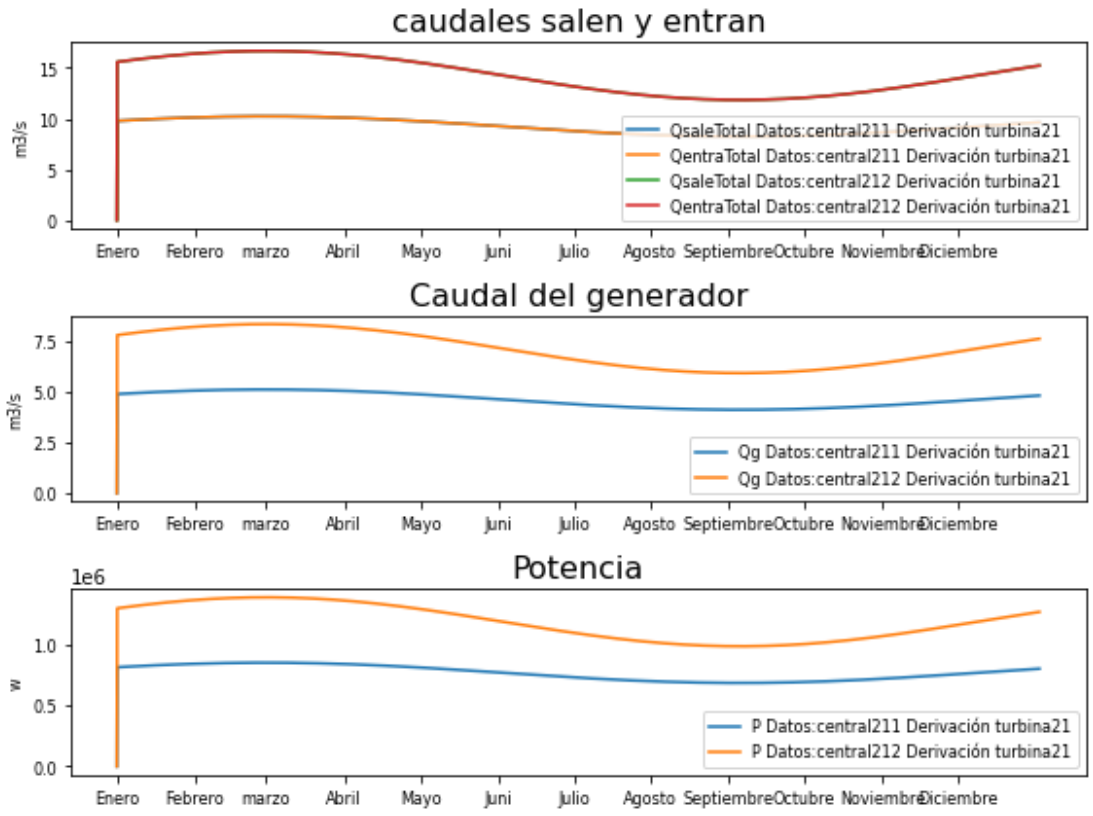
In [27]:

```
listaDatos21=hp.sistema_multiderivacionfluyente(t_total,
listPresa=listPresa21, necesidad=Necesidades0)
```

Gráficas

In [28]:

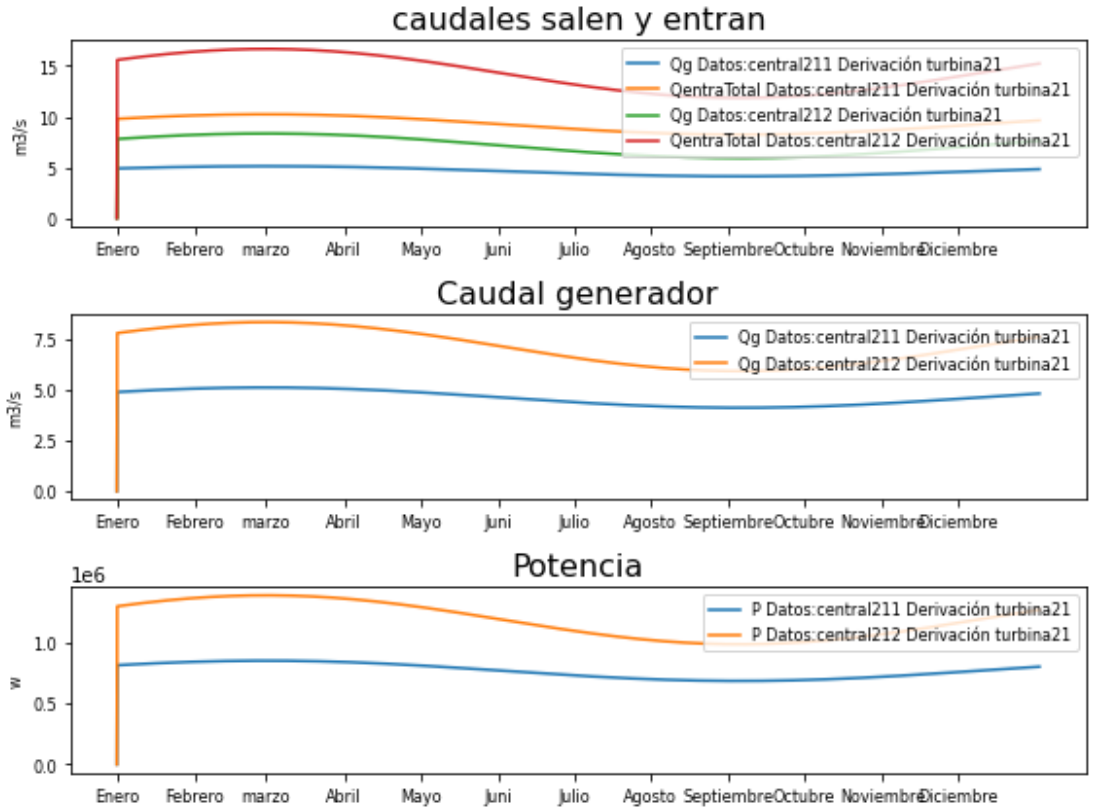
```
hp.representar_FLUYENTE_DERIVACION(listaDatos21)
```



In [29]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
plt.title('caudales salen y entran',fontsize=2*LETTER_SIZE)
for dato in listaDatos21:
    string1= 'Qg '+dato.Nombre
    string2= 'QentraTotal '+dato.Nombre
    plt.plot(t_total,dato.Qg,label = string1 )
    plt.plot(t_total,dato.QentraTotal,label = string2 )
plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
```

```
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)#,rotation=40
plt.legend(loc=1,fontsize=LETTER_SIZE)


plt.subplot(312)
plt.title('Caudal generador',fontsize=2*LETTER_SIZE)
for dato in listaDatos21:
    string1= 'Qg '+dato.Nombre
    plt.plot(t_total,dato.Qg,label = string1 )
    string1= 'Qdesbordamiento '+dato.Nombre
plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.subplot(313)
plt.title('Potencia',fontsize=LETTER_SIZE*2)
for dato in listaDatos21:
    string1= 'P '+dato.Nombre
    plt.plot(t_total,dato.Pg_ms,label = string1)
plt.ylabel('w', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE)
plt.legend(loc=1,fontsize=LETTER_SIZE)


plt.tight_layout()
plt.show()
```



2.2 RETARDO

In [30]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central221',Presa_salida='cen
tral222',P_Qg=0.5,Presa_salida_t=RETARDO,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Derivac
ion',H_des=35,
                  Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central222',Presa_salida='',P
_Qg=0.5,

base=1000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Derivac
ion',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [31]:

```
#Inicializamos las turbinas
```

```
turbina22 = hp.Turbina(Nombre='turbina21',r=0.85)
```

In [32]:

```
presa1.turbina=turbina22
presa2.turbina=turbina22
```

3. Creamos los caudales

In [33]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2

#mit=0.3

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
```

4. Lista de centrales

In [34]:

```
listPresa22 = [presa1,presa2]
```

5. Calcular

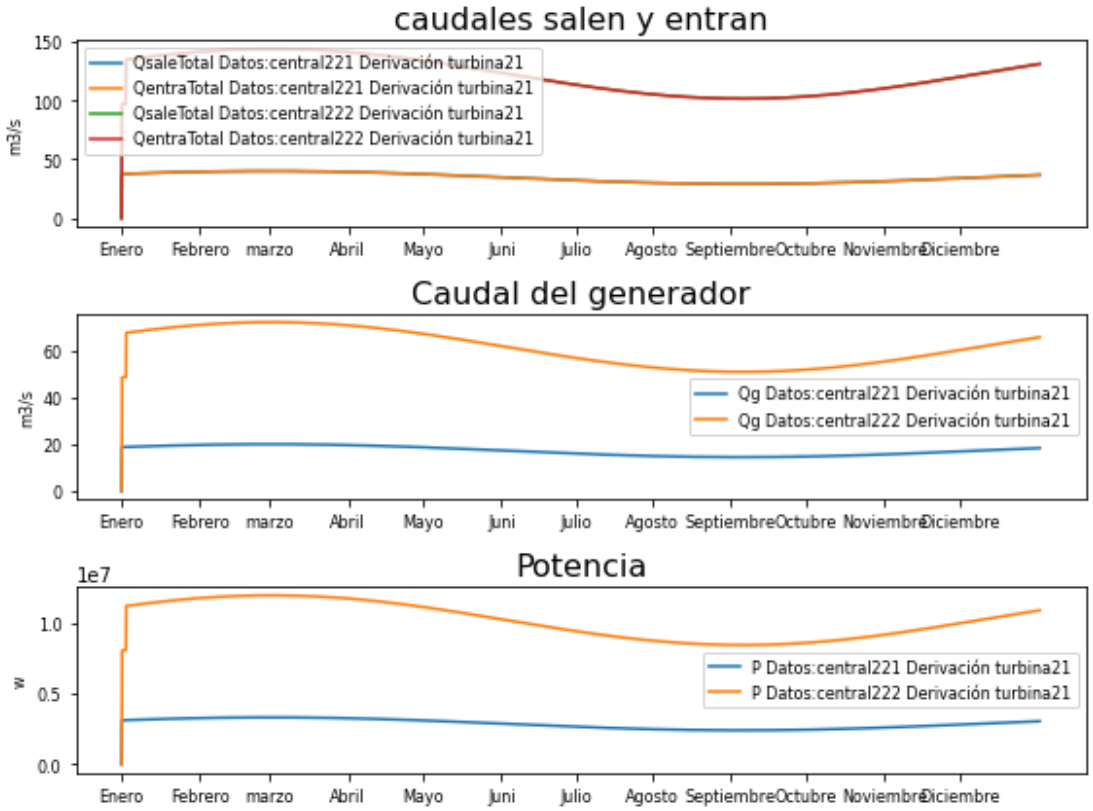
In [35]:

```
listaDatos22=hp.sistema_multiderivacionfluyente(t_total,
listPresa=listPresa22, necesidad=Necesidades0)
```

Gráficas

In [36]:

```
hp.representar_FLUYENTE_DERIVACION(listaDatos22)
```



3. REGULACIÓN

3.1 INTERCONEXIONES

1. Inicializamos las centrales

In [37]:

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central311',Presa_salida='cen
tral312',P_Qg=1,Presa_salida_t=RETARDO,

base=10000,base_absorcion=5000,Hi=20,H_max=25,H_min=10,Tipo_centra
l='Regulación',H_des=35,
                  Qdesbordado=10.0)
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central312',Presa_salida='',P
_Qg=1,

base=10000,base_absorcion=5000,Hi=20,H_max=30,H_min=10,Tipo_centra
l='Regulación',H_des=40,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [38]:

```
def funcion_turbina_313(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [39]:

```
#Inicializamos las turbinas
turbina313 =
hp.Turbina(Nombre='turbina313',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_313)
```

In [40]:

```
def funcion_turbina_314(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 14
    else:
        self.Qg[t] = 17
```

In [41]:

```
#Inicializamos las turbinas
turbina314 =
hp.Turbina(Nombre='turbina314',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_314)
```

3. Creamos los caudales

In [42]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
```

```
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2

# caudalTámega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión

Q_med2 = 5.03
Q_A2 =2.83

#mit=0.3

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

4. Calcular

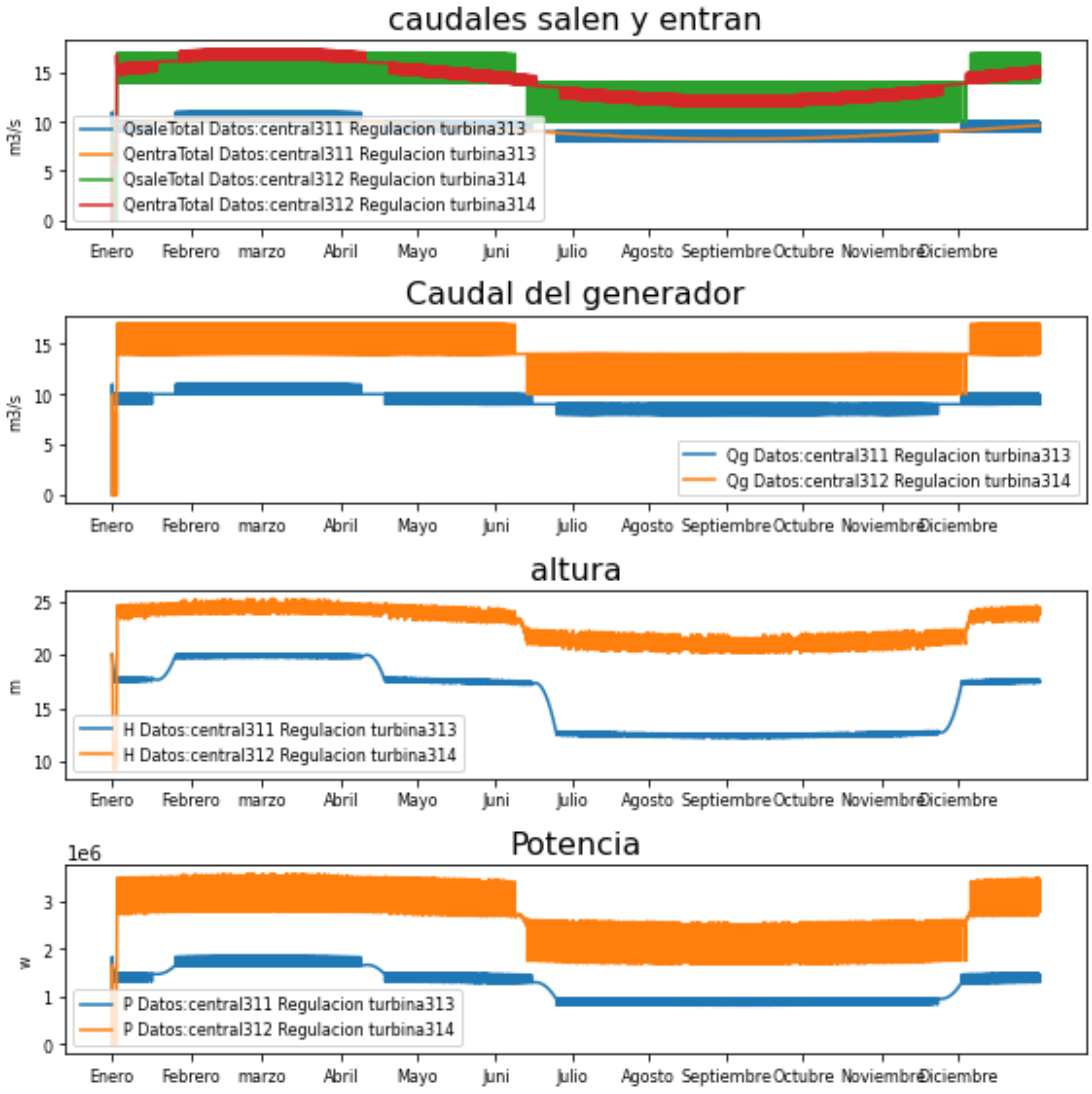
In [43]:

```
presa1.turbina = turbina313
presa2.turbina = turbina314

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

listPresa = [presa1,presa2]

listaDatos1=hp.sistema_multiregulacion(t_total,
listPresa=listPresa, necesidad=Necesidades0)
```

Gráficas

In [44]:

```
hp.representar_REGULACION(listaDatos1)
```



3.2 RETARDO

1. Inicializamos las centrales

In [45]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central321',Presa_salida='cen
tral322',P_Qg=1,Presa_salida_t=RETARDO,

base=10000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Regula
ción',H_des=35,
                  Qdesbordado=10.0)
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central322',Presa_salida='',P
_Qg=1,

base=10000,base_absorcion=5000,Hi=20,H_max=30,Tipo_central='Regula
ción',H_des=40,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [46]:

```
def funcion_turbina_323(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [47]:

```
#Inicializamos las turbinas
turbina323 =
hp.Turbina(Nombre='turbina323',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_323)
```

In [48]:

```
def funcion_turbina_324(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 14
    else:
        self.Qg[t] = 17
```

In [49]:

```
#Inicializamos las turbinas
```

```
turbina324 =
hp.Turbina(Nombre='turbina324',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_324)
```

3. Creamos los caudales

In [50]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2


# caudal Támega
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

4. Calcular

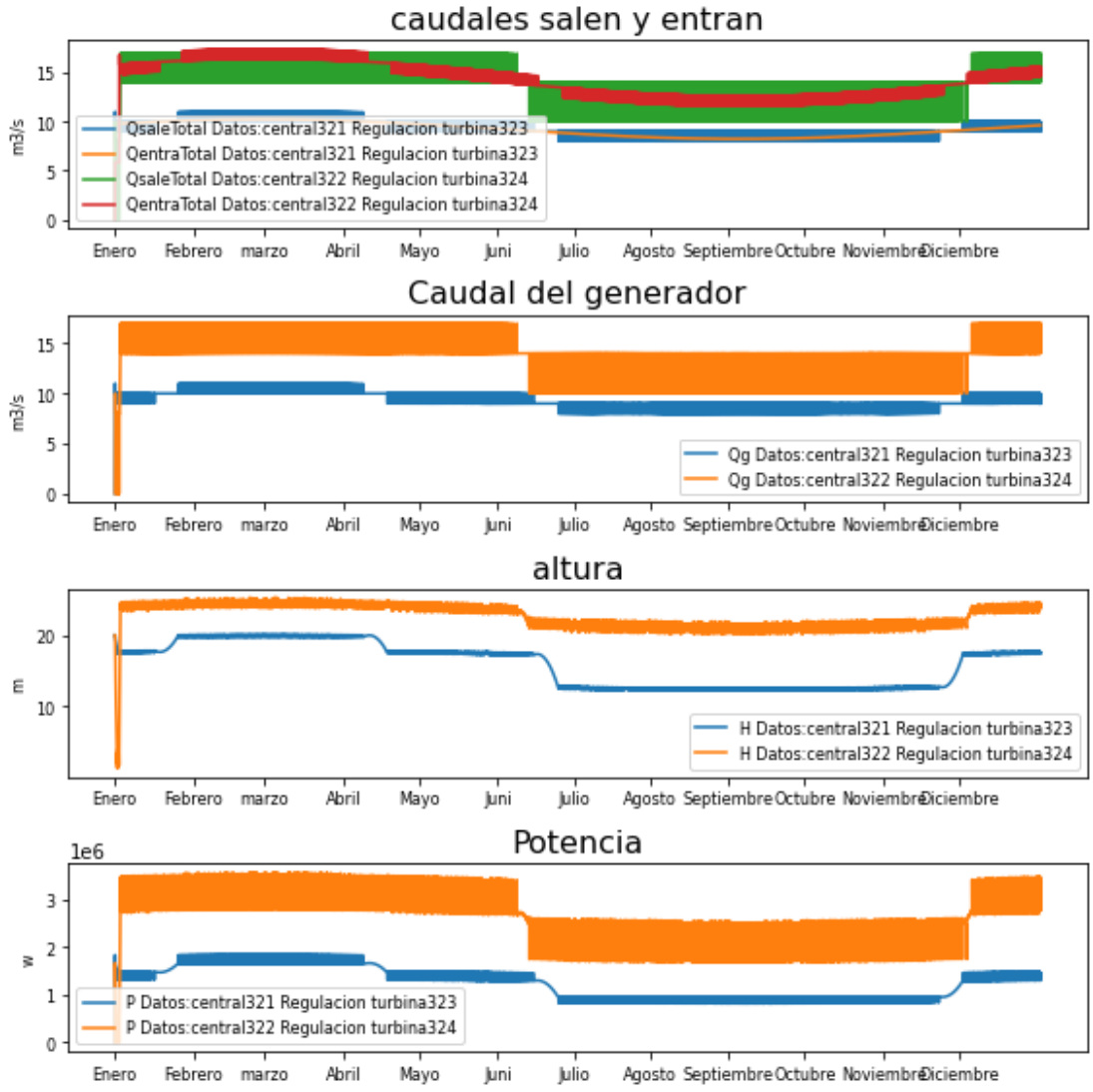
In [51]:

```
presa1.turbina = turbina323
presa2.turbina = turbina324

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

listPresa32 = [presa1,presa2]

listaDatos1=hp.sistema_multiregulacion(t_total,
listPresa=listPresa32, necesidad=Necesidades0)
```

Gráficas

In [52]:

```
hp.representar_REGULACION(listaDatos1)
```



3.3 MIXTA REGULACIÓN FLUYENTE

1. Inicializamos las centrales

In [53]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central331',Presa_salida='cen
tral332',P_Qg=1,

base=10000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Regula
ción',H_des=35,
                    Qdesbordado=10.0)
```

```
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central332',Presa_salida='',P
_Qg=1,

base=10000,base_absorcion=5000,Hi=20,H_max=25,Tipo_central='Fluyen
te',H_des=35,
                  Qdesbordado=10.0)
```

2. Inicializamos la turbina

In [54]:

```
def funcion_turbina_331(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10

    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
    else:
        self.Qg[t] = 12
```

In [55]:

```
# Inicializamos las turbinas
turbina331 =
hp.Turbina(Nombre='turbina331',r=0.85,Qg_funcion_propia=funcion_tu
rbina_331)
```

In [56]:

```
# Inicializamos las turbinas
turbina332 = hp.Turbina(Nombre='turbina332',r=0.85)
```

3. Creamos los caudales

In [57]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2

# caudal Támega
```

```
Q_med1 = 9.23
Q_A1=2

# caudal Carrión
Q_med2 = 5.03
Q_A2 =2.83

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

4. Calcular

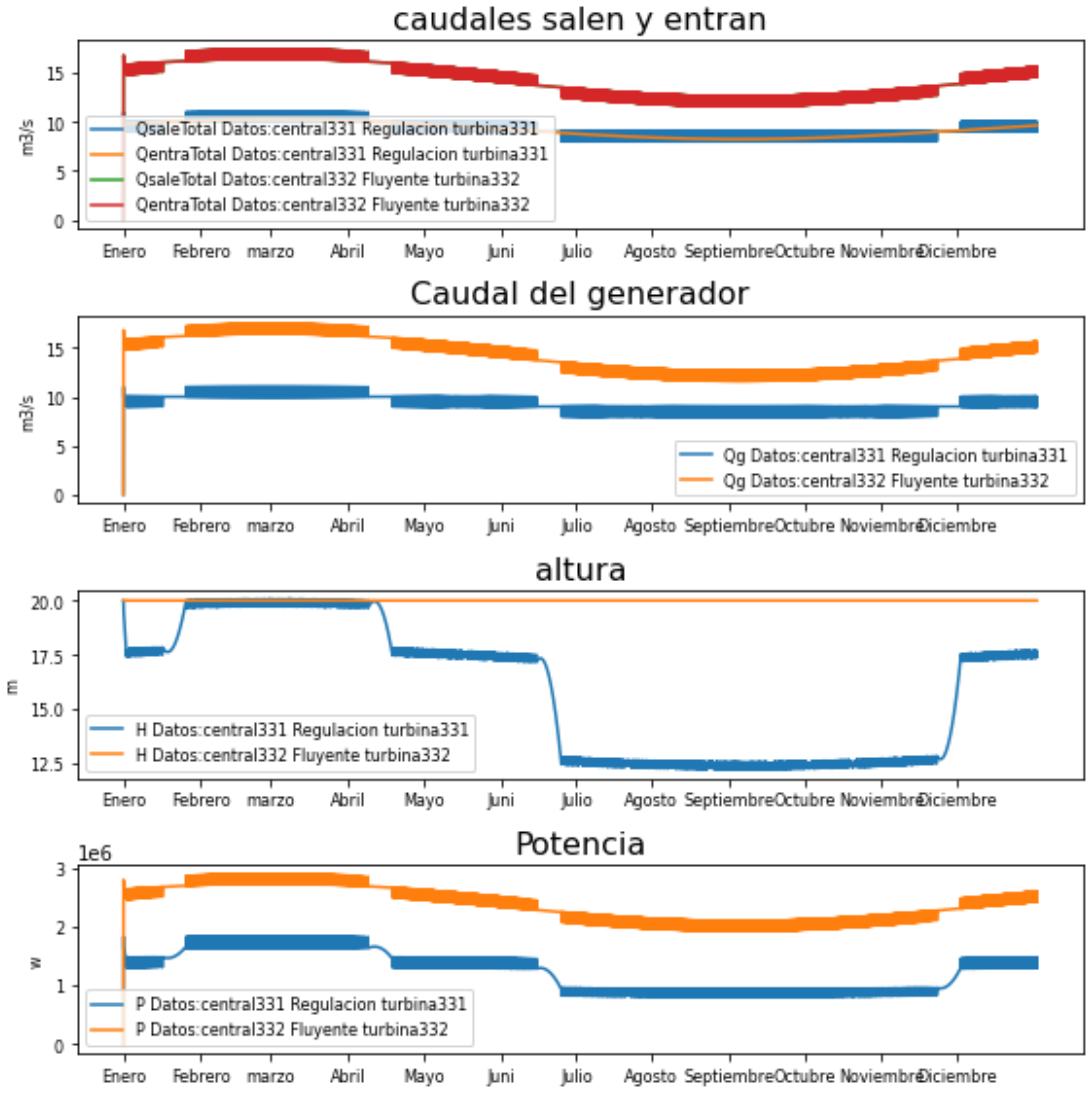
In [58]:

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)

presa1.turbina = turbina331
presa2.turbina = turbina332

listPresa33 = [presa1,presa2]

listaDatos1=hp.sistema_multicentral(t_total,
listPresa=listPresa33, necesidad=Necesidades0)
```

Gráficas

In [59]:

```
hp.representar_REGULACION(listaDatos1)
```



Gráfica caudales reducido

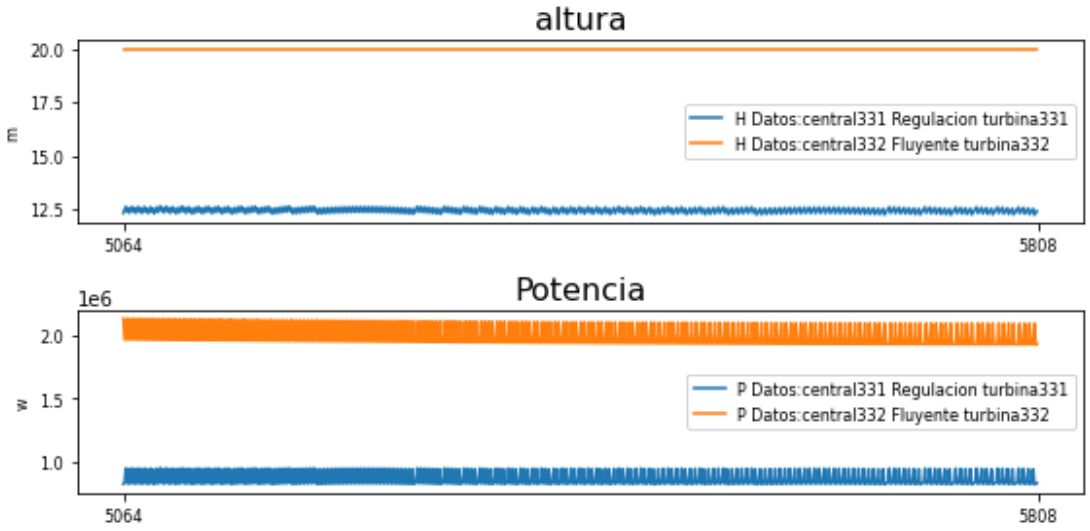
In [60]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=7
Mfin=8
tini=horaMes[Mini]
tfin=horaMes[Mfin]
```

In [61]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.subplot(211)
hp.representar_Altura(listaDatos1, t_total=t_total[tini:tfin],
horatik=[tini,tfin],nombretik=[tini,tfin], LH={})
```

```
plt.subplot(212)
hp.representar_Pgenerador(listaDatos1, t_total=t_total[tini:tfin],
horatik=[tini,tfin],nombretik=[tini,tfin])


plt.tight_layout()
plt.show()
```



ANEXO 7.3 NOTEBOOK 3

3 CENTRALES DE BOMBEO

Simulación de las centrales de bombeo del Capítulo 7 y Capítulo 8 de la Memoria

Estructura del notebook:

1. Primeras simulaciones (primeros ejemplos de bombeo)
2. Simulación de centrales de bombeo individuales
    - Simulación sin demanda
    - Simulación sin demanda con lluvia
    - Simulación con demanda de agua
    - Simulación con demanda de potencia
3. Simulación de centrales de bombeo interconectadas
    - Simulación sin demanda
    - Simulación sin demanda con lluvia
    - Simulación con demanda de agua
    - Simulación con demanda de potencia

1. Importaciones e inicializaciones

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
#%matplotlib widget
```

Módulos para el modelo de las distintas entradas y demandas:

In [2]:

```
import lib.herramientas_modelado as hm
```

Módulos para las simulaciones:

In [3]:

```
import lib.herramientas_presas as hp
```

2. Datos para las simulaciones:

In [4]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia

tini=0
tfin=horas_año

t_total= np.arange(tfin-tini)
```

3. Datos para la representación:

In [5]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]

semana =
['Lunes','Martes','Miercoles','Jueves','Viernes','Savado','Domingo
']
horaSemana = [0,24,24*2,24*3,24*4,24*5,24*6]
```

Para graficar más fácilmente haremos unos ajustes:

In [6]:

```
n=0
WIDE = 8
HEIGHT = 2
LETTER_SIZE = 8
```

4. Creamos el objeto de la clase necesidades vacío:

In [7]:

```
Necesidades0 = hp.Necesidades(t_total,Nombre='Necesidades0')
```

1. CENTRAL DE BOMBEO (PRIMEROS MODELOS)

1.1 PRIMER MODELO

1. Inicializamos las centrales

In [8]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 = hp.Presa(t_total,Q_entrada=0,Nombre='presa111',

base=1000,base_absorcion=5000,Hi=2,H_max=25,H_min=5,Tipo_central='
bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=5,Hentrada=8.000))
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa112',H_des=35,

base=1000,base_absorcion=5000,Hi=2,H_max=25,H_min=5,Tipo_central='
bombeo',Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=5,Hentrada=
8.000))
```

```
presa3 =
hp.Presa(t_total,Hi=5,Q_entrada=0,Tipo_central='bombeo',P_Qg=0.9,N
ombre='presa113',H_des=35,

base=1000,base_absorcion=5000,H_max=25,Qdesbordado=10.0,aliviadero
=hp.Aliviadero(Area=5,Hentrada=8.000))
```

2. Inicializamos la turbina

In [9]:

```
def funcion_turbina_11(self=0,t=0, necesidad=0, presa=0, g=9.8,
den=998, intervalo=3600):
    var= presa.H_max - presa.H_min

    if self.H[t]<presa.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< presa.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< presa.H_max*0.8:
        self.Qg[t] = 5
    elif  self.H[t]< presa.H_max*0.9:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7.1
```

In [10]:

```
#Inicializamos las turbinas
turbina11 =
hp.Turbina(Nombre='turbina11',Qg_funcion_propia=funcion_turbina_11
)
```

2. Inicializamos la Bomba

In [11]:

```
def funcion_Qb11(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 0.0;
```

In [12]:

```
def funcion_Pb11(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    self.Pb[t] = 0;
```

In [13]:

```
#Inicializamos las turbinas
bomba11 =
hp.Bomba(Nombre='bomba11',Qb_funcion_propia=funcion_Qb11,Pb_funcio
n_propia=funcion_Pb11)
```

3. Creamos los caudales

In [14]:

```
# Caudal eresma Bernardos
Q_med1 = 5.3
Q_A1=2.08

# Caudal Támega
Q_med2 = 9.23
Q_A2=2

# Caudal Carrión

Q_med3 = 5.03
Q_A3 =2.83


Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
Q_rio3=hm.caudal_rio(t_total,Q_med3,variacion_anual=Q_A3,max_an=(3
1+28)*24)
```

Creamos el objeto de necesidades. aunque en este caso su valor estará practicamente vacío dado que queremos observar el efecto del caudal del generador y no otros parámetros

4. Calcular

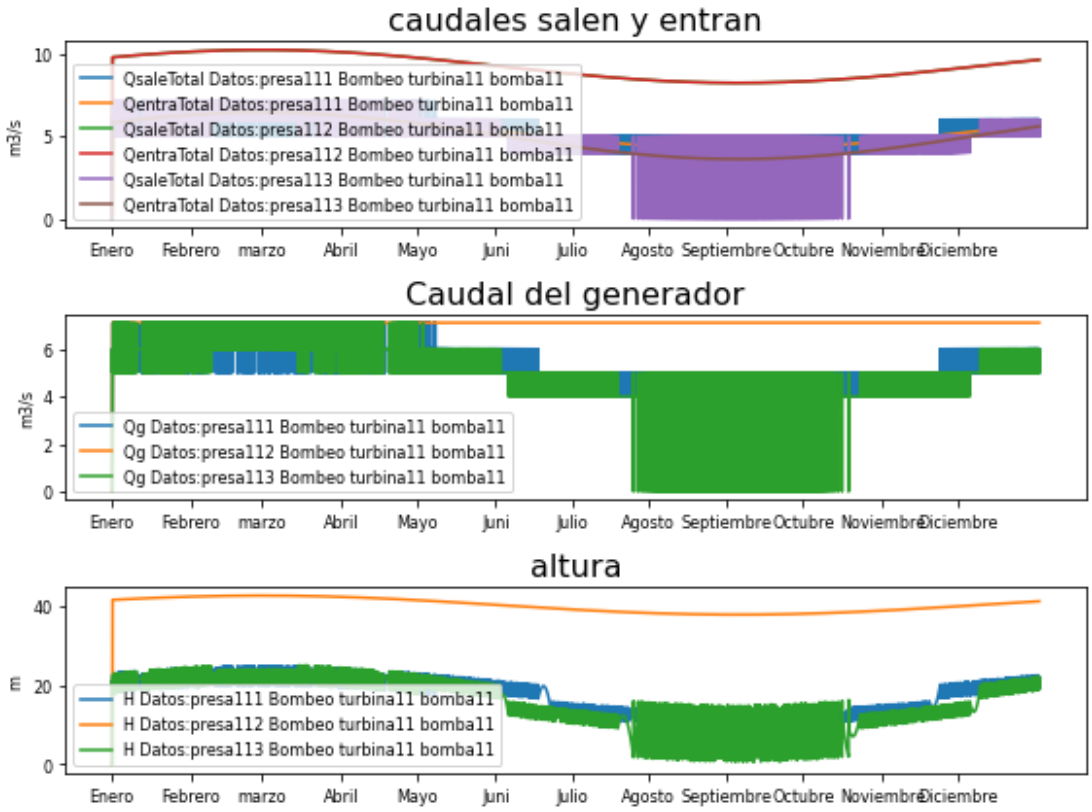
In [15]:

```
presa1.turbina=turbina11
presa2.turbina=turbina11
presa3.turbina=turbina11

presa1.bomba=bomba11
presa2.bomba=bomba11
presa3.bomba=bomba11

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio2)
presa3.indicar_Qentra_m3s(Q_rio3)

Datos1 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=Necesidades0)
Datos2 = hp.sistema_bombeo(t_total, presa=presa2,
necesidad=Necesidades0)
Datos3 = hp.sistema_bombeo(t_total, presa=presa3,
necesidad=Necesidades0)
ListaDatos=[Datos1,Datos2,Datos3]
```

Caudales

In [16]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_QentraQsale(ListaDatos)

plt.subplot(312)
hp.representar_Qgenerador(ListaDatos)

plt.subplot(313)
hp.representar_Altura(ListaDatos)

plt.tight_layout()
plt.show()
```



Caudales reducidos

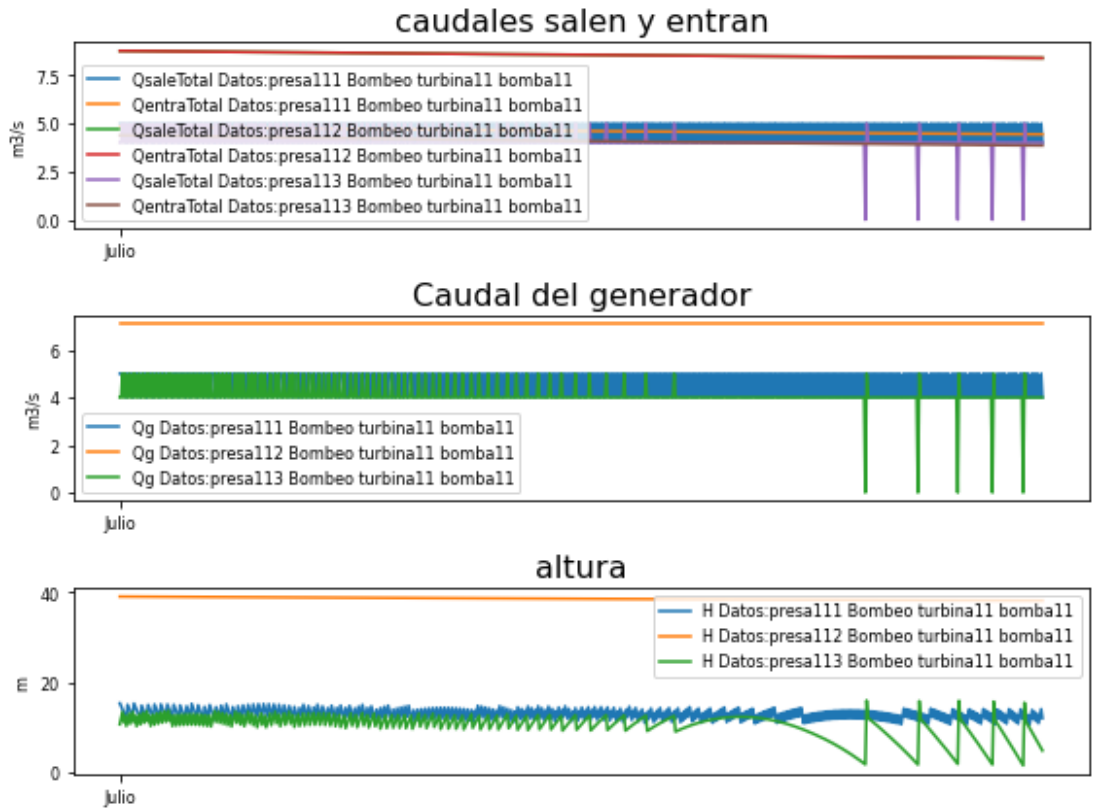
In [17]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=6
Mfin=7
```

```
tini=horaMes[Mini]
tfin=horaMes[Mfin]
```

In [18]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_QentraQsale(ListaDatos,t_total=t_total[tini:tfin],h
oratik=horaMes[Mini:Mfin] , nombretik=meses[Mini:Mfin])


plt.subplot(312)

hp.representar_Qgenerador(ListaDatos,t_total=t_total[tini:tfin],ho
ratik=horaMes[Mini:Mfin] , nombretik=meses[Mini:Mfin])

plt.subplot(313)
hp.representar_Altura(ListaDatos,t_total=t_total[tini:tfin],horati
k=horaMes[Mini:Mfin] , nombretik=meses[Mini:Mfin])

plt.tight_layout()
plt.show()
```



1.2 SEGUNDO MODELO

1. Inicializamos las centrales

In [19]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 = hp.Presa(t_total,Q_entrada=0,Nombre='presa121',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa122',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
presa3 = hp.Presa(t_total,Q_entrada=0,Nombre='presa123',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [20]:

```
def funcion_turbina_12(self=0,t=0, necesidad=0, presa=0, g=9.8,
den=998, intervalo=3600):
    var= presa.H_max - presa.H_min

    if self.H[t]<presa.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< presa.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< presa.H_max*0.8:
        self.Qg[t] = 5
    elif  self.H[t]< presa.H_max*0.9:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7.1
```

In [21]:

```
#Inicializamos las turbinas
```

```
turbina12 =
hp.Turbina(Nombre='turbina12',Qg_funcion_propia=funcion_turbina_12
)
```

In [22]:

```
presa1.turbina=turbina12
presa2.turbina=turbina12
presa3.turbina=turbina12
```

3. Inicializamos la Bomba

In [23]:

```
def funcion_Qb121(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 5;
```

In [24]:

```
#Inicializamos las turbinas
bomba121 =
hp.Bomba(Nombre='bomba121',Qb_funcion_propia=funcion_Qb121,Pb_func
ion_propia=funcion_Pb11)
```

In [25]:

```
def funcion_Qb122(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 3;
```

In [26]:

```
#Inicializamos las turbinas
bomba122 =
hp.Bomba(Nombre='bomba123',Qb_funcion_propia=funcion_Qb122,Pb_func
ion_propia=funcion_Pb11)
```

In [27]:

```
presa1.turbina=turbina12
presa2.turbina=turbina12
presa3.turbina=turbina12

presa2.bomba=bomba121
presa3.bomba=bomba122
```

4. Creamos los caudales

In [28]:

```
# caudal eresma Bernardos
Q_med1 = 5.3
Q_A1=2.08

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)
```

```
presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio1)
presa3.indicar_Qentra_m3s(Q_rio1)
```

Creamos el objeto de necesidades. aunque en este caso su valor estará practicamente vacío dado que queremos observar el efecto del caudal del generador y no otros parámetros.

4. Calcular

In [29]:

```
Datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
Datos2 = hp.sistema_bombeo(t_total, presa=presa2,
necesidad=Necesidades0)
Datos3 = hp.sistema_bombeo(t_total, presa=presa3,
necesidad=Necesidades0)

ListaDatos = hp.sistema_multiple(t_total,
listPresa=[presa1,presa2], necesidad=Necesidades0)
```

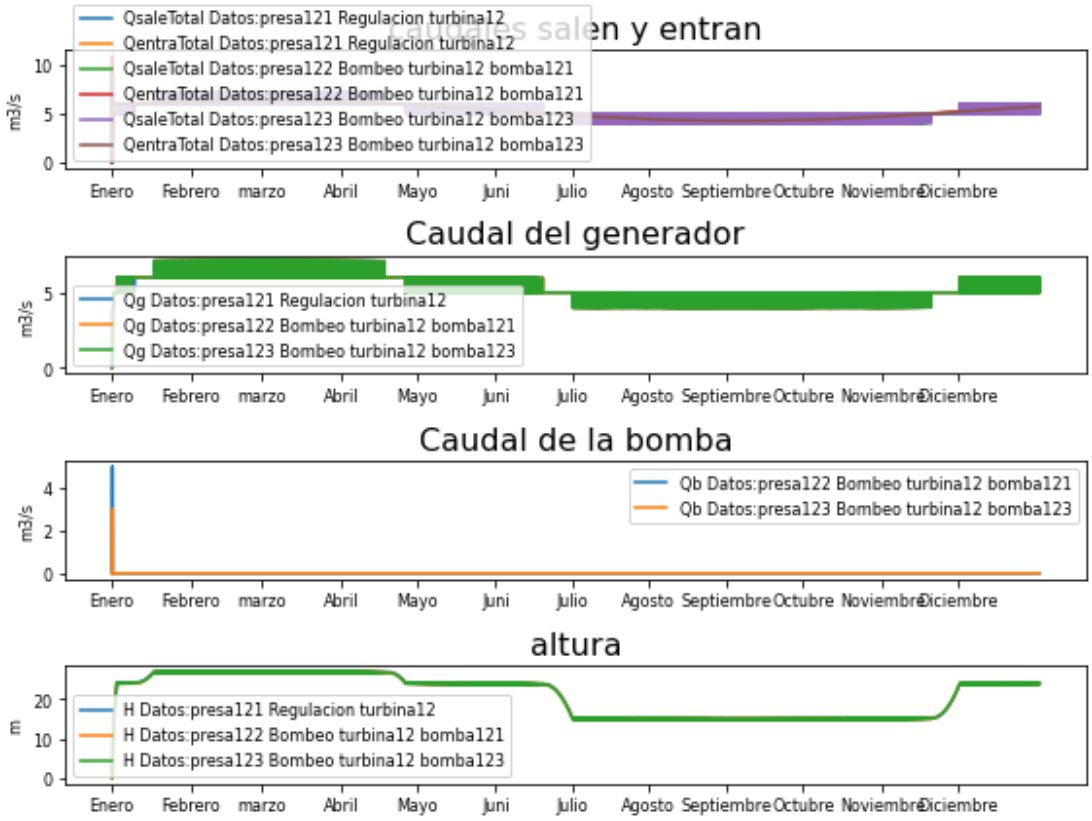
Caudales

In [30]:

```
ListaDatos=[Datos1,Datos2,Datos3]
```

In [31]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos)

plt.subplot(413)
hp.representar_Qbomba(ListaDatos)

plt.subplot(414)
hp.representar_Altura(ListaDatos)

plt.tight_layout()
plt.show()
```



Caudales reducidos

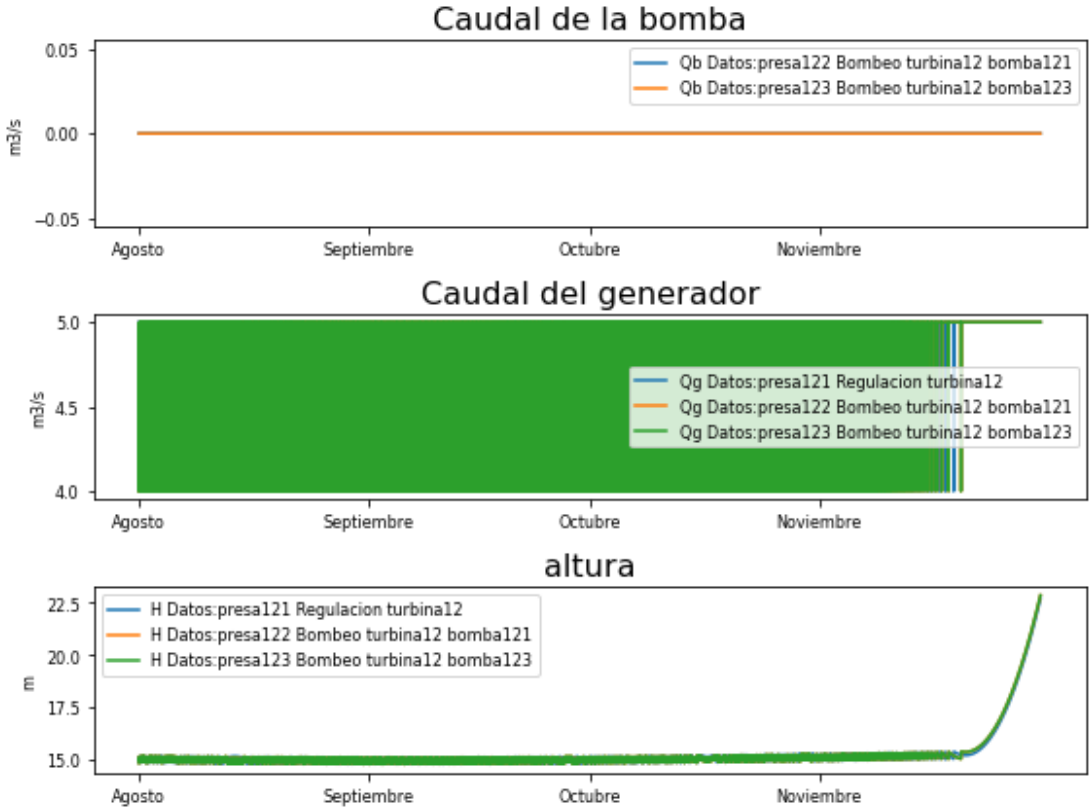
In [32]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=7
Mfin=11
tini=horaMes[Mini]
tfin=horaMes[Mfin]
```

In [33]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_Qbomba(ListaDatos, t_total=t_total[tini:tfin],
horatik=horaMes[Mini:Mfin], nombretik=meses[Mini:Mfin])


plt.subplot(312)
hp.representar_Qgenerador(ListaDatos, t_total=t_total[tini:tfin],
horatik=horaMes[Mini:Mfin], nombretik=meses[Mini:Mfin])
```

```
plt.subplot(313)
hp.representar_Altura(ListaDatos, t_total=t_total[tini:tfin],
horatik=horaMes[Mini:Mfin], nombretik=meses[Mini:Mfin])

plt.tight_layout()
plt.show()
```



1.3 SEGUNDO MODELO

1. Inicializamos las centrales

In [34]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 = hp.Presa(t_total,Q_entrada=0,Nombre='presa131',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa132',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'bombeo',H_des=35,
```

```
Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
presa3 = hp.Presa(t_total,Q_entrada=0,Nombre='presa133',

base=10000,base_absorcion=5000,Hi=2,H_max=30,H_min=5,Tipo_central=
'bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [35]:

```
def funcion_turbina_13(self=0,t=0, necesidad=0, presa=0, g=9.8,
den=998, intervalo=3600):
    var= presa.H_max - presa.H_min

    if self.H[t]<presa.H_min:
        self.Qg[t] = 0
    #elif  self.H[t]< presa.H_max/2:
    #    self.Qg[t] = 4
    elif  self.H[t]< presa.H_max*0.8:
        self.Qg[t] = 5
    elif  self.H[t]< presa.H_max*0.9:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7.1
```

In [36]:

```
#Inicializamos las turbinas
turbina13 =
hp.Turbina(Nombre='turbina13',Qg_funcion_propia=funcion_turbina_13
)
```

In [37]:

```
presa1.turbina=turbina13
presa2.turbina=turbina13
presa3.turbina=turbina13
```

3. Inicializamos la Bomba

In [38]:

```
def funcion_Qb121(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 3;
```

In [39]:

```
#Inicializamos las turbinas
```

```
bomba121 =
hp.Bomba(Nombre='bomba121',Qb_funcion_propia=funcion_Qb121,Pb_func
ion_propia=funcion_Pb11)
```

In [40]:

```
def funcion_Qb122(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 1;
```

In [41]:

```
#Inicializamos las turbinas
bomba122 =
hp.Bomba(Nombre='bomba123',Qb_funcion_propia=funcion_Qb122,Pb_func
ion_propia=funcion_Pb11)
```

In [42]:

```
presa2.bomba=bomba121
presa3.bomba=bomba122
```

4. Creamos los caudales

In [43]:

```
# caudal Eresma Bernardos
Q_med1 = 5.3
Q_A1=2.08

Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(3
1+28)*24)

presa1.indicar_Qentra_m3s(Q_rio1)
presa2.indicar_Qentra_m3s(Q_rio1)
presa3.indicar_Qentra_m3s(Q_rio1)
```

Creamos el objeto de necesidades. aunque en este caso su valor estará practicamente vacío dado que queremos observar el efecto del caudal del generador y no otros parámetros.

5. Calcular

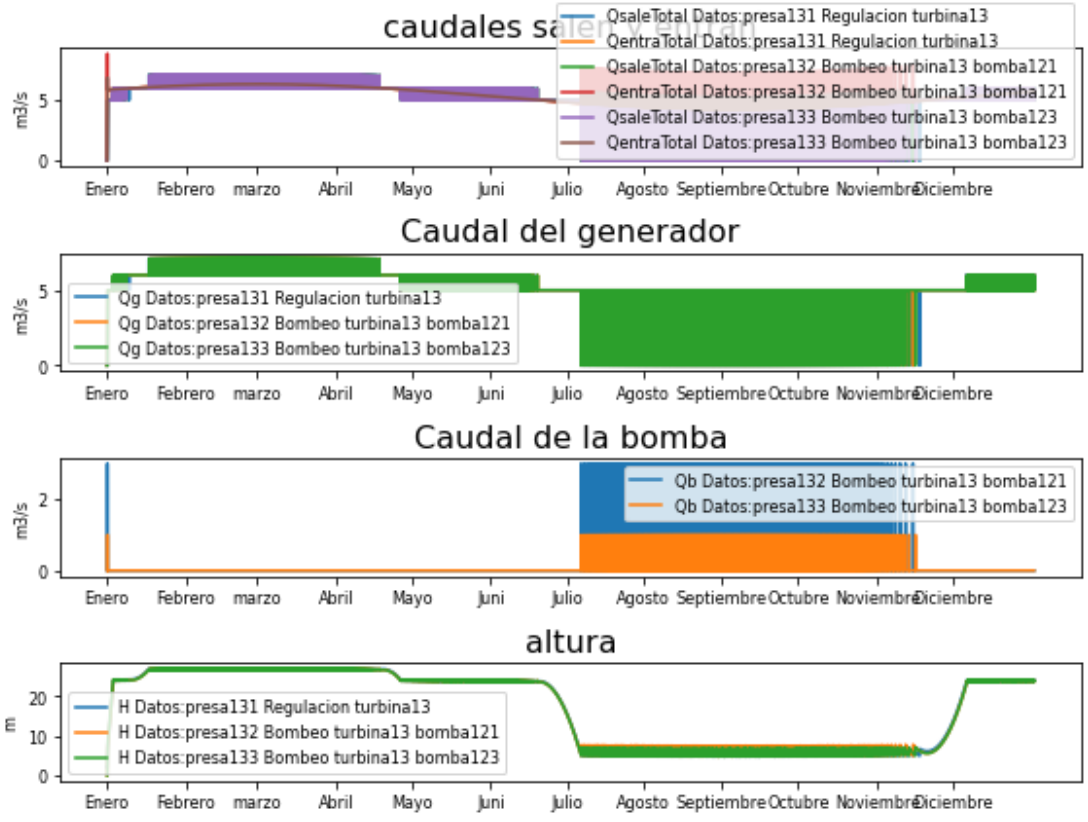
In [44]:

```
Datos1 = hp.sistema_regulacion(t_total, presa=presa1,
necesidad=Necesidades0)
Datos2 = hp.sistema_bombeo(t_total, presa=presa2,
necesidad=Necesidades0)
Datos3 = hp.sistema_bombeo(t_total, presa=presa3,
necesidad=Necesidades0)
ListaDatos=[Datos1,Datos2,Datos3]
```

Caudales

In [45]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos)

plt.subplot(413)
hp.representar_Qbomba(ListaDatos)

plt.subplot(414)
hp.representar_Altura(ListaDatos)

plt.tight_layout()
plt.show()
```



Caudales reducidos 1

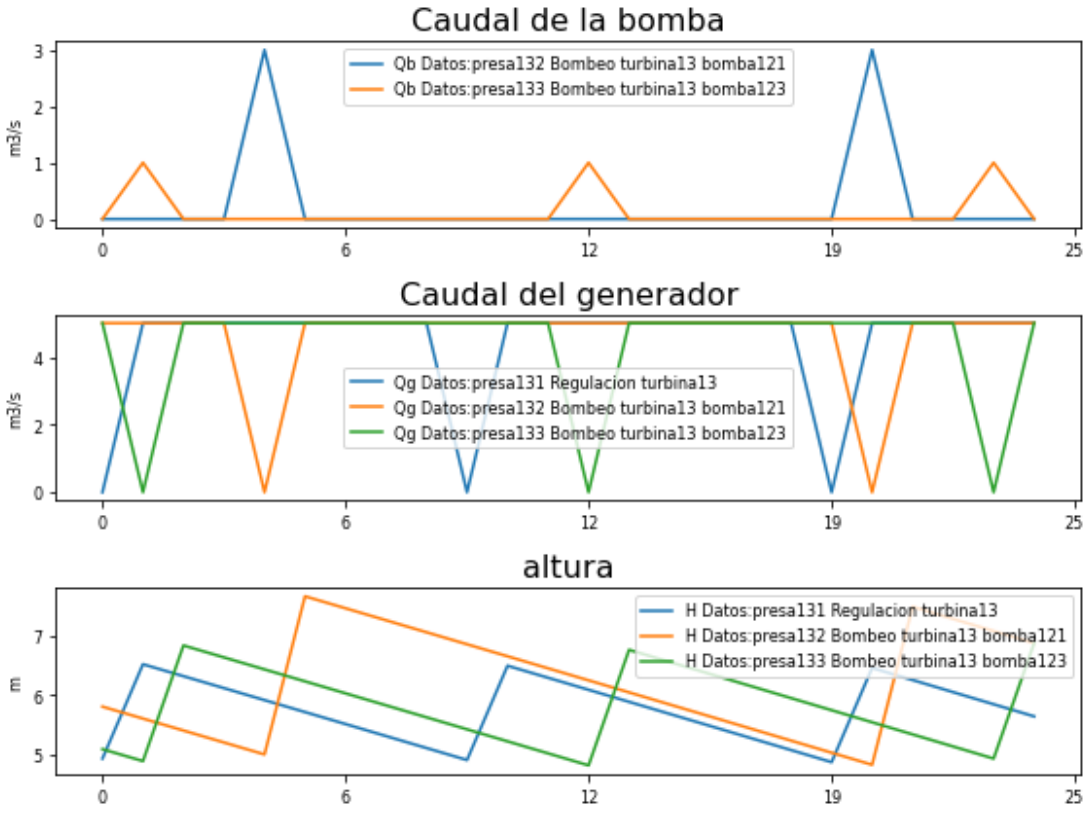
In [46]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=7
```

```
Mfin=11
tini=horaMes[Mini]
tfin=horaMes[Mfin]
tfin=horaMes[Mini]+24*1
t_representar_1 = [round(n) for n in np.linspace(tini,tfin,5)]
t_representar_2 = [round(n) for n in np.linspace(0,25,5)]
```

In [47]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_Qbomba(ListaDatos, t_total=t_total[tini:tfin],
horatik=t_representar_1, nombretik=t_representar_2)

plt.subplot(312)
hp.representar_Qgenerador(ListaDatos, t_total=t_total[tini:tfin],
horatik=t_representar_1, nombretik=t_representar_2)

plt.subplot(313)
hp.representar_Altura(ListaDatos, t_total=t_total[tini:tfin],
horatik=t_representar_1, nombretik=t_representar_2)

plt.tight_layout()
plt.show()
```

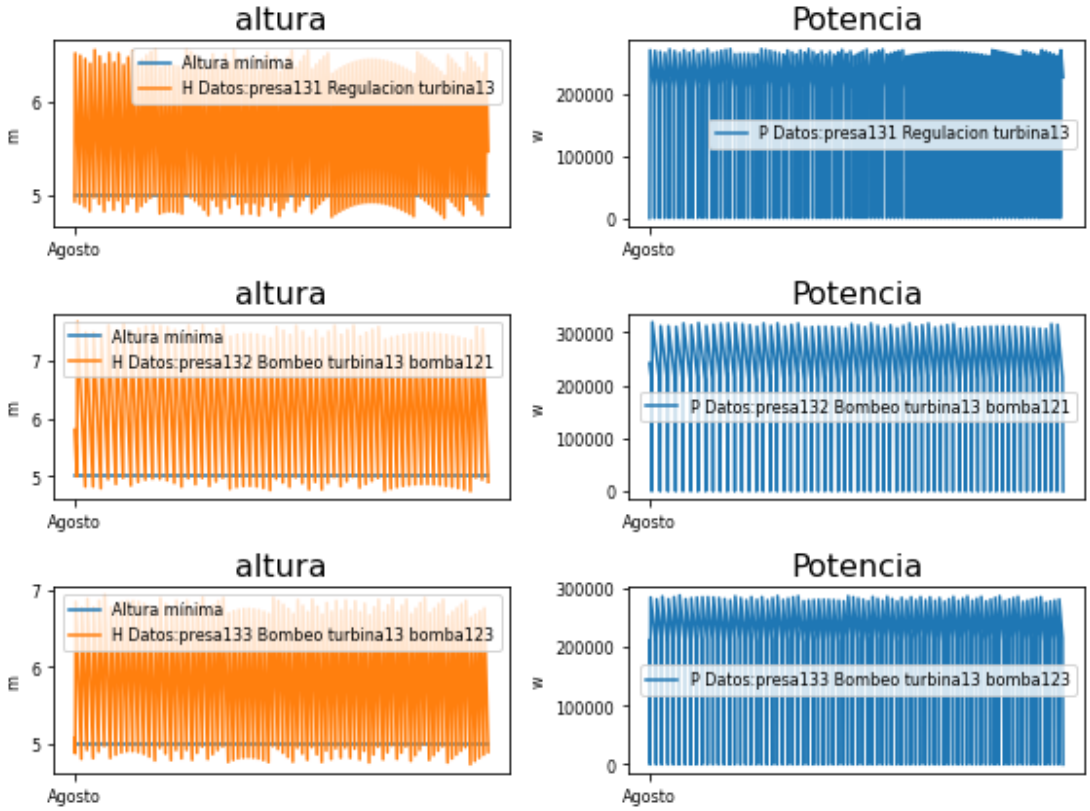


In [48]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=7
Mfin=8
tini=horaMes[Mini]
tfin=horaMes[Mfin]
```

In [49]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))
n=0
i=1
for dato in ListaDatos:
    plt.subplot(3,2,i)
    i+=2

plt.plot(t_total[tini:tfin],np.linspace(presa1.H_min,presa1.H_min,
len(t_total))[tini:tfin],label = 'Altura mínima')
    hp.representar_Altura([dato], t_total=t_total[tini:tfin],
horatik=horaMes[Mini:Mfin], nombretik=meses[Mini:Mfin])

i=2
for dato in ListaDatos:
    plt.subplot(3,2,i)
    i+=2
    hp.representar_Pgenerador([dato], t_total=t_total[tini:tfin],
horatik=horaMes[Mini:Mfin], nombretik=meses[Mini:Mfin])

plt.tight_layout()
plt.show()
```



2. CENTRAL DE BOMBEO SIMPLE

2.1 Río

1. Inicializamos las centrales

In [50]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central211',Presa_salida='',P
_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central212',Presa_salida='',P
_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,
```

```
Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa3 =
hp.Presa(t_total,Q_entrada=0,Nombre='central213',Presa_salida='',P
_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='regulable',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [51]:

```
def funcion_turbina_211(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 55
    else:
        self.Qg[t] = 60


def funcion_turbina_212(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
```

```
        self.Qg[t] = 125
```

In [52]:

```
#Inicializamos las turbinas
turbina211 =
hp.Turbina(Nombre='turbina211',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_211)
turbina212 =
hp.Turbina(Nombre='turbina212',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_212)
```

3. Inicializamos la Bomba

In [53]:

```
def funcion_Pb211(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    area=3
    if self.H[t]<0.9:
        self.Pb[t] =
den*g*self.Qb[t]*(self.H[t]+(self.Qb[t]/area)**2/(2*g)*presa.bomba
.alpha)/presa.bomba.r
```

In [54]:

```
def funcion_Qb211(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t]=5
```

In [55]:

```
#Inicializamos las turbinas
bomba211 =
hp.Bomba(Nombre='bomba211',Qb_funcion_propia=funcion_Qb211,Pb_func
ion_propia=funcion_Pb211)
```

4. Creamos los caudales

In [56]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [57]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]
```

```
Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
Q_rioP2=hm.poly_odd(t_total,hora,caudal2,horas_año)

presa1.indicar_Qentra_m3s(Q_rioP1*0.5)
presa2.indicar_Qentra_m3s(Q_rioP2)
```

5. Calcular

In [58]:

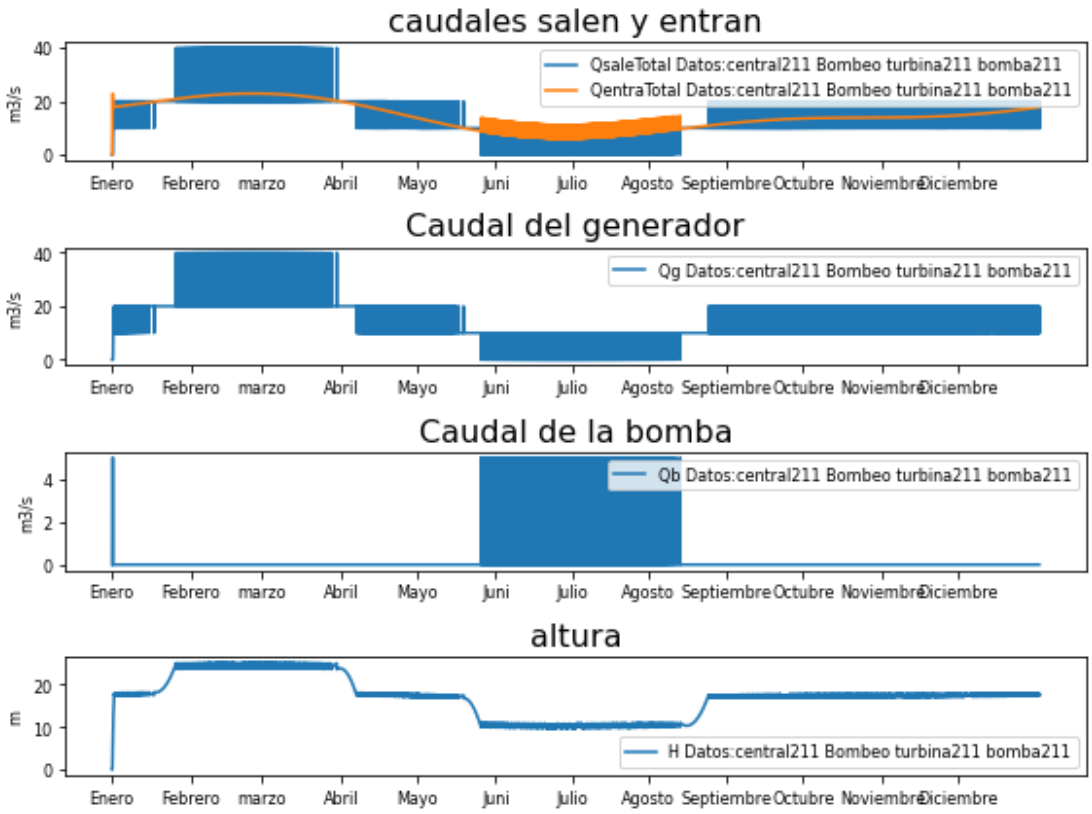
```
presa1.turbina=turbina211
presa2.turbina=turbina212
presa3.turbina=turbina211

presa1.bomba=bomba211
presa2.bomba=bomba211

presa1.indicar_Qentra_m3s(Q_rioP1*0.4)
presa2.indicar_Qentra_m3s(Q_rioP2*0.4)
presa3.indicar_Qentra_m3s(Q_rioP1*0.4)

datos1 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=Necesidades0)
datos2 = hp.sistema_regulacion(t_total, presa=presa3,
necesidad=Necesidades0)
#datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
ListaDatos=[datos1]
ListaPresa=[presa1]

ListaDatos1=[datos1,datos2]
ListaPresa1=[presa1,presa3]
```

Caudales

In [59]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos)

plt.subplot(413)
hp.representar_Qbomba(ListaDatos)

plt.subplot(414)
hp.representar_Altura(ListaDatos)
```

```
plt.tight_layout()
plt.show()
```



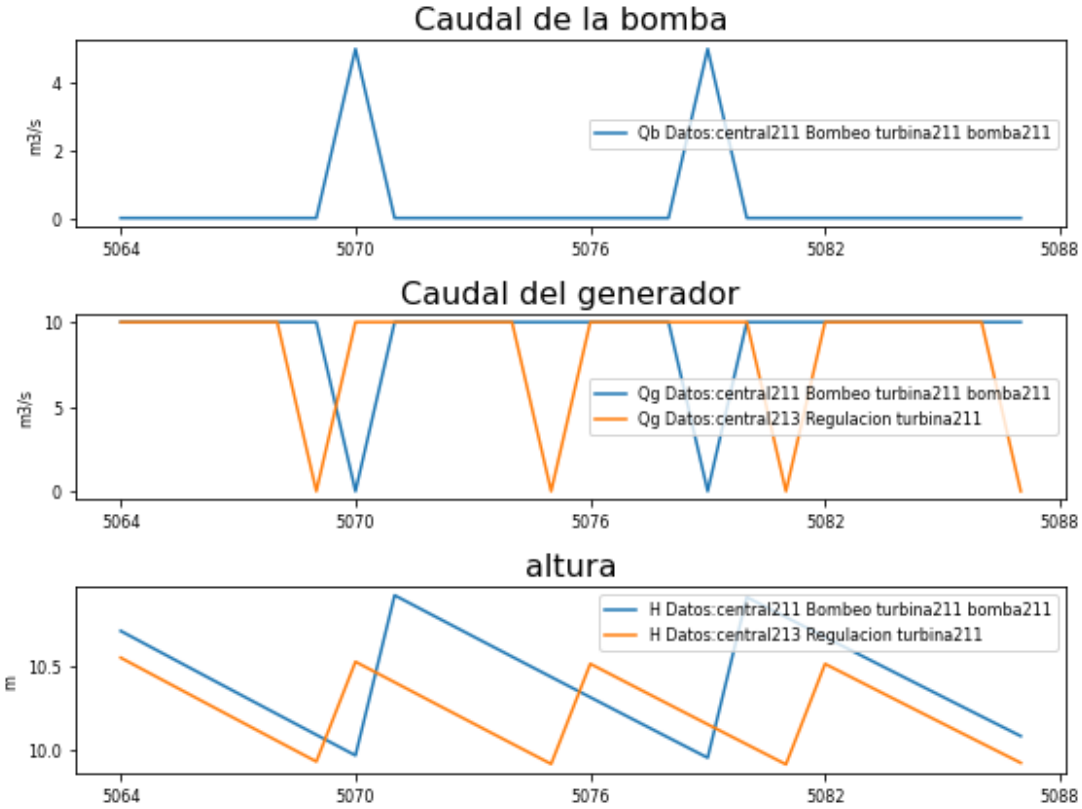
In [60]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=7
Mfin=11
tini=horaMes[Mini]
tfin=horaMes[Mfin]
tfin=horaMes[Mini]+24*1
t_representar = [round(n) for n in np.linspace(tini,tfin,5)]
```

In [61]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(311)
hp.representar_Qbomba(ListaDatos1, t_total=t_total[tini:tfin],
horatik=t_representar, nombretik=t_representar)

plt.subplot(312)
hp.representar_Qgenerador(ListaDatos1, t_total=t_total[tini:tfin],
horatik=t_representar, nombretik=t_representar)
```

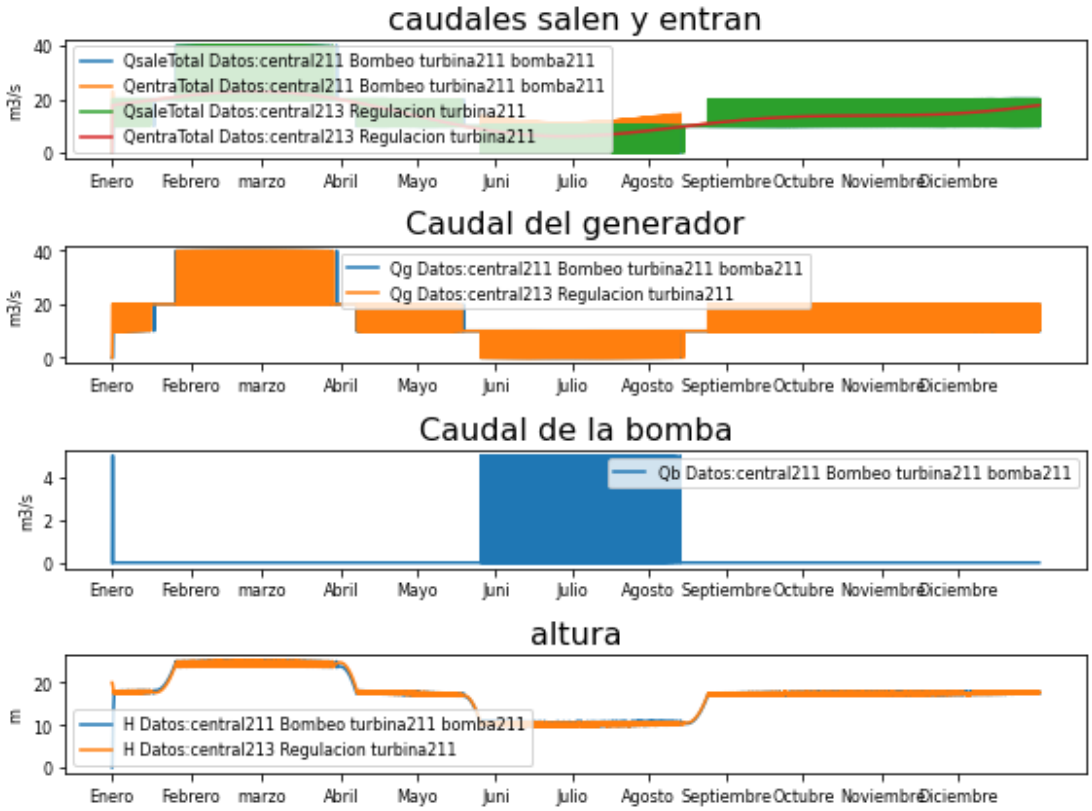
```
plt.subplot(313)
hp.representar_Altura(ListaDatos1, t_total=t_total[tini:tfin],
horatik=t_representar, nombretik=t_representar)

plt.tight_layout()
plt.show()
```



Comparación bombeo y regulable

In [62]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos1)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos1)

plt.subplot(413)
hp.representar_Qbomba(ListaDatos1)

plt.subplot(414)
hp.representar_Altura(ListaDatos1)

plt.tight_layout()
plt.show()
```



2.2 LLUVIA

1. Inicializamos las centrales

In [63]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central211',Presa_salida='',P
_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central211',Presa_salida='',P
_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [64]:

```
def funcion_turbina_211(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 55
    else:
        self.Qg[t] = 60

def funcion_turbina_212(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [65]:

```
#Inicializamos las turbinas
turbina211 =
hp.Turbina(Nombre='turbina211',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_211)
turbina212 =
hp.Turbina(Nombre='turbina212',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_212)
```

3. Inicializamos la Bomba

In [66]:

```
def funcion_Pb211(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    area=3
    self.Pb[t] =
den*g*self.Qb[t]*(self.H[t]+(self.Qb[t]/area)**2/(2*g)*presa.bomba
.alpha)/presa.bomba.r
```

In [67]:

```
def funcion_Qb211(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t]=5
```

In [68]:

```
#Inicializamos las turbinas
bomba211 =
hp.Bomba(Nombre='bomba211',Qb_funcion_propia=funcion_Qb211,Pb_func
ion_propia=funcion_Pb211)
```

4. Creamos los caudales

In [69]:

```
Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Valladolid')
Q_lluvia1 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,

area_abastecida=presa1.base,#presa1.base_absorcion,

unidades_preci='mm',unidades_area='m2')

Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Zamora')
Q_lluvia2 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,
                                   area_abastecida=presa2.base,

unidades_preci='mm',unidades_area='m2')
```

Introducimos las lluvias

In [70]:

```
presa1.indicar_Qentra_m3s(Q_lluvia1)
presa2.indicar_Qentra_m3s(Q_lluvia2)
```

5. Calcular

In [71]:

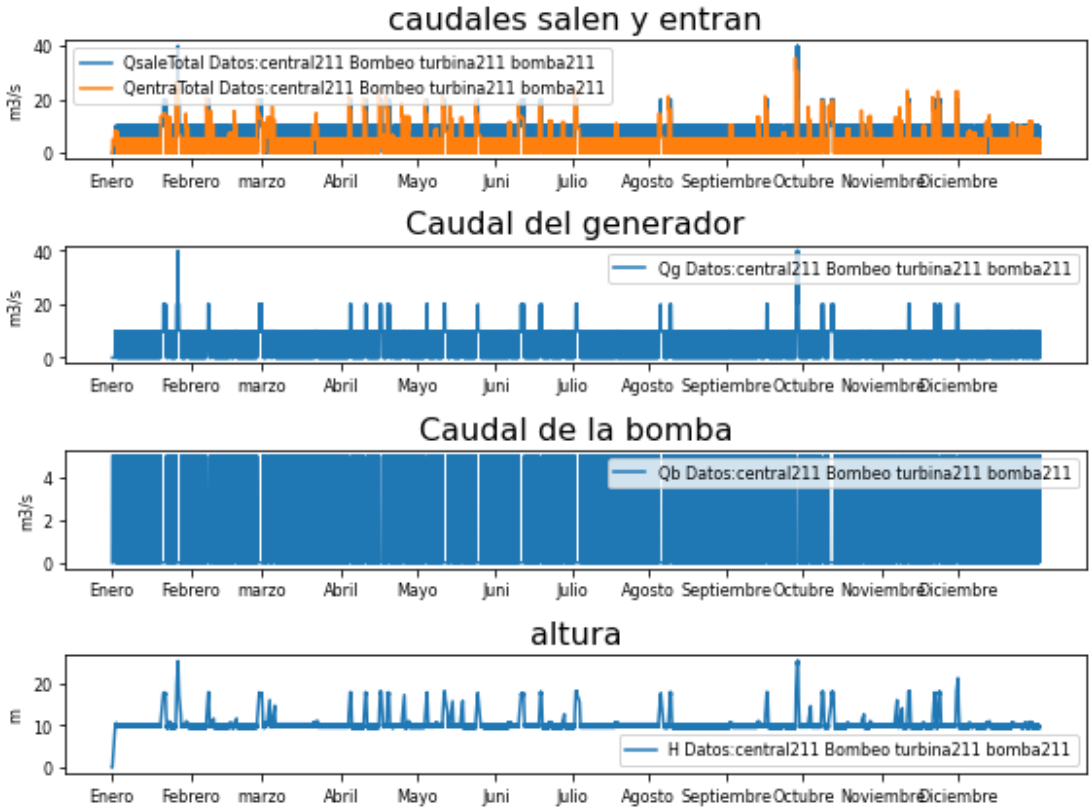
```
presa1.turbina=turbina211
presa2.turbina=turbina212
```

```
presa1.bomba=bomba211
presa2.bomba=bomba211

datos1 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=Necesidades0)
#datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
ListaDatos=[datos1]
ListaPresa=[presa1]
```

Caudales

In [72]:

```
plt.figure(figsize=(WIDE, HEIGHT*3))

plt.subplot(411)
hp.representar_QentraQsale(ListaDatos)

plt.subplot(412)
hp.representar_Qgenerador(ListaDatos)

plt.subplot(413)
hp.representar_Qbomba(ListaDatos)

plt.subplot(414)
hp.representar_Altura(ListaDatos)

plt.tight_layout()
plt.show()
```



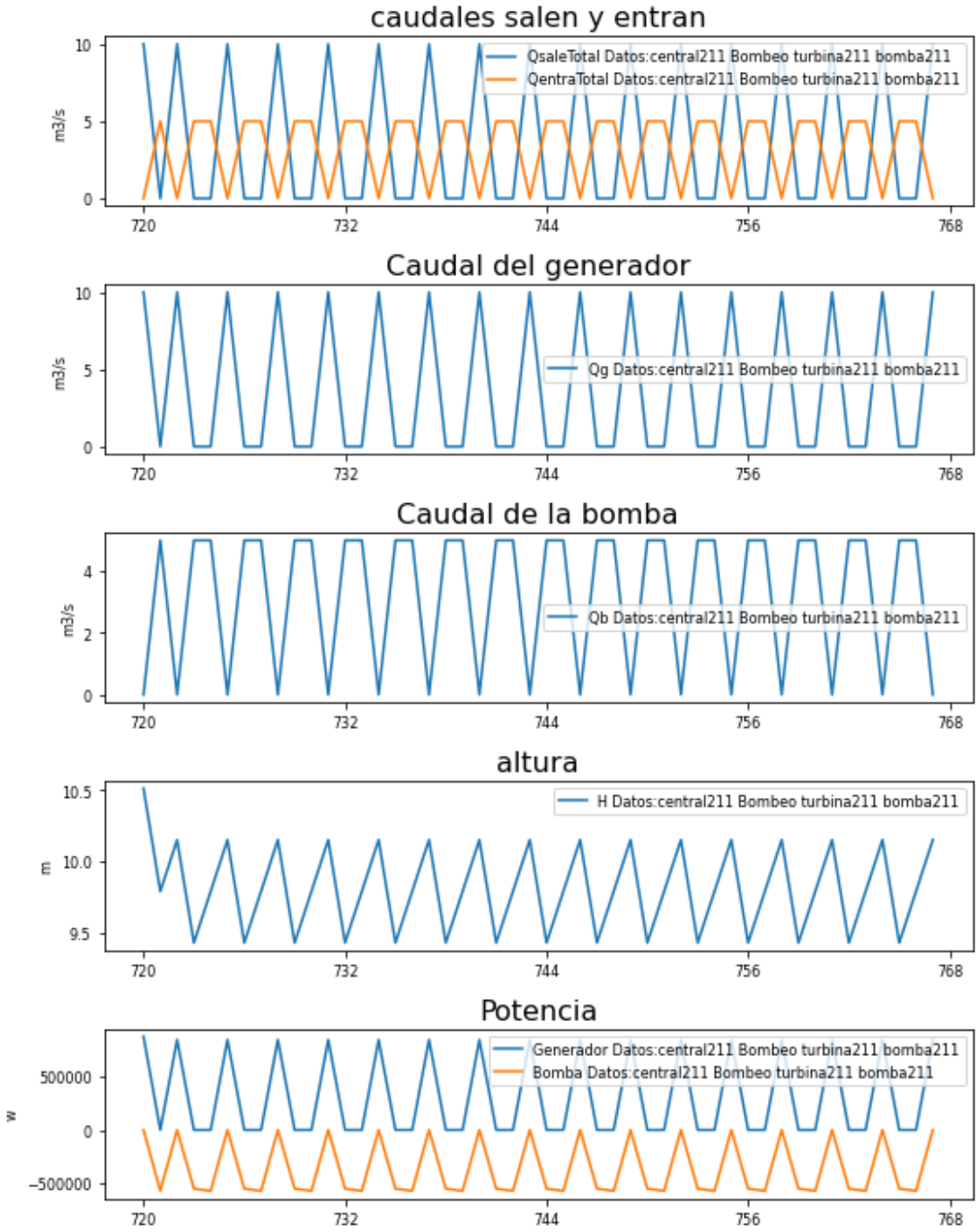
Caudales reducidos 1

In [73]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
Mini=1
Mfin=1
tini=horaMes[Mini]-24
tfin=horaMes[Mfin]
tfin=horaMes[Mini]+24
```

In [74]:

```
t_representar = [round(n) for n in np.linspace(tini,tfin,5)]
hp.representar_REVERSIBLE(ListaDatos, t_total=t_total[tini:tfin],
horatik=t_representar, nombretik=t_representar)
```



2.3 AGUA

1. Inicializamos las centrales

In [75]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
```

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='central231',Presa_salida='',P
_Qg=1,

base=20000,base_absorcion=20000,Hi=20,H_max=35,H_min=15,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total,Q_entrada=0,Nombre='central231',Presa_salida='',P
_Qg=1,

base=20000,base_absorcion=20000,Hi=20,H_max=35,H_min=15,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [76]:

```
def funcion_turbina_231(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 45
    else:
        self.Qg[t] = 60

def funcion_turbina_233(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 5
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.9:
```

```
        self.Qg[t] = 25
    else:
        self.Qg[t] = 30

def funcion_turbina_232(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [77]:

```
#Inicializamos las turbinas
turbina231 =
hp.Turbina(Nombre='turbina231',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_231)
turbina232 =
hp.Turbina(Nombre='turbina232',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_232)
```

In [78]:

```
presa1.turbina=turbina231
presa2.turbina=turbina232
```

In [79]:

```
def funcion_turbina_233(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 8
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 9
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 11
```

```
    else:
        self.Qg[t] = 12
```

In [80]:

```
#Inicializamos las turbinas
turbina233 =
hp.Turbina(Nombre='turbina233',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_233)
```

In [81]:

```
def funcion_turbina_234(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 4
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 5
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 6
    else:
        self.Qg[t] = 7
```

In [82]:

```
#Inicializamos las turbinas
turbina234 =
hp.Turbina(Nombre='turbina234',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_234)
```

3. Inicializamos la Bomba

In [83]:

```
def funcion_Pb231(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    area=3
    if self.Qb[t]>0:
        self.Pb[t] =
den*g*self.Qb[t]*(self.H[t]+(self.Qb[t]/area)**2/(2*g)*presa.bomba
.alpha)/presa.bomba.r
```

In [84]:

```
def funcion_Qb231(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t]=5
```

In [85]:

```
#Inicializamos las turbinas
bomba231 =
hp.Bomba(Nombre='bomba231',Qb_funcion_propia=funcion_Qb231,Pb_func
ion_propia=funcion_Pb231)
```

In [86]:

```
presa1.bomba=bomba231
presa2.bomba=bomba231
```

4. Creamos los caudales

In [87]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [88]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
#Q_rioP2=hm.poly_odd(t_total,hora,caudal2,horas_año)
```

In [89]:

```
# caudal Támega
lista_meses=[1,4,5,9,11]
hora2=hm.lista_mes_hora(lista_meses)
Q_med3 = 9.23

Q_max3 =10.63
Q_min3 =8.63

Q_A3=Q_max3-Q_min3
caudal3=[Q_max3,Q_med3,Q_min3*0.98 ,Q_min3*0.99,Q_med3  ]

Q_rioP3=hm.poly_odd(t_total,hora2,caudal3,horas_año)
# caudal Carrión

Q_med4 = 5.03
Q_max4 =6.89
Q_min4 =4.06
Q_A4=Q_max4-Q_min4
caudal4=[Q_max4,Q_med4,Q_min4*1.02,Q_min4*1.03,Q_med4 ]

Q_rioP4=hm.poly_odd(t_total,hora2,caudal4,horas_año)
```

In [90]:

```
presa1.indicar_Qentra_m3s(Q_rioP1*0.4)
```

```
presa1.indicar_Qentra_m3s(Q_rioP1*1-10)
#presa2.indicar_Qentra_m3s(Q_rioP2*0.4)
```

5. Creamos las necesidades

Población 2021 Instituto Nacional de Estadística:

In [91]:

```
Poblacion_Bercero = 191
Poblacion_Toro = 8532
Poblacion_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en España por persona

In [92]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [93]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [94]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [95]:

```
necesidad231=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad232=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad233=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
```

6. Calcular

In [96]:

```
presa1.turbina=turbina233
#presa2.turbina=turbina364

presa1.indicar_Qentra_m3s(Q_rioP3)
#presa2.indicar_Qentra_m3s(Q_rioP4)
presa1.Nombre='central231'
datos1 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad231)
presa1.Nombre='central232'
datos2 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad232)
presa1.Nombre='central233'
datos3 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad233)

ListaDatos2=[datos1,datos2,datos3]
```

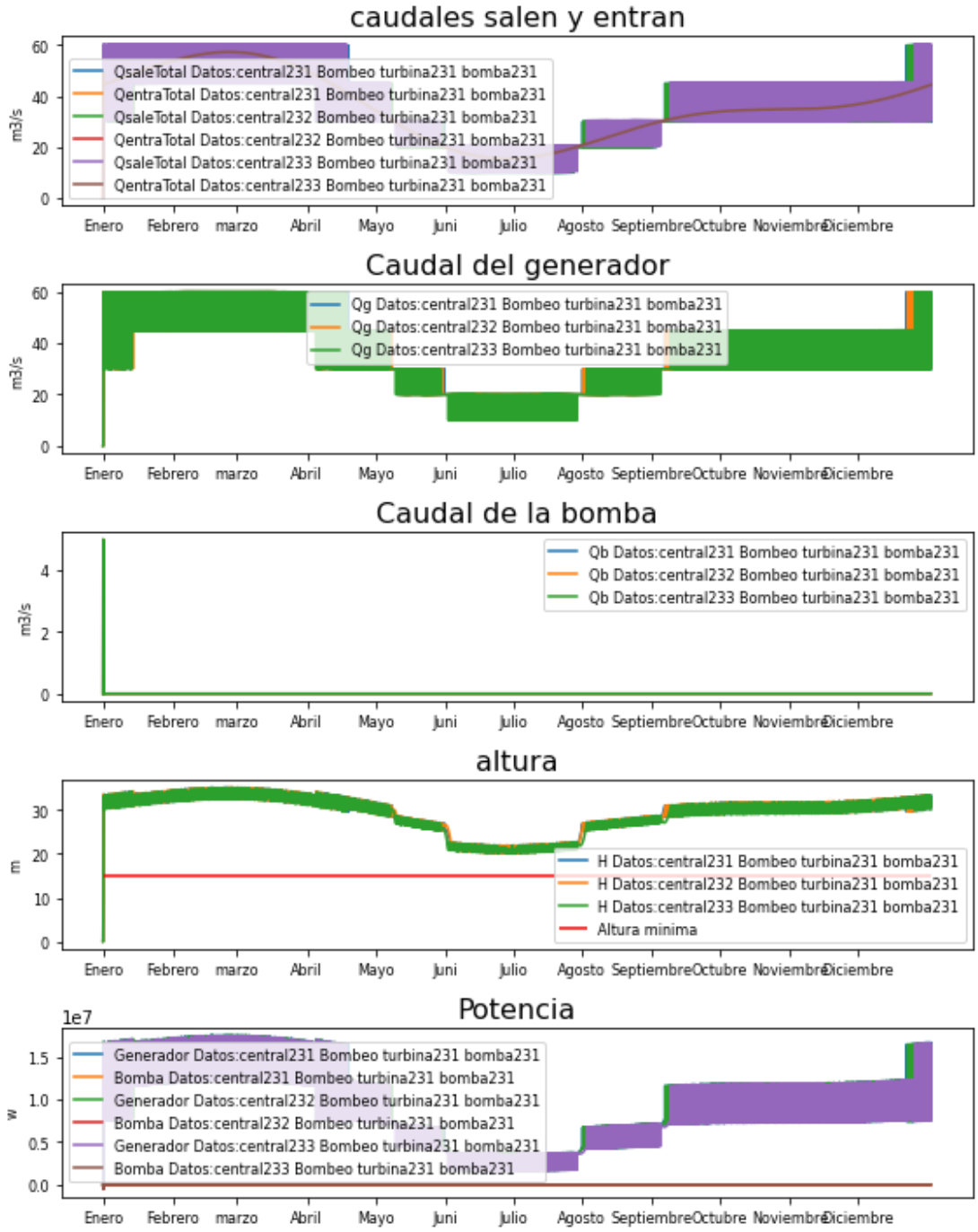
In [97]:

```
presa1.turbina=turbina231
#presa2.turbina=turbina362

presa1.indicar_Qentra_m3s(Q_rioP1)
#presa2.indicar_Qentra_m3s(Q_rioP2)
presa1.Nombre='central231'
datos1 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad231)
presa1.Nombre='central232'
datos2 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad232)
presa1.Nombre='central233'
datos3 = hp.sistema_bombeo(t_total, presa=presa1,
necesidad=necesidad233)

ListaDatos1=[datos1,datos2,datos3]
```

Gráfica de ríos grandes simple

In [98]:

```
hp.representar_REVERSIBLE(ListaDatos1,LH_Altura={'Altura
minima':presa1.H_min})
```



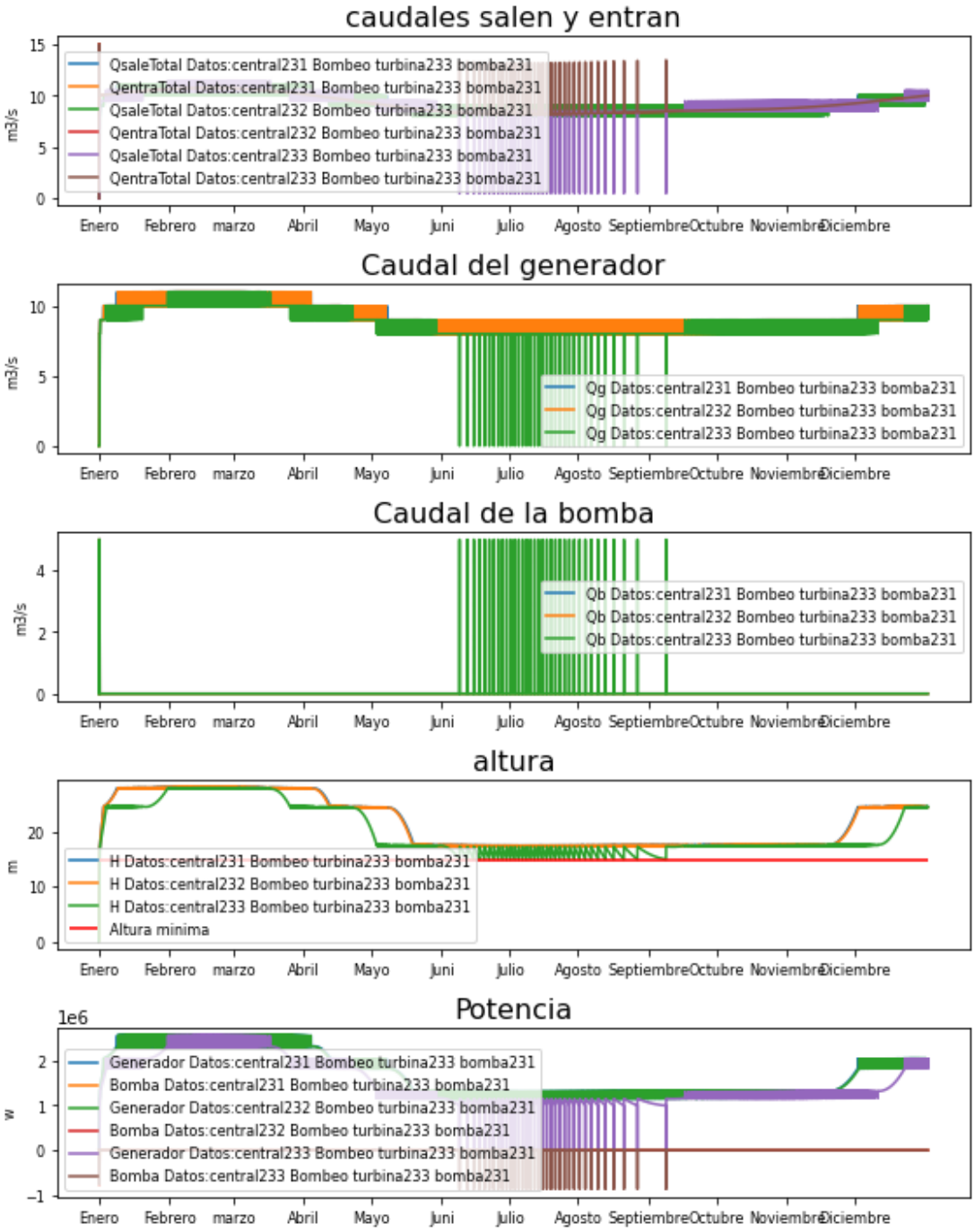
Gráfica de ríos pequeños simple

In [99]:

```
ListaDatos = ListaDatos2
```

In [100]:

```
hp.representar_REVERSIBLE(ListaDatos,LH_Altura={'Altura
minima':presa1.H_min})
```



2.4 POTENCIA

In [101]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia

tini=0
```

```
tfin=7*horas_dia
t_total_semana= np.arange(tfin-tini)
```

In [102]:

```
from sympy.solvers import solve
from sympy import Symbol
```

In [103]:

```
from sympy.abc import x
```

1. Inicializamos las centrales

In [104]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total_semana,Q_entrada=0,Nombre='central241',Presa_sali
da='',P_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
presa2 =
hp.Presa(t_total_semana,Q_entrada=0,Nombre='central241',Presa_sali
da='',P_Qg=1,

base=50000,base_absorcion=10000,Hi=20,H_max=35,H_min=10,Tipo_centr
al='Bombeo',H_des=45,

Qdesbordado=50.0,aliviadero=hp.Aliviadero(Hentrada=35,Area=20))
```

2. Inicializamos la turbina

In [105]:

```
def funcion_turbina_241(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 55
    else:
        self.Qg[t] = 60
```

```
def funcion_turbina_242(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [106]:

```
turbina241 =
hp.Turbina(Nombre='turbina241',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_241)
turbina242 =
hp.Turbina(Nombre='turbina242',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_242)
```

3. Inicializamos la Bomba

In [107]:

```
def funcion_Qb24(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    #print(t)
    self.selector_Pb(t, presa=presa, necesidad=necesidad, g=9.8,
den=998)
    if self.Pb[t]>0:

a,b,c=solve([den*g*x*(self.H[t]+(x/presa.bomba.area)**2/(2*g)*pres
a.bomba.alpha)-self.Pb[t]*presa.bomba.r],x)

        res = float('.'.join(str(a) for a in a))
        if res > 0:
            self.Qb[t]=res
        else:
            res = float('.'.join(str(b) for b in b))
            if 0<res:
                self.Qb[t]=res
            else:
```

```
            res = float('.'.join(str(c) for c in c))
            if 0<res:
                self.Qb[t]=res
```

In [108]:

```
def funcion_Pb24(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    if necesidad.Pgn_ms[t]<0:
        self.Pb[t] = necesidad.Pgn_ms[t]*-1;
```

In [109]:

```
bomba24 =
hp.Bomba(Nombre='bomba24',Qb_funcion_propia=funcion_Qb24,Pb_funcio
n_propia=funcion_Pb24)
```

4. Creamos los caudales de entrada

In [110]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [111]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total_semana,hora,caudal1,horas_año)
Q_rioP2 = hm.poly_odd(t_total_semana,hora,caudal2,horas_año)
```

5. Creamos las necesidades

In [112]:

```
hora= [4,9,13,17,22]
potencia=[4.9,6.5,6.6,5.8,6.9]
```

In [113]:

```
t_sem = np.linspace(0, 24, num=25)
demanda_ElHierro = hm.poly_odd(t_total_semana,hora,potencia,24)
```

In [114]:

```
demanda_ElHierro_centrado =demanda_ElHierro-
np.mean(demanda_ElHierro)
```

Creamos los objetos necesidades

In [115]:

```
necesidad_potencia_1=
hp.Necesidades(t_total_semana,Nombre='Necesidad_ElHierro_diario',P
otencia_necesaria=demanda_ElHierro_centrado)
```

In [116]:

```
poblacion =11154
necesidad242=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P100',Potencia_nec
esaria=demanda_ElHierro_centrado/poblacion * 100)
necesidad243=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P1000',Potencia_ne
cesaria=demanda_ElHierro_centrado/poblacion * 1000)
necesidad244=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P10000',Potencia_n
ecesaria=demanda_ElHierro_centrado/poblacion * 10000)
necesidad245=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P50000',Potencia_n
ecesaria=demanda_ElHierro_centrado/poblacion * 50000)
necesidad246=
hp.Necesidades(t_total_semana,Nombre='Necesidad_P100000',Potencia_
necesaria=demanda_ElHierro_centrado/poblacion * 100000)
```

6. Calcular

In [117]:

```
presa1.turbina=turbina241
presa2.turbina=turbina241

presa1.bomba=bomba24
presa2.bomba=bomba24

presa1.indicar_Qentra_m3s(Q_rioP1)
presa2.indicar_Qentra_m3s(Q_rioP1*0.5)

datos1 = hp.sistema_bombeo(t_total_semana, presa=presa1,
necesidad=necesidad_potencia_1)
datos2 = hp.sistema_bombeo(t_total_semana, presa=presa2,
necesidad=necesidad_potencia_1)
#datos2 = hp.sistema_regulacion(t_total, presa=presa2,
necesidad=Necesidades0)
```

In [118]:

```
ListaDatos1=[datos1]
ListaPresa1=[presa1]

ListaDatos2=[datos2]
ListaPresa2=[presa2]
```

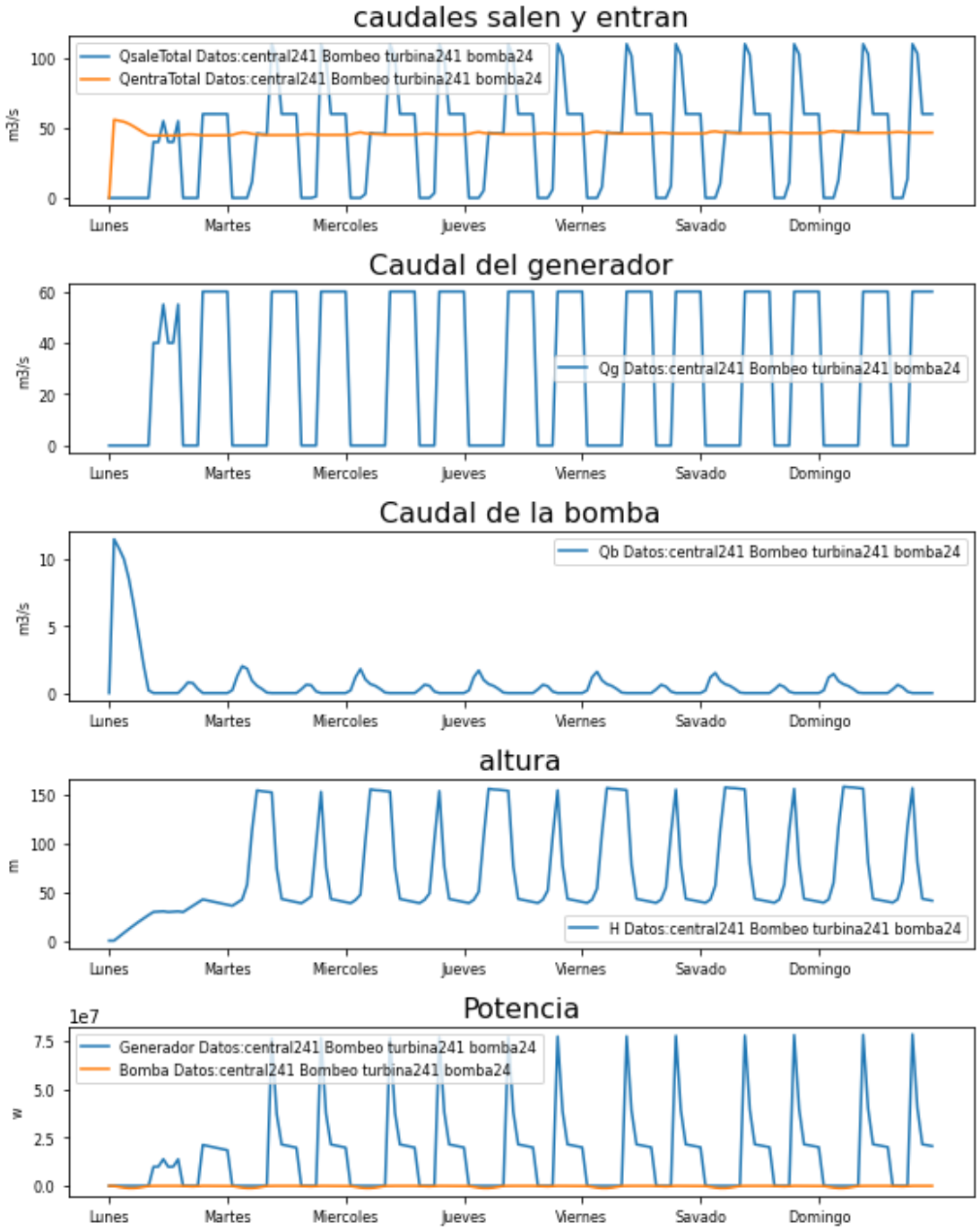
In [119]:

```
presa1.turbina=turbina241
presa2.turbina=turbina241

presa1.bomba=bomba24
presa2.bomba=bomba24

presa1.indicar_Qentra_m3s(Q_rioP1)
presa2.indicar_Qentra_m3s(Q_rioP1*0.5)

datos1 = hp.sistema_bombeo(t_total_semana, presa=presa1,
necesidad=necesidad242)
datos2 = hp.sistema_bombeo(t_total_semana, presa=presa2,
necesidad=necesidad243)
datos3 = hp.sistema_bombeo(t_total_semana, presa=presa2,
necesidad=necesidad244)
datos4 = hp.sistema_bombeo(t_total_semana, presa=presa2,
necesidad=necesidad245)
datos5 = hp.sistema_bombeo(t_total_semana, presa=presa2,
necesidad=necesidad246)
ListaDatos3=[datos1,datos2,datos3,datos4,datos5]
```

Caudales

In [120]:

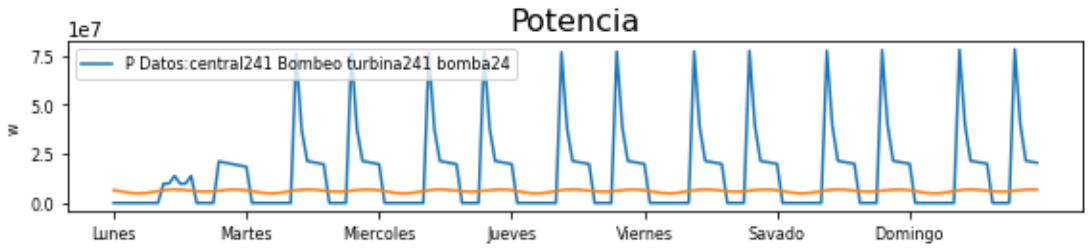
```
ListaDatos = ListaDatos1
hp.representar_REVERSIBLE(ListaDatos, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
```



In [121]:

```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos1, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)
```

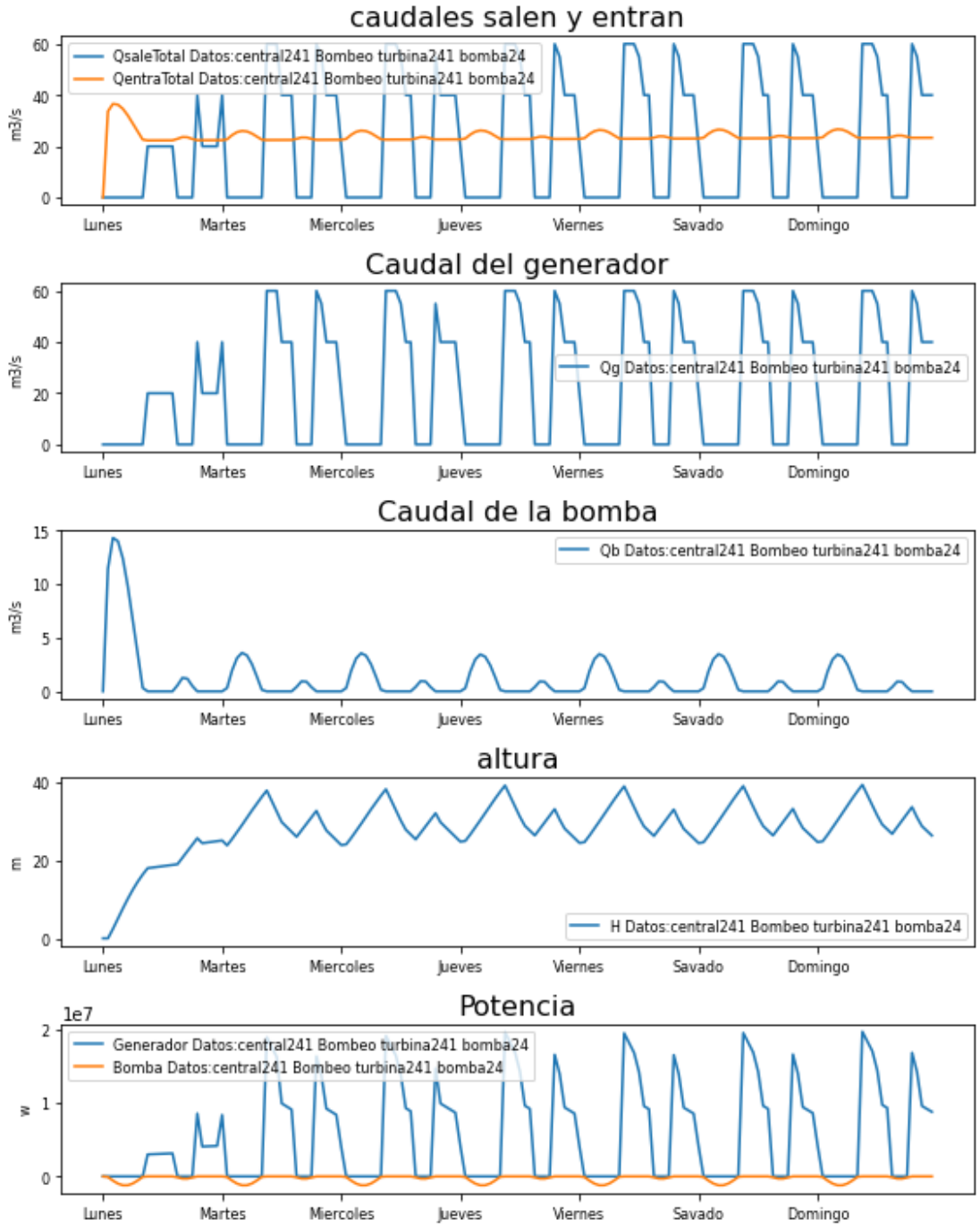
```
plt.tight_layout()
plt.show()
```



Caudales $(Q_e \times 0.5)$

In [122]:

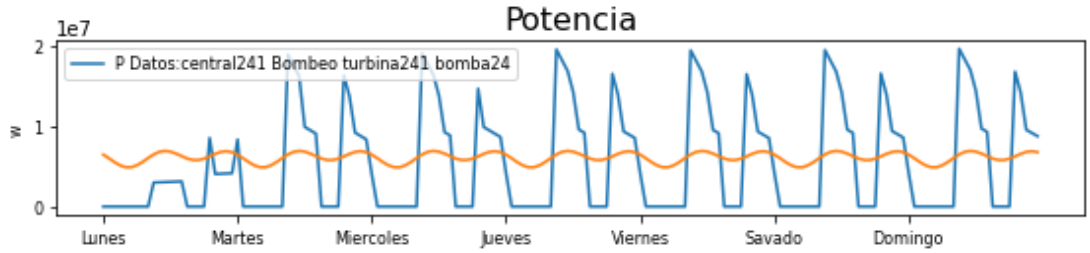
```
hp.representar_REVERSIBLE(ListaDatos2, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
```



In [123]:

```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos2, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)
```

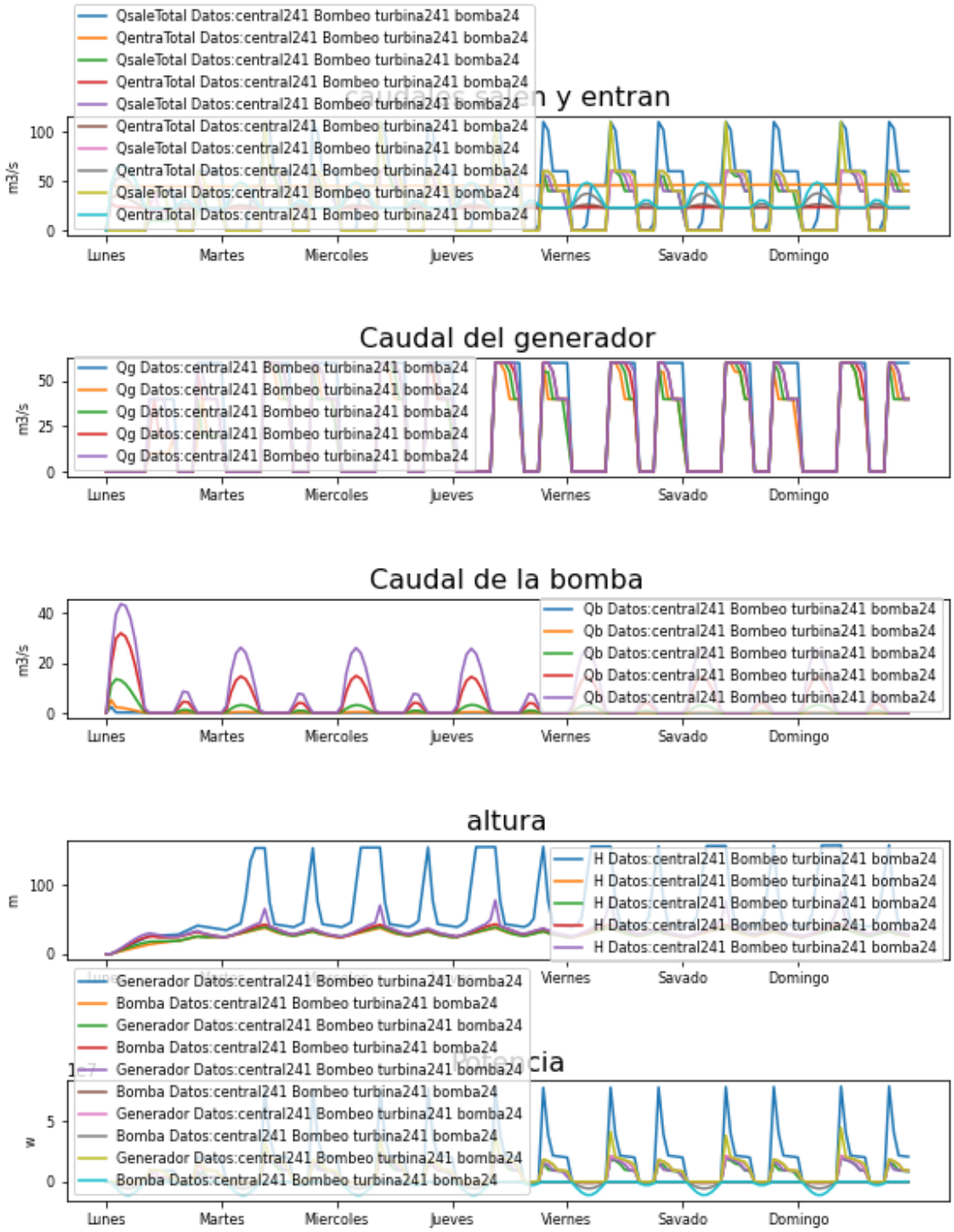
```
plt.tight_layout()
plt.show()
```



Con demandas distintas

In [124]:

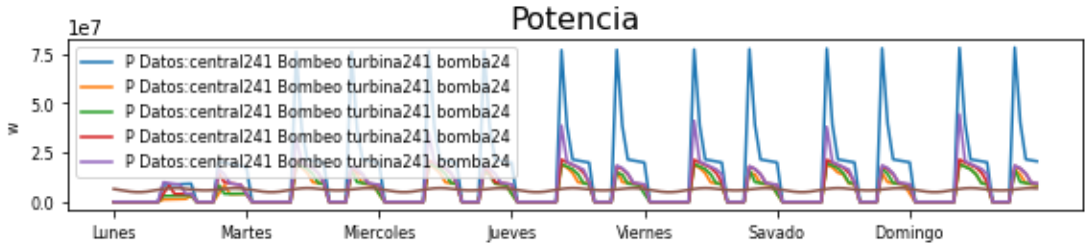
```
hp.representar_REVERSIBLE(ListaDatos3, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
```



In [125]:

```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos3, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)
```

```
plt.tight_layout()
plt.show()
```



3. CENTRAL DE BOMBEO MÚLTIPLE

3.1 SISTEMA SIMPLE

1. Inicializamos las centrales

In [126]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='presa311',Presa_salida='presa
312',Presa_de_absorcion='presa312',

base=50000,base_absorcion=50000,Hi=2,H_max=30,H_min=10,Tipo_centra
l='bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
#5hectarea
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa312',

base=70000,base_absorcion=50000,Hi=2,H_max=30,H_min=5,Tipo_central
='regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=40,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [127]:

```
def funcion_turbina_311(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
```

```
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 50

def funcion_turbina_312(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.2:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.4:
        self.Qg[t] = 50
    elif  self.H[t]< pres.H_min + var*0.55:
        self.Qg[t] = 80
    elif  self.H[t]< pres.H_min + var*0.7:
        self.Qg[t] = 100
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 120
    else:
        self.Qg[t] = 160
```

In [128]:

```
#Inicializamos las turbinas
turbina311 =
hp.Turbina(Nombre='turbina311',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_311)
turbina312 =
hp.Turbina(Nombre='turbina312',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_312)
def funcion_turbina_313(self=0,t=0, necesidad=0, presa=0, g=9.8,
den=998, intervalo=3600): var= presa.H_max - presa.H_min if
self.H[t]
```

3. Inicializamos la Bomba

In [129]:

```
def funcion_Qb311(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 5;
```

In [130]:

```
def funcion_Pb311(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    self.Pb[t] = 0;
```

In [131]:

```
#Inicializamos las turbinas
bomba311 =
hp.Bomba(Nombre='bomba311',Qb_funcion_propia=funcion_Qb311,Pb_func
ion_propia=funcion_Pb311)
```

In [132]:

```
def funcion_Qb312(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t] = 3;
```

In [133]:

```
#Inicializamos las turbinas
bomba312 =
hp.Bomba(Nombre='bomba312',Qb_funcion_propia=funcion_Qb312,Pb_func
ion_propia=funcion_Pb311)
```

4. Creamos los caudales

In [134]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [135]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
#Q_rioP2=hm.poly_odd(t_total,hora,caudal2,horas_año)

Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(3
1+28)*24)
```

In [136]:

```
presa1.indicar_Qentra_m3s(Q_rioP1 - 10)
presa2.indicar_Qentra_m3s(Q_rio2 * 0.5)
```

 # caudal eresma Bernardos Q_med1 = 5.3 Q_A1=2.08 # caudalTámega Q_med2 = 9.23 Q_A2=2 # caudal Carrión Q_med3 = 5.03 Q_A3 =2.83 #mit=0.3
*Q_rio1=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(31+28)*24)*
*Q_rio2=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(31+28)*24)*
*Q_rio3=hm.caudal_rio(t_total,Q_med3,variacion_anual=Q_A3,max_an=(31+28)*24)*
presa1.indicar_Qentra_m3s(Q_rio1) presa2.indicar_Qentra_m3s(Q_rio2)
presa3.indicar_Qentra_m3s(Q_rio3)

Creamos el objeto de necesidades. aunque en este caso su valor estará practicamente vacío dado que queremos observar el efecto del caudal del generador y no otros parámetros

5. Calcular

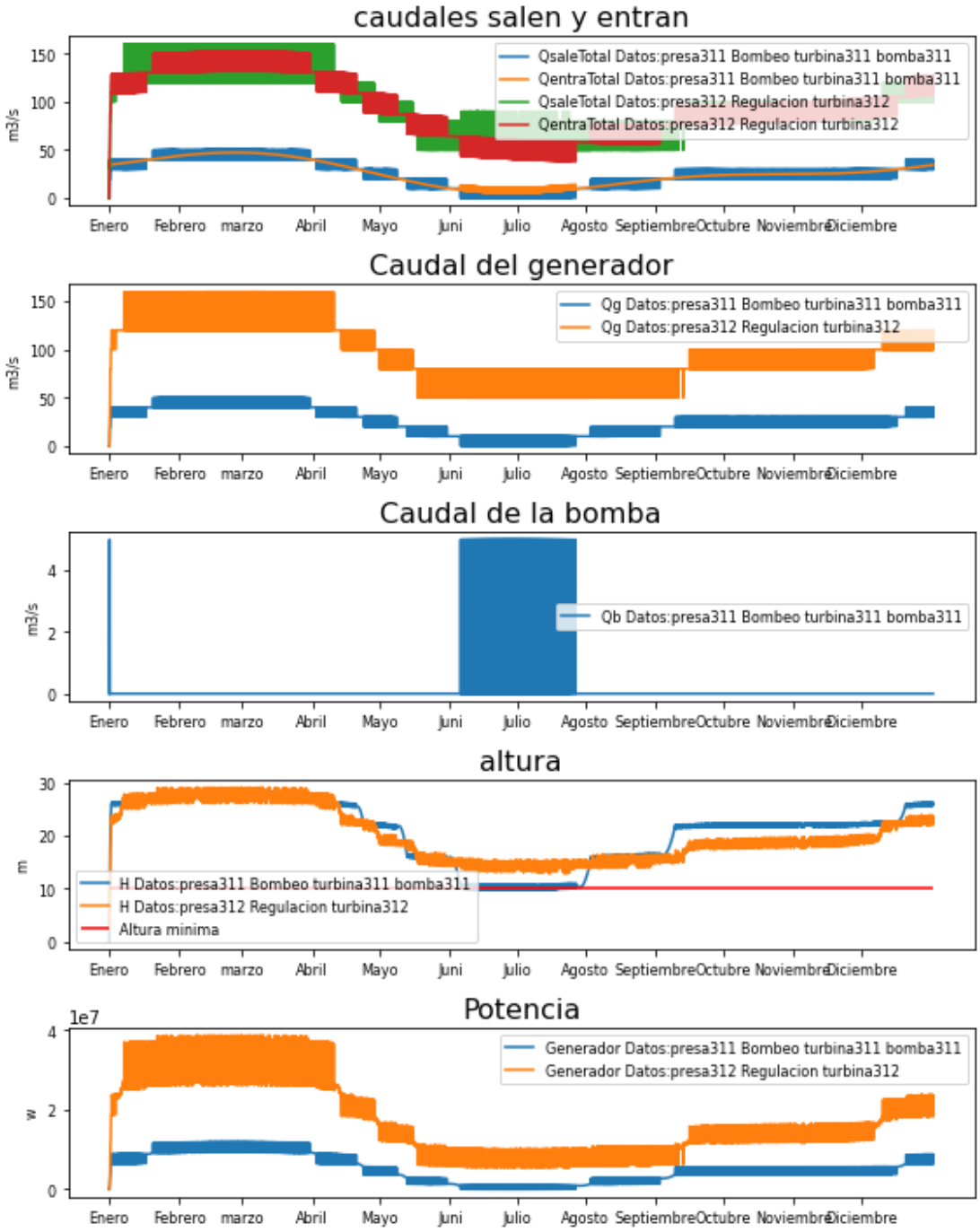
In [137]:

```
presa1.indicar_Qentra_m3s(Q_rioP1 - 10)
presa2.indicar_Qentra_m3s(Q_rio2 * 0.5)

presa1.bomba=bomba311
presa2.bomba=bomba311

presa1.turbina=turbina311
presa2.turbina=turbina312
ListaDatos = hp.sistema_multiple(t_total,
listPresa=[presa1,presa2], necesidad=Necesidades0)
```

Gráficas

In [138]:

```
hp.representar_REVERSIBLE(ListaDatos,LH_Altura={'Altura
minima':presa1.H_min})
```



3.2 LLUVIA

1. Inicializamos las centrales

In [139]:

```
#1hectarea
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='presa321',Presa_salida='presa
322',Presa_de_absorcion='presa312',
```

```
base=10000,base_absorcion=10000,Hi=2,H_max=30,H_min=10,Tipo_centra
l='bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
#5hectarea
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa322',

base=50000,base_absorcion=10000,Hi=2,H_max=30,H_min=10,Tipo_centra
l='regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=40,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [140]:

```
def funcion_turbina_321(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 50

def funcion_turbina_322(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_max*0.6:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 90
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 100
    #elif  self.H[t]< pres.H_max*0.9:
    #   self.Qg[t] = 100
```

```
    else:
        self.Qg[t] = 160
```

In [141]:

```
#Inicializamos las turbinas
turbina321 =
hp.Turbina(Nombre='turbina321',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_321)
turbina322 =
hp.Turbina(Nombre='turbina322',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_322)
```

3. Inicializamos la Bomba

In [142]:

```
def funcion_Pb321(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    area=3
    self.Pb[t] =
den*g*self.Qb[t]*(self.H[t]+(self.Qb[t]/area)**2/(2*g)*presa.bomba
.alpha)/presa.bomba.r
```

In [143]:

```
def funcion_Qb321(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t]=5
```

In [144]:

```
#Inicializamos las turbinas
bomba321 =
hp.Bomba(Nombre='bomba321',Qb_funcion_propia=funcion_Qb321,Pb_func
ion_propia=funcion_Pb321)
```

4. Creamos los caudales

In [145]:

```
Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Valladolid')
Q_lluvia1 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,

area_abastecida=presa1.base,#presa1.base_absorcion,

unidades_preci='mm',unidades_area='m2')

Q_precipitaciones=hm.precipitacion_hora_ciudad(hm.DatosPluviales,'
Zamora')
Q_lluvia2 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones ,
                                   area_abastecida=presa2.base,
```

```
unidades_preci='mm',unidades_area='m2')
```

5. Calcular

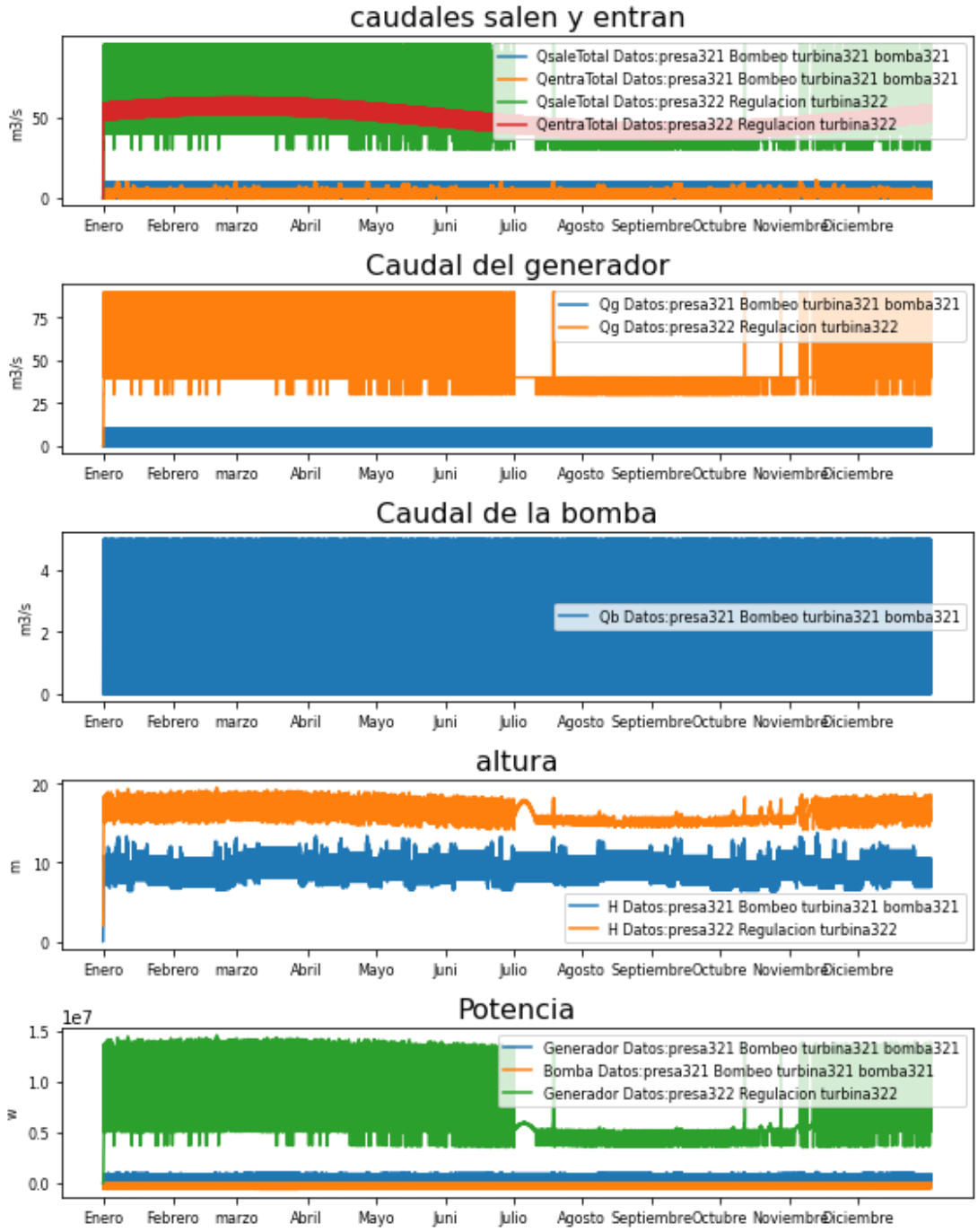
In [146]:

```
presa1.indicar_Qentra_m3s(Q_lluvia1 )
#presa2.indicar_Qentra_m3s(Q_lluvia2)
presa2.indicar_Qentra_m3s(Q_rio2 * 0.5)

presa1.bomba=bomba321
presa2.bomba=bomba321

presa1.turbina=turbina321
presa2.turbina=turbina322

ListaDatos = hp.sistema_multiple(t_total,
listPresa=[presa1,presa2], necesidad=Necesidades0)
```

Caudales

In [147]:

```
hp.representar_REVERSIBLE(ListaDatos)
```



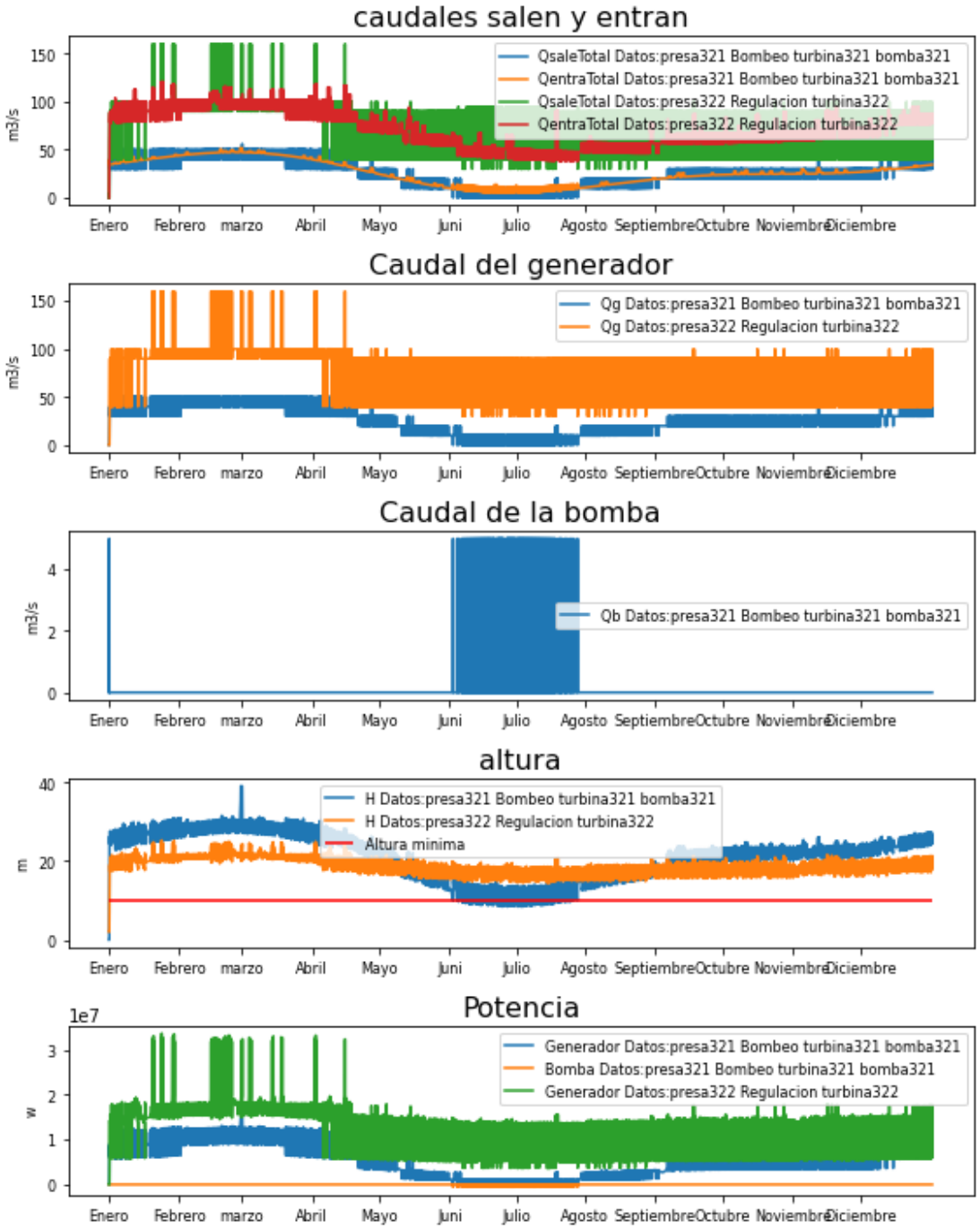
5. Calcular

In [148]:

```
presa1.indicar_Qentra_m3s(Q_lluvia1 + (Q_rioP1-10))
#presa2.indicar_Qentra_m3s(Q_lluvia2)
presa2.indicar_Qentra_m3s(Q_lluvia2 + Q_rio2 * 0.5)

ListaDatos = hp.sistema_multiple(t_total,
listPresa=[presa1,presa2], necesidad=Necesidades0)
```

In [149]:

```
hp.representar_REVERSIBLE(ListaDatos,LH_Altura={'Altura
minima':presa1.H_min})
```



3.3 DEMANDA DE AGUA

1. Inicializamos las centrales

In [150]:

```
#Las distintas centrales a simular:
#3.603 km² islas baleares
```

```
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='presa331',Presa_salida='presa
332',

base=10000,base_absorcion=10000,Hi=2,H_max=30,H_min=5,Tipo_central
='bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)

presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa332',

base=10000,base_absorcion=10000,Hi=2,H_max=30,H_min=5,Tipo_central
='regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [151]:

```
def funcion_turbina_231(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 55
    else:
        self.Qg[t] = 60

def funcion_turbina_232(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max*0.4:
        self.Qg[t] = 15
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 20
    #elif  self.H[t]< pres.H_max*0.6:
    #    self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
```

```
        self.Qg[t] = 60
    #elif  self.H[t]< pres.H_max*0.8:
    #    self.Qg[t] = 80
    elif  self.H[t]< pres.H_max*0.9:
        self.Qg[t] = 100
    else:
        self.Qg[t] = 125
```

In [152]:

```
#Inicializamos las turbinas
turbina231 =
hp.Turbina(Nombre='turbina231',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_231)
turbina232 =
hp.Turbina(Nombre='turbina232',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_232)
```

3. Inicializamos la Bomba

In [153]:

```
def funcion_Pb231(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):
    area=3
    if self.H[t]<0.9:
        self.Pb[t] =
den*g*self.Qb[t]*(self.H[t]+(self.Qb[t]/area)**2/(2*g)*presa.bomba
.alpha)/presa.bomba.r
```

In [154]:

```
def funcion_Qb231(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    if self.H[t]<presa.H_min:
        self.Qb[t]=5
```

In [155]:

```
#Inicializamos las turbinas
bomba231 =
hp.Bomba(Nombre='bomba231',Qb_funcion_propia=funcion_Qb231,Pb_func
ion_propia=funcion_Pb231)
```

4. Creamos los caudales

In [156]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [157]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]
```

```
# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total,hora,caudal1,horas_año)
Q_rioP2=hm.poly_odd(t_total,hora,caudal2,horas_año)

presa1.indicar_Qentra_m3s(Q_rioP1*0.5)
presa2.indicar_Qentra_m3s(Q_rioP2)
```

In [158]:

```
presa1.indicar_Qentra_m3s(Q_rioP1*0.4)
presa2.indicar_Qentra_m3s(Q_rioP2*0.4)
```

5. Creamos las necesidades

Población 2021 Instituto Nacional de Estadística:

In [159]:

```
Poblacion_Bercero = 191
Poblacion_Toro = 8532
Poblacion_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en españa por persona

In [160]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [161]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [162]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                      variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [163]:

```
necesidad231=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad232=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad233=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
#Necesidad = [necesidad231,necesidad232,necesidad233]
Necesidad = [necesidad233,necesidad232]
```

6. Calcular

In [164]:

```
ListaPresa=[presa1,presa2]
```

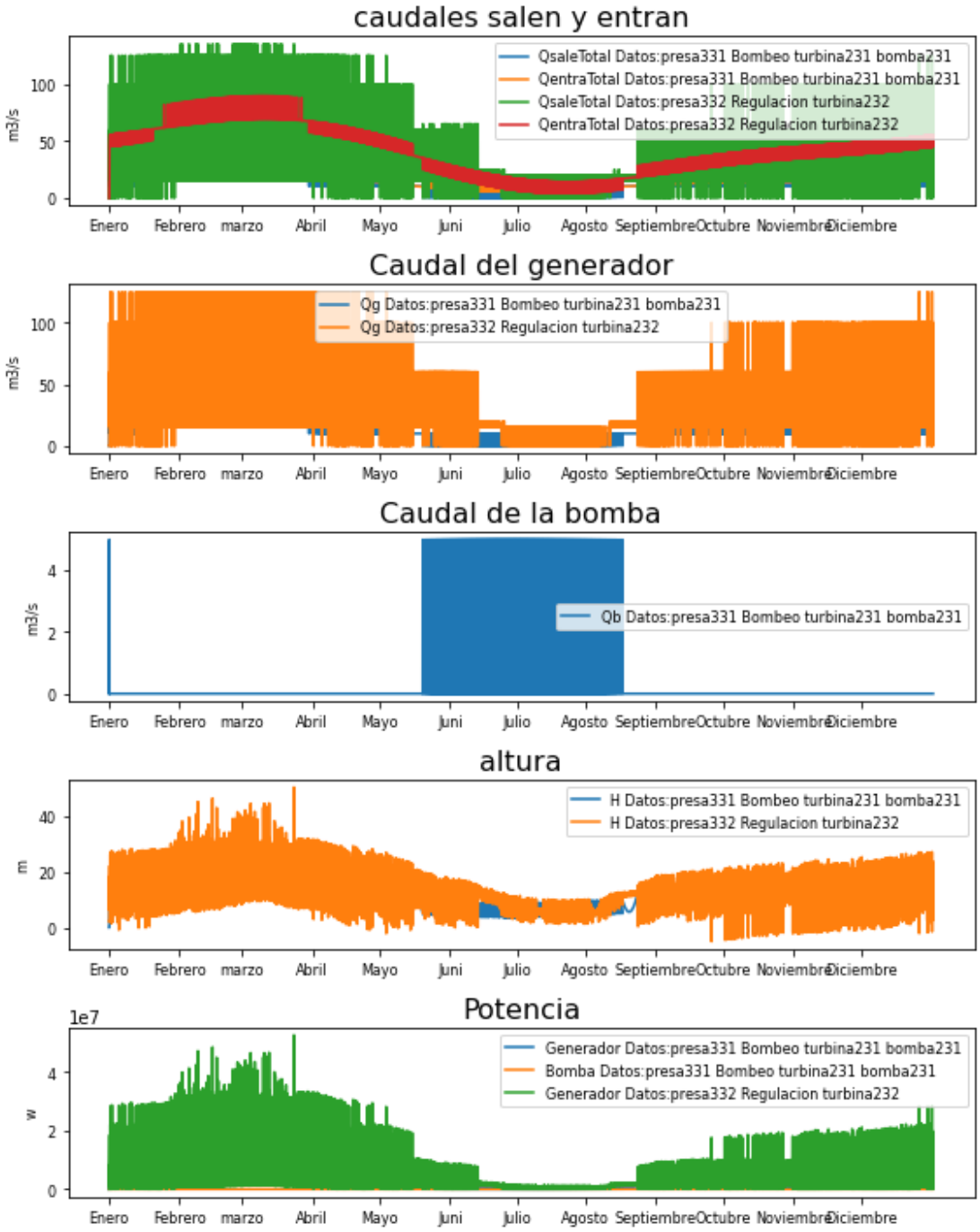
In [165]:

```
presa1.bomba=bomba231
presa2.bomba=bomba231

presa1.turbina=turbina231
presa2.turbina=turbina232

ListaDatos = hp.sistema_multiple(t_total,
listPresa=[presa1,presa2], necesidad=Necesidad)
```

Caudales

In [166]:

```
hp.representar_REVERSIBLE(ListaDatos)
```



3.4 DEMANDA DE POTENCIA

In [167]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia

tini=0
```

```
tfin=7*horas_dia
t_total_semana= np.arange(tfin-tini)
```

In [168]:

```
from sympy.solvers import solve
from sympy import Symbol
```

In [169]:

```
from sympy.abc import x
```

1. Inicializamos las centrales

In [170]:

```
#1hectarea
presa1 =
hp.Presa(t_total,Q_entrada=0,Nombre='presa341',Presa_salida='presa
342',Presa_de_absorcion='presa342',

base=10000,base_absorcion=10000,Hi=2,H_max=30,H_min=10,Tipo_centra
l='bombeo',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=20,Hentrada=30.000)
)
#5hectarea
presa2 = hp.Presa(t_total,Q_entrada=0,Nombre='presa342',

base=50000,base_absorcion=10000,Hi=2,H_max=30,H_min=10,Tipo_centra
l='regulación',H_des=35,

Qdesbordado=10.0,aliviadero=hp.Aliviadero(Area=40,Hentrada=30.000)
)
```

2. Inicializamos la turbina

In [171]:

```
def funcion_turbina_341(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    var= pres.H_max - pres.H_min
    if self.H[t]<pres.H_min or self.Qb[t]>0:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_min + var*0.3:
        self.Qg[t] = 10
    elif  self.H[t]< pres.H_min + var*0.6:
        self.Qg[t] = 20
    elif  self.H[t]< pres.H_min + var*0.8:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_min + var*0.9:
        self.Qg[t] = 40
    else:
        self.Qg[t] = 50
```

```
def funcion_turbina_342(self=0,t=0, necesidad=0, pres=0, turb=0
,g=9.8, den=998, intervalo=3600):
    if self.H[t]<pres.H_min:
        self.Qg[t] = 0
    elif  self.H[t]< pres.H_max/2:
        self.Qg[t] = 30
    elif  self.H[t]< pres.H_max*0.6:
        self.Qg[t] = 40
    elif  self.H[t]< pres.H_max*0.7:
        self.Qg[t] = 90
    elif  self.H[t]< pres.H_max*0.8:
        self.Qg[t] = 100
    #elif  self.H[t]< pres.H_max*0.9:
    #   self.Qg[t] = 100
    else:
        self.Qg[t] = 160
```

In [172]:

```
#Inicializamos las turbinas
turbina241 =
hp.Turbina(Nombre='turbina241',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_341)
turbina242 =
hp.Turbina(Nombre='turbina242',Q_max=10,Q_min=0.1,Qg_funcion_propi
a=funcion_turbina_342)
```

3. Inicializamos la Bomba

In [173]:

```
def funcion_Qb24(self=0,t=0, necesidad=0,listDatos=0,
presa=0,g=9.8, den=998, intervalo=3600):
    #print(t)
    self.selector_Pb(t, presa=presa, necesidad=necesidad, g=9.8,
den=998)
    if self.Pb[t]>0:

a,b,c=solve([den*g*x*(self.H[t]+(x/presa.bomba.area)**2/(2*g)*pres
a.bomba.alpha)-self.Pb[t]*presa.bomba.r],x)

        res = float('.'.join(str(a) for a in a))
        if res > 0:
            self.Qb[t]=res
        else:
            res = float('.'.join(str(b) for b in b))
            if 0<res:
                self.Qb[t]=res
            else:
                res = float('.'.join(str(c) for c in c))
```

```
                if 0<res:
                    self.Qb[t]=res
```

In [174]:

```
def funcion_Pb24(self=0,t=0, necesidad=0, presa=0,g=9.8, den=998,
intervalo=3600):

    if necesidad.Pgn_ms[t]<0:
        self.Pb[t] = necesidad.Pgn_ms[t]*-1;
```

In [175]:

```
#Inicializamos las turbinas
bomba24 =
hp.Bomba(Nombre='bomba24',Qb_funcion_propia=funcion_Qb24,Pb_funcio
n_propia=funcion_Pb24)
```

4. Creamos los caudales

In [176]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [177]:

```
# caudal Pisuerga (Valladolid)
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

# caudal Duero (Toro)
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rioP1 = hm.poly_odd(t_total_semana,hora,caudal1,horas_año)
Q_rioP2 = hm.poly_odd(t_total_semana,hora,caudal2,horas_año)
```

5. Creamos las necesidades

In [178]:

```
hora= [4,9,13,17,22]
potencia=[4.9,6.5,6.6,5.8,6.9]
```

In [179]:

```
t_sem = np.linspace(0, 24, num=25)
demanda_ElHierro = hm.poly_odd(np.arange(tfin-
tini),hora,potencia,24)
```

In [180]:

```
demanda_ElHierro_centrado =demanda_ElHierro-
np.mean(demanda_ElHierro)
```

Creamos los objetos necesidades

In [181]:

```
necesidad_potencia_1=
hp.Necesidades(t_total_semana,Nombre='Necesidad_ElHierro_diario',P
otencia_necesaria=demanda_ElHierro_centrado)
```

6. Calcular

In [182]:

```
presa1.turbina=turbina241
presa2.turbina=turbina242

presa1.bomba=bomba24
presa2.bomba=bomba24

presa1.indicar_Qentra_m3s(Q_rioP1 - 10)
presa2.indicar_Qentra_m3s(Q_rio2 * 0.5)
```

In [183]:

```
ListaDatos1 = hp.sistema_multiple(t_total_semana,
listPresa=[presa1,presa2], necesidad=necesidad_potencia_1)
```

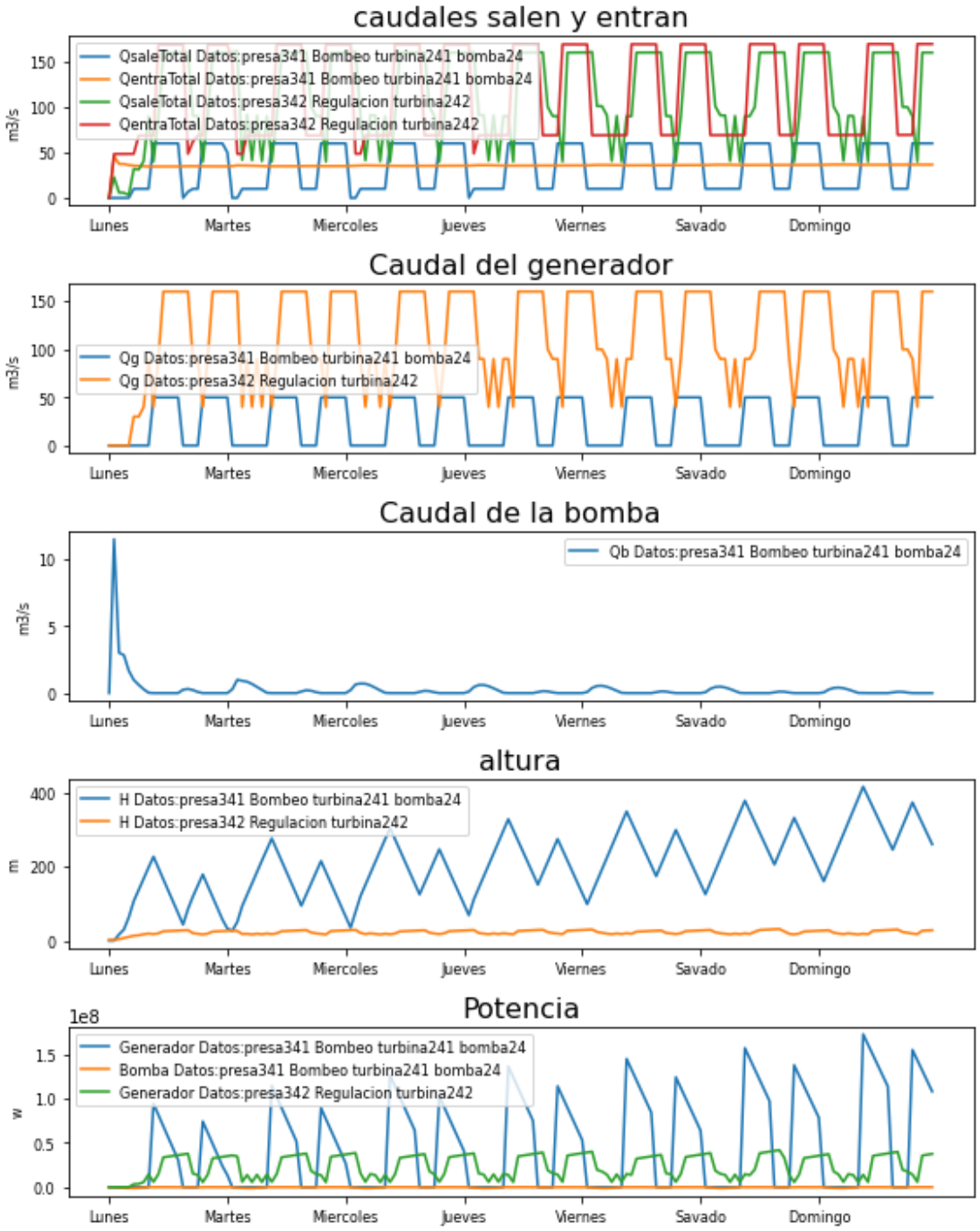
In [184]:

```
ListaDatos2 = hp.sistema_multiple(t_total_semana,
listPresa=[presa1,presa2],
necesidad=[necesidad_potencia_1,Necesidades0])
```

Caudales

In [185]:

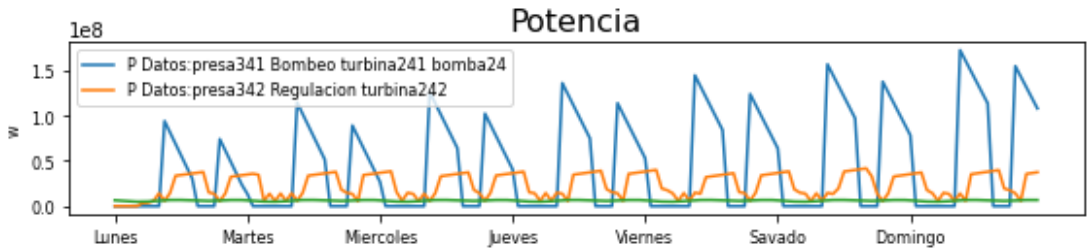
```
hp.representar_REVERSIBLE(ListaDatos1, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
```



In [186]:

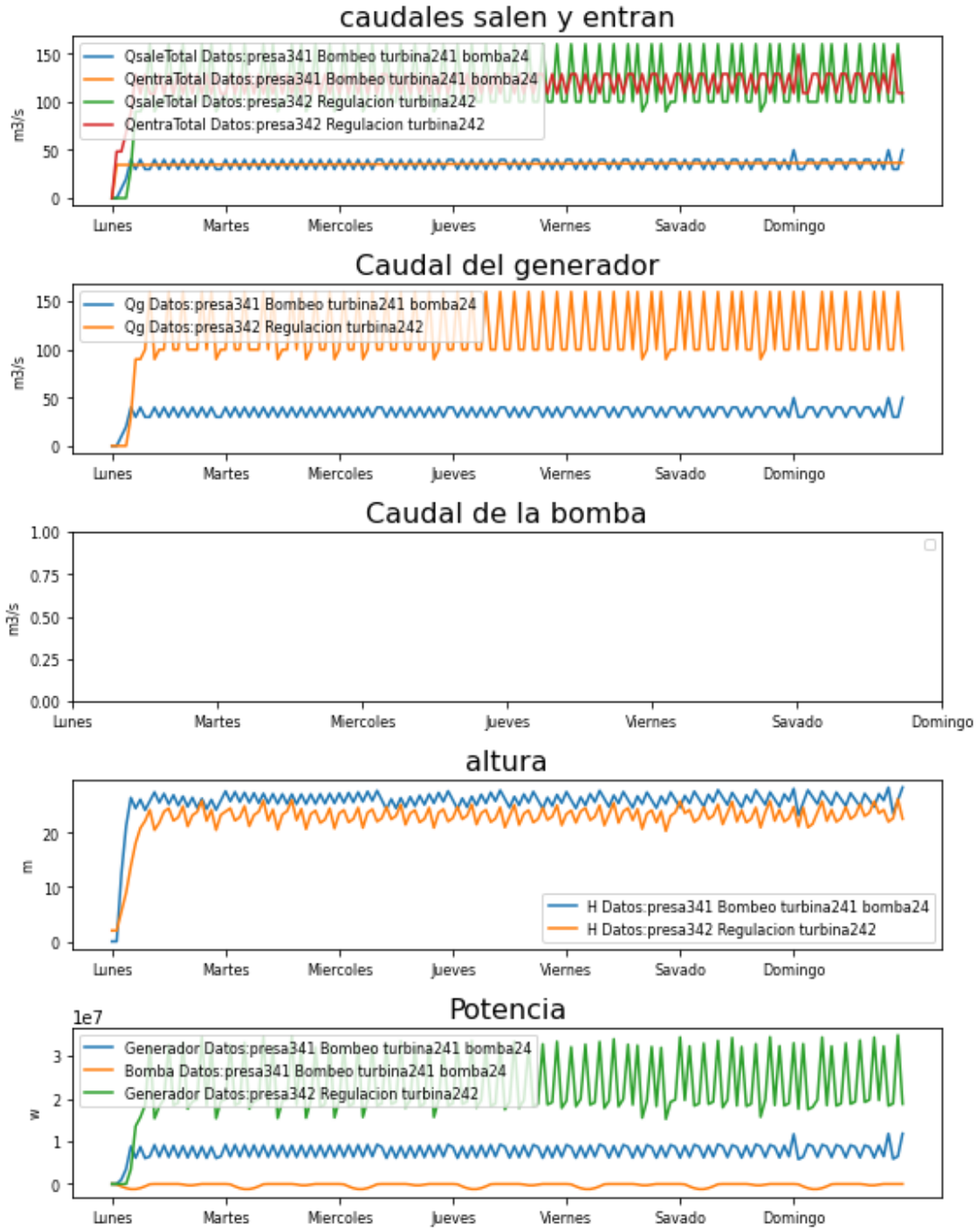
```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos1, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)
```

```
plt.tight_layout()
plt.show()
```



In [187]:

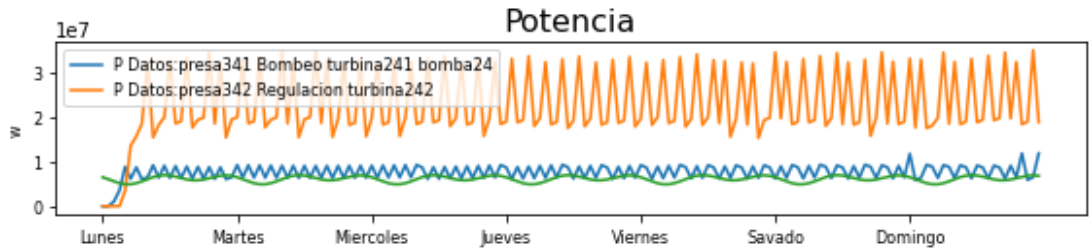
```
hp.representar_REVERSIBLE(ListaDatos2, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana, LH_Altura={})
No artists with labels found to put in legend.  Note that artists
whose label start with an underscore are ignored when legend() is
called with no argument.
```



In [188]:

```
plt.figure(figsize=(WIDE, HEIGHT))

plt.subplot(111)
hp.representar_Pgenerador(ListaDatos2, t_total=t_total_semana,
horatik=horaSemana, nombretik=semana)
string1= 'P demanda_ElHierro '
plt.plot(t_total_semana,demanda_ElHierro*1000000,label = string1)
```

```
plt.tight_layout()
plt.show()
```



ANEXO 7.4 NOTEBOOK 4

AJUSTE DE DATOS

Modelos de distintos parámetros necesarios para la realización de las simulaciones.

Estructura del notebook:

- 0_MODELOS DE DEMANDA
    0. Demanda de potencia
        - Diario
        - Semanal
        - Anual
        - Conjunto
    1. Demanda de agua
- 0_MODELOS DE CAUDALES DE ENTRADA
    0. Caudal de los ríos
        - Senoidal
        - Polinomial
        - Comparación
        - Río Sella
    1. Precipitaciones

1. Importaciones e inicializaciones

In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
#%matplotlib widget
```

Módulos para el modelo de las distintas entradas y demandas:

In [2]:

```
import lib.herramientas_modelado as hm
```

Módulos para las simulaciones:

In [3]:

```
import lib.herramientas_presas as hp
```

2. Datos para la simulación:

In [4]:

```
dias_año=365
horas_dia=24
horas_año=dias_año*horas_dia
```

```
tini=0
tfin=horas_año
t_total= np.arange(tfin-tini)
```

3. Datos para la representación:

In [5]:

```
meses=['Enero','Febrero','marzo','Abril','Mayo','Juni','Julio','Ag
osto','Septiembre','Octubre','Noviembre','Diciembre']
horaMes=[0,31*24,59*24,90*24,120*24,151*24,181*24,211*24,242*24,27
2*24,303*24,333*24]
semana=['lunes','Martes','Miércoles','Jueves','Viernes','Sábaso','
Domingo']
horaSemana=[0,24,48,72,96,120,144]
L=[12]*7
horaSemana2=np.array(horaSemana)+L
horaSemana2
```

Out[5]:

```
array([ 12,  36,  60,  84, 108, 132, 156])
```

Para graficar más fácilmente haremos unos ajustes:

In [6]:

```
n=0
WIDE = 8
HEIGHT = 2
LETTER_SIZE = 8
```

4. Creamos el objeto de la clase necesidades vacío:

In [7]:

```
Necesidades0 = hp.Necesidades(t_total,Nombre='Necesidades0')
```

0 MODELO DE LA DEMANDA

1. DEMANDA DE POTENCIA

In [8]:

```
v=np.array([0,1,2,3,4,5,6,7,8,9,10,11])
Dias_Mes=[31,28,31,30,31,30,30,31,30,31,30,31]
V_DM=np.array(Dias_Mes)
Dias_Mes_acumulado=[0,31,59,90,120,151,181, 211, 242, 272, 303,
333]
V_DMA=np.array(Dias_Mes_acumulado)
```

1.1 DIARIO

1.2.1 Isla de El Hierro

In [9]:

```
hora= [4,9,13,17,22]
potencia=[4.9,6.5,6.6,5.8,6.9]
```

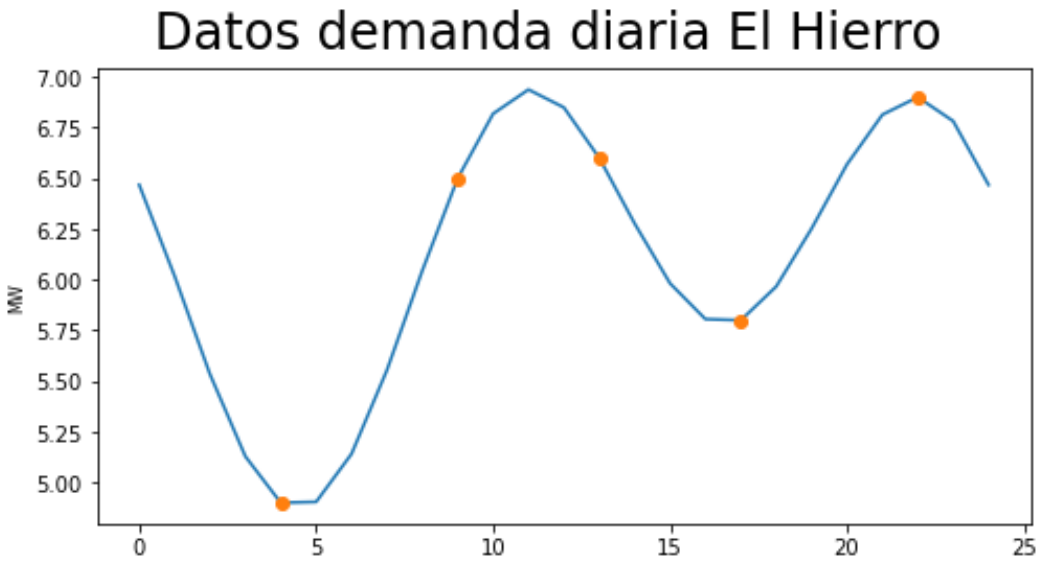
In [10]:

```
t_sem = np.linspace(0, 24, num=25)
demanda_ElHierro = hm.poly_odd(t_total,hora,potencia,24)
```

9.1.1.1 Representar

In [11]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.suptitle('Datos demanda diaria El
Hierro',fontsize=LETTER_SIZE*3)
plt.subplot(111)
plt.plot(t_total[:25],demanda_ElHierro[:25])
plt.plot(hora,potencia,'o')
plt.xticks( rotation=0)
plt.ylabel('MW', fontsize=LETTER_SIZE)

plt.show()
```



Datos procedentes de https://www.ree.es/es

1.2.1 Isla de El Hierro

In [12]:

```
hora= [4,9,13,17,22]
potencia=[2200,31000,32500,31000,29000]
```

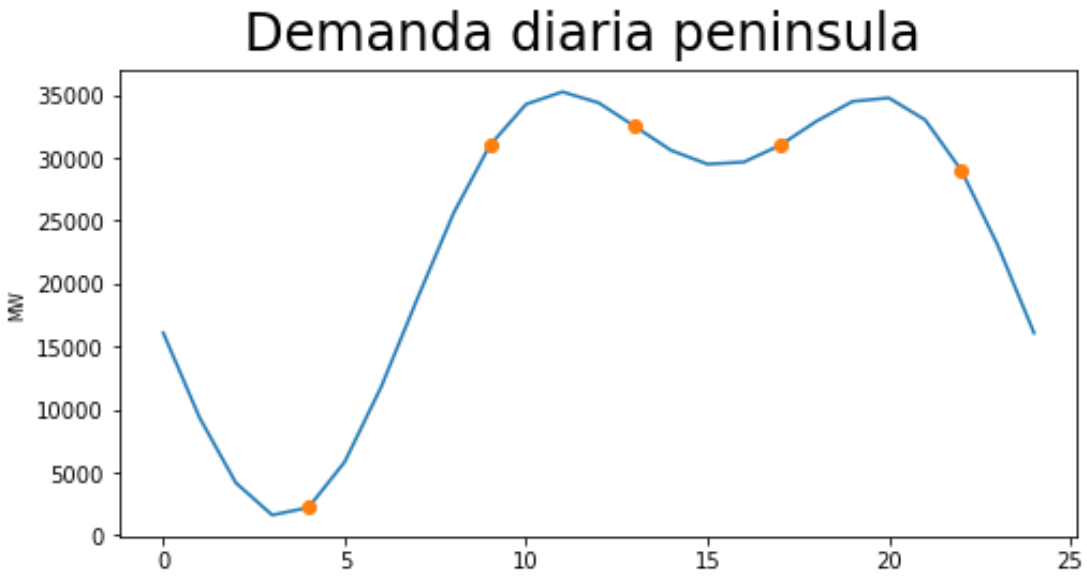
In [13]:

```
t_sem = np.linspace(0, 24, num=25)
demandaD_peninsula = hm.poly_odd(t_total,hora,potencia,24)
```

9.1.1.2 Representar

In [14]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.suptitle('Demanda diaria peninsula',fontsize=LETTER_SIZE*3)
plt.subplot(111)
plt.plot(t_total[:25],demandaD_peninsula[:25])
plt.plot(hora,potencia,'o')
plt.xticks( rotation=0)
plt.ylabel('MW', fontsize=LETTER_SIZE)

plt.show()
```



Datos procedentes de https://www.ree.es/es

1.2 SEMANAL

In [15]:

```
df_Potencia_Mensual= pd.read_excel('datos/DEMANDA_MENSUAL.xlsx')

print ('Data read into a pandas dataframe!')
Data read into a pandas dataframe!
```

In [16]:

```
df_Potencia_Mensual.set_index('Fecha', inplace=True)
```

In [17]:

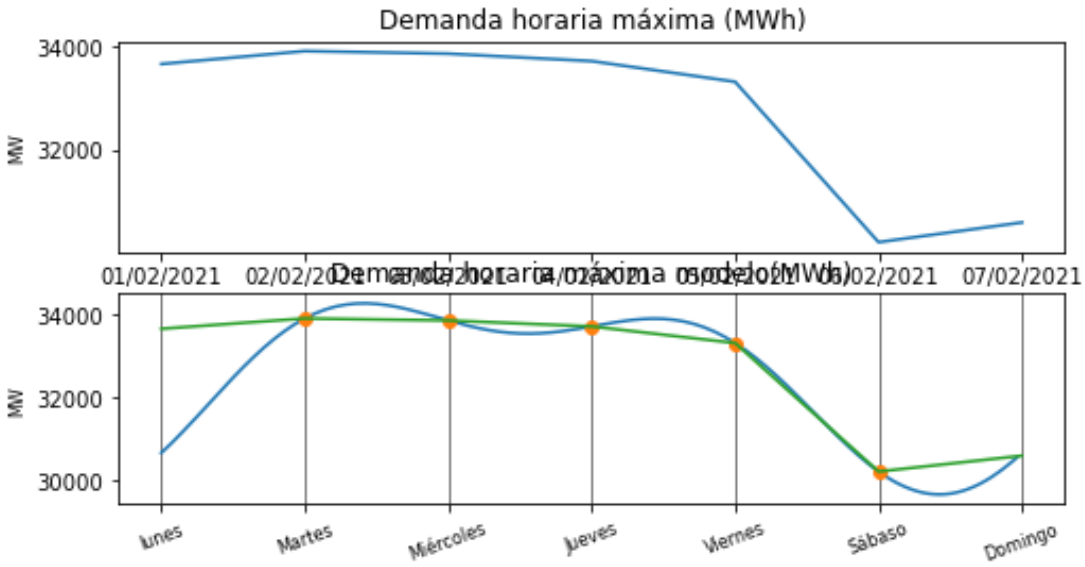
```
df_Potencia_Mensual.columns = list(map(str,
df_Potencia_Mensual.columns))
```

In [18]:

```
dias=[1,2,4,3,5]
semana_entera=df_Potencia_Mensual['Demanda horaria máxima
(MWh)'].tolist()[0:7]
semana1=df_Potencia_Mensual['Demanda horaria máxima
(MWh)'][dias].tolist()[0:7]
demandaS_peninsula = hm.poly_odd(t_total,np.linspace(0, 7*24,
num=7)[dias],semana1,T=7*24)
```

In [19]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.subplot(211)
df_Potencia_Mensual['Demanda horaria máxima (MWh)'][:7].plot()
plt.xticks(rotation=0)
plt.title('Demanda horaria máxima (MWh)')
plt.ylabel('MW', fontsize=LETTER_SIZE)

plt.subplot(212)
plt.plot(t_total[:7*24],demandaS_peninsula[:7*24])
plt.plot(np.linspace(0, 7*24, num=7)[dias],semana1,'o')
plt.plot(np.linspace(0, 7*24, num=7),semana_entera)
plt.xticks(np.linspace(0, 7*24, num=7),
semana,fontsize=LETTER_SIZE,rotation=20)
plt.ylabel('MW', fontsize=LETTER_SIZE)
plt.title('Demanda horaria máxima modelo(MWh)')
plt.grid(color='k',axis='x', linestyle='-', linewidth=0.5)

plt.show()
```



1.3 MENSUAL

In [20]:

```
df_demanda_mensua_penunsilar=
pd.read_excel('datos/Demanda_Peninsular_Mensual_GWh.xlsx')

print ('Data read into a pandas dataframe!')
Data read into a pandas dataframe!
```

In [21]:

```
df_demanda_mensua_penunsilar.head()
```

Out[21]:

	Mes	2019	2020	2021
0	Enero	23296.649046	22577.217377	22705.774443
1	Febrero	20154.629677	19840.085662	19192.438320
2	Marzo	20726.895805	19808.362302	20684.554316
3	Abril	19514.052023	16160.449329	8600.369600
4	Mayo	19899.136009	17368.389883	NaN

In [22]:

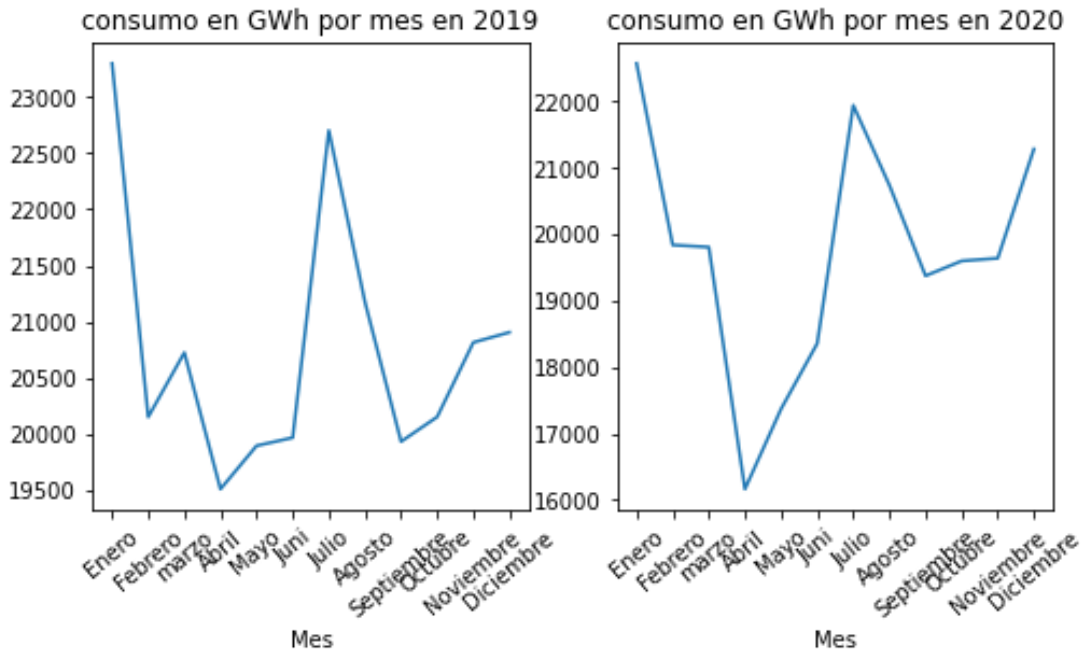
```
df_demanda_mensua_penunsilar.set_index('Mes', inplace=True)
```

In [23]:

```
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.subplot(121)
df_demanda_mensua_penunsilar[2019].plot(kind='line')
plt.title('consumo en GWh por mes en 2019')
plt.xticks( [0,1,2,3,4,5,6,7,8,9,10,11],meses,rotation=40)

plt.subplot(122)
df_demanda_mensua_penunsilar[2020].plot(kind='line')
plt.title('consumo en GWh por mes en 2020')
plt.xticks( [0,1,2,3,4,5,6,7,8,9,10,11],meses,rotation=40)
plt.show
```

Out[23]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



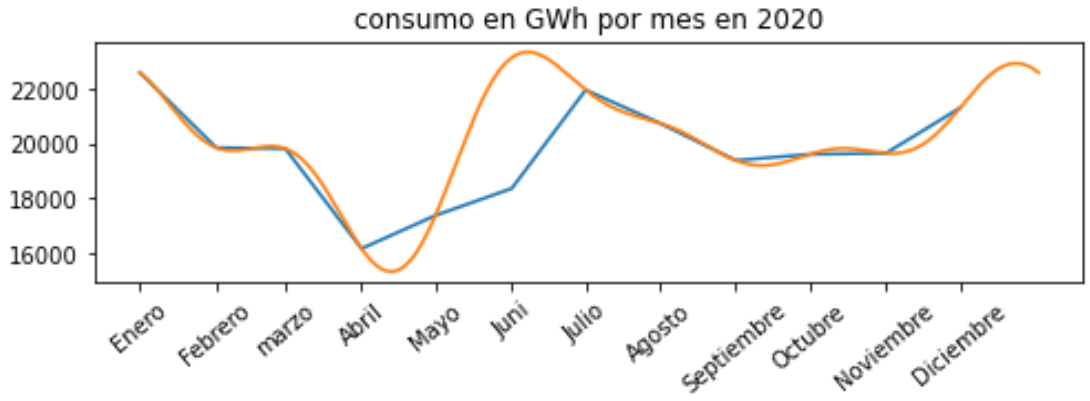
Modelo 1

In [24]:

```
demandaA_peninsula2020 = hm.poly_odd(t_total,

np.concatenate((V_DMA[0:5]*24,V_DMA[6:12]*24), axis=None),

np.concatenate((df_demanda_mensua_penunsilar[2020].tolist()[0:5],d
f_demanda_mensua_penunsilar[2020].tolist()[6:12]),
axis=None),horas_año)
```

In [25]:

```
plt.figure(figsize=(WIDE, HEIGHT))

plt.plot(V_DMA*24,df_demanda_mensua_penunsilar[2020])
plt.plot(t_total,demandaA_peninsula2020)
plt.title('consumo en GWh por mes en 2020')
plt.xticks( V_DMA*24,meses,rotation=40)

plt.show()
```



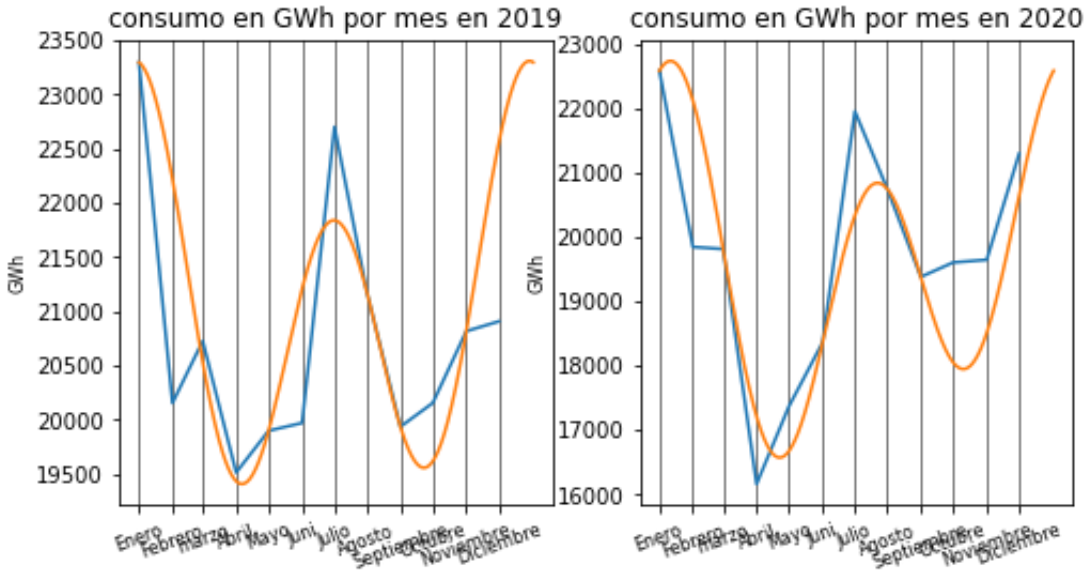
Modelo 2

In [26]:

```
indice=[[0,4,7,8,10],[0,2,5,7,8]]
List_Ano=[2019,2020]
lista_demandaA_peninsula=[[]]*len(List_Ano)
for i in range(len(List_Ano)):
    lista_demandaA_peninsula[i] = hm.poly_odd(t_total,
                          V_DMA[indice[i]]*24,

df_demanda_mensua_penunsilar[List_Ano[i]][indice[i]].tolist(),hora
s_año)
```

In [27]:

```
FILAS=round(len(List_Ano)/2)

plt.figure(figsize=(WIDE, HEIGHT*FILAS*2))
for i in range(len(List_Ano)):
    plt.subplot(FILAS,2,i+1)
    plt.plot(V_DMA*24,df_demanda_mensua_penunsilar[List_Ano[i]])
    plt.plot(t_total,lista_demandaA_peninsula[i])
    plt.title('consumo en GWh por mes en '+str(List_Ano[i]))
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.grid(color='k',axis='x', linestyle='-', linewidth=0.5)
    plt.ylabel('GWh', fontsize=LETTER_SIZE)


plt.show()
```



1.4 CONJUNTO

In [28]:

```
diario_medio = demandaD_peninsula-np.mean(demandaD_peninsula)
```

In [29]:

```
demandaD_peninsula-np.mean(demandaD_peninsula)
```

Out[29]:

```
array([ -7698.29777637, -14431.94770079, -19611.05603664, ...,
         9237.81216435,   5215.97385039,   -729.66883979])
```

In [30]:

```
semanal_medio= demandaS_peninsula-
[np.mean(demandaS_peninsula)]*len(demandaS_peninsula)
```

In [31]:

```
ano_media= demandaA_peninsula2020-
[np.mean(demandaA_peninsula2020)]*len(demandaA_peninsula2020)
```

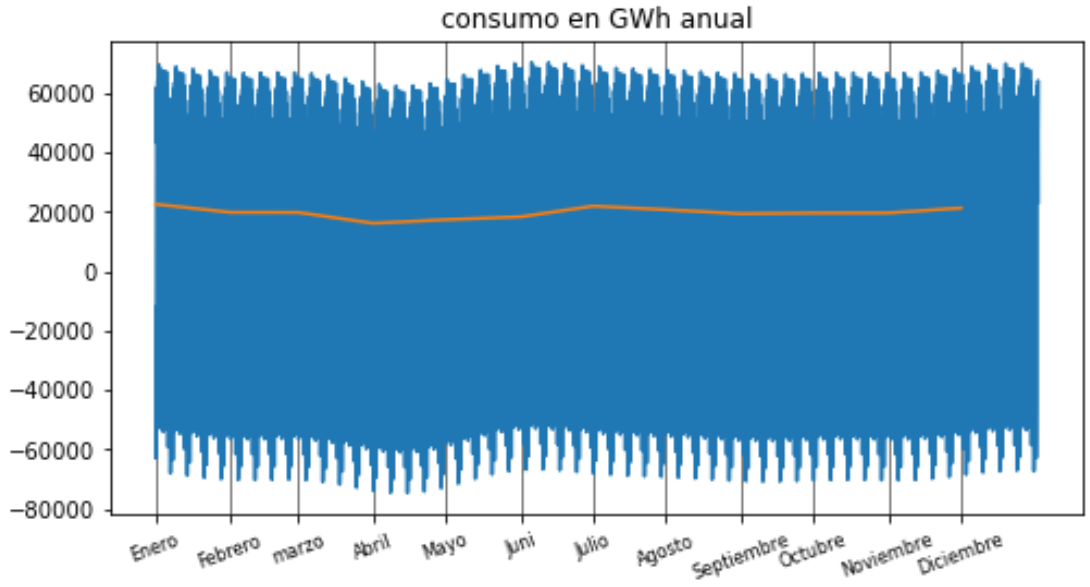
In [32]:

```
total=diario_medio*3.6+semanal_medio*3.6+demandaA_peninsula2020
```

In [33]:

```
FILAS=1#round(len(List_Ano)/2)

plt.figure(figsize=(WIDE*1, HEIGHT*FILAS*2))
for dat in [total]:
    plt.subplot(FILAS,1,1)
    plt.plot(t_total,dat)
    plt.plot(V_DMA*24,df_demanda_mensua_penunsilar[2020])
    plt.title('consumo en GWh anual')
```

```
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.grid(color='k',axis='x', linestyle='-', linewidth=0.5)

plt.show()
```



2. DEMANDA DE AGUA

Población 2021 Instituto Nacional de Estadística:

In [34]:

```
Poblacion_Bercero = 191
Poblacion_Toro = 8532
Poblacion_Valladolid= 297225
Nombres_Poblacion=['Bercero','Toro','Valladolid']
```

Consumo de agua medio en españa por persona

In [35]:

```
Consumo_Medio = 0.136
```

Se calcula cuánto se consume a la hora

In [36]:

```
CB = hm.consumo_hidrico_habitantes(Poblacion_Bercero,
promedio=Consumo_Medio, periodo_horas=24)
CT = hm.consumo_hidrico_habitantes(Poblacion_Toro,
promedio=Consumo_Medio, periodo_horas=24)
CV = hm.consumo_hidrico_habitantes(Poblacion_Valladolid,
promedio=Consumo_Medio, periodo_horas=24)
```

Cuánto se consume cada hora del año

In [37]:

```
Consumo_Bercero = hm.consumo_hidraulico(t_total, CB/6,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Toro = hm.consumo_hidraulico(t_total, CT,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
Consumo_Valladolid = hm.consumo_hidraulico(t_total, CV,
variacion_anual=0, max_an=4344,
                        variacion_dia=0, max_dia=0)
```

Creamos los objetos necesidades

In [38]:

```
necesidad1=
hp.Necesidades(t_total,Nombre='Necesidad_Bercero',Caudal_necesario
=Consumo_Bercero)
necesidad2=
hp.Necesidades(t_total,Nombre='Necesidad_Toro',Caudal_necesario=Co
nsumo_Toro)
necesidad3=
hp.Necesidades(t_total,Nombre='Necesidad_Valladolid',Caudal_necesa
rio=Consumo_Valladolid)
```

REPRESENTAR

Lista para las representaciones

In [39]:

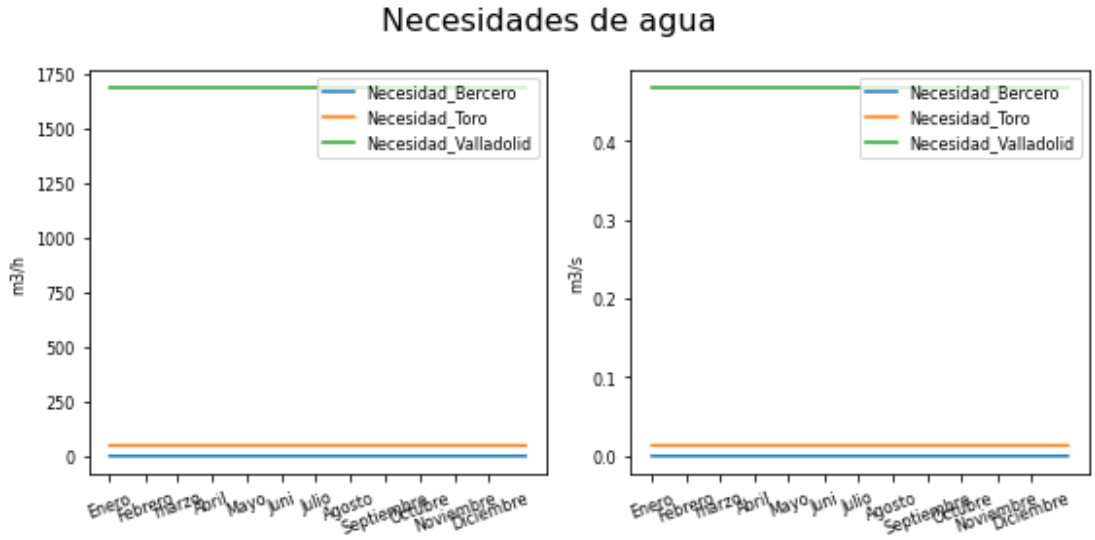
```
Lista_Necesidades = [necesidad1,necesidad2,necesidad3]
```

Representacion senoidal

In [40]:

```
filas =  round(len(Nombres_Poblacion)/2)
plt.figure(figsize=(WIDE, HEIGHT*2))

plt.suptitle('Necesidades de agua',fontsize=LETTER_SIZE*2)
plt.subplot(1,2,1)
for Necesidad in Lista_Necesidades:
    plt.plot(t_total,Necesidad.Caudal_necesario,label =
Necesidad.Nombre )
plt.ylabel('m3/h', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.subplot(1,2,2)
for Necesidad in Lista_Necesidades:
```

```
    plt.plot(t_total,Necesidad.Qsale_m3s,label = Necesidad.Nombre
)
plt.ylabel('m3/s', fontsize=LETTER_SIZE)
plt.yticks(fontsize=LETTER_SIZE)
plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



0 MODELO DE LOS CAUDALES DE ENTRADA

3. CAUDALES DE RÍOS

3.1 SIMULACIÓN SENOIDAL

Río Pisuerga (Valladolid)

In [41]:

```
Q_med1 = (54+32)/2
Q_A1 =(54-32)

Q_rio11=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(
31+28)*24)
```

Río Duero (Toro)

In [42]:

```
Q_med2 = (109.18+46.58)/2
Q_A2 =(109.18-46.58)
```

```
Q_rio12=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(
31+28)*24)
```

Río Támega

In [43]:

```
Q_med1 = 9.23
Q_A1=2

Q_rio13=hm.caudal_rio(t_total,Q_med1,variacion_anual=Q_A1,max_an=(
31+28)*24)
```

Río Carrión

In [44]:

```
Q_med2 = 5.03
Q_A2 =2.83

Q_rio14=hm.caudal_rio(t_total,Q_med2,variacion_anual=Q_A2,max_an=(
31+28)*24)
```

Representar

Lista para las representaciones

In [45]:

```
Nombre_Rio1=['Pisuerga','Duero',]
Lista_Rio_sin1=[Q_rio11,Q_rio12]
```

In [46]:

```
Nombre_Rio2=['Támega','Carrión']
Lista_Rio_sin2=[Q_rio13,Q_rio14]
```

In [47]:

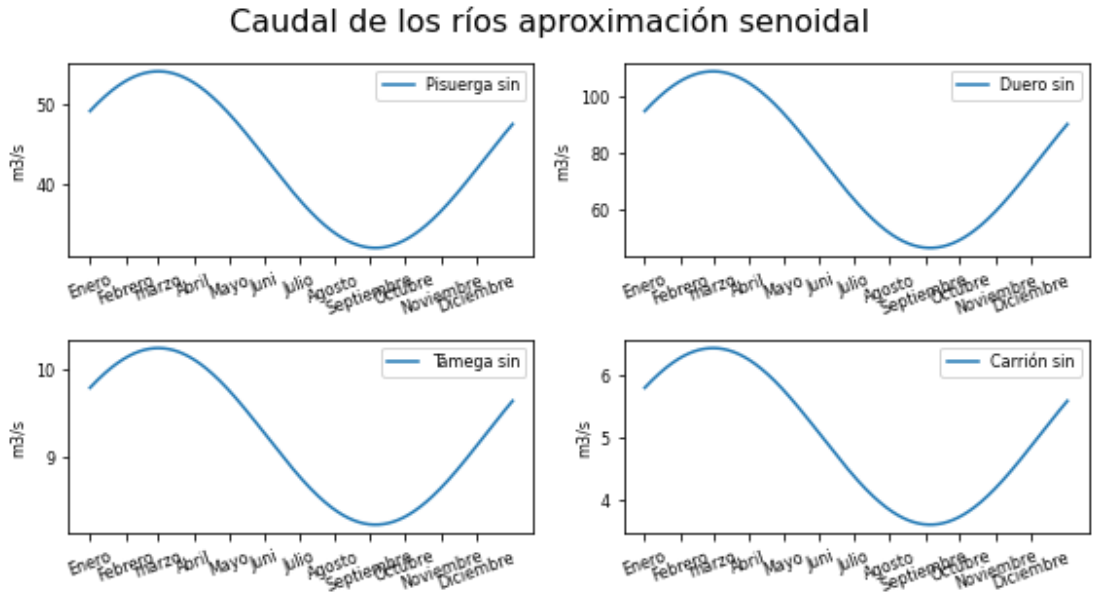
```
Lista_Rio_sin = Lista_Rio_sin1 + Lista_Rio_sin2
Nombre_Rio = Nombre_Rio1 + Nombre_Rio2
```

Gráfica

In [48]:

```
filas =  round(len(Nombre_Rio)/2+1)
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('Caudal de los ríos aproximación
senoidal',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Rio_sin)):
    plt.subplot(filas,2,i+1)
    plt.plot(t_total,Lista_Rio_sin[i],label = Nombre_Rio[i]+' sin'
)


    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
```

```
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



3.2 SIMULACIÓN POLINOMIAL

In [49]:

```
lista_meses=[0,1,4,5,9]
hora1=hm.lista_mes_hora(lista_meses)
```

Río Pisuerga (Valladolid)

In [50]:

```
Q_med1 = 34.49
Q_A1 =(54-32)/2
caudal1=[44.49,54,34.49,20,34]

Q_rio21 = hm.poly_odd(t_total,hora1,caudal1,horas_año)
```

Río Duero (Toro)

In [51]:

```
Q_med2 = 88.19
Q_A2 =(109.18-46.58)/2
caudal2=[88.19,109.18,88.19,46.58,46.58]

Q_rio22=hm.poly_odd(t_total,hora1,caudal2,horas_año)
```

Río Támega

In [52]:

```
lista_meses=[1,4,5,9,11]
hora2=hm.lista_mes_hora(lista_meses)
Q_med3 = 9.23

Q_max3 =10.63
Q_min3 =8.63

Q_A3=Q_max3-Q_min3
caudal3=[Q_max3,Q_med3,Q_min3*0.98 ,Q_min3*0.99,Q_med3  ]

Q_rio23=hm.poly_odd(t_total,hora2,caudal3,horas_año)
```

Río Carrión

In [53]:

```
Q_med4 = 5.03
Q_max4 =6.89
Q_min4 =4.06
Q_A4=Q_max4-Q_min4
caudal4=[Q_max4,Q_med4,Q_min4*1.02,Q_min4*1.03,Q_med4 ]

Q_rio24=hm.poly_odd(t_total,hora2,caudal4,horas_año)
```

Representar

Lista para las representaciones

In [54]:

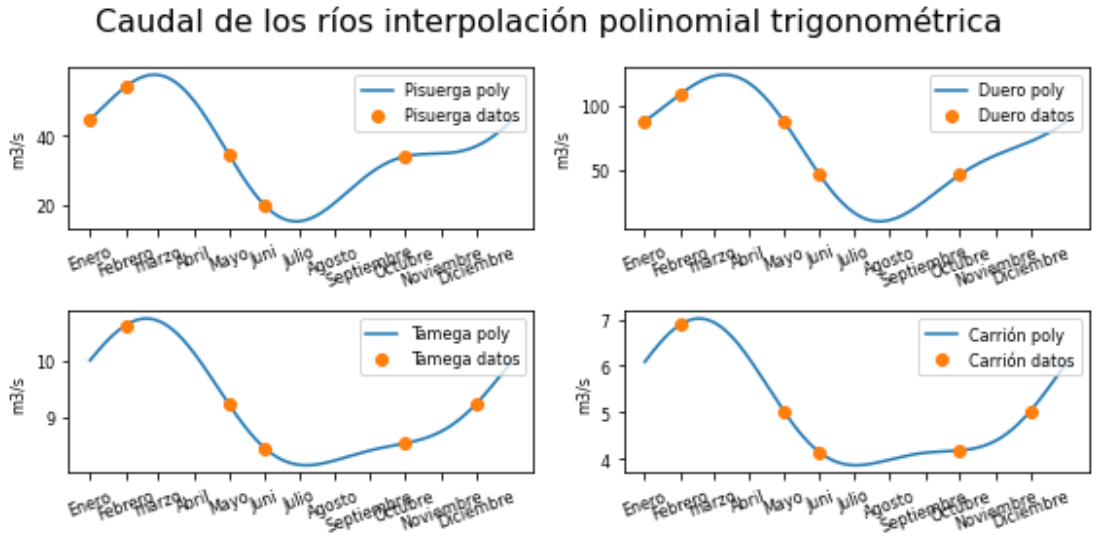
```
Nombres_Rios=['Pisuerga','Duero','Tamega','Carrión']
Lista_Rio_poly=[Q_rio21,Q_rio22,Q_rio23,Q_rio24]
Lista_Caudal=[caudal1,caudal2,caudal3,caudal4]
hora=[hora1,hora1,hora2,hora2]
```

Representacion senoidal

In [55]:

```
filas =  round(len(Nombres_Rios)/2)
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('Caudal de los ríos interpolación polinomial
trigonométrica',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Rio_poly)):
    plt.subplot(filas,2,i+1)

    plt.plot(t_total,Lista_Rio_poly[i],label = Nombres_Rios[i]+'
poly' )
    plt.plot(hora[i],Lista_Caudal[i],'o',label = Nombres_Rios[i] +
' datos')
```

```
    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```

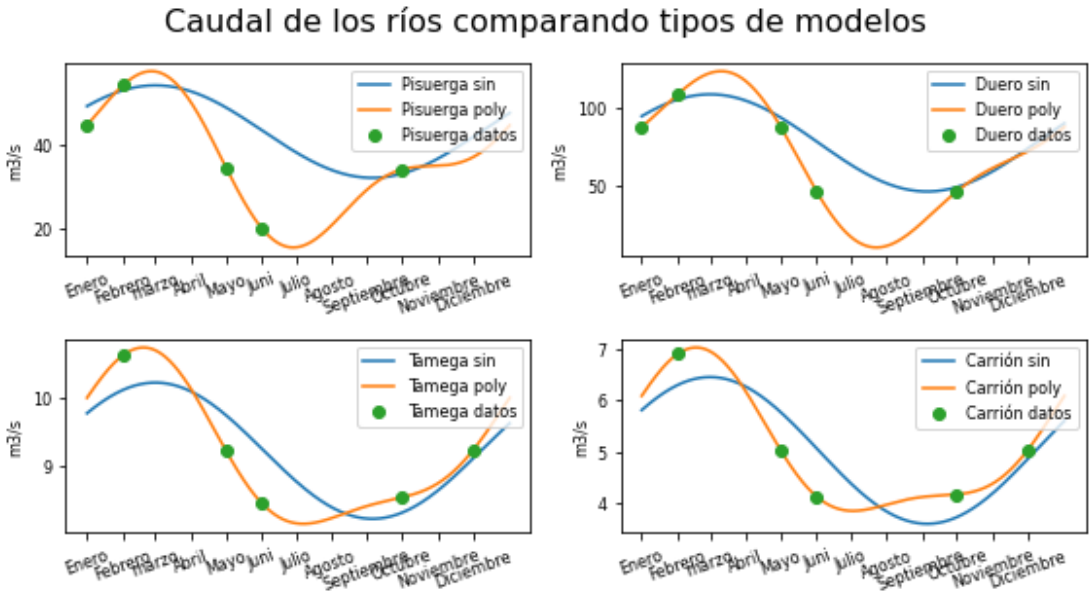


3.3 COMPARACIONES

In [56]:

```
filas =  round(len(Nombres_Rios)/2+1)
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('Caudal de los ríos comparando tipos de
modelos',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Rio_poly)):
    plt.subplot(filas,2,i+1)
    plt.plot(t_total,Lista_Rio_sin[i],label = Nombres_Rios[i]+'
sin' )
    plt.plot(t_total,Lista_Rio_poly[i],label = Nombres_Rios[i]+'
poly' )
    plt.plot(hora[i],Lista_Caudal[i],'o',label = Nombres_Rios[i] +
' datos')


    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)
```

```
plt.tight_layout()
plt.show()
```



3.4 RÍO SELLA

Datos del río Sella en las Cangas de Onís. En 2020 un caudal medio de 9,84 y en 2021 un caudal medio de 10,09. Su caudal máximo fue 18,42 y su caudal mínimo fue 4,16

In [57]:

```
lista_meses=[0,1,4,5,9]
hora=hm.lista_mes_hora(lista_meses)
```

In [58]:

```
Q_med1 = 10.09
Q_A1 =(18.42-4.16)/2
caudal1=[17,18.42,17,5,5]

Q_rio21 = hm.poly_odd(t_total,hora,caudal1,horas_año)
```

Lista para las representaciones

In [59]:

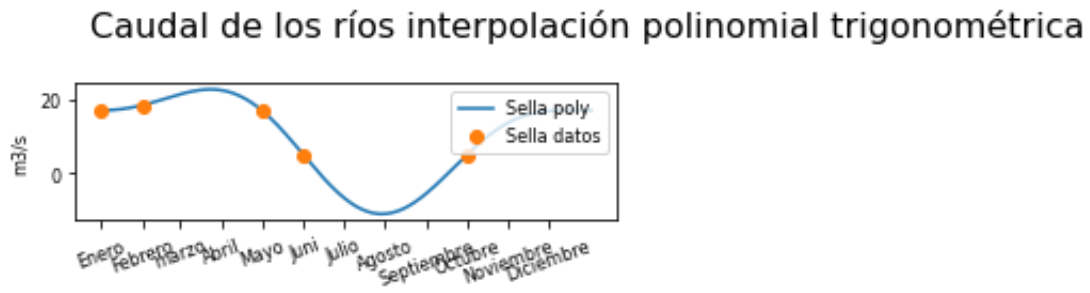
```
Nombres_Rios=['Sella']
Lista_Rio_poly=[Q_rio21]
Lista_Caudal=[caudal1]
Nombres_Rios_poly = Nombres_Rios
```

Representacion senoidal

In [60]:

```
filas =  round(len(Lista_Rio_poly)/2)+1
plt.figure(figsize=(WIDE, HEIGHT*filas))
```

```
plt.suptitle('Caudal de los ríos interpolación polinomial
trigonométrica',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Rio_poly)):
    plt.subplot(filas,2,i+1)

    plt.plot(t_total,Lista_Rio_poly[i],label =
Nombres_Rios_poly[i]+' poly' )
    plt.plot(hora,Lista_Caudal[i],'o',label = Nombres_Rios_poly[i]
+ ' datos')


    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



4. PRECIPITACIONES

Lluvia en Valladolid

In [61]:

```
Q_precipitaciones41=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Valladolid')
Q_lluvia1 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones41 ,
                                   area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

Lluvia en Zamora

In [62]:

```
Q_precipitaciones42=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Zamora')
Q_lluvia2 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones42 ,
```

```
                                    area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

Lluvia en Málaga

In [63]:

```
Q_precipitaciones43=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Malaga')
Q_lluvia3 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones43 ,
                                    area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

Lluvia en Segovia

In [64]:

```
Q_precipitaciones44=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'Segovia')
Q_lluvia4 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones44 ,
                                    area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

Lluvia en San Sebastián

In [65]:

```
Q_precipitaciones45=hm.precipitacion_hora_ciudad(hm.DatosPluviales
,'San Sebastian')
Q_lluvia5 = hm.volumen_preci_total(precipitacion_hora
=Q_precipitaciones45 ,
                                    area_abastecida=10000, #
presa2.base,

unidades_preci='mm',unidades_area='m2')
```

9.2 REPRESENTAR

Lista para las representaciones

In [66]:

```
Nombres_Provincia1=['Valladolid','Zamora']
Lista_precipitacion41=[Q_precipitaciones41,Q_precipitaciones42]
Lista_Caudal41=[Q_lluvia1,Q_lluvia2]
```

In [67]:

```
Nombres_Provincia2=['Malaga','Segovia','San Sebastian']
Lista_precipitacion42=[Q_precipitaciones43,Q_precipitaciones44,Q_p
recipitaciones45]
Lista_Caudal42=[Q_lluvia3,Q_lluvia4,Q_lluvia5]
```

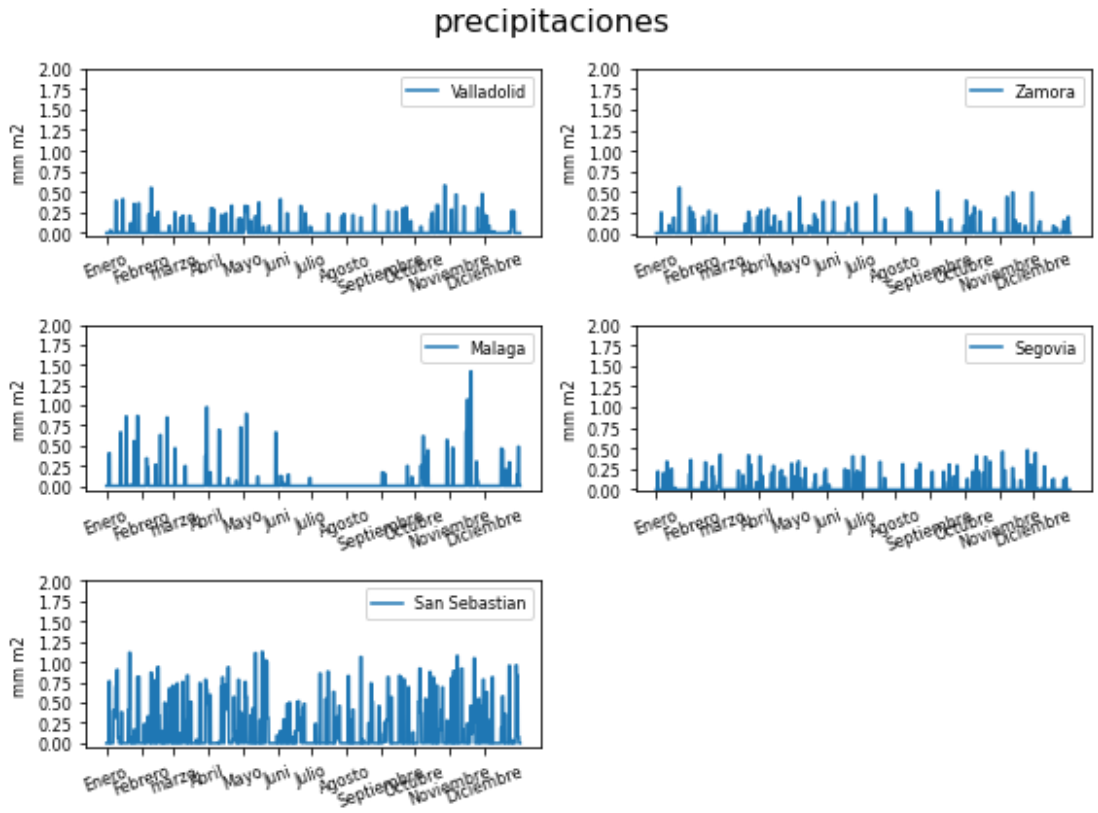
Representación senoidal

In [68]:

```
Nombres_Provincia = Nombres_Provincia1 + Nombres_Provincia2
Lista_precipitacion = Lista_precipitacion41 +
Lista_precipitacion42
Lista_Caudal = Lista_Caudal41 + Lista_Caudal42
```

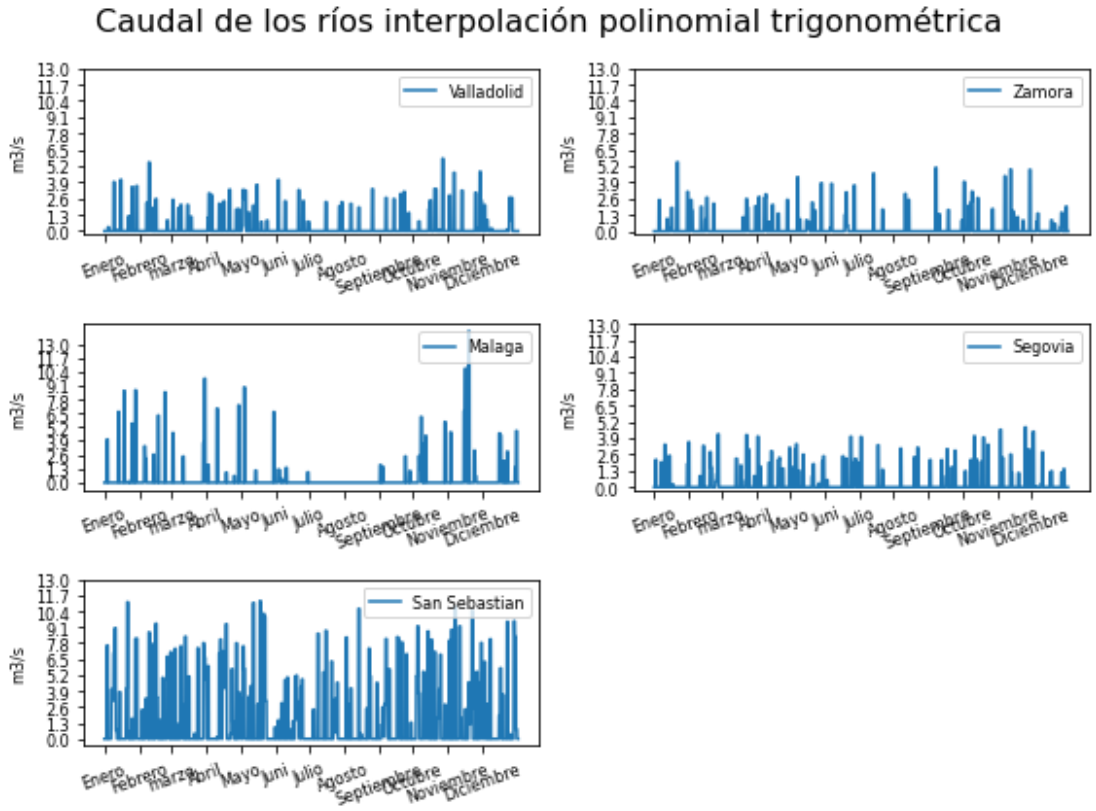
In [69]:

```
filas =  round(len(Nombres_Provincia)/2) +1
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('precipitaciones',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Caudal)):
    plt.subplot(filas,2,i+1)

    plt.plot(t_total,Lista_precipitacion[i],label =
Nombres_Provincia[i] )

    plt.ylabel('mm m2', fontsize=LETTER_SIZE)
    plt.yticks(np.linspace(0,2,9),fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



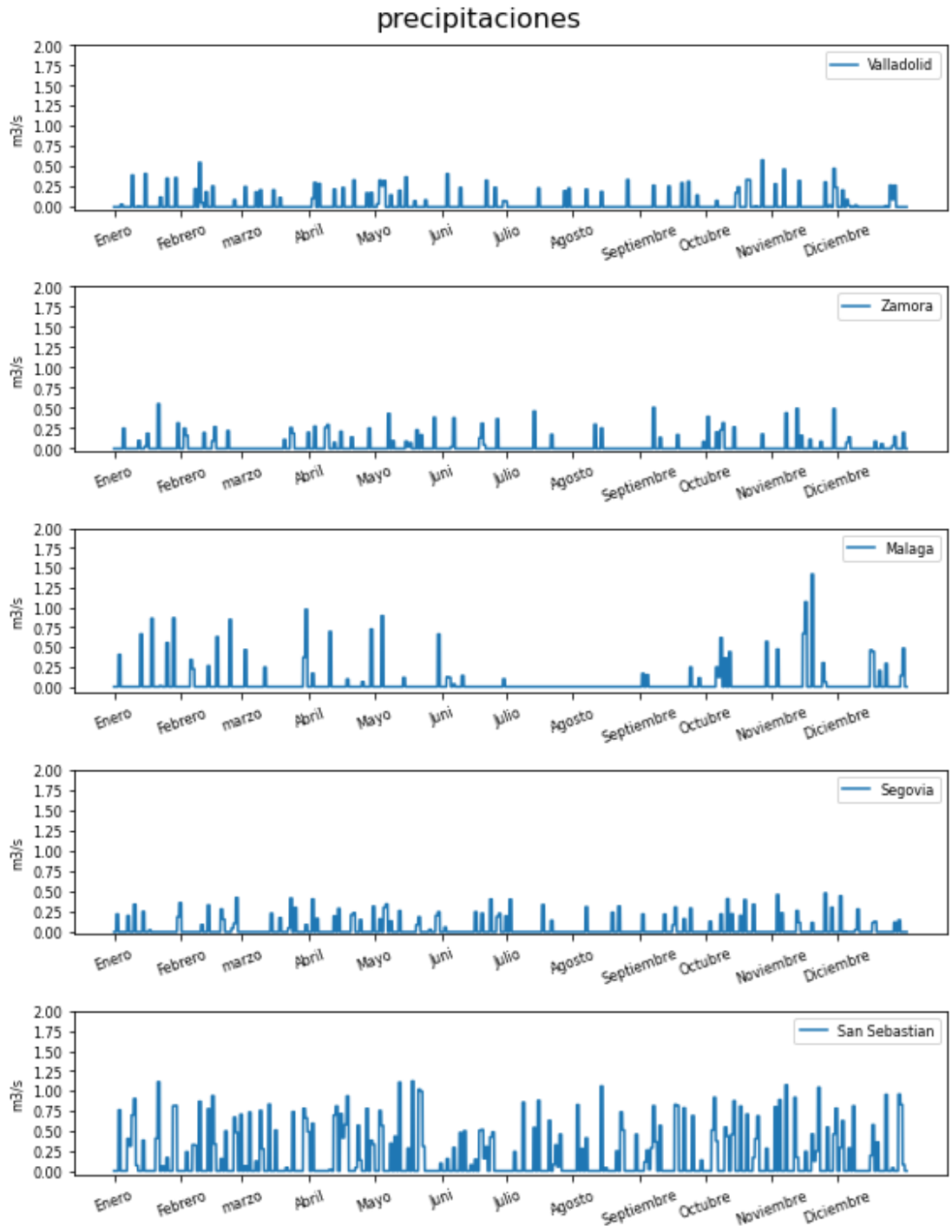
In [70]:

```
filas =  round(len(Nombres_Provincia)/2)+1
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('Caudal de los ríos interpolación polinomial
trigonométrica',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Caudal)):
    plt.subplot(filas,2,i+1)

    plt.plot(t_total,Lista_Caudal[i],label = Nombres_Provincia[i]
)

    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(np.linspace(0,13,11),fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



In [71]:

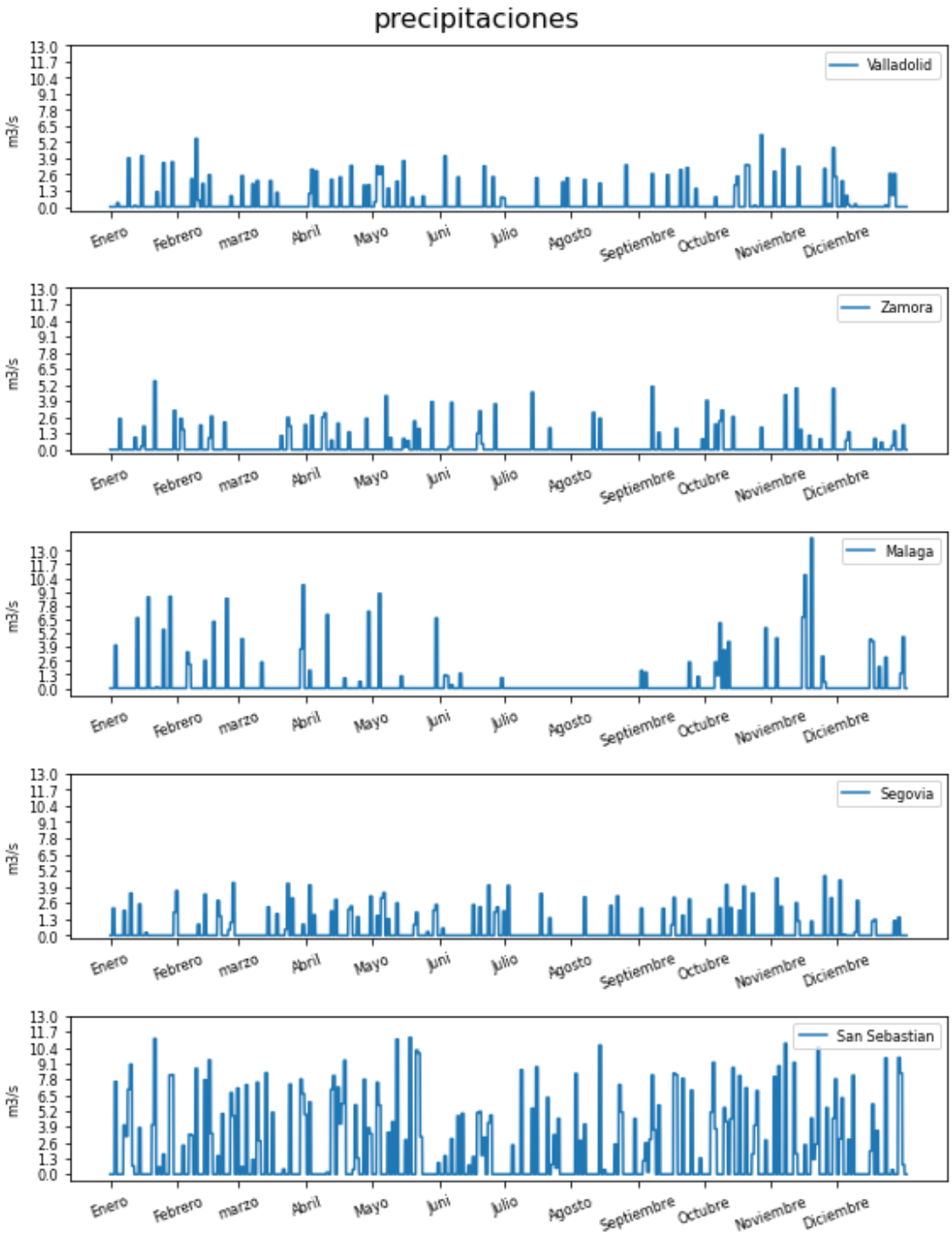
```
filas =  round(len(Nombres_Provincia))+1
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('precipitaciones',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Caudal)):
    plt.subplot(filas,1,i+1)

    plt.plot(t_total,Lista_precipitacion[i],label =
Nombres_Provincia[i] )

    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(np.linspace(0,2,9),fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```



In [72]:

```
filas =  round(len(Nombres_Provincia))+1
plt.figure(figsize=(WIDE, HEIGHT*filas))

plt.suptitle('precipitaciones',fontsize=LETTER_SIZE*2)
for i in range(len(Lista_Caudal)):
    plt.subplot(filas,1,i+1)
```

```
    plt.plot(t_total,Lista_Caudal[i],label = Nombres_Provincia[i]
)

    plt.ylabel('m3/s', fontsize=LETTER_SIZE)
    plt.yticks(np.linspace(0,13,11),fontsize=LETTER_SIZE)
    plt.xticks(horaMes, meses,fontsize=LETTER_SIZE,rotation=20)
    plt.legend(loc=1,fontsize=LETTER_SIZE)

plt.tight_layout()
plt.show()
```


precipitaciones
Valladolid
Zamora
Malaga
Segovia
San Sebastian
m3/s
13.0
11.7
10.4
9.1
7.8
6.5
5.2
3.9
2.6
1.3
0.0
Enero
Febrero
marzo
Abril
Mayo
Juni
Julio
Agosto
Septiembre
Octubre
Noviembre
Diciembre