UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

CARACTERIZACIÓN EXPERIMENTAL DE UN CIRCUITO GENÉTICO SINTÉTICO

AUTOR: LUIS VILLALBA PÉREZ

TUTOR: JESÚS PICÓ MARCÓ

COTUTOR: JOSÉ LUIS NAVARRO HERRERO

Curso Académico: 2016-17

TABLA DE CONTENIDO

TABLA DE IMÁGENES

ÍNDICE DE TABLAS

RESUMEN

Mediante el presente trabajo se pretende caracterizar el rango de valores adoptables por los parámetros de un modelo dinámico en un circuito biológico mediante optimización multiobjetivo.

El circuito genético utilizado para este experimento consiste en un sistema incoherente de tipo I que actuará como sensor a la introducción de una proteína. Este circuito ha sido elegido dada su simplicidad y ubicuidad como elemento básico en otros circuitos. Esto se debe a la importancia del desarrollo de circuitos biológicos que actúen como módulos estándar para su uso en el diseño de conjuntos más complejos. A su vez, su función de sensor es una representación del gran potencial y versatilidad que tiene hoy en día el campo de la biología sintética.

Durante el trabajo se describirá en detalle el modelo matemático utilizado y se indicarán los pasos realizados para su obtención. Se detallará su proceso de implementación y reducción para su uso computacional, así como los métodos utilizados para la optimización e identificación de los parámetros.

A continuación, se detallará el proceso de obtención de un *software* con la mayor estabilidad posible, sencillo de utilizar y alterar para el uso de diferentes modelos matemáticos en el proceso de optimización multiobjetivo. Este *software* deberá poder leer los datos experimentales de diversos experimentos con las menores variaciones al código posibles, obtener las diferentes soluciones dentro de la frontera de Pareto proporcionadas por la optimización y validación posterior cualitativa y cuantitativa mediante datos experimentales.

Finalmente utilizando este código se realizará la identificación de diversos parámetros del modelo y como, la variación de rangos de búsqueda puede afectar a una mejor o peor adaptación a los datos experimentales. El análisis de los conjuntos de soluciones obtenidos se llevará acabo usando técnicas de validación cualitativa y cuantitativa del error.

PARTE I:
MEMORIA

1 INTRODUCCIÓN

1.1 Biología sintética

El avance de la ciencia y la técnica ha permitido el surgimiento y la expansión de nuevos campos de investigación llenos de potencial y promesas tanto para los propios investigadores que tienen la oportunidad de dedicarse a ellos como para el resto de la sociedad. Y pocos de estos campos han alcanzado, en los últimos años, el seguimiento que ha recibido la Biología Sintética [1]. Desde su incorporación a la vida universitaria con nuevas carreras centradas en su estudio hasta la proliferación de publicaciones, equipos de investigación y artículos de divulgación a nivel global.

Este campo surgido hace poco más de unas décadas [2], ha visto un continuo aumento en el número de investigaciones y artículos publicados, habiéndose multiplicado el número de estos desde el 2008 hasta la actualidad. La capacidad para modificar y crear circuitos orgánicos a partir de piezas diseñadas individualmente ha captado la atención de campos previamente tan apartados como la biología y la ingeniería [3] debido al inmenso número de posibilidades que presentan estos nuevos circuitos. Ya que todos los ámbitos de la sociedad actual que trabajan con material biológico pueden ser revolucionados por las técnicas y descubrimientos que se están desarrollando en esta nueva disciplina.

Muchos procesos industriales basan la obtención de materia prima o elaboración de producto final en procesos biológicos que podrían ser optimizados en gran medida al aplicárseles un control más eficiente. La biología sintética ofrece, por ejemplo, la posibilidad de controlar individualmente las células que participan en un proceso de fermentación o compostaje en vez de limitarse a controlar las condiciones generales de la población. También la ciencia médica ha empezado a beneficiarse de esta nueva tecnología, desde mejoras en la tecnología de implantes hasta en la producción de medicamentos… las posibilidades de aplicaciones son casi ilimitadas.

Como se podrá observar en el presente trabajo, una de las mayores posibilidades de la Biología Sintética es la elaboración de células que sirvan como sensores o reguladores. Esto aprovecha la gran aptitud que las células biológicas han ido desarrollando a lo largo de siglos de evolución para detectar los cambios más variados en su entorno y realizar todo tipo de funciones en consecuencia. Así, se podrían utilizar para fines muy variados como la sustitución de técnicas médicas de identificación más invasivas y perjudiciales para los pacientes. Como se ha indicado anteriormente, las aplicaciones de este campo son innumerables.

1.2 Optimización Multiobjetivo

La modelización de los nuevos circuitos es esencial en el campo de la biología sintética [4] [5]. La capacidad de describir correctamente el comportamiento de experimentos biológicos, caracterizados por una gran incertidumbre y cantidad de factores que influyen en su respuesta, es un gran reto de los investigadores actuales.

Para resolver este problema, se propone el uso de Optimización Multiobjetivo en la identificación y control de circuitos biológicos. Esta forma de diseñar afecta tanto a la optimización del sistema como a la forma misma de plantear los objetivos. Está técnica proporcionará al usuario más información útil sobre su sistema (rangos cualitativos de sus parámetros, indicaciones de su comportamiento en su implementación…) y más detallada que con otros métodos tradicionales de optimización como la asignación de prioridad a objetivos. Esto se debe a que en la mayoría de circuitos dinámicos con una cierta complejidad, se presenta una gran cantidad de objetivos a optimizar y por lo tanto es necesario una compensación entre ellos.

1.3 Objetivo del trabajo

El presente trabajo nace dentro de la investigación llevada a cabo por el Instituto Universitario ai2 [4], para desarrollar y consolidar el método de Optimización Multiobjetivo para circuitos de biología sintética, con el fin de ayudar en el desarrollo de este método. De una forma más concreta, el presente trabajo se centrará en la caracterización de los parámetros sin valor conocidos en un modelo matemático que describe el comportamiento de un circuito biológico incoherente de tipo I.

1.4 Alcance del trabajo

Durante la realización del presente trabajo se utilizarán diferentes técnicas de tratamientos de datos experimentales para seleccionar datos utilizables para la posterior identificación del modelo. Además, pese a que el modelo matemático ya ha sido proporcionado por el equipo de investigación, se han usado técnicas de reducción matemática del modelo para facilitar su implementación computacional. Modelo que se optimizará mediante, el proceso multiobjetivo descrito anteriormente.

A su vez, para el presente trabajo, se ha utilizado el código proporcionado por el equipo de investigación del Instituto Universitario ai2, para realizar la optimización. Será competencia del presente trabajo el optimizar este *software* y crear las partes faltantes para aumentar el alcance del mismo para ser posible su utilización en otros experimentos de optimización y facilitar su uso a aquellos no familiarizados con este. Para ello se aplicarán conocimientos de programación, diseño de interfaces y tratamiento de bases de datos.

Dado al objetivo de acercar el *software* a usuarios nuevos que vayan a colaborar con el proyecto de investigación posteriormente o en otros proyectos similares, el presente trabajo no requiere un conocimiento profundo del campo de la biología sintética. Sí se recomienda una familiaridad básica con los procesos de identificación, optimización y modelización de sistemas dinámicos.

1.5 Resumen

2. Normativa.

Reglamento y legislación vigente durante la realización del presente trabajo y pertinente a él. Referente, en su mayoría, a precauciones de seguridad necesarias durante el trabajo de laboratorio, necesario para la obtención de datos experimentales, en el que intervengan microorganismos, sustancias químicas, poblaciones celulares u otro tipo de material biológico.

3. Planteamiento

El modelo que se ha optimizado en el presente trabajo corresponde a un sistema incoherente de tipo I que describe la concentración de 3 proteínas dentro de una célula tras una entrada de escalón. En este apartado se describirán las ecuaciones, las variables y los parámetros que comprenden este sistema y se comentará brevemente su obtención. También se introduce el experimento que se ha realizado para obtener los datos experimentales con los que realizaremos la identificación.

4. Selección de datos experimentales

Como se ha indicado anteriormente, los circuitos biológicos se caracterizan por actuar con una gran incertidumbre. Por ello, para realizar una identificación eficaz, como requiere el presente trabajo, será necesario realizar una selección de los datos provenientes de aquellos experimentos que presenten menos perturbaciones o desviaciones de la respuesta teórica del sistema. En este apartado se describen las características de dicha respuesta, así como los criterios utilizados durante la selección.

5. Elaboración del *software* de optimización

En esta sección se describe el funcionamiento de la versión final del *software* elaborado para la realización del presente trabajo, así como las instrucciones de uso para sus interfaces, destinadas a nuevos usuarios. También se indican los pasos seguidos durante su elaboración a partir del código original proporcionado por el equipo e investigación del Instituto Universitario ai2.

5. Caracterización

Incluye los resultados obtenidos tras utilizar el *software* descrito en el apartado anterior con los datos experimentales seleccionados y la información que se ha obtenido de ellos. Tanto sobre la validez del modelo utilizado, como sobre funcionalidades adicionales necesarias para el *software* e indicaciones de modificaciones necesarias para los intervalos de valores posibles para los parámetros.

6. Conclusión

Conclusiones obtenidas a partir de toda la información mencionada. Asimismo, se incluye información adicional sobre las dificultades encontradas durante la realización del trabajo y medidas utilizadas para solucionarlas y simplificar el desarrollo de otros proyectos o la continuación del trabajo con el *software* que se ha elaborado en este.

7. Presupuesto

Detalla los gastos concurridos durante el trabajo y a que se han dedicado, también se Incluye el cómputo de las horas dedicadas a la realización del mismo por la mano de obra y el coste del equipo necesario para realizar los experimentos y crear el circuito biológico sobre el que se basa el proceso. Finalmente concluye con el cálculo del coste total incluyendo el impuesto sobre el valor añadido.

8. Bibliografía

Documentación que forma la base teórica sobre la que se construye el presente trabajo.

2 NORMATIVA

La mayor parte del presente trabajo se centra en trabajo de elaboración y modificación de *software* existente. Todos los scripts y funciones cedidos por el equipo de investigación del Instituto Universitario ai2 tienen su respectivo Copyright (Copyright 2006-2012-CPOH) que debe respetarse durante su uso, modificación y, en caso de ser necesario, distribución a nuevos usuarios.

Los datos experimentales se obtienen mediante trabajo en el laboratorio que implica el uso de agentes biológicos y substancias químicas que pueden presentar un riesgo para la salud. Es necesario la lectura, entendimiento y aplicación, por parte de los responsables del trabajo en el laboratorio, de la normativa vigente en la Comunitat Valenciana. De esta forma se podrán evitar y comprender los posibles riesgos en su totalidad. Esta normativa se encuentra reunida en los siguientes decretos y manuales.

Guía técnica para la evaluación y prevención de los riesgos relacionados con la exposición a agentes biológicos. Real Decreto 667/1997, de 12 de mayo BOE nº124, de 24 de mayo. Ministerio de Trabajo y Asuntos Sociales & Instituto Nacional de Seguridad e Higiene en el Trabajo.

Marca obligaciones para trabajos en el laboratorio y, más concretamente, proyectos educativos universitarios. Estas incluyen la presencia de personal docente e investigación y preparación previa de las instalaciones antes de realizar el trabajo de laboratorio.

También define los agentes biológicos utilizados en trabajos de investigación como «*microorganismos, incluyendo aquellos modificados genéticamente, culturas celulares y endoparásitos humanos, responsables de cualquier tipo de infección, alergia o toxicidad.»* Y clasifica los microorganismos «*toda entidad biológica, celular o no, capaz de replicar o transferir material genético»*, según su riesgo. La población celular utilizada en este experimento corresponde al grupo de **agentes biológicos de grupo 1**, «*aquellos poco probables de causar enfermedades a un ser humano».*

Esta clasificación concuerda con la encontrada en el **Real decreto 363/1995** del 10 de Marzo, que indica que en este experimento no se utilizará ningún *compuesto peligroso*.

Manual de Bioseguridad en el Laboratorio 3ra Edición Organización Mundial de la Salud (OMS), 1983.

De acuerdo a la clasificación descrita anteriormente, durante nuestro trabajo se utilizarán los **Métodos de control de grupo 1** (1º Nivel de Control) descritos en este manual. Que incluye medidas generales como el uso de material protector, incluyendo bata de laboratorio y protección ocular y medidas de higiene básicas, incluyendo la descontaminación de superficies de trabajo al

menos una vez al día y la obligación de que el personal se lave las manos antes de salir del laboratorio si se ha manipulado material infeccioso.

También incluye medidas más concretas para el uso de material biológico, que debe estar correctamente etiquetados durante su almacenaje y transporte, el cual no debe realizarse sin las precauciones adecuadas de forma que, en caso de caída, no salpique y no debe hacerse a mano. En caso de que sea necesario el almacenaje en nitrógeno líquido, el personal utilizará protección ocular y facial. Si se rompiera el contenedor, el nitrógeno tiene que evaporarse antes de proceder con la limpieza.

Reglamento Regulador de la Gestión de los Residuos Sanitarios. Orden del 14 de Julio 1997 de la Conselleria de Medio Ambiente de la C.V. incluyendo el Decreto 240/1994 de la Ley 10/2000 del 12 de Diciembre sobre **Residuos de la Comunidad Valenciana**

Clasifica los agentes biológicos que se desechan del laboratorio en:

- Residuos biológicos sólidos, como residuos urbanos.
- Residuos biológicos sólidos especiales.
- Residuos sólidos no patogénicos provenientes de culturas microbiológicas. - Residuos líquidos biológicos.

Las formas de tratamiento de residuos para el **Nivel de Control 1** se describen también en el Manual de Bioseguridad en el Laboratorio 3ra Edición. Este manual encomienda el reciclaje y reutilización de todo el material posible. Aquel material biológico que sea potencialmente infeccioso debe ser descontaminado o desinfectado mediante un procedimiento aprobado, o ser empaquetado o incinerado de forma adecuada.

«*El método de descontaminación estándar es la esterilización por autoclave a vapor. Utilizando métodos alternativos solo si se ha comprobado su eficacia para eliminar y/o matar microorganismos*.»

Manual de seguridad para operaciones en laboratorios de biotecnología y de tipo biológico. UPV.

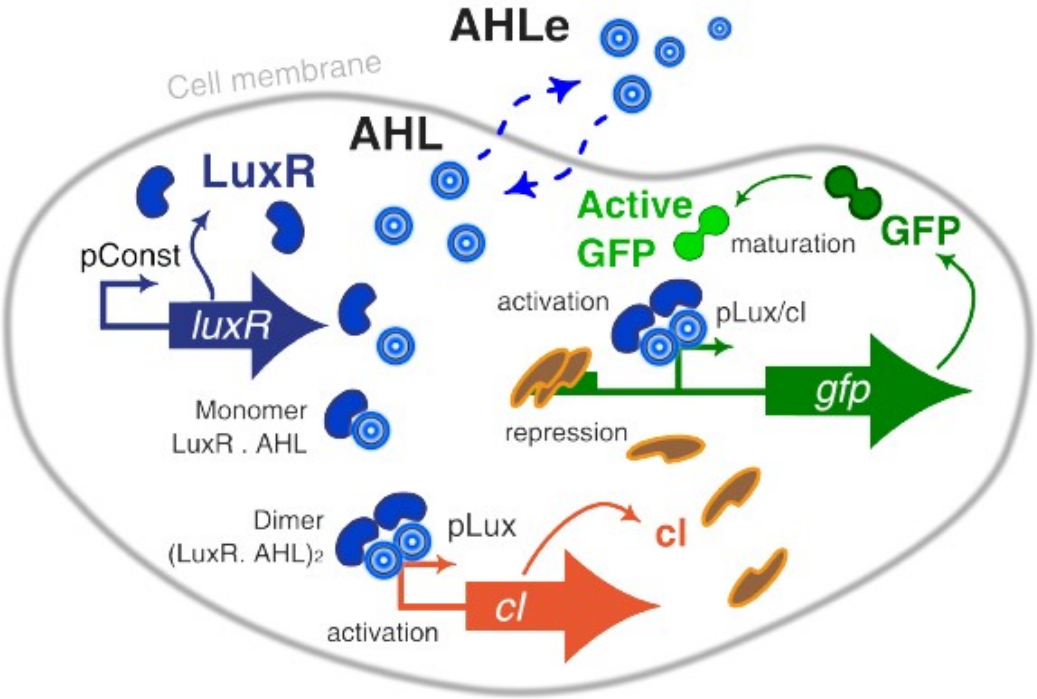
Proporciona normas de seguridad generales para cumplir durante la realización del experimento. Las más relevantes para nuestro caso corresponden con la necesidad de mantener el máximo orden y pulcritud durante y tras la realización del experimento. Mantener la normativa de espacio mínimo por trabajador ($2m^2$ y $10m^3$ sin usar por personal alguno), así como cumplir la ley de temperaturas descrita en el **Real Decreto 486/1997 Anexo III** e iluminación del **Anexo IV** del mismo decreto.

3 PLANTEAMIENTO

3.1 Circuito biológico

Antes de proceder a la descripción del trabajo realizado durante este proyecto, es necesario detallar el circuito biológico utilizado y el modelo matemático que se ha obtenido de él. Para ello hay que recalcar la importancia de la adaptación frente a señales externas en los circuitos biológicos y recordar, como se ha indicado en la introducción, las posibilidades que este tipo de sistemas ofrecen como sensores y reguladores. La adaptación a su medio y a los agentes que hay en este es clave para la supervivencia de los microorganismos, el conjunto de reacciones que estos hacen para volver a un estado de equilibrio tras un cambio en el medio se conoce como homeóstasis [6] y es en lo que se basa el circuito utilizado.

Figura 1: Circuito biológico con tres nodos. Imagen obtenida de [7]



El presente trabajo se ha llevado a cabo en colaboración con el equipo de investigación del Instituto Universitario ai2 en uno de sus proyectos, por lo que el circuito biológico y su correspondiente modelo diferencial han sido elaborados previamente como se recoge en [7]. Pero antes de describirlo hay que introducir el circuito que describe.

Se ha utilizado un sistema incoherente de tipo I que actúa sobre tres nodos: los genes gA, gB y gC que producirán las proteínas A, B y C respectivamente. En el medio donde se encuentran las células se introduce una biomolécula como inductor. Diferenciamos a la hora del modelo el inductor externo Ie, que se encuentra fuera de las células y, por lo tanto, no interactúa directamente con los genes y el inductor que entra en la célula I.

El gen gA permanece en constante funcionamiento, produciendo proteína A. Cuando la concentración de Ie empieza a subir, estas dos proteínas forman el monómero AI. Así mismo, cuando la concentración de este monómero sube, empiezan a formarse dímeros $(AI)_2$ que activan el funcionamiento de gB y gC. La proteína C que produce gC causa la fluorescencia de la célula y la proteína B dimerizará consigo misma para actuar como represor de gC. Tras un tiempo, el circuito vuelve al equilibrio inicial donde la producción de C se ha detenido y la que se había producido se ha degradado por completo, volviendo a la fluorescencia inicial.

Para montar este circuito durante la obtención de datos experimentales se utilizan piezas estándar (obtenidas del Registry of Standard Biological Parts de la Biobrick Foundation, 2004) que nos permitirán determinar los parámetros del modelo dependiendo de las cadenas de sus nucleótidos en diferentes regiones:

- **Región de código**: contiene la información de una proteína y su velocidad de degradación. Es decir, de esta región depende la producción de un gen y la desaparición gradual del producto creado.

- **Promotor**: controla la unión de una proteína determinada a un gen para aumentar o detener la producción de este gen. Controlan los parámetros de degradación y transcripción de mRNA.

- **Región de unión del ribosoma**: en la cual se puede unir el mRNA para empezar la traducción. De esta zona depende la velocidad de traducción.

- **Terminador**: finaliza la traducción del mRNA.

Estas regiones se introducen en las células utilizando los ***Biobricks*** (Ladrillos biológicos) estandarizados para el fin de crear nuevos sistemas biológicos. Estos ladrillos incluyen promotores, terminadores… que se pueden consultar en diversos catálogos [8]. Para inocular estos ladrillos en las células E. Coli se utiliza un vector que se reproduce dentro de la célula huésped.

Los *Biobricks* utilizados en este circuito son los siguientes (obtenidos del catálogo indicado anteriormente) y cuyo precio se desglosa en el apartado de presupuesto:

Tabla 1: Labriollos biológicos utilizados para formar el circuito

Gen	Nombre	Promotor	RBS	Región de código	Terminador
gA	LuxR	BBa_J23106	BBa_B0034	BBa_C0062	BBa_B1006
gB	CI-LVA	BBa_R0062	BBa_B0034	BBa_K327018	BBa_B1006

gC	GFP-LVA	BBa_K415032	BBa_B0034	BBa_K18001	BBa_K259006

3.2 Modelo matemático

El modelo original se desarrolló mediante un acercamiento determinístico tal y como como se describe en [9], se ha alcanzado el siguiente modelo (para una sola célula):

Tabla 2: Modelo matemático original [7]

$$\dot{x}_1 = k_{mA} C_{gA} - d_{mA} x_1$$

$$\dot{x}_2 = k_{pA} x_1 - d_A x_2 - k_2 x_2 x_3 + k_{-2} M$$

$$\dot{x}_3 = -k_2 x_2 x_3 + k_{-2} M + k_d (x_9 - x_3) - d_I x_3$$

$$\dot{x}_4 = k_3 M^2 - k_{-3x_4} - d_{AI2} x_4$$

$$\dot{x}_5 = K_{mB} C_{gB} \frac{x_4}{\gamma_1 + x_4} - d_{mB} x_5$$

$$\dot{x}_6 = k_{pB} x_5 - d_B x_6$$

$$\dot{x}_7 = K_{mC} C_{gC} \frac{x_4 + \beta_1 \gamma_4 x_6 + \beta_2 \gamma_5 x_4 x_6}{\gamma_2 + \gamma_3 x_4 + \gamma_5 x_4 x_6} - d_{mC} x_7$$

$$\dot{x}_8 = k_{pC} x_7 - d_C x_8$$

$$\dot{x}_9 = K_{cells} k_d (-x_9 + x_3) - d_{Ie} x_9$$

$$M = -\frac{d_{AI} + k_{-2}}{4k_3} + \frac{1}{4k_3}\sqrt{(d_{AI} + k_{-2})^2 + 8k_3(k_2 x_2 x_3 + 2k_{-3} x_4)}$$

$$K_{cells} = \frac{V_{cell} N_{cells}}{V_{medium}}$$

En este modelo los tres genes que producen las proteínas A, B y C se denominan con las letras Cg y el sufijo correspondiente. M es el monómero de concentración en el medio de una célula y K_{cells} refleja el tráfico entrando o saliendo de la célula, basado en el volumen típico de una célula E.coli (10^{-15}L),un número de células N_{cells}=2,4x10^8 células/mL*0,18 mL en un medio V_{medium}=180µL.

Cada una de las 9 ecuaciones diferenciales representa la evolución en el tiempo de uno de los componentes del sistema (proteínas, monómeros o dímeros compuestos por estas) y se utilizan 27 parámetros, como se describe en las tablas 3 y 4.

Tabla 3: Variables del modelo [10]

VARIABLE	DESCRIPCIÓN	UNIDADES	SÍMBOLO
x_1	$mRNA_{gA}$	nM	mA
x_2	Proteína A	nM	A
x_3	Inductor	nM	I
M	Monómero AI	nM	AI
x_4	Dímero $(AI)_2$	nM	$(AI)_2$
x_5	$mRNA_{gB}$	nM	mB
x_6	Proteína B	nM	B
x_7	$mRNA_{gC}$	nM	mC
x_8	Proteína C	nM	C
x_9	Inductor extracelular	nM	I_e

Tabla 4: Parámetros del modelo [10]

PARÁMETRO	DESCRIPCIÓN	UNIDADES
C_{gA}, C_{gB}, C_{gC}	Genes productores de A, B y C	---
k_{mA}, k_{mB}, k_{mC}	Velocidad de transcripción de los genes	1/min
d_{mA}, d_{mB}, d_{mC}	Velocidad de degradación de los genes	1/min
k_{pA}, k_{pB}, k_{pC}	Velocidad de traducción de los genes	1/min
d_A, d_B, d_C	Velocidad de degradación de las proteínas	1/min
k_d	Velocidad de difusión del inductor	1/min
k_2, k_3	Velocidad de asociación de (AI) y $(AI)_2$	1/min
k_{-2}, k_{-3}	Velocidad de disociación de (AI) y $(AI)_2$	1/min
γ_1	Constante de promoción Hill de gB	nM
$\gamma_2, \gamma_3, \gamma_4, \gamma_5$	Coeficientes de promoción de gC	nM,---,---,1/nM
β_1, β_2	Coeficientes de promoción basales de gC	---, 1/nM
d_I, d_{Ie}	Velocidad de degradación del inductor	1/min
d_{AI}, d_{AI2}	Velocidad de degradación de (AI) y $(AI)_2$	1/min

Con el objetivo de facilitar la implementación del modelo y, utilizando nueva información sobre el comportamiento del circuito, se ha alcanzado una nueva iteración del modelo más simple. Las variaciones en los valores de las concentraciones del mRNA de las proteínas son muy rápidas, por lo que se puede asumir estado estacionario e igualar sus derivadas a 0 (ecuaciones 1, 5 y 7) de forma que se puede obtener la ecuación de dichas concentraciones y eliminar tres expresiones del modelo anterior.

Tabla 5: Reducción del modelo

$\mathbf{0} = \boldsymbol{k_{mA} C_{gA} * K_{pA} - d_{mA} x_1}$	$\boldsymbol{KA = \frac{k_{mA} C_{gA} * K_{pA}}{d_{mA}}}$
	$\boldsymbol{KB = \frac{k_{mB} C_{gB} * K_{pB}}{d_{mB}}}$
	$\boldsymbol{KC = \frac{k_{mC} C_{gC} * K_{pC}}{d_{mC}}}$

Al sustituir estas nuevas expresiones en el resto de ecuaciones obtenemos un modelo reducido. Este modelo se puede simplificar agrupando los parámetros antiguos en unos nuevos que los engloben. También se puede sacar factor común en la expresión de M e introducir la definición de Kcells en la ecuación del inductor extracelular.

También se ha añadido al modelo el comportamiento correcto de la proteína B. Esta forma un dímero con otros ejemplares de proteína B para actuar sobre el gen C. Esto no estaba incluido en el modelo original y se ha introducido elevando al cuadrado la variable x_5, como se puede ver en la ecuación de la concentración de la proteína C sobre la que actúa (Tabla 6).

Tabla 6: Modelo para el sistema incoherente de tipo I reducido

$$\boldsymbol{A = \dot{x}_2 = k_A - k_2 x_2 x_3 + k_{-2} M - (d_A + \mu) x_2}$$

$$\boldsymbol{I = \dot{x}_3 = -k_2 x_2 x_3 + k_{-2} M + k_d (x_9 - x_3) - (d_I + \mu) x_3}$$

$$\boldsymbol{(AI)_2 = \dot{x}_4 = -k_3 M^2 - k_{-3} x_4 - (d_{AI2} + \mu) x_4}$$

$$\boldsymbol{B = \dot{x}_6 = k_B \frac{x_1}{\alpha_1 + x_1} - (d_B + \mu) x_6}$$

$$\boldsymbol{C = \dot{x}_8 = k_C \frac{\gamma_1 x_1 + \gamma_2 x_1 {x_5}^2}{1 + \gamma_3 x_1 + \gamma_4 {x_5}^2} - (d_C + \mu) x_8}$$

$$\boldsymbol{Ie = \dot{x}_9 = -\frac{k_d V_{cell} x_7}{V_{culture}} (-x_9 + x_3) - d_{Ie} x_9}$$

$$N = \mu x_7 \frac{(1 - x_7)}{k_{max}}$$

$$M = \frac{1}{4k_3}\left[-(d_{AI} + k_{-2} + \mu) + \sqrt{(d_{AI} + k_{-2} + \mu)^2 + 8k_3(k_2 x_2 x_3 + 2k_{-3} x_4)}\right]$$

Donde debe cumplirse: $\gamma_0 < 1$, $\gamma_1 < \gamma_3$ y $\gamma_2 < \gamma_5$. Esta versión reducida del modelo cuenta solo con 7 ecuaciones, ya que el valor de M será utilizado como otro parámetro que se calcula en cada iteración y se introduce en las ecuaciones que lo utilizan.

Muchos de los parámetros de este modelo reducido ya tienen un valor fijo estimado obtenido mediante experimentos anteriores y datos publicados en la biografía publicada sobre las partes estándar utilizadas. Estos valores se mantienen de la versión anterior del modelo. Los valores de dichos parámetros se encuentran en la siguiente tabla:

Tabla 7: Valores de parámetros en el modelo final [7], [10]

Parámetro	Valor	Unidades	Significado
μ	0.028	$\frac{1}{min}$	Velocidad de crecimiento de E.Coli
k_d	2	$\frac{1}{min}$	Velocidad de difusión a través de la membrana celular
d_I	0.01	$\frac{1}{min}$	Velocidad de degradación del inductor
γ_2	0.02	$\frac{1}{min}$	$\beta_2 * \gamma_5$
V_{cell}	$1 * 10^{-9}$	$\frac{microlitros}{célula}$	Volumen por célula
$V_{culture}$	180	$microlitros$	Volumen de la población
d_{Ie}	0.01	$\frac{1}{min}$	Velocidad de degradación del inductor externo
k_{max}	Depende de la cantidad de células usadas en cada experimento	$N\ células$	Capacidad máxima de células en un lote de células
d_{AI}	0.0174	$\frac{1}{min}$	Velocidad de degradación del monómero

Lo que deja una cantidad mucho menor de parámetros a identificar con el nuevo modelo. Concretamente tenemos 17 parámetros cuyo valor es desconocido:

Tabla 8: Parámetros a estimar en el modelo final

Parámetro	Unidades
$\boldsymbol{\gamma_1}$	$\frac{1}{nM}$
$\boldsymbol{\gamma_3}$	---
$\boldsymbol{\gamma_4}$	---
$\boldsymbol{\gamma_5}$	$\frac{1}{nM^2}$
$\boldsymbol{k_A}$	---
$\boldsymbol{k_B}$	---
$\boldsymbol{k_C}$	---
$\boldsymbol{k_{-2}}$	$\frac{1}{min}$
$\boldsymbol{k_2}$	$\frac{1}{min}$
$\boldsymbol{k_{-3}}$	$\frac{1}{min}$
$\boldsymbol{k_3}$	$\frac{1}{min}$
$\boldsymbol{d_A}$	$\frac{1}{min}$
$\boldsymbol{d_B}$	$\frac{1}{min}$
$\boldsymbol{d_C}$	$\frac{1}{min}$

Para la optimización multiobjetivo utilizaremos la fluorescencia de las células como salida del sistema. Dado que esta propiedad está relacionada con la concentración de proteína C en la célula, su ecuación correspondiente será la utilizada para la validación de los parámetros identificados. La aparición de múltiples objetivos para la optimización se deberá al hecho de que en un mismo experimento están presentes diversas poblaciones de células con diferentes concentraciones de inoculador I. Realizando los experimentos de esta manera se puede buscar

valores para los parámetros que cubran diferentes comportamientos del circuito aumentando su versatilidad.

3.3 Estado inicial

Para acelerar la optimización que realizará el *software*, le introduciremos las expresiones de las ecuaciones en el estado permanente inicial. Este cálculo resulta muy sencillo dado que conocemos todas las condiciones de inicio del experimento.

La población de células permanecerá en un estado permanente hasta la inoculación del inductor I, y las ecuaciones que describan la concentración del inductor empezarán a 0 ya que no está de forma natural en el medio hasta su inoculación. Esto incluye también la concentración del monómero y el dímero que el inductor forma con la proteína A y la ecuación M ya que depende de la degradación del inductor.

El resto de ecuaciones tendrán expresiones muy simplificadas debido a estas condiciones:

Tabla 9: Ecuaciones en estado inicial

$$\boldsymbol{A = \frac{k_A}{(d_A + \mu)}}$$
$$\boldsymbol{I = 0}$$
$$\boldsymbol{(AI)_2 = 0}$$
$$\boldsymbol{B = \dot{x}_6 = k_B \frac{\alpha_0}{\alpha_1 (d_B + \mu)}}$$
$$\boldsymbol{C = \dot{x}_8 = k_C \frac{\gamma_0}{(d_C + \mu)}}$$
$$\boldsymbol{Ie = Condición\ de\ inoculación\ inicial\ experimental}$$
$$\boldsymbol{N = N_{cells}}$$
$$\boldsymbol{M = 0}$$

Como se puede observar, también desaparecen varios parámetros del modelo, ya que hacen referencia a interacciones entre las proteínas que no se darán acabo que se produzca la inoculación.

3.4 Experimento

El objetivo es medir la evolución de la concentración de proteína C, causante de la fluorescencia de la célula, en la población cuando se introduce el inductor. Dada la imposibilidad de hacer un conteo exacto de proteínas, se elegirá una proteína fluorescente para que se pueda medir el nivel de luz que emite la población de células.

A la hora de medir la fluorescencia se introduce en el medidor una placa que contiene las diferentes poblaciones. Cada población recibe una cantidad diferente de inductor. Por lo tanto, en cada colección de datos podremos observar la evolución de la proteína C para diferentes cantidades de inductor I que corresponderán a los diferentes objetivos de optimización.

Cada experimento se ha realizado con diferentes distribuciones de concentraciones de inoculación en la población de células (separadas en pocillos) para obtener la mayor cantidad de información posible sin necesidad de repetir experiencias. A nivel de tratamiento de datos esto requerirá mejorar la estabilidad del *software* para poder leer las lecturas de diferentes experimentos con la mínima cantidad de cambios en el código.

Las células se mantienen refrigeradas la noche anterior al experimento para prepararlas y se mantienen a temperatura ambiente antes de pasar a la inoculación. Este tiempo de espera debe ser constante en las medidas utilizadas para la identificación ya que, como se indicará en el apartado correspondiente, afecta claramente a la respuesta del circuito.

Dado que los valores de las lecturas son bastante bajos, es necesario aplicar un factor de ganancia que multiplique las lecturas. Igual que en el caso anterior, es imprescindible que se mantenga la misma ganancia para los diferentes experimentos. En caso contrario, estaríamos trabajando con datos experimentales en diferentes escalas.

4 SELECCIÓN DE DATOS EXPERIMENTALES

Durante los experimentos hemos introducido el inductor I en el sistema incoherente de tipo I de forma que actúe como una entrada de escalón, es decir, tras la primera variación en su valor al introducirlo al medio se mantiene el nivel constante para que las células puedan alcanzar un nuevo estado permanente. Esta evolución es la que debería verse reflejada en los datos obtenidos durante el experimento.

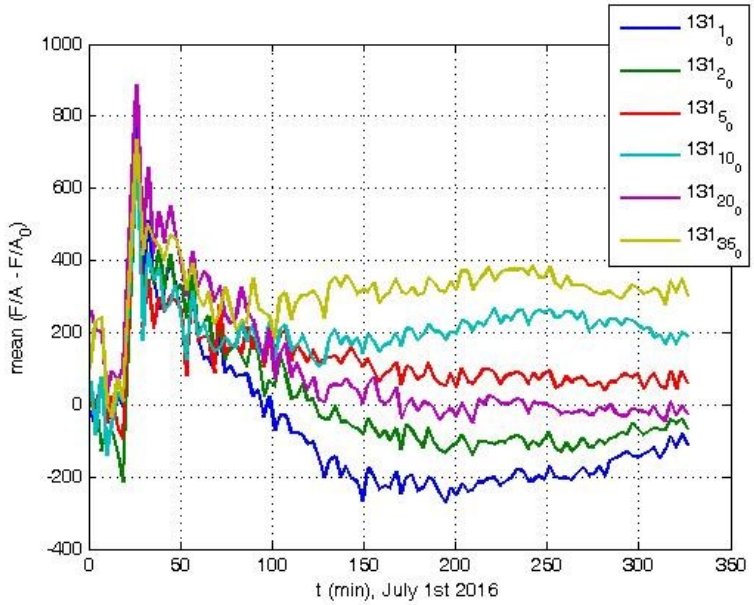
En el presente trabajo utilizaremos datos correspondientes a experimentos realizados en varios días, para poder contrastar los valores de los parámetros identificados entre ellos. Todos los experimentos utilizados para la caracterización deben haber sido realizados con las mismas condiciones, es decir, mismo tiempo de espera inicial y mismo factor de ganancia. Debido a la gran dependencia de las circunstancias de contorno en experimentos con elementos biológicos, las lecturas recopiladas varían ampliamente entre ellas; aunque siempre mantengan un comportamiento similar al esperado. Por ello, es necesario mantener, en la máxima medida posible, las condiciones constantes para todas las experiencias.

Durante la realización de los experimentos se ha ido variando la relación de las concentraciones de los genes B y C buscando la forma de acercar el comportamiento del sistema con el comportamiento teórico ideal. Si bien es cierto que muchas de las relaciones utilizadas daban un comportamiento similar al deseado, algunas son más sensibles a perturbaciones o aparición de ruido en las lecturas.

La primera selección, la selección de la relación B/C, se hace mediante un criterio cualitativo utilizando la representación gráfica de las medidas experimentales. Mediante la evolución del cociente de la fluorescencia sobre la absorbancia menos el cociente de las células sin inocular (población de control) $((F/A)-(F/A)_0)$ en el tiempo, se puede diferenciar rápidamente aquellos casos que más se alejen del comportamiento esperado. El caso ideal correspondería a un pico en el cociente que duraría hasta que las células alcancen el nuevo estado de equilibrio. En esta gráfica podremos, además, identificar los objetivos de la optimización en el comportamiento de las poblaciones con diferente concentración de molécula I inoculada

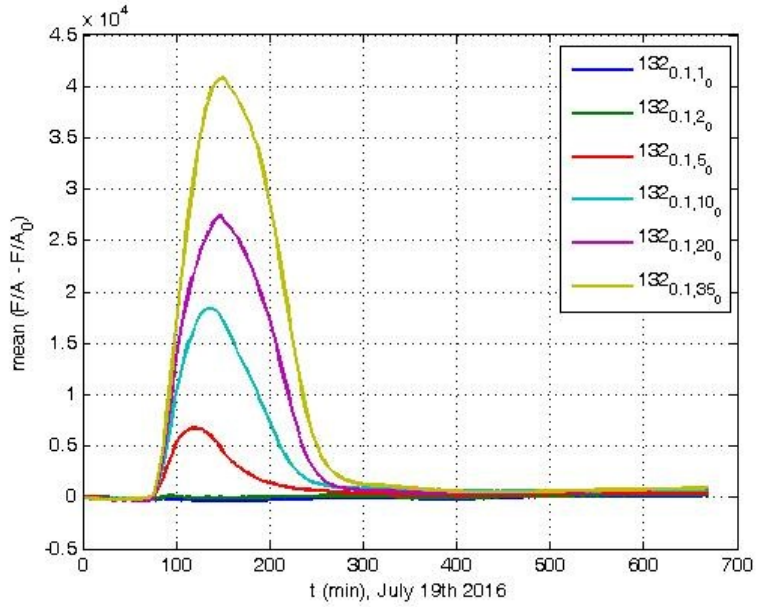
En la figura 2 se observa una de estas combinaciones que proporcionaba lecturas inutilizables para el proceso de identificación dada la cantidad de ruido y error en el estado permanente final que contienen. Esta combinación, denominada 131, correspondía a una cantidad media de represor B y una cantidad media de gen C.

Figura 2: Gráfica F/A 131-010716



En algunos experimentos, en cambio, se puede apreciar un comportamiento mucho más parecido al descrito teóricamente, sobre todo para las concentraciones más altas de inoculación. Esto se da normalmente, para una combinación de grandes cantidades tanto de represor B como de gen C productor de la proteína fluorescente. Dicha combinación, denominada 132, es la que, por tanto, se ha elegido para realizar la identificación. En la figura 3 se puede ver como para concentraciones de inoculación superiores a 5nM el pico de la fluorescencia sube al inocular la población y luego desciende hasta alcanzar el estado inicial con un error prácticamente nulo.

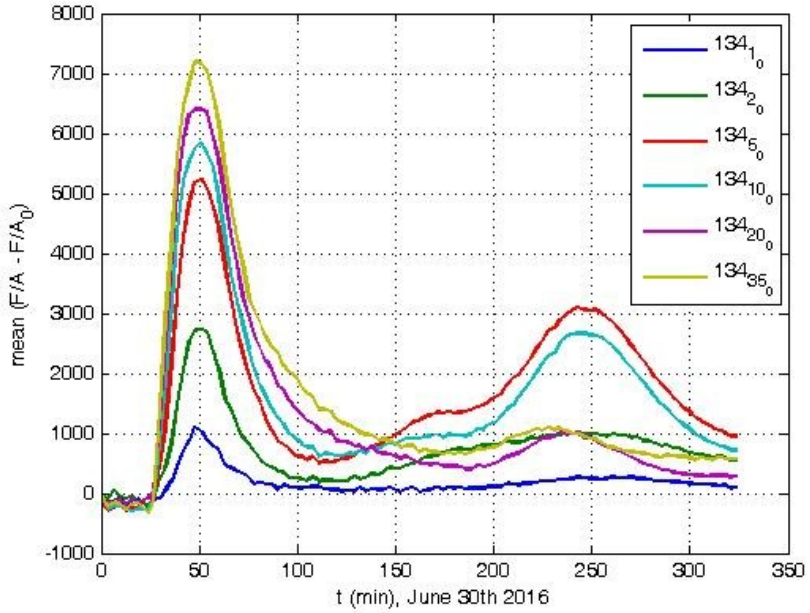
Figura 3: Gráfica F/A 132-190716(01)



La combinación 132 también tiene la ventaja de no presentar tantos comportamientos anómalos no explicados por el sistema. Muchas otras combinaciones han presentado la aparición de un segundo crecimiento del cociente tras alcanzar el estado permanente final. Esto dificulta el trabajo de optimización del modelo y falsifica los resultados de la caracterización. Para trabajar con

estos datos habría que detectar la causa exacta de la aparición de este “segundo pico” e introducirla en la modelización, lo que se sale del alcance del presente trabajo.

Figura 4: Gráfica F/A 134-300616



En la figura 4 se puede apreciar perfectamente esta segunda crecida de la fluorescencia en las células. Además de perturbar el nuevo equilibrio causa un mayor error final entre los estados iniciales y finales de equilibrio. También requeriría realizar experimentos con más tiempo de toma de datos, ya que el “segundo pico” ralentiza bastante la llegada del equilibrio final.

Una vez elegida la combinación 132, hay que seleccionar cuales de los experimentos realizados sirve para la identificación. Para ello tenemos que seleccionar los que más información proporcionen para la identificación, es decir, aquellos que se hayan realizado con más número de concentraciones. También tenemos que mantener los criterios descritos en el apartado anterior de mantener tiempo de espera y coeficiente de ganancia constantes durante los experimentos.

Como se puede observar en la figura 3, el tiempo de espera son 60 minutos. Pero revisando los experimentos realizados, se ha concluido que la mayoría de ellos se han hecho con un tiempo de espera de 20 minutos, de forma que vamos a descartar las gráficas que hayan sido realizadas con un tiempo diferente a este.

También se ha observado que la mayoría de experimentos que cumplen los criterios anteriores tienen un factor de ganancia de 80. Por lo que experiencias con otros valores también debe ser descartadas. La diferencia de magnitud en el cociente F/A es obvia observando las gráficas. La figura 5 corresponde a un experimento realizado con un factor de ganancia 70 mientras que la figura 3 tiene el factor de ganancia 80 mencionado anteriormente.

Tras aplicar todos los criterios mencionados el proceso de criba nos proporciona la selección de 4 experimentos realizados en junio de 2016, con la combinación B/C 132, 20 minutos de espera antes de la inoculación y un factor de ganancia de 80. Todos los experimentos seleccionados

cumplen, además, con un comportamiento muy similar al descrito teóricamente y muy similar entre ellos, por lo que facilitarán mucho el trabajo de identificación (Figuras 5, 6, 7 y 8).

Figura 5: Gráfica F/A 132-210616

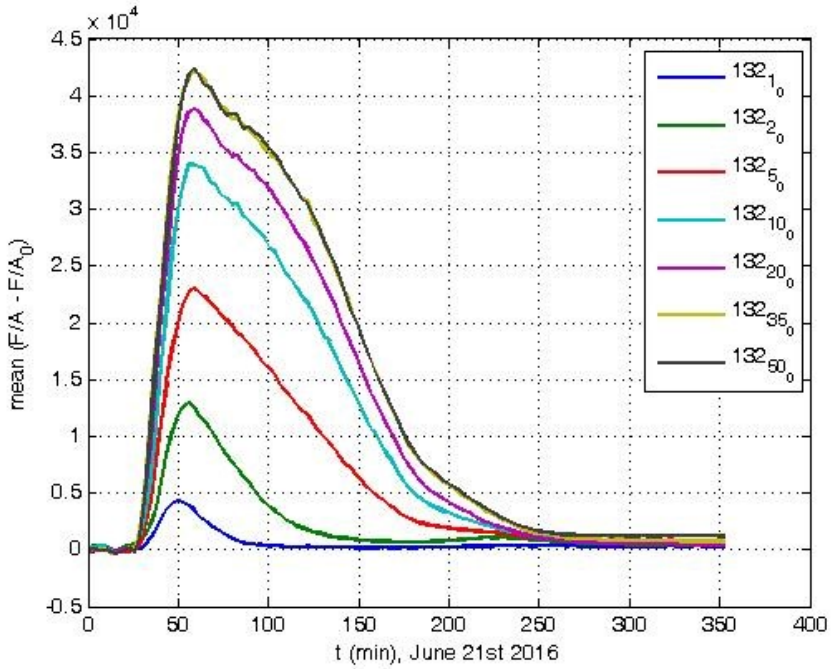


Figura 6: Gráfica F/A 132-230616

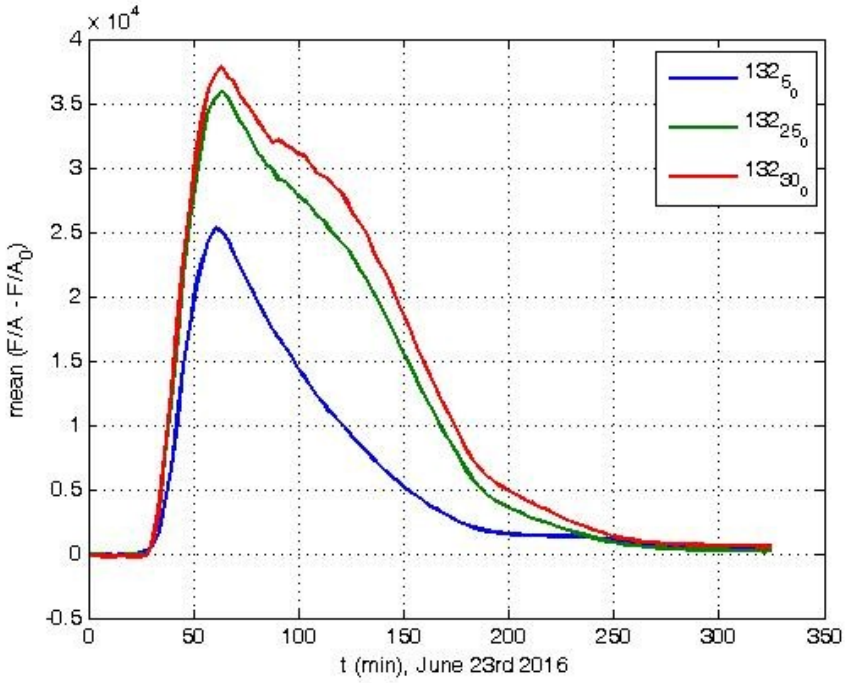


Figura 7: Gráfica F/A 132-280616

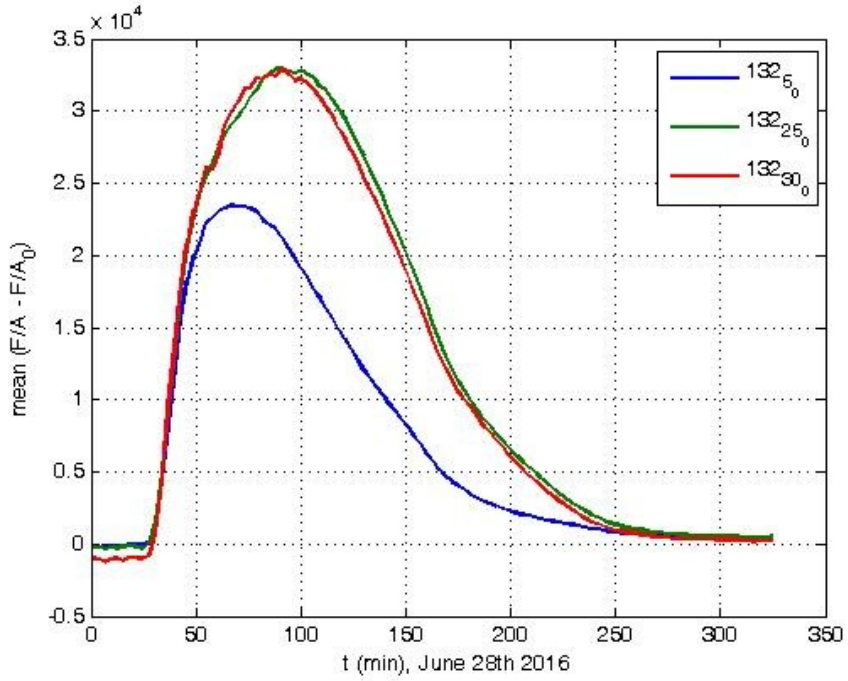
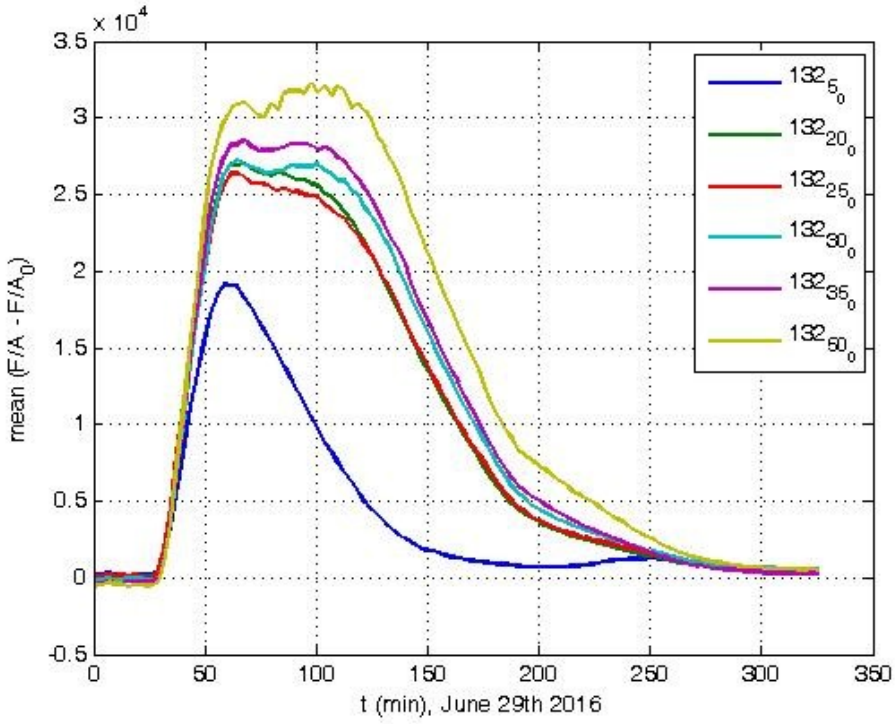


Figura 8: Gráfica F/A 132-290616



Como se indicaba en el apartado referido a la obtención experimental de datos cada uno de los experimentos ha sido realizado con una distribución diferente de concentraciones de inoculación. Las distribuciones para los experimentos que se han seleccionado tras la criba son las siguientes:

Figura 9: Distribución de la placa para el experimento del 21-06-16

% 1 2	3	4	5	6	7	8	9	10	11	12	
% M9	M9	M9	M9	M9	M9	M9	M9	M9	M9	M9	M9
% 0	0	0	0	0	0	0	0	0	0	0	0
% 1	1	1	1	1	1	1	1	1	1	1	1
% 2	2	2	2	2	2	2	2	2	2	2	2
% 5	5	5	5	5	5	5	5	5	5	5	5
% 10	10	10	10	10	10	10	10	10	10	10	10
% 20	20	20	20	20	20	20	20	20	20	20	20
% 35	35	35	35	35	35	35	35	35	35	35	35
% ID132-0,2					ID132-0,1					ID134 0,2	
%											

Figura 10: Distribución de la placa del experimento del 23-06-16

% 1	2	3	4	5	6	7	8	9	10	11	12
% A								M9	M9	M9	M9
% B								0	0	0	0
% C								5	5	5	5
% D								25	25	25	25
% E								30	30	30	30
% F											
% G											
% H											
%									ID132		

Figura 11:Distribución de la placa del experimento del 28-06-16

% 1	2	3	4	5	6	7	8	9	10	11	12
% M9	M9	M9	M9	M9	M9	M9	M9				
% 0	0	0	0	0	0	0	0				
% 1	1	1	1	5	5	5	5				
% 2	2	2	2	20	20	20	20				
% 5	5	5	5	25	25	25	25				
% 10	10	10	10	30	30	30	30				
% 20	20	20	20								
% 35	35	35	35								
%	ID133					ID132					

Figura 12:Distribución de la placa del experimento del 29-06-16

% 1 2	3	4	5	6	7	8	9	10	11	12	
% M9	M9	M9	M9	M9	M9	M9	M9	M9	M9	M9	M9
% 0	0	0	0	0	0	0	0	0	0	0	0
% 1	1	1	1	1	1	1	1	1	1	1	1
% 2	2	2	2	2	2	2	2	2	2	2	2
% 5	5	5	5	5	5	5	5	5	5	5	5
% 10	10	10	10	10	10	10	10	10	10	10	10
% 20	20	20	20	20	20	20	20	20	20	20	20
% 35	35	35	35	35	35	35	35	35	35	35	35
% ID132-0,2					ID132-0,1					ID134 0,2	

Como se puede observar las distribuciones varían en muchos aspectos. Tres de los experimentos (todos excepto el realizado el fechado a día 23) incluían poblaciones de células de otra especie (indicadas por su diferente ID) que deben obviarse durante la adquisición de datos del

software para evitar falsos resultados en la optimización ya que las diferentes especies requerirían diferentes parámetros para modelar su comportamiento.

También se observa que todos los experimentos comparten el mismo medio habitado por las células (M9) que facilita la comparación de los datos obtenidos al eliminar factores a tener en cuenta.

Otra ventaja de elegir estos datos es que la mayoría de concentraciones de inducciones se repite en diferentes experimentos. Esto permite comparar directamente la eficacia de los intervalos elegidos para la optimización en cuanto a reducir el error de optimización. Dado que la cantidad de proteína variará en gran medida el comportamiento del circuito, es de esperar que sea necesario la obtención de diferentes intervalos de parámetros para concentraciones extremas.

Fijándonos en las representaciones gráficas se puede apreciar una variación importante de comportamiento entre las concentraciones bajas y las altas de inoculación (Figura 5). Por ello se deduce la necesidad de encontrar diferentes conjuntos de valores de parámetros. La imposibilidad de un modelo que describa el comportamiento de todas las concentraciones, teniendo en cuenta la gran incertidumbre y complejidad del circuito, se evita con el uso de parámetros diferentes según la concentración de inoculación utilizada.

Por último, la inclusión de los datos obtenidos el día 23 y 28 acelerará la obtención de parámetros identificados. Esto se debe a la menor cantidad de concentraciones presentes en el experimento. Facilitando el trabajo durante la optimización ya que reduce los puntos que definen la frontera de Pareto. Con estos datos podremos obtener una primera indicación de la validez de los intervalos seleccionados, menos precisa, pero con menor trabajo computacional.

5 DESARROLLO DE CÓDIGO PARA IDENTIFICACIÓN Y VALIDACIÓN

El *software* utilizado para la identificación y validación del modelo con los datos experimentales se basará en el proporcionado por el equipo de investigación [7] del Instituto Universitario ai2. Este código deberá adaptarse para facilitar su uso y acelerar su funcionamiento, así como automatizarlo lo máximo posible para el uso en experimentos con diferentes condiciones iniciales. Analizando cada una de las funciones que debe realizar el *software* en su conjunto, será más sencillo encontrar y aplicar las posibles mejoras.

5.1 Lectura y representación de los datos experimentales (Anexo I)

Tras el experimento los datos han sido almacenados en una hoja de cálculo junto a una descripción de las condiciones del experimento (tiempo previo a la inoculación y posterior, medios y cepas utilizadas, tipo de placa… De esta hoja de cálculo lo que utilizará el *software* son las lecturas de los sensores a lo largo de la duración del experimento.

Read	Absorbance Endpoint
	A1..H4
	Wavelengths: 600
	Read Speed: Normal, Delay: 100 msec, Measurements/Data Point: 8
Read	Fluorescence Endpoint
	A1..H4
	Filter Set 1
	Excitation: 485, Emission: 528
	Optics: Top, Gain: 70
	Light Source: Xenon Flash, Lamp Energy: High
	Read Speed: Normal, Delay: 100 msec, Measurements/Data Point: 10
	Read Height: 7 mm
End Kinetic	

Results

Actual Temperature: 32,5

	1	2	3	4	5	6	7	8	9	10	11	12	
A	0,045	0,045	0,046	0,047	0,048	0,048	0,048	0,048	0,049	0,048	0,049	0,052	Read 1450
B	0,11	0,108	0,111	0,111	0,049	0,048	0,049	0,049	0,048	0,048	0,048	0,048	Read 1450
C	0,109	0,11	0,11	0,11	0,053	0,048	0,048	0,048	0,048	0,049	0,048	0,047	Read 1450
D	0,109	0,109	0,11	0,11	0,049	0,049	0,049	0,049	0,048	0,05	0,048	0,048	Read 1450
E	0,11	0,112	0,11	0,111	0,049	0,049	0,049	0,049	0,049	0,049	0,048	0,048	Read 1450
F	0,112	0,111	0,112	0,111	0,049	0,049	0,049	0,05	0,052	0,049	0,047	0,049	Read 1450
G	0,109	0,11	0,113	0,112	0,049	0,049	0,049	0,049	0,049	0,049	0,047	0,048	Read 1450
H	0,109	0,111	0,11	0,11	0,049	0,048	0,048	0,049	0,049	0,048	0,049	0,049	Read 1450

Read 2:600

Time	T° Read 2:600	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	B1	B2	B3	B4	B5	B6
0:00:00	36,6	0,037	0,037	0,037	0,037									0,083	0,084	0,085	0,084		
0:03:00	37	0,037	0,037	0,037	0,037									0,084	0,084	0,085	0,084		
0:06:00	37	0,037	0,037	0,037	0,037									0,085	0,085	0,086	0,085		
0:09:00	37	0,037	0,037	0,037	0,037									0,085	0,086	0,086	0,086		
0:12:00	37	0,037	0,037	0,037	0,037									0,086	0,087	0,087	0,087		
0:15:00	37	0,037	0,037	0,037	0,037									0,088	0,088	0,089	0,088		
0:18:00	37	0,037	0,037	0,037	0,037									0,09	0,09	0,091	0,09		

Figura 13: Extracto de hoja de cálculos donde se recogen los datos experimentales

En un principio se crearon scripts únicos para cada experimento a partir de una plantilla. En estos scripts se indicaba la ubicación de los datos en el Excel cambiando directamente las celdas que leía el código. Esto concluía en una colección de scripts muy específicos que leían, representaba

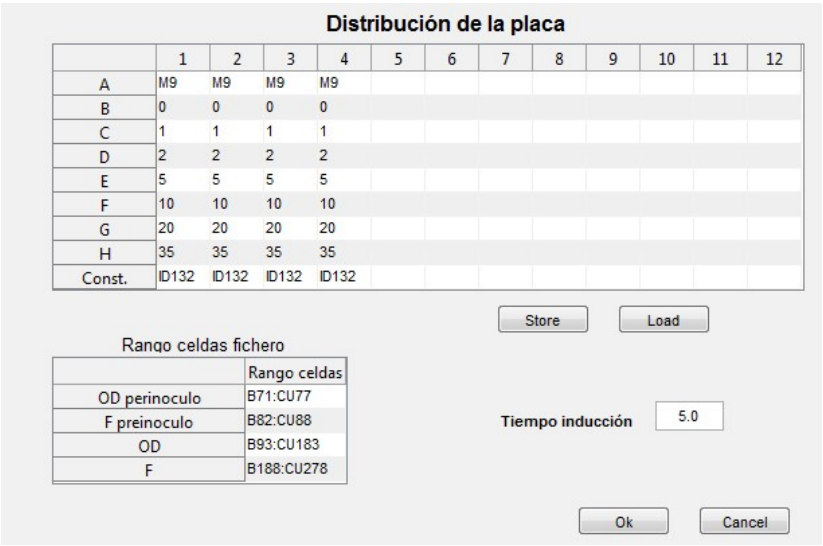
y guardaban la información del experimento para el uso posterior. Si bien este proceso funciona perfectamente, resulta laborioso para usuarios no familiarizados con el código y una solución más general podría evitar la necesidad de código muy específico.

Figura 14: Ejemplo de función específica de lectura de datos de un experimento

```
[dades_2906_OD_pre] = xlsread('./290616-ID128-ID132-AHL-M9-PST.xlsx','D71:CU77');
time_2906_OD_pre = [0:3:18];  % Time in minutes

[dades_2906_F_pre] = xlsread('./290616-ID128-ID132-AHL-M9-PST.xlsx','D82:CU88');
time_2906_F_pre = [34/60:3:18+34/60];  % Time in minutes. Fluorescence started to be measured 51 seconds after OD.

[dades_2906_OD_post] = xlsread('./290616-ID128-ID132-AHL-M9-PST.xlsx','D93:CU193');
time_2906_OD_post = time_2906_F_pre(end)+5 + 10/60 + [3/60:3:5*60+3/60];  % Started 3 seconds after the order was done. Lasted 10.0 hours.
                    %Induction took 7 min 10 sec, but notice for 132-0,2 it
                    %was 3 min 10 secs, and 5 min 10 sec for 132-0,1
[dades_2906_F_post] = xlsread('./290616-ID128-ID132-AHL-M9-PST.xlsx','D198:CU298'); % Started 54 seconds after the order was done. Lasted 10.0 hours.
time_2906_F_post = time_2906_F_pre(end)+5 + 10/60 + [37/60:3:5*60+37/60];

time_2906_OD = [time_2906_OD_pre, time_2906_OD_post];
time_2906_F = [time_2906_F_pre, time_2906_F_post];

Time_lag_2906 = 60;

%Absorbance  Pre %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dades_2906_OD_M9_pre = dades_2906_OD_pre(:,5:8); % Used as background absorbance
mean_OD_M9_pre = mean(dades_2906_OD_M9_pre,2)*ones(1,4);
dades_2906_OD_132_01_0_pre = dades_2906_OD_pre(:,13:20);
dades_2906_OD_132_01_1_pre = dades_2906_OD_pre(:,25:32);
dades_2906_OD_132_01_2_pre = dades_2906_OD_pre(:,37:44);
dades_2906_OD_132_01_5_pre = dades_2906_OD_pre(:,49:56);
dades_2906_OD_132_01_10_pre = dades_2906_OD_pre(:,61:68);
```

Para una mayor automatización del proceso se aprovecha las similitudes entre los experimentos. El tamaño de la placa es constante y de él depende la distribución de los datos en el Excel. Por lo que se ha elegido una forma sencilla para introducir la distribución de la placa.

Figura 15: Interfaz gráfica de la función readExperiment



Esto se hace mediante el script **readExperiment,** con una interfaz gráfica en la cual se debe introducir la distribución de cepas y concentraciones de inductor y medio en la placa del experimento, así como las filas que ocupan los datos y el tiempo de inducción. De esta forma no es necesario cambiar directamente el código y se evitan posibles fallos. Además, para facilitar futuros usos, se incluye la posibilidad de guardar distribuciones de placas concretas para usarlas posteriormente sin ser necesario volver a introducirlas a mano.

Se considera que la representación gráfica del experimento no será siempre una prioridad, por lo que la función de representación se ha relegado a un script diferente **plotExperiment**, que se llama de forma independiente. Este script utiliza los datos leídos por el anterior para crear tres gráficas: evolución de la absorbancia, la fluorescencia y el cociente de ambas en el tiempo. Una comprobación rápida de la corrección de la lectura de datos es comparar estas gráficas con las que acompañan a los datos experimentales. Como se observa en las figuras 17 y 18, las lecturas de los datos se hacen correctamente.

Figura 16: Lectura de datos para el experimento 29-06

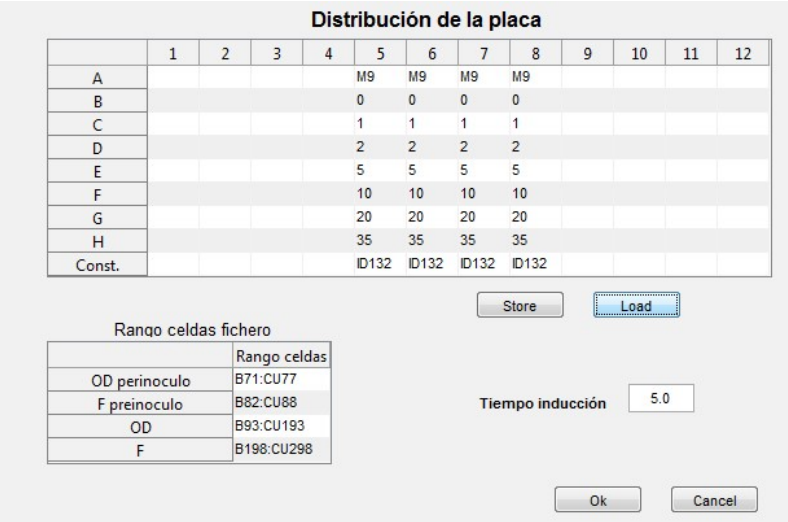


Figura 17: Fluorescencia del experimento 29-06 usando plotExperiment

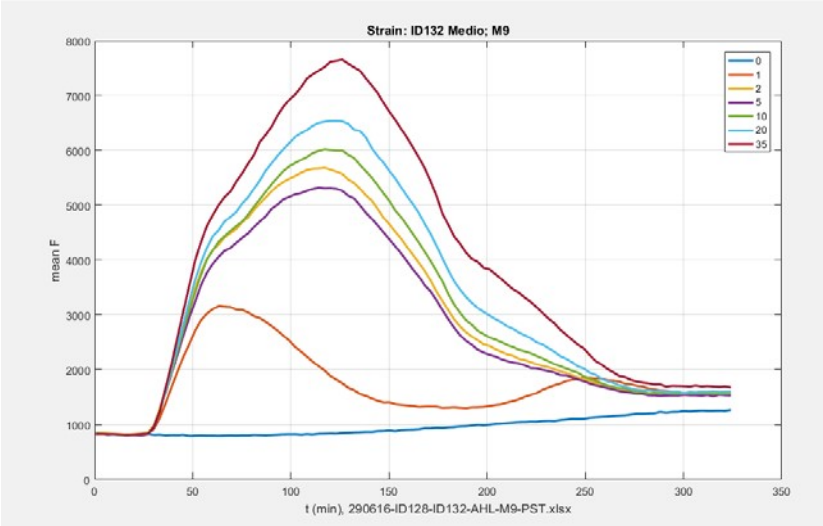
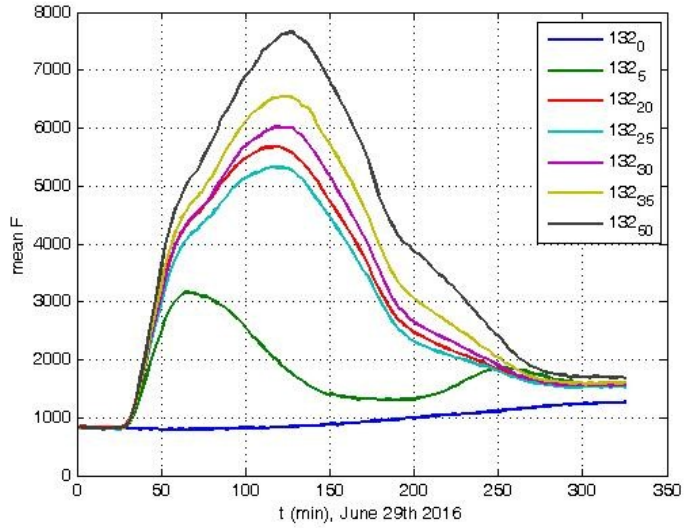


Figura 18: Fluorescencia del experimento 29-06 obtenida en el experimento



También se han alterado el formato de salida de los datos experimentales tras ser leídos por el programa. Tras varias iteraciones diferentes, se ha decidido que el mejor formato de salida es una estructura de datos (variable **evol**) que almacenará en su interior todos los datos experimentales. Dado que cada inducción presente en el experimento ocupa varias poblaciones de células, la cantidad de datos se multiplica. Tener toda la información en una sola estructura de datos mejora la estabilidad del *software* frente a la opción de crear matrices diferentes, que producían por los scripts específicos (figura 20). También facilita su acceso durante el resto del trabajo.

Trabajar con matrices más pequeñas reduce el tiempo de cálculo y la estructura de datos más ordenada permite trabajar seleccionar los datos de inducciones específicas con más sencillez. El formato de estructura de datos permite almacenar, también, información del experimento (fecha, distribución de la placa, inducciones presentes…). Se ha añadido la función de leer experimentos con diferentes especies y trabajar con ellas por separado, sin necesidad de cambiar el código interno.

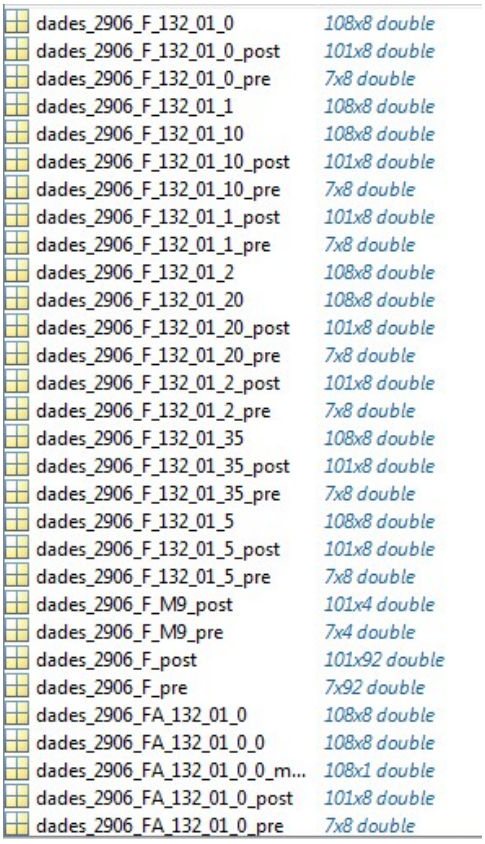
Figura 19: Estructura de datos evol para el experimento del 29-06

evol | evol.medios | evol.medios.data

evol.medios.data

ds	Inducc	OD	F	OD_medio	F_medio	FOD	FA
	'0'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'1'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'2'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'5'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'10'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'20'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double
	'35'	108x8 double	108x8 double	108x1 double	108x1 double	108x1 double	108x8 double

Figura 20: Datos del experimento 29-06 producido por su script específico



Como se ve en la figura 19, además de la absorbancia (OD) y la fluorescencia (F) se almacenan sus valores medios y su cociente (FA), así como el cociente de sus valores medios (FOD). Aunque para la optimización se utilizan los datos de absorbancia y el cociente.

5.2 Preparación de los datos para la optimización (Anexo II)

Antes de poder aplicar el modelo matemático es necesario añadir cierta información a la estructura de datos y elegir que queremos identificar y con qué información vamos a validar las soluciones obtenidas posteriormente. Esta función se realiza mediante la función **spMODEparam2.** El primer trabajo de esta función no ha cambiado mucho respecto a la original, se genera una nueva estructura de datos (variable **Dat**) en la que se incluyen el número de concentraciones usadas en el experimento, número de objetivos, número de parámetros, rangos de estos durante el proceso de optimización, el modelo y función de coste que deberá utilizar y parámetros relevantes para el proceso de implementación; como el número máximo de evaluaciones. También se le introduce la estructura entera evol creada anteriormente para que los datos se puedan acceder directamente desde Dat.

Una ventaja de esta segmentación de scripts y estructuras de datos es que permite una automatización más estable. De la estructura evol se puede obtener fácilmente las concentraciones presentes en el experimento y cuantas poblaciones de células tiene cada una. Esto evita tener que cambiarlas a mano dentro del código como se hacía en la versión original del código.

Para la realización de la optimización se ha eliminado la población de control (con inducción 0) ya que las lecturas experimentales de estas células corresponden con un estado permanente de fluorescencia nula. La optimización con esta población incluida falsificaría la frontera de Pareto obtenida.

Tabla 10: Rango de las variables durante la optimización

Parámetros	Rango
γ_1	$[0.001\ 0.1]$
γ_3	$[0.1\ 1]$
γ_4	$[0.0005\ 5]$
γ_5	$[0.01\ 10]$
k_A	$[6\ 4500]$
k_B	$[6\ 4500]$
k_C	$[6\ 4500]$
α_1	$[10\ 100]$
k_2	$[0.5\ 3]$
k_3	$[0.1\ 1]$
k_2	$[6e^{-4}\ 6e^{-2}]$
k_3	$[6e^{-4}\ 6e^{-2}]$
d_A	$[0.01\ 0.3]$
d_B	$[0.01\ 0.3]$
d_C	$[0.01\ 0.3]$
k_{gain}	$[0.1\ 100]$

Los rangos de los parámetros mostrados en la tabla 10 se han obtenido a partir de los recogidos en [7] para el modelo diferencial original, siendo muchos de los parámetros equivalentes a los usados en dicho modelo. La mayor diferencia se encuentra en los nuevos parámetros K_A, K_B y K_C cuya obtención se ha indicado anteriormente. Sus rangos se han obtenido matemáticamente a partir de los valores mínimos y máximos que pueden tomar sus definiciones mencionadas anteriormente.

En esta estructura se seleccionan los datos con los que se trabaja. Para aumentar la información obtenida durante la optimización, se utilizan todos los datos de las diferentes poblaciones de las inducciones presentes tanto para la identificación como para la valoración. En la versión original del código se utilizaban solo una parte de los datos para la identificación usando los restantes para la validación. Esta opción se descarta debido a las ventajas de la opción anterior ya mencionadas y la falta de estabilidad que causaba la selección de datos para experimentos con diferente número de poblaciones por concentración de inducción.

5.3 Implementación del modelo y optimización (ANEXO III)

Dada la naturaleza multiobjetivo del problema [4], nuestra solución óptima se obtendrá mediante la compensación del error individual de cada objetivo. En nuestro caso, se utiliza un algoritmo diferencial evolutivo con poda esférica. Esto significa que a todos nuestros objetivos se le aplica la misma prioridad, en vez de aplicarles diferentes pesos para guiar la optimización. Durante el presente trabajo se considera como objetivo, la fluorescencia de cada concentración presente en el experimento. Este número varía entre 3 y 6 para los problemas optimizados durante el presente trabajo.

La función **spMODE2** inicia la población de células simuladas mediante una distribución uniforme y llama a la función de coste (**CostFunction_AdaptationGFP2**) que obtendrá el error del modelo (almacenado en **model_adaptation_GFP3**) utilizando el algoritmo ode15s [11]. Tras obtener la solución, se evalúa su posición en el frente de Pareto dinámico y se descarta o se conserva como un nuevo punto. Cabe destacar que este software proporciona tantas soluciones como puntos encuentre en la frontera de Pareto que una los objetivos presentes en el problema. De ahí la necesidad de un *software* de validación entre las diferentes soluciones.

En cada iteración el *software* optimizará el modelo para todos los datos experimentales introducidos. Por lo que, para simplificar la información transmitida al usuario, se presenta el error medio para cada concentración de inductor. Cada punto en la frontera de Pareto se caracteriza por su error medio con todas las concentraciones presentes y en cada iteración se informa al usuario del error mínimo presente para cada una para presentar el avance de la optimización.

Este paso es el que más más tiempo requerirá, dada la cantidad de parámetros, ecuaciones y variables de decisión que tiene el modelo. Por ello, es una prioridad acelerar lo máximo posible el trabajo de las funciones que realizan esta función, sin reducir su eficacia con respecto a las originales.

Figura 21: Implementación del modelo reducido

```
dxdt=zeros(size(x));

%M
c1 = param.dAI + param.k_2; % + param.mu;
c2 = c1^2 + 8*param.k3*(param.k2*x(1)*x(2) + 2*param.k_3*x(3));
c3 = 4*param.k3;
M= (1/c3)*(-c1 + sqrt(c2));

%x(1):A
dxdt(1)=param.kA(k)-param.k2*x(1)*x(2)-param.k_2*M-(param.dA(k)+param.mu)*x(1);

%x(2):I
dxdt(2)= - param.k2*x(1)*x(2) + param.k_2*M + param.kd*(x(6)-x(2)) - (param.dI + param.mu)*x(2);

%x(3):(AI)2
dxdt(3)= param.k3*M^2 - param.k_3*x(3) - (param.mu + param.dAI2)*x(3);

%x(4):B
dxdt(4)= param.kB(k)*(param.alpha0 + x(3))/(param.alpha1(k) + x(3)) -  (param.mu + param.dB(k))*x(4);

%x(5):C
a=param.gamma0 + param.gamma1(k)*x(3) + param.gamma2*x(3)*(x(4)^2);
b=1 + (param.gamma3(k))*x(3)+ param.gamma4(k)*(x(4)^2) + (param.gamma5(k))*x(3)*(x(4)^2);
c=param.kC(k)*a/b;
dxdt(5)= c -  (param.mu + param.dC(k))*x(5);

%x(6):Ie
dxdt(6)= - (param.Vcell*x(7)/param.Vculture)*param.kd*(x(6) - x(2)) - param.dIe*x(6);

%x(7):N
dxdt(7)= param.mu*x(7)*(1 - x(7)/param.Kmax);
```

La dificultad de esta optimización ha requerido la elaboración de este algoritmo específico en vez de utilizar métodos más tradicionales, como el muestreo Monte Carlo, que en comparación requiere un peso computacional que resulta inviable si se suma al peso considerable inherente en un problema tan complejo.

Se ha eliminado de cada evaluación todos los procesos irrelevantes, aquellos que no es necesario repetir y que bastará con hacer al inicio de la implementación. Concretamente, la inicialización de parámetros conocidos y constantes se ha asignado a una nueva función **set_parameters3** (Figura 21) que se llama antes de aplicar el modelo a cada concentración y no individualmente en cada uno de estos casos creando el struct **param** que se utilizará en la optimización para evitar volver a calcular los parámetros con valor fijo. En esta función se diferencia entre aquellos parámetros fijos, que mantienen un valor constante y aquellos que nos interesa estimar debido a que se desconoce su valor exacto y describen con más detalle el comportamiento del circuito. Todos los parámetros a estimar se acumulan en un vector dentro de la estructura de parámetros que genera este script.

Figura 22: Vector de parámetros a estimar

```
param.gamma1 = X(:,1);
param.gamma3 = X(:,2);
param.gamma4 = X(:,3);
param.gamma5 = X(:,4);
param.kA = X(:,7);
param.kB = X(:,8);
param.kC = X(:,9);
param.alpha1=X(:,10);
param.k_2 = X(:,11);
param.k2 = X(:,12);
param.k_3 = X(:,13);
param.k3 = X(:,14);
param.dA = X(:,15);
param.dB = X(:,16);
param.dC = X(:,17);
```

Las desigualdades necesarias para que el modelo reducido sea válido se han implementado en forma de suma. Por ejemplo: $\gamma\gamma_1 < \gamma\gamma_3$ implica que $\gamma\gamma_3$ aparece en el código como $(\gamma\gamma_3 + \gamma\gamma_1)$ y que el resultado proporcionado para este parámetro será el resultado de dicha suma, de forma que siempre se cumplirá la desigualdad.

También se ha mejorado la estabilidad de la función de coste para trabajar con diferentes números de objetivos sin tener que cambiar manualmente su código. Y se ha creado otra función independiente **set_initial3** (figura 23) que inicializará las variables quitando más trabajo en cada iteración.

Figura 23: Implementación del estado inicial

```
Ncells_0 = param.Vculture*OD_0*param.Ccells_OD600_1;    % initial number of cells

%Initial Conditions of variables (except x9 = input)
Nstates = 7;   % model number states
Npop=length(param.kB);
aux=ones(1,Npop);

X0 = zeros(Nstates,Npop);

%  x(1): A
%  x(2): I
%  Monomer M: algebraic equation
%  x(3): (AI)2
%  x(4): B
%  x(5): C
%  x(6): Ie
%  x(7): N

X0(1,:) = ((param.kA/(param.dA + param.mu))*aux')';
%X0(2,:) = 0;
%X0(3,:) = 0;
X0(4,:) = (((param.kB'*param.alpha0)/(param.alpha1'*(param.mu+param.dB)))*aux')';
X0(5,:) = (((param.kC*param.gamma0)/(param.dC + param.mu))*aux')';
X0(6,:) = (INPUT*aux')';
X0(7,:) = (Ncells_0*aux')';
```

Para ahorrar trabajo computacional se crea una matriz de dimensiones iguales a la población celular en el experimento donde se introducen los valores iniciales por columnas. De esta

forma se podrán hacer los cálculos mediante operaciones matriciales en vez de usando ciclos de repeticiones.

Por último, se ha modificado el código para que utilice la función **parallel pool** [12] de Matlab. De esta forma se utilizarán paralelamente todos los núcleos del procesador en el que se realice la implementación, lo que tiene un gran potencial a la hora de acelerar el proceso.

Figura 24: Campos de datos que contiene la estructura de datos OUT tras la optimización.

Field	Value
Ini	'10-May-2017 11:47:20'
PSet	*34x17 double*
PFront	*34x6 double*
SubParent	*50x17 double*
Parent	*90x17 double*
JxParent	*90x6 double*
JxSubParent	*50x6 double*
Param	*1x1 struct*
Fin	'10-May-2017 11:51:31'

Una vez las evaluaciones terminan se obtiene la estructura de datos final **OUT** (Figura 24). En esta se incluyen todos los puntos de la frontera de Pareto obtenidos durante la optimización (PFront) y los valores de los parámetros en dichos puntos (PSet) que se utilizan para la validación posterior. También incluye las estructuras de datos Dat y evol, la fecha de inicio y fin de la optimización, así como información de la población simulada.

La elección de separar el código en diferentes scripts también nos aporta la ventaja de un código modular. El modelo, la función de coste y los parámetros del sistema son los elementos que particularizan un experimento concreto. Al estar cada uno separado en sus respectivas funciones, se pueden cambiar independientemente para adaptar el código a otros experimentos con el mínimo trabajo necesario y con gran estabilidad, ya que el resto de código se ha adaptado para trabajar establemente con la información que le proporciona estas funciones.

5.4 Validación de las soluciones obtenidas (Anexo IV)

Una de las características de este método de optimización es que las soluciones obtenidas no serán únicas en ningún caso, dependiendo del grado de equilibrio objetivo que se utilice para su generación. De esta forma se considera una prioridad que el *software* tenga capacidad para comparar las diferentes soluciones. La comparación se hará de manera cualitativa mediante representación gráfica, y cuantitativa mediante los vectores de errores relativos de cada solución para los diferentes objetivos.

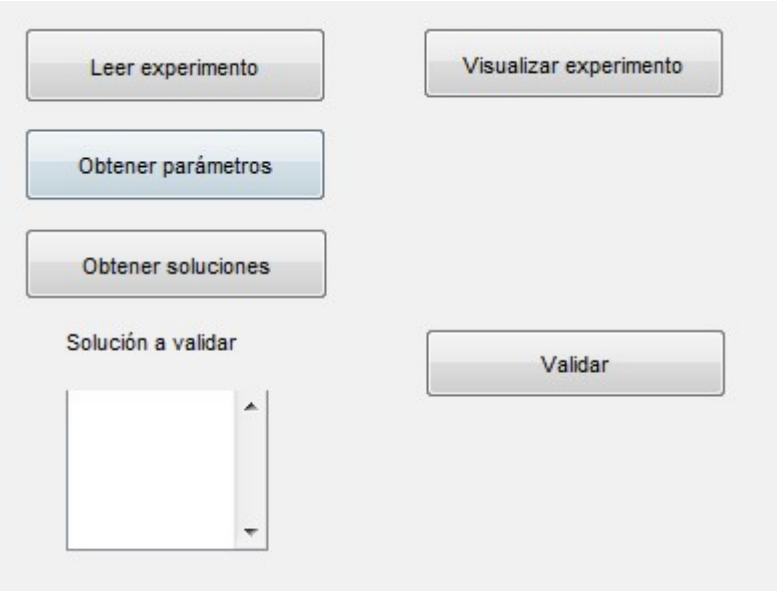
En esta última función (**Validation_Adaptation_GFP2av**) se selecciona una de las posibles soluciones del frente de Pareto obtenidos anteriormente, se aplica al modelo para simular todo el

experimento y se compara gráficamente con una representación de los datos experimentales. Este proceso se repite para todas las concentraciones. Para facilitar la visualización de los datos relevantes para el presente trabajo se ha prescindido de gráficas, en este caso irrelevantes, que proporcionaba la antigua función y se han juntado todas las restantes en una sola ventana.

5.5 Interfaz gráfica (Anexo V)

Para facilitar el uso de todas las funciones y scripts descritos anteriormente se ha elaborado una interfaz que automatiza el proceso (figura 25). Esta interfaz consiste en botones que llaman a las funciones requeridas pasándoles los datos necesarios según el paso que se esté realizando. También permite la selección de la solución a validar mediante una lista para que se puedan visualizar diferentes soluciones rápidamente y se pueda evaluar su validez.

Figura 25: Interfaz para controlar todo el proceso.



5.6 Funciones adicionales y control de errores (Anexo II)

Para la comprobación de errores en la optimización y lectura de datos se han tenido que elaborar funciones adicionales y funcionalidades extra a las descritas anteriormente creando diferentes versiones, aumentando la estabilidad y versatilidad del código.

Dado que a veces no será necesario guardar una distribución determinada de la placa de un experimento porque solo se deseará leerlo una vez o ya se ha leído mediante su función específica; la función spMODEparam2 está preparada para trabajar también con estos datos directamente. Los cambios necesarios están incluidos en la función como comentario, solo es necesario sustituir con estos comentarios el código que cumple la misma función en el funcionamiento descrito de la función. Se añade como anexo adicional el archivo spMODEparam2C donde esta sustitución se ha realizado para demostrar lo versátil que es esta función.

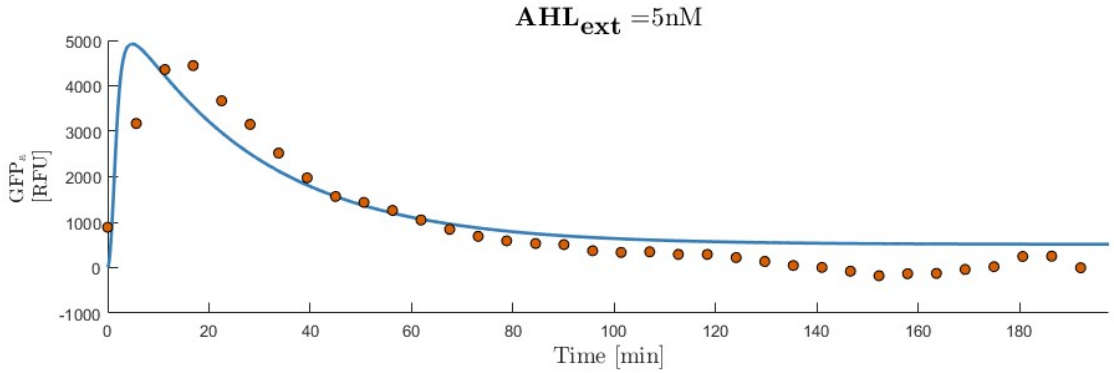
En caso de que aparezca un error durante la optimización y se requiera acceso a las variables internas de las funciones, el funcionamiento en paralelo que acelera el procedimiento impediría este acceso. Para solucionarlo se ha incluido la línea comentada que sustituye el bucle de optimización en paralelo por uno sin esta función.

Para la solución de errores en otras partes del código será necesario trabajar sin la interfaz gráfica, dado que esta esconde las variables internas del usuario. Esto se puede realizar llamando a las funciones por separado siguiendo la siguiente línea de comandos:

- Lectura de datos: evol = readExperiment;
- Representación de datos: plotExperiment(evol);
- Tratamiento de datos para optimización: spMODEDat = spMODEparam2(evol);
- Optimización: OUT=spMODE2(spMODEDat);
- Validación de la solución deseada: Validation_Adaptation_GFP2av(OUT,X) siendo X la solución a identificar.

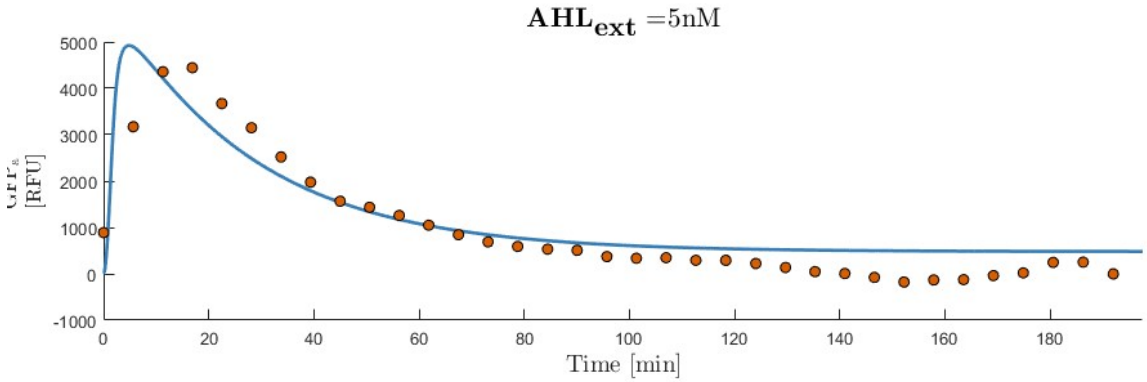
También se ha elaborado una versión paralela a las funciones descritas que utiliza el modelo matemático no reducido, usada principalmente como herramienta de comprobación de errores. Mediante estas versiones se ha podido corroborar el correcto funcionamiento de la optimización mediante parámetros conocidos para un experimento concreto. Este paso inicial también nos ofrece la oportunidad de guardar los puntos de la frontera de Pareto como población inicial para futuras optimizaciones, reduciendo significativamente el tiempo requerido para las nuevas identificaciones.

Figura 26: Solución de optimización para 5nM del experimento 1706



Como se observa en la figura 26, realizando una optimización rápida con pocas evaluaciones y usando valores conocidos podremos obtener una indicación inicial para las siguientes caracterizaciones y una comprobación del buen funcionamiento del modelo reducido, como se ve en la figura 27. Una sencilla traducción de los parámetros entre ambos modelos demuestra que ambos son equivalentes en la descripción del circuito biológico.

Figura 27: Modelo optimizado para 5nM del experimento 1706 usando el modelo reducido



6 CARACTERIZACIÓN DE PARÁMETROS

Esta parte del trabajo se centra en el tratamiento de la información de las soluciones aportadas por el software de optimización para la validación de los parámetros obtenidos. Para ello se han realizado varias optimizaciones modificando el límite de evaluaciones a alcanzar y los rangos de búsqueda de parámetros para reducir el error y el tiempo necesario para alcanzar una aproximación viable al comportamiento del circuito.

Como ya se ha indicado en el punto anterior, las identificaciones realizadas con el modelo reducido se han creado a partir de una traducción de posibles valores para parámetros del modelo anterior. Para simplificar el trabajo de optimización se ha fijado también el parámetro Υ_1 aumentando la equivalencia entre ambos modelos y la ganancia k_{gain} (ambos a 1) para poder centrar los esfuerzos de identificación en los parámetros más desconocidos y relevantes para el modelo.

También se han fijado las velocidades de disociación y asociación del monómero y el dímero dado que estos son reacciones químicas no regulables y sus valores son conocidos del modelo anterior. Esto, además de reducir la cantidad de parámetros a identificar, evita el trabajo matemático de extraer los parámetros en el nuevo modelo matemático. En la siguiente tabla se detallan los valores de los parámetros fijados durante la simplificación previa a la identificación.

Tabla 11: Parámetros fijados antes de la identificación

Parámetros	Valor
γ_1	1
k_2	0.2
k_3	0.0006
k_2	0.5
k_3	1
k_{gain}	1

6.1 Herramienta Level Diagram y hoja de cálculo (Anexo VII)

Para la identificación de parámetros se utilizará una función elaborada en Matlab previa a el presente trabajo. Esta herramienta representa gráficamente todos los puntos de la frontera de Pareto obtenidos en la optimización multiobjetivo y sus respectivos parámetros.

En la figura 28 se observa una frontera de Pareto de un experimento con 5 concentraciones. En el eje Y se indica el valor asignado a cada solución para que se puedan identificar en los siguientes gráficos y en el X el error relativo respecto a cada objetivo. Al seleccionar un punto la herramienta lo resalta en la representación asignada a cada objetivo, de forma que se puede observar rápidamente la diferencia de valores óptimos para concentraciones altas y bajas de inoculación. Reflejo de la compensación de error mínimo alcanzable entre objetivos en la frontera de Pareto.

En el ejemplo de la figura 28 se han seleccionado las soluciones que podrían tener mejores valores de parámetros para la menor de las concentraciones, pero dichos valores tendrán el mayor error posible para el resto de concentraciones. En la figura 29 se observan los valores parámetros de estas soluciones y su posición dentro del intervalo posible fijado durante esta optimización, estas gráficas comparten eje Y con las de la figura 28 pero en el eje X se encuentra el valor de cada uno de los parámetros. Las 12 gráficas corresponden a los 12 parámetros a identificar del modelo reducido.

Como se ve claramente en este ejemplo, muchas de las soluciones comparten valores de parámetros pegados a los límites (asíntotas verticales). Así se pueden detectar casos donde un aumento de los rangos de búsqueda puede ayudar a la identificación de parámetros más exactos.

También se ha elaborado una hoja de cálculo sencilla (figura 30) que calcula los valores mínimos, máximos, la cantidad de soluciones que toman estos valores y la media aritmética. Con esta hoja de cálculo se puede obtener información concreta de un grupo de solucione seleccionadas para encontrar rangos de búsqueda más ajustados a una concentración en concreto.

Figura 28: Ejemplo de frontera de Pareto de 5 objetivos con Level Diagram

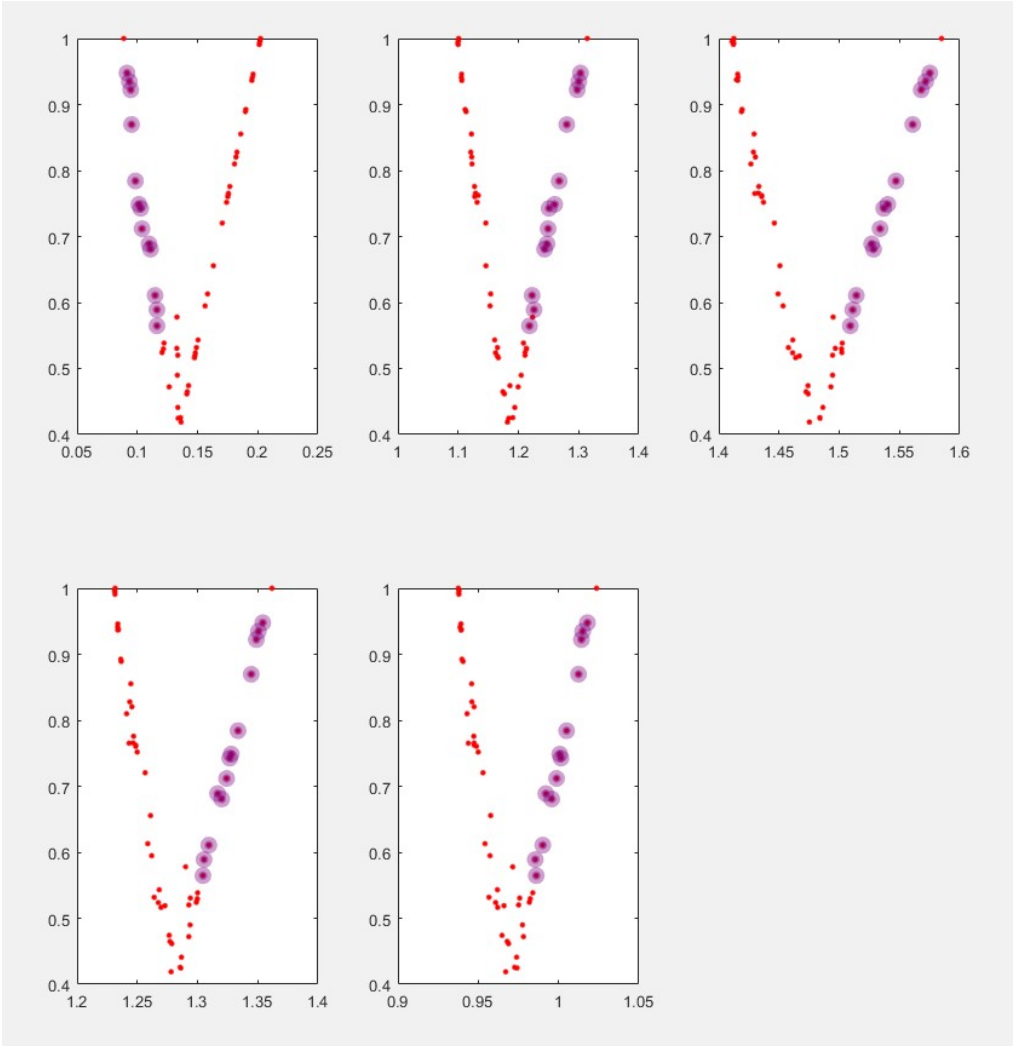


Figura 29: Ejemplo de parámetros de la frontera de Pareto de 5 objetivos con Level Diagram

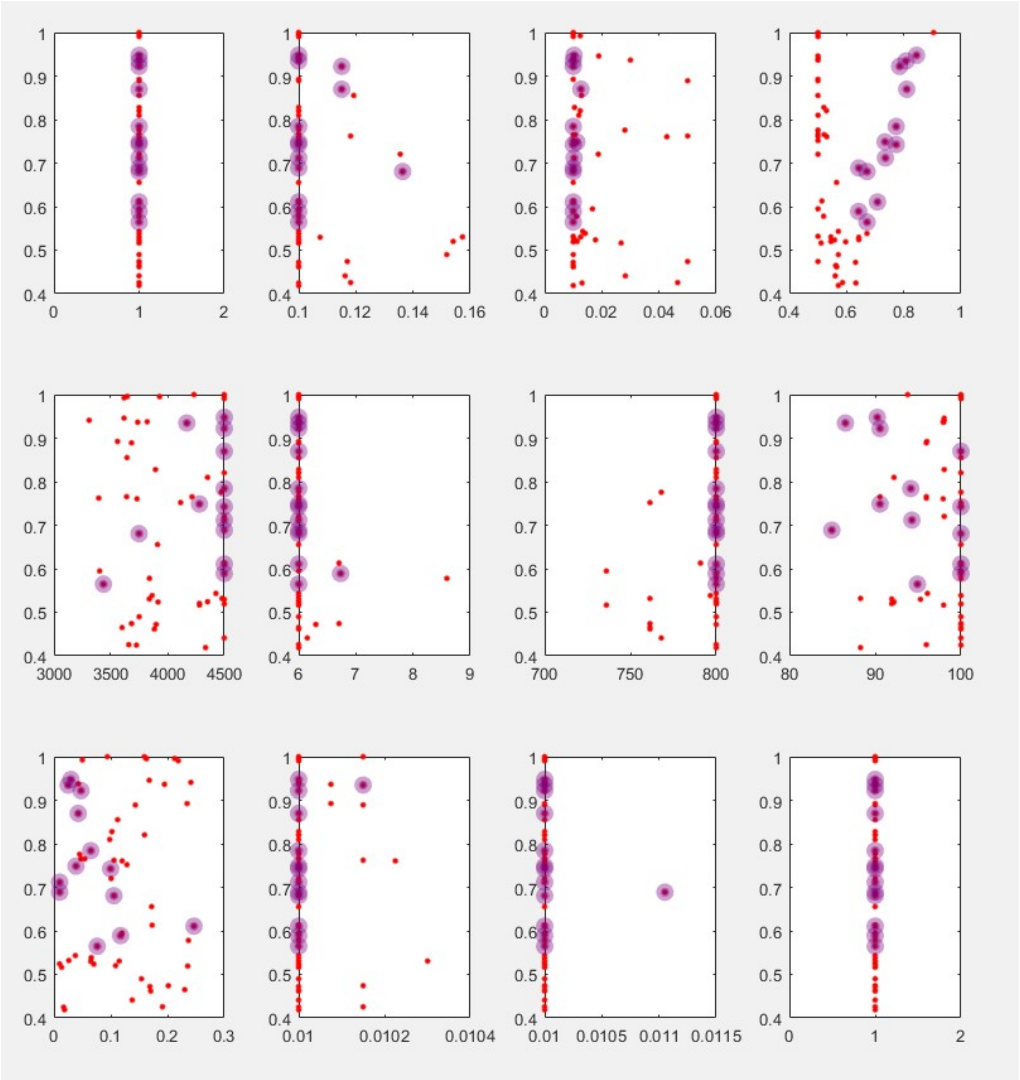


Figura 30: Ejemplo de parámetros identificados tratados con excel

	gamma1	gamma2	gamma4	gamma5	kA	kB	kC	alpha1	dA	dB	dC	Kgain
	1,00E-04	0,1	0,0005	1,13E-02	2036,81569	3441,39232	4500	67,6508526	0,3	0,3	1,20E-01	59,0315309
	0,00691948	0,1	0,0005	0,01	2766,47027	3603,37637	4500	94,162386	0,3	0,3	2,00E-01	73,4437285
	0,00570501	0,1	0,0005	0,01	4026,2046	2873,11513	4080,77662	53,3576569	0,3	0,3	2,89E-01	55,556842
	5,33E-02	0,46472884	0,0005	1,00E-02	2962,05369	773,187079	2518,31109	10	0,01	0,3	2,62E-01	64,4393376
	0,0686256	0,64832984	5	1,00E-02	1744,86228	308,709919	2735,97337	94,162386	0,01	0,3	0,221	73,4437285
	0,09470692	0,1	0,75295304	1,53E-02	4500	829,694782	3590,02234	70,7184744	0,3	0,23	2,00E-01	100
	0,02935495	0,18586395	2,01030751	0,01	2972,91176	207,4998	1859,19657	25,005445	0,01	0,21	3,00E-01	100
	0,03412524	0,26341327	4,32788166	0,01	3685,44163	1540,38413	2936,38428	100	0,3	0,26	1,88E-01	59,4200984
	1,00E-04	0,1	0,0005	0,01	3110,74701	194,648292	4500	100	1,00E-02	0,3	3,00E-01	53,7394163
	0,01615813	0,37618612	1,87156985	0,01	4131,06874	4221,1158	2134,96268	44,6623274	0,01	0,2469	3,00E-01	100
	0,0001	0,34536311	0,0005	0,01	3612,41217	283,246918	4500	100	0,01	0,3	3,00E-01	25,5813723
	0,01663475	0,46685083	0,0005	0,01	4500	2260,28305	2991,31417	100	0,3	0,234	3,00E-01	52,4566875
	4,63E-03	0,1	0,0005	0,01	4500	683,930743	2107,41149	97,4612269	0,3	0,28	0,29	93,1816641
	0,0001	0,26195271	1,88545646	0,01	3618,61799	6	4438,55459	80,91455	0,01	0,07	0,3	88,7664014

	gamma1	gamma2	gamma4	gamma5	kA	kB	kC	alpha1	dA	dB	dC	Kgain
Min	1,00E-04	0,1	0,0005	0,01	1744,86228	6	1859,19657	10	0,01	7,1761685	0,5504377	25,5813723
NºMin	4,00E+00	6	8	12	1	1	1	1	7	1	1	1
Max	0,09470692	0,64832984	5	0,01529052	4500	4221,1158	4500	100	30	30	30	100
NºMax	1	1	1	1	3	1	4	4	1	3	1	3
Media	0,00456463	0,2051806	0,01824093	0,01039983	3318,78295	683,61547	3230,77431	64,1377517	0,45544794	23,0155165	8,01841257	67,4573757

6.2 Proceso de identificación

Mediante las traducciones de parámetros en los modelos que ya se ha mencionado anteriormente se han obtenido los rangos de búsqueda iniciales (tabla 9) que aplicar a los experimentos seleccionados tras la criba. La estrategia a seguir ha sido encontrar los rangos de búsqueda más aproximados a los parámetros que ajusten correctamente el modelo a las respuestas experimentales de un experimento. Luego se ha aplicado ese conjunto de rangos al resto de experimentos para comprobar que el modelo es válido para diferentes conjuntos de datos y comprobar las diferencias que se presentan.

Tras varios procesos de optimización se han alcanzado los rangos de búsqueda finales que se muestran en la tabla 11:

Tabla 12: Rangos de búsqueda finales

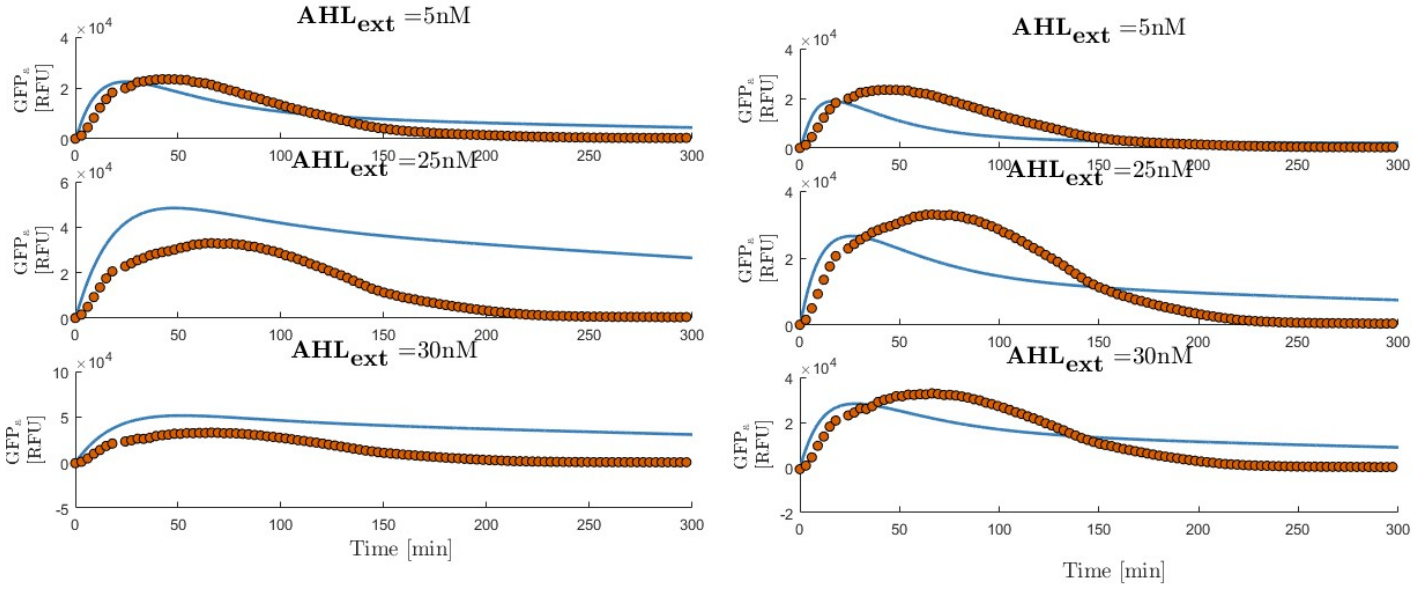
Parámetro	Rango
γ_3	[1.0001 1.01]
γ_4	[5 200]
γ_5	[0.02 2.02]
k_A	[100 6500]
k_B	[0.001 1]
k_C	[1000 4500]
α_1	[0.1 100]
d_A	[0.01 0.3]
d_B	[0.001 0.3]
d_C	[0.01 0.3]

Estos límites no consiguen una modelización perfecta con el tiempo de computación disponible para el presente trabajo. Pero dada la dificultad del problema de optimización se ha llegado a un compromiso entre reducción del error y alteración de rangos de búsqueda. La mayoría de parámetros biológicos no se pueden alterar con total libertad y, como se ha ido describiendo en este documento, se han realizado varias simplificaciones, que aumentan ligeramente el error.

El proceso de optimización se ha iniciado optimizando los datos del experimento 2806 dado su menor desviación del comportamiento teórico (Figura 7). Como se puede ver en la figura 31 la optimización es más eficaz para la concentración de 5nM de inductor. Esto se debe a que, dado que

esta es la única concentración que comparten los cuatro experimentos con los que se ha trabajado, se han modificado los rangos de búsqueda para reducir el error de esta inoculación con el fin de obtener más información en el aspecto de comparación entre los datos experimentales.

Figura 31: Optimización de datos 2806.



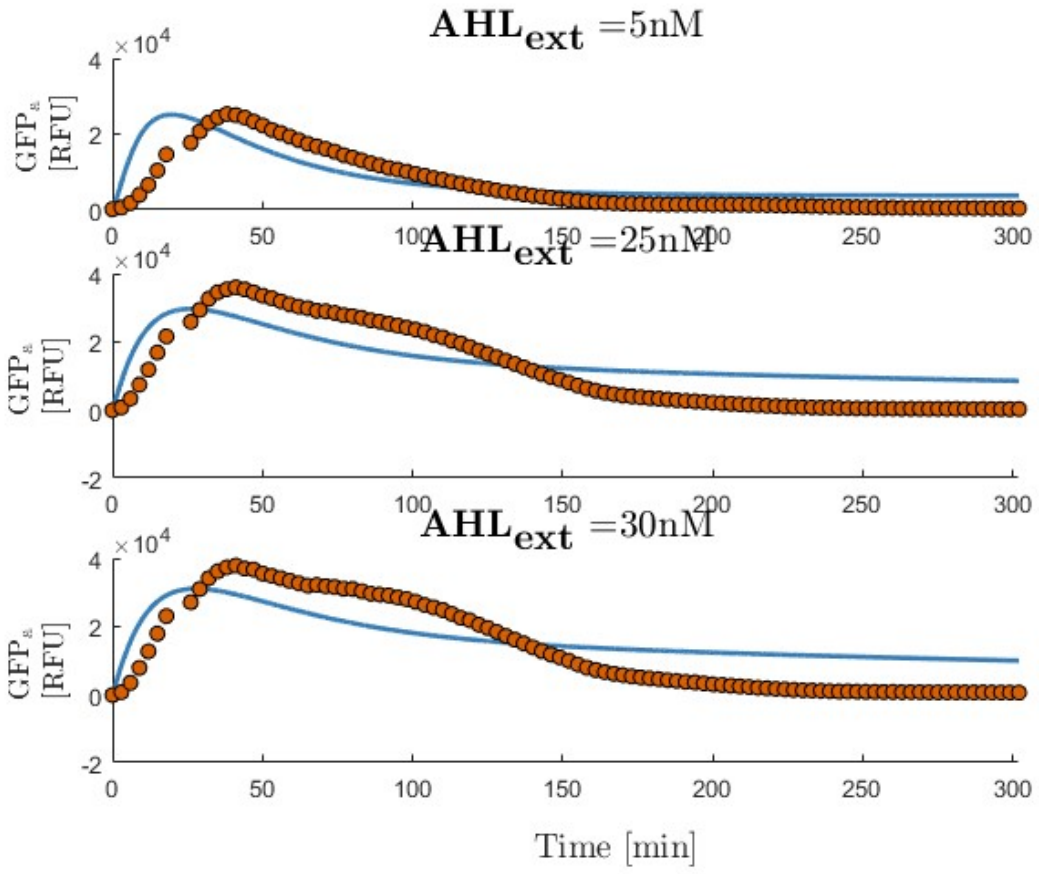
Las dos gráficas de la figura 31 corresponden a los puntos extremos de la frontera de Pareto obtenida, a la izquierda, con error mínimo para 5nM y a la derecha con error mínimo para 30nM. Para ambas concentraciones se encuentran rangos más concretos, que ejemplifican la compensación entre objetivos.

Tabla 13: Parámetros en extremos de frontera de Pareto para 2806

Parámetro	Rangos 5nM	Rangos 30nM
γ_3	[1.004 1.01]	[1.002 1.01]
γ_4	[150 200]	[190 200]
γ_5	[0.02 0.077]	[0.02 0.0014]
k_A	[4400 6800]	[5500 7000]
k_B	[0.04 0.05]	[0.101 0.107]
k_C	[2600 3300]	[2700 3500]
α_1	[0.1 9]	[0.1 47]
d_A	[0.07 0.28]	[0.1 0.2]
d_B	[0.001 0.001]	[0.001 0.001]
d_C	[0.01 0.01]	[0.01 0.01]

La mayoría de parámetros crecen a medida que crece la concentración de molécula inoculadora. También se observan límites que seguramente mejorarían la optimización al ampliarse. Las velocidades de degradación de las proteínas B y C se dirigen a valores inferiores de los permitidos. Este intervalo no se ha ampliado más por razones biológicas, ya que queremos obtener soluciones que sean factibles de reproducir en el laboratorio.

Figura 32: Optimación de datos del experimento 2306



Aplicando el proceso de optimización se obtiene la figura 32, donde se aprecia el segundo pico presente en los datos experimentales (Figura 6), lo que ejemplifica la gran diferencia de comportamientos entre el mismo experimento realizado en diferentes días. En este caso, pese que los datos experimentales se alejan más de los teóricos ideales, el error se ha reducido en todas las concentraciones. Las tendencias presentes en la optimización de los datos 2806, por lo que la posibilidad de utilizar la frontera obtenida anteriormente como población inicial de esta optimización, aumenta la eficacia del proceso

Al optimizar los experimentos con más concentraciones (2106 y 2906), se aprecian algunos comportamientos con más claridad. En esta memoria se muestra como ejemplo los resultados del 2106, dado que el 2906 se desvía mucho del comportamiento ideal (Figura 8) y se ha usado solo como herramienta de confirmación. En la figura 33 se representan los puntos con menor error relativo para 5nM y para 35nM. En este caso, durante la optimización se obtuvieron una cantidad de puntos de la frontera mucho mayor que en los anteriores, por lo que la herramienta Level Diagram presenta una representación clara de la compensación entre los valores de parámetros mientras se recorre la frontera (Figura 34).

Figura 33: Optimización de datos 2106. Diferentes extremos de la frontera de Pareto.

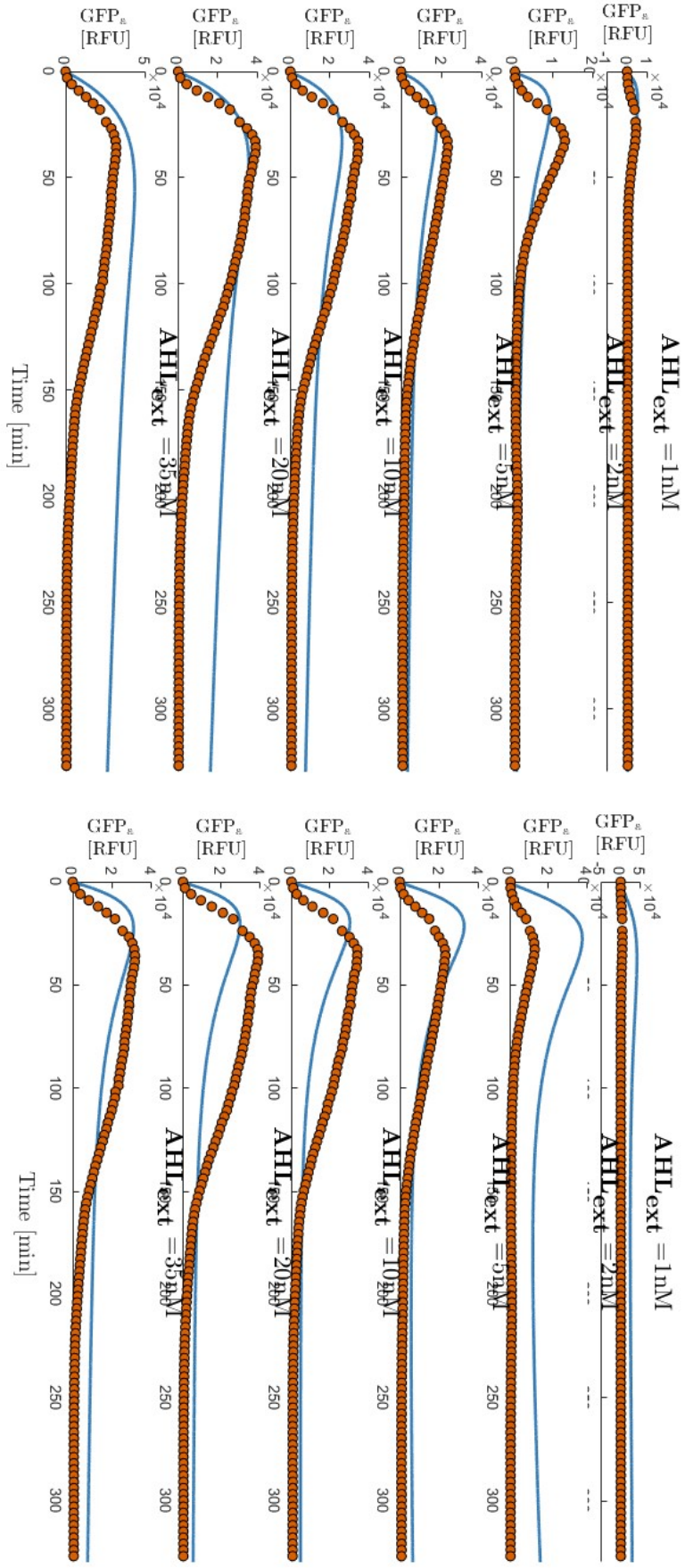
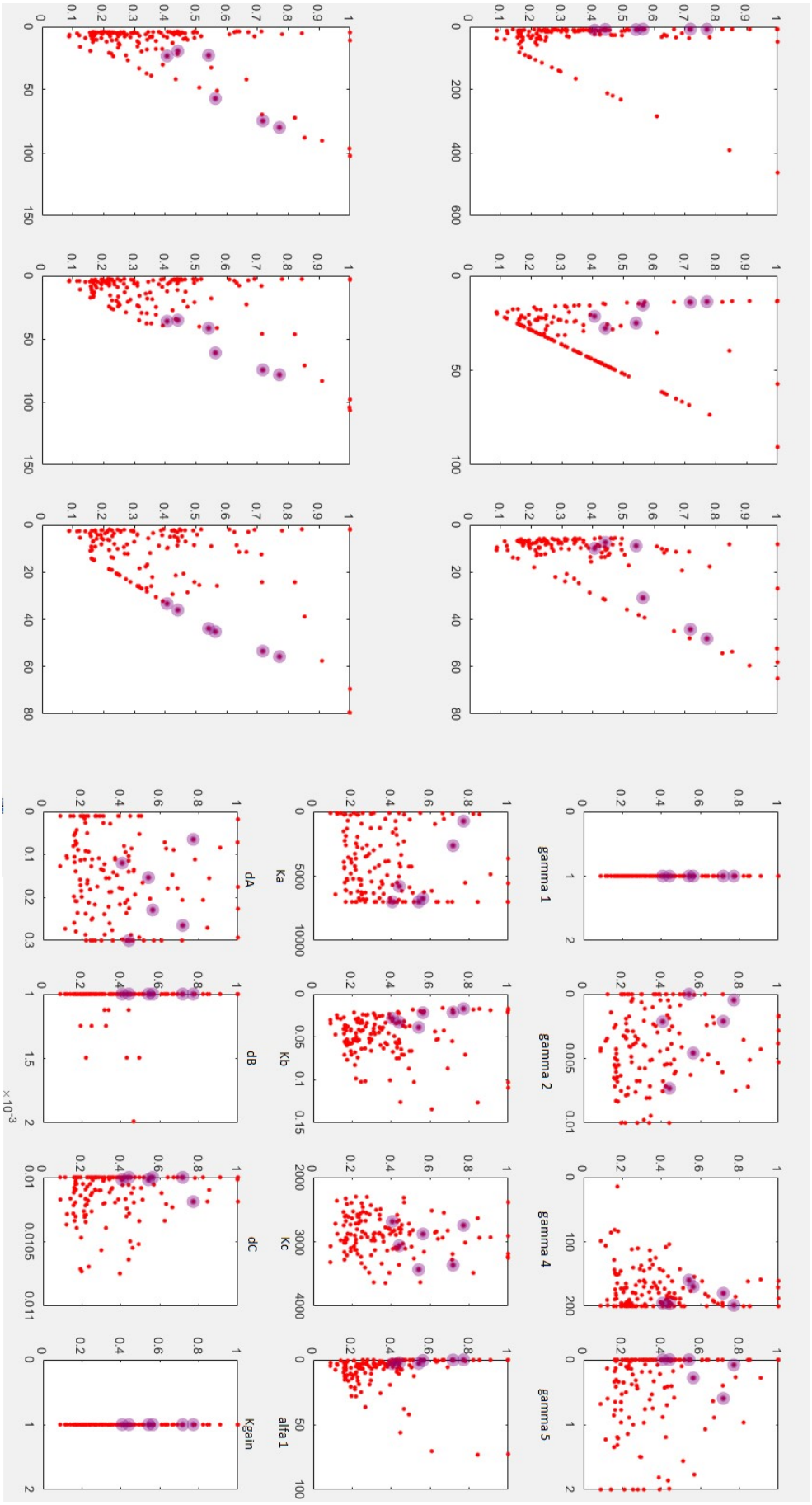


Figura 34: Level Diagram de 2106. Puntos de menor error para 1nM seleccionados.



7 CONCLUSIÓN

Tras la realización de la identificación se puede apreciar el valor de la elaboración de *software* propio para la optimización de modelos. Un *software* estable y modular que sea versátil acelera casi todas las etapas del trabajo por lo que ha sido una de las prioridades durante todo el trabajo. Por ello se han pasado por diversas iteraciones del *software* utilizando diferentes estructuras de datos y forma de tratarlos.

Antes de alcanzar la versión final de las funciones utilizadas para la identificación y dadas las diversas variables que se han ido elaborando, ha proliferado la aparición de errores. Mayormente causados por la gran dependencia de operaciones matriciales cuyos rangos dependían de los datos experimentales y por el cambio de órdenes en variables y parámetros al cambiar el modelo, hasta que se ha alcanzado una solución casi completamente modular.

También cabe resaltar la utilidad de mantener el código comprensible y comentado. Al realizar el trabajo a partir de *software* existente con el fin de optimizarlo, una buena comprensión del código original evita errores a lo largo del desarrollo.

La necesidad de comprensión también se traslada a los cálculos. Aunque no muy extensos durante la realización del proyecto, el hecho de trabajar con modelos con gran número de ecuaciones y variables requiere que todas se mantengan bien identificadas durante procesos como la reducción del modelo y que todas sean consistentes en los diferentes documentos y *software* que se utiliza.

Tras tener en cuenta todos los resultados obtenidos en las optimizaciones se pueden llegar a conclusiones generales a tener en cuenta para la continuación del trabajo de caracterización. La primera anomalía a tener en cuenta es un adelantamiento en la subida de la fluorescencia en todos los datos simulados. Esto se debe al hecho que la proteína C tiene un periodo de maduración que no se ha tenido en cuenta en el modelo, y que deberá ser tenido en cuenta en el futuro. El retraso observado es de unos 20 minutos en casi todos los casos, valor que es consistente con los tiempos de maduración de la proteína gfp usada como salida del circuito.

La caracterización final del circuito con un único conjunto de valores fijos para los parámetros es imposible dada la presencia de fuerte variación paramétrica y dinámicas no modeladas. La gran dificultad computacional de la optimización requeriría una inversión mayor de tiempo de computación y selección de rangos de búsqueda de parámetros para conseguir mejores resultados que los obtenidos, por lo que se pretende que los intervalos que se han señalado en el apartado anterior sirvan simplemente como punto de partida para la continuación del trabajo.

Como conclusión final, se puede afirmar que el *software* modificado reduce considerablemente el tiempo requerido para cada optimización, en muchos casos pudiendo realizar en horas el número de evaluaciones que antes se realizaba en días. También se ha confirmado su buen funcionamiento y estabilidad, así como se ha comprobado la validez del modelo reducido, en relación con el modelo original.

BIBLIOGRAFÍA

[1] A. Cheng y T. K. Lu. Synthetic Biology: An Emerging Engineering Discipline. *Annual Review of Biomedical Engineering.* 10:14, 2012

[2] D. E. Cameron, C. J. Bashor y J. J. Collins. A brief history of synthetic biology. *Nature Reviews*, volumen 12, páginas 381:390, 2014

[3] G. M. Church, M. B. Elowitz, C. D. Smolke, C. A. Voigt y R. Weiss. Realizing the potential of synthetic biology. www.nature.com/reviews/molcellbio. Accedido el: 01-07-2017.

[4] Y.Boada, G.Reynoso-Meza, J. Picó y A. Vignoni. Multi-objective optimization framework to obtain model-based guidlines for tunint biological synthetic devices: an adaptive network case. *BMC Systems Biology*. 10:27, 2016

[5] D. D. Vecchio, A. J. Dy y Y. Qian. Control Theory Meets Synthetic Biology. https://www.researchgate.net/publication/305480045_Control_theory_meets_synthetic_biology . Accedido el: 02-07-2017.

[6] https://es.wikipedia.org/wiki/Homeostasis. Accedido el: 2-07-2017

[7] Y. Boada, A. Vignoni, G. Reynoso-Meza, J. Picó. Parameter identification in synthetic biological circuits using multi-objective optimization. I.U. de Automática e Informática Industrial (ai2), Universitat Politècnica de Valencia.

[8] D. E. R. P. Shetty y T. F. Knight. Engineering biobrick vectors from biogrick parts. *Biological Engineering.* 2008

[9] J. Picó, A. Vignoni. E Picó-Marco, Y. Boada. Modelado de sistemas bioquímocos: de la Ley de Acción de Masas a la Aproximación Lineal del Ruido. *Revista Iberoamericana de Automática e Informática Industrial 12.* Páginas 241:252, 2015.

[10] A. G. Boscá, J. Picó, Y. Boada y A. Vignoni. Design and Characterization of a Robust Incoherent Feedforward Synthetic Genetic Circuit. Universitat Politècnica de València. 2014.

[11] https://es.mathworks.com/help/Matlab/ref/ode15s. Accedido el: 01-07-2017

[12] https://es.mathworks.com/help/distcomp/run-code-on-parallel-pools.html. Accedido el: 01-07-2017

PARTE II:
PRESUPUESTO

1 PRESUPUESTO

1.1 Introducción

En esta parte del documento se detallará los costes del equipo usado en los experimentos, la realización de estos, el desarrollo del *software* necesario para la identificación de los parámetros y las horas de mano de obra requeridas para la realización del presente trabajo. Estos apartados se han clasificado en las siguientes unidades de trabajo (UT) y ordenado según el orden de realización.

- **Obtención de datos experimentales**
  - UT 1: Preparación del experimento.
  - UT 2: Verificación final de los plasmas
  - UT 3: Experimento de medición de fosforescencia.

- **Identificación del modelo**
  - UT 4: Reducción del modelo matemático.
  - UT 5: Adaptación del código existente al nuevo modelo.
  - UT 6: Identificación mediante software.

- **Creación de documentación**

1.2 Coste de mano de obra

En el presente trabajo se utilizan dos tipos de mano de obra: técnicos de laboratorio biotecnológico para la obtención de datos experimentales e ingenieros industriales para el desarrollo del modelo, *software*, identificación y documentación.

Cada experiencia utilizada para el presente trabajo requiere un día y medio de trabajo para el técnico de laboratorio. Aunque tras la criba de datos experimentales solo se han utilizado los datos de 4 experimentos, se ha realizado un total de 10 para poder desarrollar los criterios necesarios para la selección y para determinar las condiciones iniciales ideales.

Por lo tanto, el cálculo total de horas de trabajo para el técnico de laboratorio biotecnológico:

$$10\ experimentos * (24 + 12)\frac{horas}{experimento} = 360\ horas$$

El resto de unidades de obra han sido realizadas por 2 ingenieros industriales durante 4 meses (Marzo, Abril, Mayo y Junio) con un horario de 5 días a la semana, 4 horas al día. Dando un total de:

$$5\ meses * 3.5 semanas * 6\frac{días}{semana} * 7\frac{horas}{día} = 750 horas$$

Tabla 14: Coste de mano de obra

Unidades	Concepto	Horas de trabajo	Precio unitario (€/h)	Precio
1	Coste de técnico de laboratorio incluyendo retribución y cuotas patronales.	360	3.13	1126.8
2	Coste de ingeniero industrial incluyendo retribución y cuotas patronales	750	3.13	2347.5
			TOTAL	3474.3

1.3 Coste de recursos materiales

En esta parte se incluyen los costes del equipo necesario para realizar las mediciones y las partes biológicas estándares que componen el circuito genético utilizado. Estos costes incluyen los indicados en los catálogos normalizados y, en caso de ser necesarios, repuestos para aquellas partes que presenten defectos.

El circuito está formado por 2 conjuntos de 3 promotores, 3 zonas de codificación y 2 terminadores. Cada uno de estos está estimado en 40pb. Es necesario duplicar la cantidad ya que se requieren dos conjuntos por plásmido.

Una unidad de base pura de HPSF cuesta 0.49 €, de forma que el presupuesto se puede calcular de la siguiente manera:

- Síntesis de las partes estándar, correspondiente a 100pb por parte:

$$100pb * 0.49\frac{€}{pb} * 2 = 98€$$

- Creación de los conjuntos:

$$40pb * 0.49\frac{€}{pb} * 16 = 313.6€$$

- Se necesitan 4 *primers* para la secuenciación final de los plásmidos:

$$20pb * 0.49\frac{€}{pb} * 4 = 39.2€$$

Se añaden como fungibles los gastos necesarios para equipamiento variado usado en el laboratorio (pipetas, microtubos, recipientes de cultivación…) a los que se le ha asignado un 30% del coste de recursos materiales de cada unidad de trabajo en la que sean necesarios.

Tabla 15: Coste de recursos materiales

UT	**Concepto**	**Cantidad**	**Precio unitario (€)**	**Coste**
1	Reparto de los componentes	1	100	100
1	Síntesis de las partes estándar	2	49	98
1	*Primers*	16	19.6	313.6
1	Kit de reactante para clonación	2	20	40
2	*Primers* para secuenciación	4	9.8	39.2
	Fungibles (30%)	3		47.64
			TOTAL	638.44

7.4 Presupuesto parcial por unidad de trabajo

Tabla 16: UT 1 Preparación del experimento.

Ud	Descripción	Cantidad	Precio unitario (€)	Coste
h	Ingeniero	48	3.13	150.24
h	Técnico	120	3.13	375.6
u	Reparto de los componentes	1	100	100
u	Síntesis de las partes estándar	2	49	98
u	Purificación de plásmidos	8	1.8	14.4
u	*Primers*	16	19.6	313.6
u	Kit de reactantes para clonación	2	20	40
u	Partes de la purificación del PCR	8	1.6	12.8
	Fungibles(30%)			173.64
			TOTAL	1278.28

Tabla 17: UT 2 Verificación de los plásmidos.

Ud	Descripción	Cantidad	Precio unitario (€)	Coste
h	Ingeniero	48	3.13	639.84
h	Técnico	120	3.13	375.6
u	Purificación de plásmidos	2	1.8	3.6
u	Verificación de plásmidos	4	7.5	30
u	*Primers* para secuenciación	4	9.8	39.2

	Fungibles(30%)		7.28
		TOTAL	1095.52

Tabla 18: UT 3 Experimento de medición de fosforescencia.

Ud	**Descripción**	**Cantidad**	**Precio unitario (€)**	**Coste**
h	Ingeniero	48	3.13	639.84
h	Técnico	120	3.13	375.6
u	Purificación de plásmidos	2	1.8	3.6
u	Verificación de plásmidos	4	7.5	30
u	*Primers* para secuenciación	4	9.8	39.2
	Fungibles(30%)			7.28
			TOTAL	1095.52

Tabla 19: UT 4 Reducción del modelo matemático.

Ud	**Descripción**	**Cantidad**	**Precio unitario (€)**	**Coste**
h	Ingeniero	106	3.13	331.78
			TOTAL	331.78

Tabla 20: UT 5 Adaptación del código existente al nuevo modelo.

Ud	Descripción	Cantidad	Precio unitario (€)	Coste
h	Ingeniero	350	3.13	1095.5
			TOTAL	1095.5

Tabla 21: UT 6 Identificación mediante software.

Ud	Descripción	Cantidad	Precio unitario (€)	Coste
h	Ingeniero	100	3.13	313
			TOTAL	313

Tabla 22: UT 7 Desarrollo de documentos.

Ud	Descripción	Cantidad	Precio unitario (€)	Coste
h	Ingeniero	50	3.13	156.5
			TOTAL	156.5

7.5 Presupuesto final

Obtenido sumando los presupuestos parciales de cada unidad de trabajo y añadiendo los impuestos necesarios mediante el IVA.

Tabla 23: Presupuesto final

01	Obtención de datos experimentales	3469.32
02	Identificación del modelo	1740.28
03	Creación de documentación	156.5
	Presupuesto total material	5366.1
	21% IVA	1126.881
	Presupuesto total	6592.981

El coste total del trabajo asciende a SEIS MIL QUINIENTOS NOVENTA Y DOS EUROS CON NOVECIENTOS OCHENTA Y UN CÉNTIMOS.

PARTE III:
ANEXOS

1 ANEXO 1

1.1 Función readExperiment

```
function evol=readExperiment
% Programa para visualización experimentos Cytation

% minutos en un día
mind = 60*24;

% Temperatura experimento
Texp= 37;

% Nombre fichero. Se puede abrir diálogo para introducir nombre
% file='./260516-ID126-132-127-AHL-PST.xlsx';
file=uigetfile('*.xlsx;*.xls');
if isequal(file,0)
    return;
end

% Abre diálogo para definición de las placas
[placa,rangos,tind]=DefPlaca;

if isempty(placa)
    return;
end

pdim=size(placa);
pdim(1)=pdim(1)-1;
plc=struct('strain',{},'Id',{},'isConc',{});
% Genera estructura de datos con la información de la placa

for i=1:pdim(1)
    for j=1:pdim(2)
        plc(i,j).strain=placa{end,j};
        aux=placa{i,j};
        plc(i,j).Id=aux;
        plc(i,j).isConc=(sum(isletter(aux))==0);
        plc(i,j).medio=placa{1,j};
        plc(i,j).i=i;
        plc(i,j).j=j;
        plc(i,j).col=(i-1)*12+j+2;
    end
end

%Convierte el struct array en tabla para facilitar consulta
tplaca=struct2table(plc(:));
% Busca índices cuyo Id sea igual a un str
%find(strcmp(a.Id,'LB'))
% Selecciona elemontos cuyo Id sea igual a str
%a(strcmp(a.Id,'LB'),:)
%a(strcmp(a.Id,'10'),:)

% Elimina las filas vacias
TF = ismissing(tplaca);
tplaca=tplaca(~any(TF,2),:);
```

```
% Distribución celdas experimento. Anaizar si se puede automatizar
OD_preRef=rangos{1};
F_preRef=rangos{2};
OD_postRef=rangos{3};
F_postRef=rangos{4};

OD_pre = xlsread(file,OD_preRef);
OD = xlsread(file,OD_postRef);
F_pre = xlsread(file,F_preRef);
F = xlsread(file,F_postRef);

OD(:,1)=OD(:,1)+F_pre(end,1)+tind/mind;
F(:,1)=F(:,1)+F_pre(end,1)+tind/mind;

OD_exp=[OD_pre;OD];
F_exp=[F_pre;F];

% Almacenamiento experimento en struct array
% Permite el acceso estructurado a los datos del experimento

% Nombres de strains
cStraini=unique(tplaca.strain,'stable');
% Tipos de experientos realizados
mediosi=unique(tplaca.medio,'stable');
%Concentraciones. Revisar poque podría haber distinto número
%concentraciones en función de strain/medios
Ci=unique(tplaca(tplaca.isConc,:).Id,'stable');


%Genera un struct anidado strain->medios->datos para almacenar datos
%experimento

evol=cell2struct(cStraini','strain');
evol.placa=placa;
evol.rangos=rangos;
evol.tinduccion=tind;
evol.file=file;

for i=1:length(evol)
    % Primer nivel: Array para cada strain: campo strain mantiene cepa y
medios
    % es un array con los medios usados

    % Almacena el vector con el tiempo de las medidas
    % Si se dispone de información de tiempo para cada celda se podría
    % incluir el tiempo en la matriz de datos del pocillo

    evol(i).time_OD=OD_exp(:,1)*mind;
    evol(i).time_F=F_exp(:,1)*mind;

    evol(i).medios=cell2struct(mediosi','medio');
    for k=1:length(evol(i).medios)
        % segundo nivel nivel: Cada medio tiene su Id (medio) y los datos
para
        % cada concentracion (data)

        % Obtiene el índice de columnas con medios

jm=table2array(tplaca(strcmp(tplaca.Id,evol(i).medios(k).medio),'col'))';
        % Almacena los datos de experimentos
        evol(i).medios(k).OD=OD_exp(:,jm);
```

```
        evol(i).medios(k).F=F_exp(:,jm);
        % Obtiene la evolución media
        evol(i).medios(k).OD_medio=mean(OD_exp(:,jm),2);
        evol(i).medios(k).F_medio=mean(F_exp(:,jm),2);

        evol(i).medios(k).data=cell2struct(Ci','Inducc');
        % Busca medio con concentración 0 para referencia medida
        for j=1:length(evol(i).medios(k).data)
            if(strcmp(evol(i).medios(k).data(j).Inducc,'0'))
                h=j;
            end
        end
        % El siguiente bucle se inicia siempre con concentración 0
        auxFOD0=[]; % almacena temporalmente ratio sin inductor
        for j=[h, 1:(h-1), (h+1):length(evol(i).medios(k).data)]
            % Obtiene el índice de columnas con medios

jm=table2array(tplaca(strcmp(tplaca.strain,evol(i).strain)&...
                strcmp(tplaca.medio,evol(i).medios(k).medio)&...

strcmp(tplaca.Id,evol(i).medios(k).data(j).Inducc),'col'))';
            % Obtiene la evolución media
            evol(i).medios(k).data(j).OD=OD_exp(:,jm);
            evol(i).medios(k).data(j).F=F_exp(:,jm);
            meanOD=mean(OD_exp(:,jm),2);
            meanF=mean(F_exp(:,jm),2);
            evol(i).medios(k).data(j).OD_medio=meanOD;
            evol(i).medios(k).data(j).F_medio=meanF;
            if j==1
%                 auxFOD0=meanF./meanOD;
                auxFOD0=meanF;
            end
%             evol(i).medios(k).data(j).FOD=meanF./meanOD-auxFDO0;
            evol(i).medios(k).data(j).FOD=(meanF-auxFOD0)./meanOD;
        end
    end

    for ii=1:length(evol(1).medios(k).data)

evol.medios.data(ii).FA=evol.medios.data(ii).F./evol.medios.data(ii).OD;
    end

end
```

1.2 Función plotExperiment

```
function plotExperiment(evol)
%Visualiza resultados de los experimentos con Cytation

% Tamaño pantalla para escalado figuras
scrsz = get(groot,'ScreenSize');

% Visualiza resultados experimento

%Ventana para OD
hndOD=figure(1);
set(hndOD, 'OuterPosition',[1,scrsz(4)/2, scrsz(3)*2/3, scrsz(4)/2] );
```

```
% Ventana para F
hndF=figure(2);
set(hndF, 'OuterPosition',[1,1, scrsz(3)*2/3, scrsz(4)/2] )

% Ventana para ratio F/OD
hndFOD=figure(3);
set(hndFOD, 'OuterPosition',[1,1, scrsz(3)*2/3, scrsz(4)/2] )

for i=1:length(evol)
    for k=1:length(evol(i).medios)
        auxOD=[]; auxF=[];
        for j=1:length(evol(i).medios(k).data)

            auxOD=[auxOD,evol(i).medios(k).data(j).OD_medio];
            auxF=[auxF,evol(i).medios(k).data(j).F_medio];
            auxFOD=[auxF,evol(i).medios(k).data(j).FOD];
        end
        figure(hndOD), subplot(length(evol(i).medios),length(evol),(k-
1)*length(evol)+i)
            plot(evol(i).time_OD,auxOD,'LineWidth',2), grid on,
ylabel('mean OD600'), xlabel(['t (min), ', evol.file])
            legend(evol(i).medios(k).data.Inducc, 'Location','northwest')
            title(['Strain: ', evol(i).strain , ' Medio;
',evol(i).medios(k).medio])
        figure(hndF), subplot(length(evol(i).medios),length(evol),(k-
1)*length(evol)+i)
            plot(evol(i).time_F,auxF,'LineWidth',2), grid on,
ylabel('mean F'), xlabel(['t (min), ', evol.file])
            legend(evol(i).medios(k).data.Inducc, 'Location','northeast')
            title(['Strain: ', evol(i).strain , ' Medio;
',evol(i).medios(k).medio])
        figure(hndFOD), subplot(length(evol(i).medios),length(evol),(k-
1)*length(evol)+i)
            plot(evol(i).time_F,auxFOD,'LineWidth',2), grid on,
ylabel('mean F/OD'), xlabel(['t (min), ', evol.file])
            legend(evol(i).medios(k).data.Inducc, 'Location','northeast')
            title(['Strain: ', evol(i).strain , ' Medio;
',evol(i).medios(k).medio])
    end
end
```

1.3 Código de la interfaz gráfica de readExperiment

```
function varargout = DefPlaca(varargin)
% DEFPLACA MATLAB code for DefPlaca.fig
%      DEFPLACA, by itself, creates a new DEFPLACA or raises the existing
%      singleton*.
%
%      H = DEFPLACA returns the handle to a new DEFPLACA or the handle to
%      the existing singleton*.
%
%      DEFPLACA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DEFPLACA.M with the given input
arguments.
%
%      DEFPLACA('Property','Value',...) creates a new DEFPLACA or raises
the
```

```
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before DefPlaca_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to DefPlaca_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DefPlaca

% Last Modified by GUIDE v2.5 05-Jul-2016 13:40:40

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DefPlaca_OpeningFcn, ...
                   'gui_OutputFcn',  @DefPlaca_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before DefPlaca is made visible.
function DefPlaca_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to DefPlaca (see VARARGIN)

% Choose default command line output for DefPlaca
handles.output = hObject;
handles.ok=0;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes DefPlaca wait for user response (see UIRESUME)
uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = DefPlaca_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = [];
varargout{2} = [];
varargout{3} = [];
if ~isempty(handles)
    if handles.ok
        varargout{1} = get(handles.placa,'Data');
        varargout{2} = get(handles.rangos,'Data');
        varargout{3} = str2num(get(handles.Tind,'String'));
    end
    close(handles.figure1);
end

function Tind_Callback(hObject, eventdata, handles)
% hObject    handle to Tind (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Tind as text
%        str2double(get(hObject,'String')) returns contents of Tind as a
double


% --- Executes during object creation, after setting all properties.
function Tind_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tind (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in OkButton.
function OkButton_Callback(hObject, eventdata, handles)
% hObject    handle to OkButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.ok=1;
guidata(hObject, handles);
uiresume(handles.figure1);


% --- Executes on button press in cancelButton.
function cancelButton_Callback(hObject, eventdata, handles)
% hObject    handle to cancelButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
uiresume(handles.figure1);


% --- Executes on button press in StoreButton.
function StoreButton_Callback(hObject, eventdata, handles)
% hObject    handle to StoreButton (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file=uiputfile('*.mat');
if isequal(file,0)
    return;
end
placa=get(handles.placa,'Data');
rangos=get(handles.rangos,'Data');
save(file, 'placa', 'rangos');


% --- Executes on button press in LoadButton.
function LoadButton_Callback(hObject, eventdata, handles)
% hObject    handle to LoadButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file=uigetfile('*.mat');
if isequal(file,0)
    return;
end
var=load(file);
set(handles.placa,'Data',var.placa);
set(handles.rangos,'Data',var.rangos);


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: delete(hObject) closes the figure
%uiresume(handles.figure1);
delete(hObject);
```

2 ANEXO II

2.1 Función spMODEparam2

```
function spMODEDat=spMODEparam2(evol)

%% Debug mode started
dbstop if error

 ne=1; %número de especie
 nm=1;   %número de medio
 spMODEDat.evol=evol;
 dimensions= size(evol(ne).medios(nm).data);
 Conditions = dimensions(1)-1;
 for i=1:(Conditions)
    spMODEDat.INPUTS(i)=str2num(evol(nm).medios(ne).data(i+1).Inducc);
 end
%
 [a,b]=size(evol(ne).medios(nm).data(1).OD);
%[a,b]=size(dades_2906_OD_132_01_1);
spMODEDat.Copies = b;  % Number of copies per experiment

spMODEDat.NOBJ = Conditions;                  % Number of objectives

% spMODEDat.NVAR = 18;                  % Number of decision variables
spMODEDat.NVAR = 12;

spMODEDat.mop  =...
    str2func('CostFunction_adaptation_GFP2');        % Cost Function for
adaptation at I=0nM

spMODEDat.CostProblem  = 'Adaptation_identification';    % Problem
Instance

spMODEDat.FieldD  = [50 100; ...              % gamma1
                     0.0001 0.5; ...          % gamma3
                     0.0005 5; ...            % gamma4
                     1 100; ...               % gamma5
                     30000 60000;0 20;25000 100000;...         % kA, kB, y
kC
                     10 1000;...       % alpha1, promoter hil constant
                     0.001 0.1;... %dA
                     0.0001 1;...  %dB
                     0.00001 0.01;...  %dC
                     0.1 1];                  %kgain of the model

spMODEDat.Initial = spMODEDat.FieldD;% Initialization bounds
%
%%
%% Variables regarding the optimization algorithm (Differential
Evolution)

spMODEDat.Xpop = 90;   % Population size,initial (5-10 X NVAR)

spMODEDat.SubXpop=50;  % SubPopulation size

spMODEDat.ScalingFactor = 0.5;       % Scaling factor
```

```
spMODEDat.CRrate= 0.9;                % Croosover Probability (para
problemas separables 0.2-0.3, no separables 0.8-0.9)
%
%%
%% Variables regarding spreading (spherical pruning)

spMODEDat.Strategy='SphP';            % 'Push' for a basic Dominance-based
                                      % selection; 'SphP' for the
spherical
                                      % pruning;

spMODEDat.Alphas=10;                 % Number of Arcs (Strategy='SphP').
(numero soluciones en el FPareto Alphas^(NOBJ-1))

spMODEDat.Norma='manhattan';          % Norm to be used in
Strategy='SphP';
                                      % It could be 'eucliden',
'manhattan',
                                      % 'infinite'.

spMODEDat.StopSize=15000;             % Maximum Pareto optimal solutions
                                      % required.
%
%%
%% Variables regarding convergence improving (elitism)

spMODEDat.CarElite = ...              % Solutions from the Approximated
    spMODEDat.SubXpop/2 - ...         % Pareto front in a generation
    spMODEDat.NOBJ;                   % to be merged with the population
                                      % in the evolution process.
%
%%
%% Variables regarding basic pertinency (Bounds on objectives)
%
spMODEDat.Pertinency= [];
                                      % Bounds on objectives; implemented
                                      % in the CostFunction. A row for
each
                                      % objective, minimum and maximum
                                      % values desired. If empty, the
                                      % Pareto front approximated will be
                                      % unbounded. Please, refer to:
% G. Reynoso-Meza, J. Sanchis, X. Blasco, J.M. Herrero. Multiobjective
% evolutionary algorithms for multivariable PI controller design. Expert
% Systems with Applications Volume 39, Issue 9, July 2012, Pages
7895ï¿½7907.

%
%%
%% Execution Variables

spMODEDat.MAXGEN =1e4;               % Generation bound

spMODEDat.MAXFUNEVALS = 2e3; %2e3       % Function evaluations bound

spMODEDat.PobInitial=[];             % Initial population (if any)

spMODEDat.SaveResults='yes';         % Write 'yes' if you want to
                                     % save your results after the
                                     % optimization process;
                                     % otherwise, write 'no';
```

```
spMODEDat.Plotter='yes';            % 'yes' if you want to see some
                                    % a graph at each generation.

spMODEDat.SeeProgress='yes';         % 'yes' if you want to see some
                                    % information at each generation.
%
%%
%% Initialization (don't modify)

spMODEDat.CounterGEN=0;             % Counter for generations.

spMODEDat.CounterFES=0;             % Counter for function evaluations.
%
%%

%% Choose data for identification and validation
% We take time from the experimental data

  load evol
 spMODEDat.DataTime = spMODEDat.evol.time_OD(1:end-8);


  spMODEDat.DataOD = [];
  spMODEDat.DataGFP_per_cell= [];
  for i=1:Conditions
      spMODEDat.DataOD=[spMODEDat.DataOD,
spMODEDat.evol.medios.data(i+1).OD(9:end,:)];
      spMODEDat.DataGFP_per_cell=[spMODEDat.DataGFP_per_cell,
spMODEDat.evol.medios.data(i+1).FA(9:end,:)];
  end

end
```

2.2 Función spMODEparam2C

```
function spMODEDat=spMODEparam2C

%% Debug mode started
dbstop if error


load C132_01_2306
spMODEDat.INPUTS=[5 25 30];

spMODEDat.Copies = 4;  % Number of copies per experiment

spMODEDat.NOBJ = 6;                  % Number of objectives

spMODEDat.NVAR = 12;                  % Number of decision variables

spMODEDat.mop  =...
    str2func('CostFunction_adaptation_GFP2');        % Cost Function for
adaptation at I=0nM
```

```
spMODEDat.CostProblem  = 'Adaptation_identification';    % Problem
Instance

                spMODEDat.FieldD  = [1 1; ...              % gamma1
                    0.0001 1; ...          % gamma3
                    0.001 1; ...            % gamma4
                    1 10; ...               % gamma5
                    0.1 6000;0.001 4500;1 5000;...         % kA, kB, y kC
                    0.1 100;...       % alpha1, promoter hil constant
                    0.001 0.3;... %dA
                    0.001 0.3;...  %dB
                    0.001 0.3;...  %dC
                    1 1];                   %kgain of the model

spMODEDat.Initial = spMODEDat.FieldD;% Initialization bounds
%
%%
%% Variables regarding the optimization algorithm (Differential
Evolution)

spMODEDat.Xpop = 90;   % Population size,initial (5-10 X NVAR)

spMODEDat.SubXpop=50;  % SubPopulation size

spMODEDat.ScalingFactor = 0.5;       % Scaling factor

spMODEDat.CRrate= 0.9;               % Croosover Probability (para
problemas separables 0.2-0.3, no separables 0.8-0.9)
%
%%
%% Variables regarding spreading (spherical pruning)

spMODEDat.Strategy='SphP';           % 'Push' for a basic Dominance-based
                                     % selection; 'SphP' for the
spherical
                                     % pruning;

spMODEDat.Alphas=10;                % Number of Arcs (Strategy='SphP').
(numero soluciones en el FPareto Alphas^(NOBJ-1))

spMODEDat.Norma='manhattan';         % Norm to be used in
Strategy='SphP';
                                     % It could be 'eucliden',
'manhattan',
                                     % 'infinite'.

spMODEDat.StopSize=15000;            % Maximum Pareto optimal solutions
                                     % required.
%
%%
%% Variables regarding convergence improving (elitism)

spMODEDat.CarElite = ...             % Solutions from the Approximated
    spMODEDat.SubXpop/2 - ...        % Pareto front in a generation
    spMODEDat.NOBJ;                  % to be merged with the population
                                     % in the evolution process.
%
%%
%% Variables regarding basic pertinency (Bounds on objectives)
%
spMODEDat.Pertinency= [];
```

```
                                    % Bounds on objectives; implemented
                                    % in the CostFunction. A row for
each
                                    % objective, minimum and maximum
                                    % values desired. If empty, the
                                    % Pareto front approximated will be
                                    % unbounded. Please, refer to:
% G. Reynoso-Meza, J. Sanchis, X. Blasco, J.M. Herrero. Multiobjective
% evolutionary algorithms for multivariable PI controller design. Expert
% Systems with Applications Volume 39, Issue 9, July 2012, Pages
7895ï¿½7907.

%% Execution Variables

spMODEDat.MAXGEN =1e4;              % Generation bound

spMODEDat.MAXFUNEVALS = 1e5; %2e3       % Function evaluations bound

load PInit_bajos
spMODEDat.PobInitial=PInit_bajos;           % Initial population (if
any)

spMODEDat.SaveResults='yes';        % Write 'yes' if you want to
                                    % save your results after the
                                    % optimization process;
                                    % otherwise, write 'no';

spMODEDat.Plotter='yes';            % 'yes' if you want to see some
                                    % a graph at each generation.

spMODEDat.SeeProgress='yes';         % 'yes' if you want to see some
                                    % information at each generation.
%
%%
%% Initialization (don't modify)

spMODEDat.CounterGEN=0;             % Counter for generations.

spMODEDat.CounterFES=0;             % Counter for function evaluations.

spMODEDat.DataTime=time_2306_OD(1:end-8);

 spMODEDat.DataOD = [ dades_2306_OD_132_5_01,...
                      dades_2306_OD_132_0_005, dades_2306_OD_132_5_005,...
                      ];

  spMODEDat.DataGFP_per_cell = [ dades_2306_FA_132_5_01-
dades_2306_FA_132_0_01, dades_2306_FA_132_0_005-
dades_2306_FA_132_0_01,...
                      dades_2306_FA_132_5_005-dades_2306_FA_132_01_0,
dades_2306_FA_132_01_10-dades_2306_FA_132_01_0,...
                      dades_2306_FA_132_01_20-
dades_2306_FA_132_01_0,dades_2306_FA_132_01_35-dades_2306_FA_132_01_0];

spMODEDat.DataGFP_per_cell = spMODEDat.DataGFP_per_cell(9:end,:);

end
```

3 ANEXO III

3.1 Función CostFunction_adaptation_GFP2

```
function  [J,X] = CostFunction_adaptation_GFP2(X,Dat)

%% Cost function
%global XRP
% Load data from spMODEparam
OD = Dat.DataOD;
GFP = Dat.DataGFP_per_cell;

% Number of copies per experiment
Copies = Dat.Copies;


Kmax_exp = 1.128; % max(max(OD,[],1)); %Modificado para ser igual a Y
param=set_parameters3(X,Kmax_exp);

% Simulation and experimental parameters
Tspan = Dat.DataTime;     % Each 10 minutes aprox
options = odeset('AbsTol',1e-5,'RelTol',1e-3);

OD_0=0.2;

J = zeros(size(X,1),Dat.NOBJ);

for j = 1:length(Dat.INPUTS)

    InitC=set_initial3(param,OD_0,Dat.INPUTS(j));

    parfor xpop=1:size(X,1)

        k=1+(j-1)*Copies;

        %Model
        [T,Y] = ode15s(@(t,y)
model_adaptation_GFP3(y,xpop,param),Tspan,InitC(:,xpop),options);

        %Output->GFP
        n1 = (Y(:,5)*ones(1,Copies)).*param.k_gain(xpop);   % GFP activa

        %Relative error (Yreal - Yestimated)/Yreal
        error_GFP = (( GFP(:,k:(k+Copies-1)) - n1 )).^2; % Square  error

        J(xpop,j)=mean(mean(error_GFP,1),2)/(1e6*Dat.INPUTS(j));
    end
end

if isempty(Dat.Pertinency)
    %Nothing happens
else
    for xpop=1:size(J,1)
        PenUpper=0;
        PenLower=0;
        for nobj=1:size(J,2)
            PenUpper=PenUpper+...
```

```
                max((J(xpop,nobj)-Dat.Pertinency(nobj,2)),0)/...
                Dat.Pertinency(nobj,2);
            PenLower=PenLower+...
                max((Dat.Pertinency(nobj,1)-J(xpop,nobj)),0)/...
                Dat.Pertinency(nobj,2);
        end

        if PenUpper+PenLower>0
            Penalty=max(Dat.Pertinency(:,2))+PenUpper+PenLower;
            J(xpop,:)=Penalty*ones(1,size(J,2));
        end
    end
end
```

4 ANEXO IV

4.1 Función Validation_Adaptation_GFP2av

```
function Validation_Adaptation_GFP2av(OUT,popj)
spMODEDat = OUT.Param;

Copies = spMODEDat.Copies;
% Input x9 -> AHLe [nM]
Input_ARRAY = spMODEDat.INPUTS;   % [1 2 5 10 20 35] nM
Longitud= length(Input_ARRAY);
i=0;

%%

Linewidth = 2;
Fontsize = 14;
figure1 = figure('Color',[1 1 1]);


% Color line
C_orange = [217,95,2]./255;
C_green = [55,126,184]./255;
C_purple = [117,112,179]./255;

%%
for Input_selector = 1:length(Input_ARRAY)

    Input = Input_ARRAY(Input_selector);     % select one input, i.e.
Input_selector = 3 -> 5nM
    i=i+1;

    %%
    % Load data for validation process
    OD = spMODEDat.DataOD;
    GFP = spMODEDat.DataGFP_per_cell;

    % Number of copies per experiment
    Copies = spMODEDat.Copies;

    % Absorbance from experimental data
    OD_0 = min(min(OD,[],1));         % Initial OD_600   (%0.2 fosbe)
    Kmax_exp = max(max(OD,[],1));           % Max OD_600 (1.128 fosbe)


    % Time from experiment
    time  = spMODEDat.DataTime;

    X = OUT.PSet(popj,:);

    % General parameters
     param=set_parameters3(X,Kmax_exp);

    % Initial state for simulation

     InitC=set_initial3(param,OD_0,spMODEDat.INPUTS(Input_selector));
```

```
    %Model
    Tspan = time;     % Each 3 minutes aprox
    options = odeset('AbsTol',1e-8,'RelTol',1e-6);

    Tsim_nuevo = 0:0.1:Tspan(end);
     [T,XS] = ode15s(@(t,y)
model_adaptation_GFP3(y,1,param),Tsim_nuevo,InitC,options);

    %Outputs
     GFP_sim=XS(:,5)*param.k_gain;     % GFP.*kgain


    %% Plots
    % Experimental data for each Input
     k=1+(Input_selector-1)*Copies;
    x8_exp = mean(GFP(:,k:(k+Copies-1)),2);

    Tsim = T(end);

titulo=sprintf('Solución %d', popj);
subplot(Longitud,1,i,'Parent',figure1);
%,'XTickLabel',{'','0','20','40','60','80','100','120','140','160','180'}
,...
hold on;
title(['$\textbf{AHL}_\textbf{ext}= $' num2str(Input)
'nM'],'FontSize',16,'Interpreter','latex');
        %'XTick',[0 10 20 40 60 80 100 120 140 160 180]);
hold on;
plot(Tsim_nuevo, GFP_sim,'Color',C_green,'LineWidth',Linewidth);
hold on;
suptitle(titulo);

plot(time,x8_exp,'Color',C_orange,'MarkerSize',6,...
    'MarkerFaceColor',C_orange,...
    'MarkerEdgeColor',[0 0 0],...
    'Marker','o',...
    'LineStyle','none');

xlabel('$\mathrm{Time}$
[min]','FontSize',Fontsize,'Interpreter','latex');
ylabel(['$\mathrm{GFP_a
}$',sprintf('\n'),'[RFU]'],'FontSize',12,'Interpreter','latex');
xlim([0 Tsim]);



end
```

5 ANEXO V

5.1 Código de la interfaz gráfica general

```
function varargout = PruebaGUI(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PruebaGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @PruebaGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PruebaGUI is made visible.
function PruebaGUI_OpeningFcn(hObject, eventdata, handles, varargin)


% Choose default command line output for PruebaGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);


% --- Outputs from this function are returned to the command line.
function varargout = PruebaGUI_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
elejido=contents{get(handles.listbox1,'Value')};
xpop=str2double(elejido);
Validation_Adaptation_GFP2av(handles.OUT,xpop);
guidata(hObject,handles);


% --- Executes on selection change in listbox1.
function listbox1_Callback(hObject, eventdata, handles)
% hObject    handle to listbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes during object creation, after setting all properties.
function listbox1_CreateFcn(hObject, eventdata, handles)


if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in Read.
function Read_Callback(hObject, eventdata, handles)

handles.evol=readExperiment;
guidata(hObject,handles);
disp('Leído');


% --- Executes on button press in Param.
function Param_Callback(hObject, eventdata, handles)
handles.spMODEDat=spMODEparam2(handles.evol);
guidata(hObject,handles);
disp('Hecho');


% --- Executes on button press in Sol.
function Sol_Callback(hObject, eventdata, handles)
handles.OUT=spMODE2(handles.spMODEDat);
a=length(handles.OUT.PSet);
vector=(1:a);
datos=int2str(vector');
celdas=cellstr(datos);
set(handles.listbox1,'string',celdas);
guidata(hObject,handles);


% --- Executes on button press in Vis.
function Vis_Callback(hObject, eventdata, handles)
plotExperiment(handles.evol);
guidata(hObject,handles);
```