UNIVERSIDAD
COMPLUTENSE
MADRID

Proyecto de Innovación
Convocatoria 2018/2019

Nº de proyecto: 60

Título del proyecto: Elaboración de una metodología learn to program/ program to learn para la enseñanza en el área de la Ingeniería Química empleando la herramienta Matlab Cody Coursework para fomentar el e-learning

Nombre del responsable del proyecto: Mª Isabel Guijarro Gil

Centros: Facultad de Ciencias Químicas

Departamentos: Departamento de Ingeniería Química y de Materiales

1. Objetivos propuestos en la presentación del proyecto

El objetivo general de este proyecto es la elaboración de recursos docentes tanto para estudiantes como para profesores utilizando Matlab Grader (antes, Cody Coursework). Respecto a los estudiantes, ayudándoles a pensar de forma global sin compartimentar en función de la asignatura que estudie. Para ello, aplicaran las habilidades adquiridas, en la asignatura Informática Aplicada (IA), de programación y cálculo ingenieril en el aprendizaje de la parte práctica de la asignatura de Fundamentos de Ingeniería Química (FIQ). Para consecución de este objetivo principal se plantean los siguientes objetivos específicos:

1. Desarrollar los distintos recursos docentes:
   1.1. Crear una colección de problemas tipo que sirvan de base para que el estudiante pueda elaborar sus propios generadores de problemas para la asignatura de FIQ.
   1.2. Crear problemas de secuencias de comandos para la asignatura de IA. Se pretende que el estudiante resuelva problemas de FIQ, que en muchos casos requieren la utilización de métodos numéricos con Matlab. Los métodos de cálculo desarrollados permitirán modificar las variables del proceso y ver su influencia en el mismo. Al tratarse de estudiantes de 1er curso será necesario elaborar los enunciados e instrucciones muy detallados de cómo resolver el problema.
   1.3. Desarrollar pruebas de evaluación de cada problema para la puesta en marcha del sistema de autocorrección.
2. Determinar las posibilidades docentes de la herramienta Matlab Grader dentro de las dos asignaturas elegidas y priorizar, en función de la viabilidad del proyecto, aquellas habilidades que se pretende reforzar con los recursos docentes elaborados.
3. Preparar un procedimiento de implantación de los materiales desarrollados para su posterior utilización.
4. Tras el uso de los recursos propuestos, realizar un plan de seguimiento, a través de la descarga de un mapa de progreso de cada estudiante que proporciona la aplicación, es decir, información sobre el número de aciertos en función de los criterios fijados y la “longitud” del código que han empleado.
5. Realizar una encuesta al final del curso en las dos asignaturas que permita conocer cuál ha sido el grado de utilización y la apreciación personal de cada estudiante en cuanto a la utilidad del material.
6. Difundir los resultados obtenidos a través de congresos y publicaciones docentes generales y específicas del área de Ingeniería Química.

2. Objetivos alcanzados

A continuación, se analiza el grado en que se han alcanzado los objetivos planteados en la solicitud de este Proyecto de Innovación.

Objetivo 1. Dentro de este objetivo se han alcanzado los siguientes logros a lo largo de la duración del proyecto, desglosados aquí en función de los objetivos específicos:

- Objetivo 1.1) Se ha seleccionado una muestra de la colección de problemas dentro de los problemas tipo de la asignatura, de forma que puedan servir de base para el desarrollo de los generadores de problemas. En particular, a lo largo de este proyecto, se han elegido problemas de balance de materia, cinética química y reología.
- Objetivo 1.2) Se han estructurado los generadores de problemas de forma que se han desglosado en un grupo de pequeñas funciones programadas en Matlab. Estas funciones les serán propuestas a los estudiantes como tareas en un orden creciente de dificultad, eliminando de las funciones de referencia el código que deben programar e incluyendo, en forma de comentarios, los distintos pasos a seguir hasta completarlas exitosamente. Una vez los estudiantes dispongan de todas estas funciones puedan integrarlas, de forma sencilla, en un único programa: el generador de problemas.
- Objetivo 1.3) Tras el desarrollo de las funciones de referencia y de las plantillas que han de completar los estudiantes, ambas necesarias en el autoevaluador Matlab Grader, se han diseñado las pruebas que han de superar las funciones programadas por los estudiantes para que el autoevaluador les indique que su función cumple con todos los requisitos para operar correctamente. El diseño de las pruebas incluye las comprobaciones a realizar y el código que se ha de ejecutar para llevar a cabo dichas comprobaciones de forma automática.

Objetivo 2. Se han determinado el número de generadores que los estudiantes pueden realizar a lo largo de la asignatura cuatrimestral IA, en función del número de funciones requeridas para su ejecución – cada función es una tarea de programación para los estudiantes y requiere un tiempo planificación, resolución y evaluación -. En este sentido, se ha decidido desarrollar 3 generadores, ya esto supone la programación de al menos 10-15 funciones que, posteriormente, hay que integrar para dar lugar a la herramienta final.

Objetivo 3. El procedimiento de implantación que se ha llevado a cabo durante el proyecto, por orden de ejecución, ha sido: se han generado en la herramienta Matlab Grader las tareas necesarias para la implantación del proyecto; de igual modo, se ha establecido la organización por grupos necesaria, incluyendo un profesor encargado de tutorizar los grupos y de instruirlos si el código requerido para la programación de las funciones y los generadores les resulta excesivamente complicado; finalmente, se han dispuesto de presentaciones específicas más allá de las empleadas en las asignaturas de IA y FIQ.

Objetivo 4 y 5. Estos objetivos son los menos desarrollados a lo largo del proyecto. La asignatura de informática aplicada es de primer cuatrimestre por lo que no se ha podido poner en práctica en el curso 2018/2019, y es el próximo curso 2019/2020 donde todo el material se empleará en la implementación del proyecto y hacer uso de las herramientas de seguimiento a través de Matlab Grader. No obstante, sí que se han desarrollado los cuestionarios de evaluación que se emplearán en la evaluación del grado de utilización de las herramientas desarrolladas y para conocer la apreciación personal de cada estudiante en cuanto al interés y utilidad del proyecto y de los materiales elaborados.

Objetivo 6. El último objetivo, la difusión de los resultados obtenidos a través de congresos y publicaciones científicas se ha llevado a cabo a través de la comunicación “Empleo de una metodología learn to *program/program to learn* en Ingeniería Química utilizando Matlab y Matlab Grader”, enviada al V Congreso de Innovación Docente en Ingeniería Química (22-24 de enero de 2020), solicitando su inclusión como comunicación oral. Transcurrido el congreso, se considerará la publicación de una selección de los trabajos presentados para su publicación en dos revistas internacionales de referencia en el ámbito de la educación en Ingeniería Química y Ambiental.

3. Metodología empleada en el proyecto

La metodología empleada en el proyecto ha sido la que se propuso en la solicitud. El grupo encargado del proyecto se ha dividido en equipos de trabajo coordinados de tal manera que cada uno de ellos ha tenido unas tareas asignadas para la consecución de todos los objetivos del proyecto.

Tarea 1 Planteamiento de las líneas de actuación prioritarias. Se han formado dos grupos de trabajo. El primero se encargó de revisar si los problemas generados a lo largo del proyecto anterior tenían una dificultad acorde con el nivel de los estudiantes de la asignatura de IA. El segundo grupo de trabajo realizó la selección de nuevos problemas en concordancia con la asignatura de IA

Tarea 2 La elaboración del material docente. El grupo se dividirá en tres equipos para el desarrollo de los nuevos materiales en función de los resultados de la tarea 1. Estos equipos llevaron a cabo una puesta en común de la colección de problemas posibles, seleccionando y priorizando aquellos que estaban en concordancia con el objetivo 1.

Tarea 3 Integración y evaluación del material elaborado. En esta tarea se ha evaluado el material en su conjunto, para ello ha sido necesaria la recopilación estructurada de todo el material desarrollado. Esto nos ha permitido mejorar y corregir los fallos encontrados. La implementación se realizará durante el primer cuatrimestre en la asignatura de IA y se seguirá mejorando durante el tiempo que las herramientas desarrolladas se sigan utilizando.

Tarea 4 Elaboración de un plan de implantación de la metodología. Cada grupo de trabajo llevó a cabo los generadores de problemas tipo decididos en la tarea 2, empleando Matlab Grader.

Tarea 5. Diseño de un plan de seguimiento de los resultados obtenidos y futuras actuaciones. Esta tarea, asociada a los objetivos 4 y 5 aún requiere de un mayor desarrollo que se pondrá en práctica en el curso 2019/2020 y en los siguientes cursos.

Tarea 6 Elaboración del informe del proyecto.

4. Recursos humanos

Como se indicó en la solicitud, el grupo innovador está compuesto por profesores del Departamento de Ingeniería Química y de Materiales y dos investigadoras predoctorales colaboradoras en tareas docentes del mismo Departamento. Este grupo innovador ha sido el encargado de la realización del proyecto de Innovación.

La recopilación de materiales y la puesta en común de la información obtenida ha sido llevada a cabo por todos los miembros del Grupo. El resto de las tareas se ha distribuido entre los miembros del grupo, los cuales han trabajado en grupos. Las formaciones de los equipos de trabajo se han realizado atendiendo al perfil de los miembros del proyecto: su experiencia en la docencia de las asignaturas para las cuales se desarrollan los materiales docentes y conocimientos del software. Ha sido de gran valor la reciente experiencia, como estudiantes, de los colaboradores docentes.

5. Desarrollo de las actividades

Las actividades se han desarrollado, hasta el punto de desarrollo alcanzable durante el curso académico 2018/2019, de acuerdo con el cronograma indicado en la solicitud de este proyecto de innovación. Los resultados obtenidos a lo largo de estas tareas se han incluido en esta memoria como anexos.

Tarea 1. Planteamiento de las líneas de actuación prioritarias. Se identificaron, basándose en la experiencia de los profesores de las asignaturas y en la experiencia acumulada en la programación de generadores de problemas en lenguaje Python durante el proyecto de innovación nº 11 de la convocatoria 2017/2018, del que se extrajeron los conceptos que presentaban una mayor dificultad. Dada las similitudes entre la programación entre Matlab y Python, en especial en el uso de los paquetes Matplotlib, Numby y Scipy para la realización de cálculos y la representación de gráficos, se concluyó que el nivel de dificultad era similar entre los generadores programados en ambos lenguajes. De este modo, se optó finalmente por reforzar los conocimientos básicos de los estudiantes en la realización de balances de materia, cinética química y reología. Una vez seleccionados, los miembros del proyecto se centraron en estructurar el trabajo a realizar, desglosando las tareas y operaciones de cada generador de problemas en funciones, que serán las que constituyan las tareas de programación de los estudiantes.

Tarea 2. Elaboración del material docente En esta tarea se llevaron a cabo las siguientes subáreas.

Tarea 2.1. Se definieron los problemas y las estrategias de resolución de los mismos mediante funciones.

Tarea 2.2. Se programaron las funciones de referencia en Matlab y se desarrollaron, a partir de éstas, las plantillas que se les facilitan como punto de partida a los estudiantes.

Tarea 2.3. Se seleccionaron los criterios de evaluación del código y se programaron los test de ejecución correspondientes a cada función.

Tarea 3. Integración y evaluación del material elaborado. Se recopiló y distribuyó el material realizado entre todos los miembros. A su vez, una vez programadas las tareas en Matlab Grader, se incluyeron como estudiantes a distintos miembros del proyecto para que pudieran analizar y evaluar el funcionamiento de Matlab Grader y corregir problemas de programación, así como detectar la necesidad de llevar a cabo más comprobaciones del código mediante test nuevos añadidos al Grader.

Tarea 4. Elaboración de un plan de implantación de la metodología. En la actualidad el plan de implantación está diseñado, quedando pendiente su implementación en sí mismo.

Tarea 5. Diseño de un plan para el seguimiento de los resultados obtenidos y futuras actuaciones. Se ha diseñado un plan de seguimiento de los resultados, tanto en las tareas propias del proyecto como en cuanto las mejoras de los resultados académicos en las dos asignaturas objetivo de este proyecto. Además, se han elaborado unos cuestionarios que permitan conocer el grado de satisfacción de los estudiantes y conocer cuáles son las fortalezas y debilidades del proyecto. Del mismo modo, a través de cuestionarios, se pretenden conocer las mejoras propuestas por los estudiantes y los docentes. La metodología será dinámica, buscando no sólo una mejora en las tasas de éxito, sino también un incremento de las capacidades y competencias adquiridas por los estudiantes.

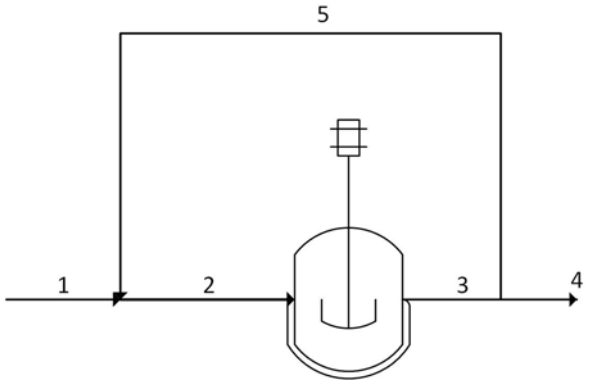
Tarea 6. Elaboración del informe final del proyecto. Finalmente, en base al desarrollo de las tareas anteriores llevadas a cabo durante el proyecto, se ha elaborado el informe final del mismo.

6. Anexos

6.1. Balances de materia.

6.1.1. Enunciado del problema.

Determine los caudales másicos y molares y las composiciones (másicas y molares) de las corrientes de un proceso en el cual tiene lugar una reacción química y hay una corriente de recirculación, tal y como se muestra el siguiente diagrama de flujo:



La reacción química que tiene lugar en el reactor es (*coeficientes variables cada vez que se ejecuta el programa*):

$$a \cdot A + b \cdot B + c \cdot C \rightarrow d \cdot D + e \cdot E + f \cdot F$$

Y variables conocidas son (*grupos de variables conocidos seleccionados al azar cada vez que se ejecuta el programa: casos*).

6.1.2. Generador de problemas (parámetros al azar y soluciones para ese caso).

```
clc
clearvars -global
%Variable del tipo estructura en donde se guardarán todos los parámetros %y variables del
problema
Resultados=struct();

%Generación de los coeficientes estequiométricos de la reacción. Al menos
%un coeficiente distinto de 0 para reactivos y productos.
% Reacción: aA+bB+cC -> dD+eE+fF

[CoefReactivos,CoefProductos]=GeneradorReaccionQuimica();
Resultados.CoefReactivos=CoefReactivos;
Resultados.CoefProductos=CoefProductos;

%Generador de composiciones y corrientes en la corriente alimento del
%reactor
[F2,Xi,TiposCorriente2,X2] =...

GeneradorCorriente(length(nonzeros(CoefReactivos)),randi([0,length(nonzeros(CoefProductos))]))
;
```

```
%Generación de pesos moleculares para los reaccionantes; se debe cumplir el
%balance de materia para la reacción
PesosMoleculares=Generador_PM_Reaccion(CoefReactivos,CoefProductos);
Resultados.PesosMoleculares=PesosMoleculares;

%Determinación del reactivo limitante para la reacción y los PMs de los
%reactivos generados al azar
ReactivoLimitante = Reactivo_limitante(Xi(1:length(CoefReactivos)),...
    CoefReactivos);
%Generación de la conversión del reactivo A (siempre presente) en función
%de la conversión máxima posible (dependiendo de cuál sea el reactivo
%limitante)
X_A = Generador_ConversionA(ReactivoLimitante,Xi(1:length(CoefReactivos)),...
        CoefReactivos);
Resultados.ReactivoLimitante=ReactivoLimitante;
Resultados.ConversionA=X_A;

%Generador al azar de la razón de recirculación al sistema
R= Generador_RazonRecirculacion();
Resultados.RazonRecirculacion=R;

%Opciones en la resolución de los balances - Función lsqnonlin
%trust-region-reflective; levenberg-marquardt
options=optimset('Display','off','MaxIter',100000,...
  'MaxFunEvals',100000,'TolFun',1E-8,'Algorithm','trust-region-reflective');

%Parámetros de la función de resolución: valores máximos y mínimos de las
%incógnitas (composiciones y caudales molares); valores iniciales de las
%incógnitas y valores de las variables conocidas (caudal y composición del
%alimento al reactor, conversión y razón de recirculación).

lb=zeros(1,17)+1E-6;
lb(1)=1E-6;lb(9)=1E-6;lb(17)=1E-6;
ub=zeros(1,17)+1;
ub(1)=1000;ub(9)=1000;ub(17)=1000;
yini=zeros(1,17)+0.2;
yini(1)=100;yini(17)=100;yini(9)=100;
V_Conocidas=[F2,X2(1:7),R];

%Resolución de los balances. El sistema de ecuaciones viene definido en la función
SistemaEcuaciones
C=lsqnonlin(@SistemaEcuaciones,yini,lb,ub, options,...
   V_Conocidas,CoefReactivos,CoefProductos,X_A);

%Asignación de valores
Incognitas=num2cell(C);
[F3,X3(1),X3(2),X3(3),X3(4),X3(5),X3(6),X3(7),...
 F1,X1(1),X1(2),X1(3),X1(4),X1(5),X1(6),X1(7),F4]=Incognitas{:};

X3(8)=(1-sum(X3));X3=round(X3,3);
X1(8)=(1-sum(X1));X1=round(X1,3);

F3=round(F3);
F1=round(F1);
F4=round(F4);
F5=round(F3*R);
```

```
%Conversión de los resultados para la obtención de los caudales y fracciones másicas
X5 = X3; X4 = X5;
for i=1:5
  s1 = 'F';x1='X';
  s2 = num2str(i);
  F= strcat(s1,s2);
  X=strcat(x1,s2);
  Resultados.Caudales.CaudalesMolares.(F)=eval(F);
  Resultados.Caudales.CaudalesMasicos.(F)=Vector_FtoM(eval(F),eval(X),PesosMoleculares);
  Resultados.Fracciones.FraccionesMolares.(X)=eval(X);
  Resultados.Fracciones.FraccionesMasicas.(X)=Vector_XtoW(eval(X),PesosMoleculares);
end

clearvars -except Resultados
```

6.1.3. Funciones empleadas en el generador que han de desarrollar los estudiantes y comprobar usando Matlab Grader.

- Función *GeneradorReaccionQuimica*:
-

```
function [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(varargin)

% Esta función genera al azar, si no se especifican como parámetros:
%  - Número de reactivos de la reacción (mín. 1 y máximo 3).
%  - Número de productos de la reacción (mín. 1 y máximo 3).
%  - Coeficiente estequiométrico de cada reactivo.

%  si se especifica un número mayor de 3 reactivos o productos la función
%  devuelve el máximo de 3.


numArgumentos = nargin;
if numArgumentos > 2
    error('La funciÛn tiene más parámetros de los permitidos: N_Reactivos y N_Productos');
end

% Se definen los valores por defecto de los parámetros
optargs = {randi([1,3]) randi([1,3])};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Cambia todos los valores especificados mayores que 3 por el valor máximo
 a=cell2mat(optargs);
 a(a>3)=3;
 a(a<1)=1;
 optargs=num2cell(a);
% Asigna los valores especificados (o por defecto) a las variables
[N_Reactivos, N_Productos] = optargs{:};

CoefReactivos=zeros(1,3);
CoefProductos=zeros(1,3);
```

```
for i=1:N_Reactivos
  CoefReactivos(i)=Generador_CoefEstq(1,3);
end
for i=1:N_Productos
  CoefProductos(i)=Generador_CoefEstq(1,3);
end
```

- Test del Grader para la función *GeneradorReaccionQuimica*.

La descripción del problema que los estudiantes ven en el evaluador es:

“Se ha de programar una función que, empleando la función ya programada para generar coeficientes estequiométricos aleatorios, dé lugar a una “reacción química” con como máximo 3 reactivos y 3 productos (y mínimo un reactivo para dar un único producto).

La función a de retornar dos vectores de coeficientes estequiómetricos, uno para reactivos y otro para productos, con 3 elementos por vector, todos ellos números enteros comprendidos entre 0 y 3 y siendo el primer elemento de cada vector distinto de 0 necesariamente.

La función podrá tener como argumentos el número reactivos y el de productos (siempre entre 1 y 3) o generar estos valores aleatoriamente si se emplea sin argumentos.

Y la plantilla de partida sobre la que incluir el código de la función es:

```
function [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(varargin)

   % Esta función genera al azar, si no se especifican como parámetros:
   %  - Número de reactivos de la reacción (mín 1 y máximo 3).
   %  - Número de productos de la reacción (mín 1 y máximo 3).
   %  - Coeficiente estequiométrico de cada reactivo.

   %  si se especifica un número mayor de 3 reactivos o productos la función
   %  devuelve el máximo de 3.


   numArgumentos = nargin;
   if numArgumentos > 2
      error('La función tiene más parámetros de los permitidos: N_Reactivos y N_Productos');
   end

   % Se definen los valores por defecto de los parámetros
   optargs = {randi([1,3]) randi([1,3])};

   % Copia de los valores por defecto
   optargs(1:numArgumentos) = varargin;
   % Cambia todos los valores especificados mayores que 3 por el valor máximo
    a=cell2mat(optargs);
    a(a>3)=3;
    a(a<1)=1;
    optargs=num2cell(a);
```

```
    % Asigna los valores especificados (o por defecto) a las variables
    [N_Reactivos, N_Productos] = optargs{:};

    % Inicia las variables CoefReactivos y CoefProductos como vectores de ceros.


    % Genera coeficientes para tantos reactivos como se hayan definido o se hayan generado
aleatoriamente


    % Genera coeficientes para tantos productos como se hayan definido o se hayan generado
aleatoriamente

end
```

Los test de evaluación de la función son los siguientes:

- *Comprueba que todos los vectores resultantes tienen 3 elementos:*

```
% Genera valores al azar de las variables de partida por defecto o no
for i=1:20

    PorDefecto=randi([0,1]);

    if PorDefecto==1

        [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica();
        vector1=length(CoefReactivos);
        vector2=length(CoefProductos);

    else
        N_Reactivos=randi([1,5]);
        N_Productos=randi([1,5]);
        [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(N_Reactivos,N_Productos);
        vector1=length(CoefReactivos);
        vector2=length(CoefProductos);
    end

    %Comprueba que el valor obtenido es un entero
    resultado(i)=(vector1==3 && vector2==3);

end

comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“Al menos uno de los vectores resultantes de la función tiene más de 3 elementos. Recuerda que no deben superar ese tamaño incluso si así se especifica en los argumentos de la función”.

- *Comprueba que los elementos de los vectores son todos enteros:*

```
% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
   PorDefecto=randi([0,1]);

   if PorDefecto==1
      % Ejecuta la función del estudiante y obtiene los resultados: y
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica();
   else
      N_Reactivos=randi([1,5]);
      N_Productos=randi([1,5]);
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(N_Reactivos,N_Productos);
   end


   %Comprueba el resultado
   resultado(i)=(all(floor(CoefReactivos) ==CoefReactivos)&& all(floor(CoefProductos)
==CoefProductos));
end


comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

"Al menos uno de los vectores resultantes contiene elementos que no son números enteros".

- *Comprueba que todos los coeficientes estequiométricos están entre 0 y 3:*

```
% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
   PorDefecto=randi([0,1]);

   if PorDefecto==1
      % Ejecuta la función del estudiante y obtiene los resultados: y
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica();
   else
      N_Reactivos=randi([1,5]);
      N_Productos=randi([1,5]);
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(N_Reactivos,N_Productos);
   end


   %Comprueba el resultado
   r1=isempty(CoefReactivos(CoefReactivos<0 & CoefReactivos>3));
   r2=isempty(CoefProductos(CoefProductos<0 & CoefProductos>3));

   resultado(i)=(r1 && r2);
end
```

```
comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante

assessVariableEqual('comprobacion', true);
```

- *Comprueba que el primer elemento de cada vector es distinto de 0:*

```
% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
   PorDefecto=randi([0,1]);

   if PorDefecto==1
      % Ejecuta la función del estudiante y obtiene los resultados: y
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica();
   else
      N_Reactivos=randi([1,5]);
      N_Productos=randi([1,5]);
      [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(N_Reactivos,N_Productos);
   end


   %Comprueba el resultado

   resultado(i)=(CoefReactivos(1)~=0 && CoefProductos(1)~=0);
end

comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“El primer elemento de uno de los dos vectores es 0”.

- Función *Generador_CoefEstq*:

```
function Coeficiciente= Generador_CoefEstq(varargin)
% Esta función genera un valor entero al azar comprendido entre 0 y 3

% Se pueden introducir un máximo de 2 parámetros de entrada: CoefMin y
% CoefMax

numArgumentos = length(varargin);
if numArgumentos > 2
   error('La funciÛn tiene m·s par·metros de los permitidos: CoefMin y CoefMax');
end

% Se definen los valores por defecto de los parámetros
```

```
optargs = {0 3};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;

[CoefMin, CoefMax] = optargs{:};

Coeficiciente=randi([CoefMin,CoefMax]);

 end
```

- Test del Grader para la función *Generador_CoefEstq.*

La descripción del problema que los estudiantes ven en el evaluador es:

“Se debe programar una función que genere un coeficiente estequiométrico, es decir un valor entero, al azar. El intervalo por defecto para estos coeficientes será de entre 0 (el reactivo o producto no interviene en la reacción) y 3. La función debe permitir incluir como argumentos opcionales el valor máximo y mínimo de los coeficientes estequiométricos (CoefMin y CoefMax).”

Y la plantilla de partida sobre la que incluir el código de la función es:

```
function Coeficiciente= Generador_CoefEstq(varargin)
    % Esta función genera un valor entero al azar comprendido entre 0 y 3

    % Se pueden introducir un máximo de 2 parámetros de entrada: CoefMin y
    % CoefMax

    numArgumentos = length(varargin);
    if numArgumentos > 2
       error('La función tiene más parámetros de los permitidos: CoefMin y CoefMax');
    end

    % Se definen los valores por defecto de los parámetros
    optargs = {0 3};

    % Copia de los valores por defecto
    optargs(1:numArgumentos) = varargin;
    % Se guardan los valores máximo y mínimo en las variables con el nombre especificado
[CoefMin, CoefMax]
    % y se genera el valor del coeficiente estequiométrico al azar (valor entero y probabilidad
uniforme).
    % Se ha de comprobar que el mayor valor del coeficiente es, efectivamente, superior al
mínimo posible y
    % que el mínimo valor posible generado es mayor o igual a 0.

end
```

Los test de evaluación de la función son los siguientes:

- *Comprueba que el número generado es un entero:*

```
% Genera valores al azar de las variables de partida por defecto o no

PorDefecto=randi([0,1]);

if PorDefecto==1
   % Ejecuta la función del estudiante y obtiene los resultados: y
   y = Generador_CoefEstq();
else
   CoefMax=randi([1,5]);
   incremento=randi([1,5]);
   CoefMin=CoefMax-incremento;
   y = Generador_CoefEstq(CoefMin,CoefMax);
end


%Comprueba que el valor obtenido es un entero
comprobacion=floor(y) ==y ;

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“El coeficiente generado no es un número entero”.

- *Comprueba que el coeficiente es mayor o igual que el mínimo y menor o igual que el máximo:*

```
%% Genera valores al azar de las variables de partida por defecto o no

PorDefecto=randi([0,1]);

if PorDefecto==1
   % Ejecuta la función del estudiante y obtiene los resultados: y
   y = Generador_CoefEstq();


   %Comprueba que el valor obtenido está en el intervalo prefijado
   comprobacion=(y<=3 && y>=0) ;
else
   CoefMax=randi([1,5]);
   incremento=randi([1,5]);
   CoefMin=CoefMax-incremento;
   y = Generador_CoefEstq(CoefMin,CoefMax);

   %Comprueba que el valor obtenido esta en el intervalo requerido
   comprobacion=(y<=CoefMax && y>=CoefMin) ;
end



% Compara los valores de referencia con los obtenidos mediante la función del estudiante

assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“El coeficiente generado no está dentro del intervalo de valores definidos”.

- Función *GeneradorCorriente*:

```
function [F,Xi,Tipos,X2] = GeneradorCorriente(varargin)

% Esta función genera al azar, si no se especifican como parámetros:
%  - El número de reactivos y productos presentes en la corriente (define
%  la reacción parcialmente, el número de productos final podría cambiar al
%  generar la reacción. Ojo en la implementación conjunta de esta función
%  y la función generadora de la reacción!
%
%  - El número de inertes
%  - El caudal molar (el generador trabaja con caudales molares, a la hora
%  de especificar el problema puede hacerse en masa empleando las
%  funciones de conversión previamente diseñadas)
%
%  - Las fracciones molares de cada uno de los productos

numArgumentos = nargin;
if numArgumentos > 3
    error('La funciÛn tiene más parámetros de los permitidos: Reactivos, Productos e Inertes');
end

% Se definen los valores por defecto de los parámetros
optargs = {randi([1,3]) randi([0,3]) randi([0,2])};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Sustituye los valores por defecto por los introducidos, si es el caso
[Reactivos, Productos, Inertes] = optargs{:};

%Genera un valor para el caudal molar de la corriente (mol)
F=round(rand*randi([0,200])+50);

%Genera las fracciones molares de los compuestos en la corriente
numeroCompuestosTotal=sum([optargs{:}]);
Xi=rand(1,numeroCompuestosTotal);
suma=sum(Xi);
Xi=Xi./suma;
Tipos=[Reactivos, Productos, Inertes];


X2(1:Reactivos)=Xi(1:Reactivos);
if Reactivos<3
    X2((Reactivos+1):3)=0;
end
X2(4:(Productos+3))=Xi((Reactivos+1):(Reactivos+Productos));
if Productos<3
    X2((4+Productos):6)=0;
end
```

```
X2(7:(6+Inertes))=Xi((Productos+Reactivos+1):end);
if Inertes<2
   X2((7+Inertes):8)=0;

end
Xi=round(Xi,3);
X2=round(X2,3);
end
```

- Función *Generador_PM_Reaccion*:

```
function PesosMoleculares= Generador_PM_Reaccion(CoefReactivos,CoefProductos)
PesosMoleculares=zeros(1,8);
for i=1:3
   PesosMoleculares(i)=Generador_PM();
end
MasaReactivos=sum(CoefReactivos.*PesosMoleculares(1:3));

if CoefProductos(2)==0
   PesosMoleculares(4)=MasaReactivos/CoefProductos(1);

elseif CoefProductos(3)==0
   PesoMaximo_D=0.85*MasaReactivos/CoefProductos(1);
   PesosMoleculares(4)=Generador_PM(10,PesoMaximo_D);
   MasaRestanteProductos=MasaReactivos-CoefProductos(1)*PesosMoleculares(4);
   PesosMoleculares(5)=(MasaRestanteProductos)/CoefProductos(2);

else
   PesoMaximo_D=0.65*MasaReactivos/CoefProductos(1);
   PesosMoleculares(4)=Generador_PM(10,PesoMaximo_D);
   MasaRestanteProductos=MasaReactivos-CoefProductos(1)*PesosMoleculares(4);
   PesoMaximo_E=0.85*(MasaRestanteProductos)/CoefProductos(2);
   PesosMoleculares(5)=Generador_PM((0.5* PesoMaximo_E),PesoMaximo_E);
   MasaRestanteProductos=MasaReactivos-sum(CoefProductos(1:2).*PesosMoleculares(4:5));
   PesosMoleculares(6)=(MasaRestanteProductos)/CoefProductos(3);
end

for i=7:8
   PesosMoleculares(i)=Generador_PM();
end
%MasaProductos=sum(CoefProductos.*PesosMoleculares(4:6));
end
```

- Test del Grader para la función *Generador_PM_Reaccion*.

La descripción del problema que los estudiantes ven en el evaluador es:

“Se ha de programar una función que, empleando la función ya programada para generar pesos moleculares aleatorios, dé lugar a pesos moleculares para los reactivos y productos involucrados en una reacción química (¡cumpliendo con el balance de materia de la reacción!) y a los correspondiente a dos compuestos inertes; de este modo, se obtendrá como retorno de la función un vector con 8 elementos.”

Y la plantilla de partida sobre la que incluir el código de la función es

```
function PesosMoleculares= Generador_PM_Reaccion(CoefReactivos,CoefProductos)

%  Inicialice en un vector de 8 elementos


% Genere pesos moleculares para los 3 reactivos

% Evalúe la masa total de reactivos involucrada en la reacción teniendo en cuenta
% los coeficientes estequiométricos de los reactivos y sus pesos moleculares



% En función de los productos de la reacción (coef no nulos) genere pesos moleculares.
% Si hay más de un producto (2) puede considerar un masa máxima (PM) para primer reactivo de
un 85 % del total de masa reaccionante.
% En el caso de 3 productos, puede ponderar las masa máxima del primero con un 65 % de la
masa total reaccionante, y
% un 85 % de la masa restante para el segundo producto como masa máxima posible.

% Genere valores aleatorios de PM para los inertes


end
```

Los test de evaluación de la función son los siguientes:

- *Comprueba que se cumple el balance de materia entre reactivos y productos para los PM generados:*

```
% Genera valores al azar de las variables de partida por defecto o no
for i=1:20

   N_Reactivos=randi([1,3]);
   N_Productos=randi([1,3]);
   [CoefReactivos,CoefProductos] = GeneradorReaccionQuimica(N_Reactivos,N_Productos);
   PesosMoleculares= Generador_PM_Reaccion(CoefReactivos,CoefProductos);


end
MasaReactivos=sum(CoefReactivos.*PesosMoleculares(1:3));
MasaProductos=sum(CoefProductos.*PesosMoleculares(4:6));
comprobacion=(MasaReactivos==MasaProductos);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“Revisa el código, no se cumple el balance de materia para la reacción: masa reaccionante igual a masa de productos”.

- Función *Generador_PM*:

```
function PM= Generador_PM(varargin)
% Esta función genera un valor real al azar comprendido, por defecto,
%entre 10 y 80 g/mol

% Se pueden introducir un máximo de 2 parámetros de entrada para modificar los valores mínimo
y máximo
% PmMin y  PmMax

numArgumentos = nargin;
if numArgumentos > 2
    error('La funciÛn tiene m·s par·metros de los permitidos: PmMin y PmMax');
end

% Se definen los valores por defecto de los parámetros
optargs = {10 80};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Sustituye los valores por defecto por los introducidos, si es el caso
[PmMin, PmMax] = optargs{:};

PM=round(rand()*(PmMax-PmMin)+PmMin,2);


end
```

- Test del Grader para la función *Generador_PM.*

La descripción del problema que los estudiantes ven en el evaluador es:

“Se debe programar una función que genere un eso molecular, es decir un valor real, con una precisión de dos cifras decimales, al azar. El intervalo por defecto para estos pesos moleculares será de entre 10 y 80. La función debe permitir incluir como argumentos opcionales el valor máximo y mínimo de los del intervalo de pesos moleculares.”

Y la plantilla de partida sobre la que incluir el código de la función es:

```
function PM= Generador_PM(varargin)
% Esta función genera un valor real al azar comprendido, por defecto,
%entre 10 y 80 g/mol

% Se pueden introducir un máximo de 2 parámetros de entrada para modificar los valores mínimo
y máximo
% PmMin y  PmMax

numArgumentos = nargin;
if numArgumentos > 2
    error('La función tiene más parámetros de los permitidos: PmMin y PmMax');
end

% Se definen los valores por defecto de los parámetros
optargs = {10 80};
```

```
% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Sustituye los valores por defecto por los introducidos, si es el caso
[PmMin, PmMax] = optargs{:};

% Genera un peso molecular dentro del intervalo especificado con una precisión
% de dos cifras decimales

end
```

Los test de evaluación de la función son los siguientes:

- *Comprueba que el valor generado está dentro del intervalo especificado:*

```
%% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
   PorDefecto=randi([0,1]);

   if PorDefecto==1
      % Ejecuta la función del estudiante y obtiene los resultados: y
      y = Generador_PM();
      resultado(i)=(y>=10 && y<=80);

   else
      PmMin=round(rand()*randi([1,80])+10,2);
      incremento=round(rand()*randi([10,40])+20,2);
      PmMax=PmMin+incremento;
      y = Generador_PM(PmMin,PmMax);

      %Comprueba que el valor obtenido esta en el intervalo requerido
      resultado(i)=(y>=PmMin && y<=PmMax);

   end

end

comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“Al menos uno de los PM obtenidos está fuera del intervalo dado (por defecto o especificado como argumentos de la función)”.

- *Comprueba que el valor generado no tiene más de dos cifras decimales:*

```
%% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
   PorDefecto=randi([0,1]);
```

```
    if PorDefecto==1
        % Ejecuta la función del estudiante y obtiene los resultados: y
        y = Generador_PM();

    else
        PmMin=rand()*randi([1,80])+10;
        incremento=rand()*randi([10,40])+20;
        PmMax=PmMin+incremento;
        y = Generador_PM(PmMin,PmMax);

    end
    % Evalua el número de cifras decimales del resultado

    caracteres=strsplit(num2str(y),'.');
    resultado(i)=length(caracteres{2})<=2;

end

comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“Al menos uno de los PM obtenidos tiene un número de cifras decimales mayor a 2.”.

- Función *Reactivo_limitante*:

```
function ReactivoLimitante = Reactivo_limitante(XiReactivos,...
    CoefReactivos)
% Esta función determina cuál de los reactivos alimentados al reactor es el
% limitante

% La conversión generada al azar se hará· con respecto al reactivo 1 (A),
% siendo el valor máximo alcanzable el correspondiente a una conversión del
% 100 % del reactivo limitante.

NReactivos=length(XiReactivos);
ReactivoLimitante=1;

for i=2:NReactivos
    ratioCaudales=(XiReactivos(ReactivoLimitante)/XiReactivos(i));
    ratioCoef=(CoefReactivos(ReactivoLimitante)/CoefReactivos(i));
    if ratioCaudales>ratioCoef
        ReactivoLimitante=i;
    end
end
```

- Función *Generador_ConversionA*:

```
function ConversionA = Generador_ConversionA(Limitante,XiReactivos,...
    CoefReactivos)

% Esta función genera una conversión al azar se hará· con respecto al reactivo 1 (A),
% siendo el valor máximo alcanzable el correspondiente a una conversión del
% 100 % del reactivo limitante.


% Se determina cual es la conversión máxima de A, i.e. caudal molar
% estequiométrico con respecto al reactivo limitante.
ratioCoef=(CoefReactivos(Limitante)/CoefReactivos(1));
FA_estequiometrico=XiReactivos(Limitante)/ratioCoef;
ConversionMax=FA_estequiometrico/(XiReactivos(1));
if ConversionMax<0.2
    ConversionMin=ConversionMax;
else
    ConversionMin=ConversionMax*0.25;
end


%Se genera un valor de la conversión: min= 25 % max(limitante)
ConversionA=round(rand()*(ConversionMax-ConversionMin)+ConversionMin,2);
end
```

- Función *Generador_RazonRecirculacion*:

```
function R= Generador_RazonRecirculacion(varargin)
% Esta función genera un valor real al azar comprendido, por defecto,
%entre 0.3 y 0.75

% Se pueden introducir un máximo de 2 parámetros de entrada para modificar los valores mínimo
y máximo
% RMin y  RMax

numArgumentos = nargin;
if numArgumentos > 2
    error('La funciÛn tiene m·s par·metros de los permitidos: RMin y RMax');
end

% Se definen los valores por defecto de los parámetros
optargs = {0.3 0.75};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Sustituye los valores por defecto por los introducidos, si es el caso
[RMin, RMax] = optargs{:};

R=round(rand()*(RMax-RMin)+RMin,2);


end
```

- Test del Grader para la función *Generador_RazonRecirculacion.*

La descripción del problema que los estudiantes ven en el evaluador es:

“Se debe programar una función que genere una razón de circulación cuyo valor esté comprendido entre 0.3 y 0.75, valores por defecto, o los incluidos como argumento en la función (siendo siempre un intervalo positivo y el máximo igual a la unidad).

Y la plantilla de partida sobre la que incluir el código de la función es:

```
function R= Generador_RazonRecirculacion(varargin)
% Esta función genera un valor real al azar comprendido, por defecto,
%entre 0.3 y 0.75

% Se pueden introducir un máximo de 2 parámetros de entrada para modificar los valores mínimo
y máximo
% RMin y  RMax

numArgumentos = nargin;
if numArgumentos > 2
    error('La función tiene más parámetros de los permitidos: RMin y RMax');
end

% Se definen los valores por defecto de los parámetros
optargs = {0.3 0.75};

% Copia de los valores por defecto
optargs(1:numArgumentos) = varargin;
% Sustituye los valores por defecto por los introducidos, si es el caso
[RMin, RMax] = optargs{:};

% Generar el valor de la razón en el intervalo correspondiente


end
```

Los test de evaluación de la función son los siguientes:

- *Comprueba que el valor generado está dentro del intervalo especificado:*

```
%% Genera valores al azar de las variables de partida por defecto o no

for i=1:20
    PorDefecto=randi([0,1]);

    if PorDefecto==1
        % Ejecuta la función del estudiante y obtiene los resultados: y
        y = Generador_RazonRecirculacion();
        resultado(i)=(y>=0.3 && y<=0.75);

    else
        a = 0.1; b = 0.5; RMin = (b-a)*rand() + a;
        RMax=round(rand()*(1-RMin)+RMin,2);
        y = Generador_RazonRecirculacion(RMin,RMax);

        %Comprueba que el valor obtenido está en el intervalo requerido
```

```
        resultado(i)=(y>=RMin && y<=RMax);

    end

end

comprobacion=all(resultado);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('comprobacion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

"Al menos en un caso la razón de recirculación obtenida está fuera del intervalo dado (por defecto o especificado como argumentos de la función)".

- Función *SistemaEcuaciones*:

```
function y=SistemaEcuaciones(Incognitas,V_Conocidas,CoefReactivos,...
    CoefProductos,X_A)


%En esta función se incluye el sistema de ecuaciones del balance de materia
%a resolver. Las ecuaciones han de expresarse de manera que cada una de
%lugar a un valor, y(i), que ha de minimizarse (hacerse cero) para obtener
%la resolución del sistema de ecuaciones (15 ecuaciones independientes).
y=zeros(1,15);

%Asignación de valores a cada una de las variables del sistema de
%ecuaciones: Ojo, han de estar todas definidas y ser coherente la
%nomenclatura seguida con la empleada en las ecuaciones de B. de materia.
%
Incog=num2cell(Incognitas);
VariablesConoc=num2cell(V_Conocidas);
[F2,X2(1),X2(2),X2(3),X2(4),X2(5),X2(6),X2(7),R]=VariablesConoc{:};
[F3,X3(1),X3(2),X3(3),X3(4),X3(5),X3(6),X3(7),...
 F1,X1(1),X1(2),X1(3),X1(4),X1(5),X1(6),X1(7),F4]=Incog{:};
F5=F3*R;

X1(8)=1-sum(X1(1:7));
X2(8)=1-sum(X2(1:7));
X3(8)=1-sum(X3(1:7));


%Balance en el reactor
%Ecuaciones para reactivos
for i=1:3
    y(i)=F2*X2(i)-F2*X2(1)*X_A*(CoefReactivos(i)/CoefReactivos(1))-F3*X3(i);
end
%Ecuaciones para productos
for i=4:6
    y(i)=F2*X2(i)+F2*X2(1)*X_A*(CoefProductos(i-3)/CoefReactivos(1))-F3*X3(i);
```

```
end
%Ecuaciones para inertes
for i=7:8
    y(i)=F2*X2(i)-F3*X3(i);
end

% Ecuaciones en el divisor
 y(9)=F3-F4-F5;

% Ecuaciones en el mezclador
for i=10:17
 y(i)=F2*X2(i-9)-F1*X1(i-9)-F5*X3(i-9);
end

end
```

- Funciones para la conversión entre caudales, totales y de compuestos, másicos (m y m_i) y molares (F y F_i), y para la conversión entre fracciones másicas y molares (W y X, respectivamente):

```
function [M] = Vector_FtoM(F,Xi,PMs)


if length(Xi)~= length(PMs)
    error('El vector de PMs tiene un tamaño distinto al vector Xi');
end

Fi=Xi.*F;
Mi=Fi.*PMs;
M=round(sum(Mi),2);
end

function [Fi] = Vector_mi_to_Fi(mi,PM)

Fi=round(mi/PM);

end


function [F] = Vector_MtoF(M,Wi,PMs)


if length(Wi)~= length(PMs)
    error('El vector de PMs tiene un tamaño distinto al vector Wi');
end

Mi=Wi.*M;
Fi=Mi./PMs;
F=round(sum(Fi),2);

end

function [Xi] = Vector_WtoX(fracciones,PMs)
```

```
if length(fracciones)~= length(PMs)
    error('El vector de PMs tiene un tamaño distinto al vector Wi');
end

cocientes=fracciones./PMs;

%Sumatorio de esos cocientes

Sumatorio=sum(cocientes);
Xi=round(cocientes./Sumatorio,3);

end

function [Wi] = Vector_XtoW(fracciones,PMs)

if length(fracciones)~= length(PMs)
    error('El vector de PMs tiene un tamaño distinto al vector Xi');
end

cocientes=round(fracciones.*PMs,3);
%Sumatorio de esos cocientes
Sumatorio=-sum(cocientes);
Wi=round(cocientes./Sumatorio,3);

end

function [mi] = Vector_Fi_to_mi(Fi,PM)

mi=round(Fi.*PM);

end
```

- Test del Grader para la función Vector_XtoW(fracciones,PMs). Para el resto de funciones de conversión las pruebas a las que se les somete a las funciones de los estudiantes son idénticas.

La descripción del problema que los estudiantes ven en el evaluador es:

“El estudiante deberá programar una función que permita convertir las fracciones molares de los compuestos pertenecientes a una corriente de proceso en fracciones másicas. Los parámetros de entrada de la función serán:

- Un vector en se incluirán los valores de todas las fracciones molares de la corriente (el sumatorio de todas ellas ha de ser 1).
- Un vector con los pesos moleculares de cada uno de los compuestos de la corriente.

La función ha de devolver como resultado un vector con las fracciones másicas de cada uno de los compuestos en la corriente (su suma debe ser 1).”

Y la plantilla de partida sobre la que incluir el código de la función es:

```
function [Wi] = Vector_XtoW(fracciones,PMs)

%Comprobación: ambos vectores, fracciones y PMs, tienen el mismo tamaño

if length(fracciones)~= length(PMs)
    error('El vector de PMs tiene un tamaño distinto al vector Xi');
end

%Ratio Xi/PMi


%Sumatorio de esos cocientes


%Cálculo de las fracciones másicas


end
```

Los test de evaluación de la función son los siguientes:

- *El resultado es un vector del mismo tamaño que los vectores de partida: fracciones y PMs:*

```
% Genera valores al azar de las variables de partida.

tamano=randi([1,6]);
fraccionesW=rand(1,tamano);
total=sum(fraccionesW);
fracciones=fraccionesW/total;
PmMax=120;
PmMin=10;
PMs=round(rand(1,tamano)*(PmMax-PmMin)+PmMin,2);

% Ejecuta la función del estudiante y obtiene los resultados: y

y = Vector_XtoW(fracciones,PMs);
solucion=size(y);

% Obtiene el valor de referencia con el que comparar los resultados del estudiante

yReference=size(fracciones);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante

assessVariableEqual('solucion', yReference);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“El resultado de la función es un vector con un número de elementos mayor que el de los vectores de partida: hay más fracciones másicas que molares.”.

- *Todos los valores del vector resultante son positivos y <=1:*

```
% Genera valores al azar de las variables de partida.
tamano=randi([1,6]);
fraccionesW=rand(1,tamano);
total=sum(fraccionesW);
fracciones=fraccionesW/total;
PmMax=120;
PmMin=10;
PMs=round(rand(1,tamano)*(PmMax-PmMin)+PmMin,2);

% Ejecuta la función del estudiante y obtiene los resultados: y
y = Vector_XtoW(fracciones,PMs);

%Comprueba que todos los valores del resultado están entre 0 y 1
comprobacion=(0<=y)&(y<=1);
solucion=all(comprobacion);

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('solucion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“El resultado de la función es un vector con valores negativos o superiores a la unidad (¡son fracciones másicas!)”.

- *La suma de las fracciones másicas (vector resultante) es igual a uno (+-0.001):*

```
% Genera valores al azar de las variables de partida.
tamano=randi([1,6]);
fraccionesW=rand(1,tamano);
total=sum(fraccionesW);
fracciones=fraccionesW/total;
PmMax=120;
PmMin=10;
PMs=round(rand(1,tamano)*(PmMax-PmMin)+PmMin,2);

% Ejecuta la función del estudiante y obtiene los resultados: y
y = Vector_XtoW(fracciones,PMs);

%Comprueba que todos los valores del resultado están entre 0 y 1
solucion=abs(sum(y)-1)<=0.001;

% Obtiene el valor de referencia con el que comparar los resultados del estudiante: vector de
unos.
%yReference=true;

% Compara los valores de referencia con los obtenidos mediante la función del estudiante
assessVariableEqual('solucion', true);
```

El evaluador muestra el siguiente mensaje cuando la función del estudiante no supera el test:

“La suma de las fracciones másicas calculadas es distinto a 1+-0.001”.

6.2. Reología.

6.2.1. Enunciado del problema.

Determine los valores de los parámetros m y n del modelo reológico potencial para los pares de valores experimentales obtenidos de la tensión causada para una cierta velocidad de deformación a la que ha sido sometida un fluido.

6.2.2. Generador del problemas (parámetros al azar y soluciones para ese caso).

```
clc
clearvars
n_ptos = input('¿Cuántos pares de datos tensión - velocidad de deformación quieres generar? ');

[tau,gamma,m,n]=generador_valores(n_ptos);
eta=tau./gamma; % viscosidad
tabla=table(eta,tau,gamma,'VariableNames',{'Viscosidad','Tension','Gamma'});

close all
Figure_1=figure();

Figure_1.Position =[200 400 600 600];
Plot1=subplot(2,1,1,'Parent',Figure_1);
hold(Plot1,'on');

tension_gamma=scatter(gamma,tau,'Parent',Plot1);
tension_gamma.LineWidth  =2;

Plot1.Title.String='Tensión vs. Velocidad de deformation';
Plot1.Title.FontSize=16;
Plot1.XLabel.String='\gamma (s^-1)';
Plot1.YLabel.String='\tau (N\cdot m^2)';
Plot1.YLabel.FontSize=16;
Plot1.XLabel.FontSize=16;
Plot1.FontSize=14;

Plot2=subplot(2,1,2,'Parent',Figure_1);
viscosidad_gamma=scatter(gamma,eta,'Parent',Plot2);
viscosidad_gamma.LineWidth  =2;
viscosidad_gamma.MarkerEdgeColor ='r';

Plot2.Title.String='Viscosidad vs. Velocidad de deformación';
Plot2.Title.FontSize=16;
Plot2.XLabel.String='\gamma (s^-1)';
Plot2.YLabel.String='\eta (Pa\cdot s)';
Plot2.YLabel.FontSize=14;
Plot2.XLabel.FontSize=14;
Plot2.FontSize=14;

% Ajuste de del modelo potencial a los datos experimentales
```

```
[y_predicha,tau_predicha_paraGrafia,gamma1,parametros,R2]=AjusteModeloPotencial(gamma,tau);
[eta_predicha,eta_predicha_paraGrafia,eta_gamma1,eta_parametros,eta_R2]=AjusteModeloPotencial(gamma,eta);


% Representación de los datos Experimentales y de los predichos

Figure_2=figure();
Figure_2.Position =[900 400 600 600];

ax=subplot(2,1,1,'Parent',Figure_2);
hold(ax,'on');

plot1=scatter(gamma,tau,'Parent',ax);
plot1.LineWidth  =2;
plot1.DisplayName='\tau Experimentales';
plot1.MarkerEdgeColor ='r';

plot2=plot(gamma1,tau_predicha_paraGrafia,'Parent',ax);
plot2.LineWidth  =2;
plot2.DisplayName=strcat('\tau Predichos - R^2: ',num2str(round(R2,4)));
plot2.Marker='none';
plot2.LineStyle='-';
plot2.Color='b';

ax.Title.String='Tensión vs. Velocidad de deformation';
ax.Title.FontSize=16;
ax.XLabel.String='\gamma (s^-1)';
ax.YLabel.String='\tau (N\cdot m^2)';
ax.YLabel.FontSize=16;
ax.XLabel.FontSize=16;
ax.FontSize=14;
lg=legend([plot1 plot2],{plot1.DisplayName,plot2.DisplayName});
lg.Location='best';
lg.Box='off';

ax1=subplot(2,1,2,'Parent',Figure_2);
hold(ax1,'on');

plot3=scatter(gamma,eta,'Parent',ax1);
plot3.LineWidth  =2;
plot3.DisplayName='\eta Experimentales';
plot3.MarkerEdgeColor ='k';

plot4=plot(eta_gamma1,eta_predicha_paraGrafia,'Parent',ax1);
plot4.LineWidth  =2;
plot4.DisplayName=strcat('\eta Predichos - R^2: ',num2str(round(eta_R2,4)));
plot4.Marker='none';
plot4.LineStyle='-';
plot4.Color='g';

ax1.Title.String='Viscosidad vs. Velocidad de deformación';
ax1.Title.FontSize=16;
ax1.XLabel.String='\gamma (s^-1)';
ax1.YLabel.String='\eta (Pa\cdot s)';
ax1.YLabel.FontSize=16;
```

```
ax1.XLabel.FontSize=16;
ax1.FontSize=14;
lg=legend([plot3 plot4],{plot3.DisplayName,plot4.DisplayName});
lg.Location='best';
lg.Box='off';
```

6.2.3. Funciones empleadas en el generador que han de desarrollar los estudiantes y comprobar usando Matlab Grader.

- Función *generador_valores*:

```
function [tau1,gamma,m,n]=generador_valores(varargin)

   % Se puede introducir como parámetro el número de puntos; si no se define se generan 10
puntos por defecto

   numArgumentos = nargin;
   if numArgumentos > 1
      error('La función tiene más parámetros de los permitidos: número de puntos');
   end

   % Se definen los valores por defecto de los parámetros: 10 puntos
   optargs = {10};

   % Copia de los valores por defecto
   optargs(1:numArgumentos) = varargin;

   n_ptos = optargs{:};

   [m, n] = generador_parametros();


   % generación de los valores inicial y final de gamma
   % gamma inicial entre 1 y 10 con una cifra decimal
   % gamma final entre 100 y 1000 con una cifra decimal

   gamma0 = numeroAleatorio(1,10,1);
   gamma_final=numeroAleatorio(100., 1000.,1);
   gamma = linspace(gamma0, gamma_final, n_ptos);



   % generación de valores de tau con un error de +- 2% para m y n

   tau=modeloPotencial(gamma, m,n);
   tau1=zeros(n_ptos,1);
   for i=1:size(tau,2)
      tau1(i)=tau(i)+numeroAleatorio(-0.04,0.04,3)*tau(i);
   end
   gamma=gamma';

end
```

- Función *generador_parametros*:

```
function [m,n]=generador_parametros()

    % generación de parámetros del modelo potencial: m y n

    % generación de m comprendido entre 10 y 1000 y con una cifra decimal
    %limite superior
    max_m=1000;
    %limite inferior
    min_m=10;
    m = round((max_m-min_m)*rand(1)+min_m,1);


    % generación de n comprendido entre 0.2 y 1.4 y con tres cifras
    % decimales
    %límite superior
    max_n=1.4;

    %límite inferior
    min_n=0.2;
    n = round((max_n-min_n)*rand(1)+min_n,3);

end
```

- Función *numeroAleatorio*:

```
function valor=numeroAleatorio(varargin)

    numArgumentos = nargin;
    if numArgumentos > 3
        error('La funciÛn tiene m·s par·metros de los permitidos: n˙mero de puntos');
    end

    % Se definen los valores por defecto de los parámetros: 10 puntos
    optargs = {0 1 0};

    % Copia de los valores por defecto
    optargs(1:numArgumentos) = varargin;

    [min, max,decimales] = optargs{:};

    % generación de n comprendido entre min y max y con decimales cifras
    valor= round((max-min)*rand(1)+min,decimales);

end
```

- Función *modeloPotencial*:

```
function tension=modeloPotencial(gamma, m,n)
```

```
    %devuelve el valor de la tensión generada por un esfuerzo cortante
    %(gamma) para un modelo reológico potencial

    tension=m*gamma.^n;

end
```

- Función *AjusteModeloPotencial*:

```
function
[y_predicha,y_predicha_paraGrafia,gamma1,parametros,R2]=AjusteModeloPotencial(x_teorica,y_
teorica)

    % como parámetros se han de introducir los valores de x e y
    % experimentales
    numArgumentos = nargin;
    if numArgumentos > 2
        error('La función tiene más parámetros de los permitidos: x e y experimentales');
    elseif numArgumentos < 2
         error('La función tiene menos parámetros de los permitidos: x e y experimentales');

    end

    % definir función de ajuste y el método de cálculo de los parámetros.
    ft = fittype( 'power1' ); % y=a*x^b
    opts = fitoptions( 'Method', 'NonlinearLeastSquares' );
    opts.Display = 'Off';
    opts.StartPoint = [500 1]; % Valores iniciales

    % Ajuste del modelo a los datos

    fitresult = fit( x_teorica, y_teorica, ft, opts );
    parametros=struct();
    parametros.m=fitresult.a;
    parametros.n=fitresult.b;


    y_predicha=feval(fitresult,x_teorica);
    R2=Calculo_R2( y_predicha, y_teorica);
    gamma1 = linspace(x_teorica(1), x_teorica(end), 200);
    y_predicha_paraGrafia=feval(fitresult,gamma1);

end
```

- Función *Calculo_R2*:

```
function R2=Calculo_R2( y, yteorica)

    % Función para calcular la R2 entre un vector de valores reales y otro
    % de valores predichos
```

```
    residuo = y - yteorica;  % Residuo: Experimentales menos predichos
    ss_res = sum(residuo.^ 2);  % SSresiduo: Suma del cuadrado de los residuos
    ss_tot = sum((y - mean(y)).^2);  % Suma de cuadrados total
    R2= 1 - (ss_res / ss_tot);  % Cálculo de R2


end
```

6.3. Cinética química – Ec. de Arrhenius.

6.3.1. Enunciado del problema.

Determine los valores de la energía de activación y el parámetro preexponencial de la ecuación de Arrhenius partiendo de los siguientes valores de la constante cinética a las temperaturas indicadas.

6.3.2. Generador del problemas (parámetros al azar y soluciones para ese caso).

```
clc
clearvars
n_ptos = input('øCu·ntos pares de datos tensiÛn - velocidad de deformaciÛn quieres generar? ');

[T,k,Ea,Ln_K]=generador_Valores(n_ptos);
T_Kelvin=T+273.15;
tabla=table(T_Kelvin,k,'VariableNames',{'Temperatura','k'});

close all
Figure_1=figure();

Figure_1.Position =[200 400 600 400];
ax=axes('Parent',Figure_1);
hold(ax,'on');

plot1=scatter(T_Kelvin,k,'Parent',ax);
plot1.LineWidth  =2;

ax.Title.String='Constante cinÈtica vs. temperatura';
ax.Title.FontSize=16;
ax.XLabel.String='Temperatura (K)';
ax.YLabel.String='k (-)';
ax.YLabel.FontSize=16;
ax.XLabel.FontSize=16;
ax.FontSize=14;

% Ajuste de del modelo potencial a los datos experimentales
[y_predicha,k_paraGrafia,T_grafica,parametros,R2]=AjusteArrhenius_lineal(T,k);

% Representación de los datos Experimentales y de los predichos
Figure_2=figure();
Figure_2.Position =[900 400 600 400];

ax1=axes('Parent',Figure_2);
```

```
hold(ax1,'on');

plot2=scatter(T,k,'Parent',ax1);
plot2.LineWidth  =2;
plot2.DisplayName='k Experimentales';
plot2.MarkerEdgeColor ='r';

ax1.Title.String='Constante cinética vs. temperatura';
ax1.Title.FontSize=16;
ax1.XLabel.String='Temperatura (K)';
ax1.YLabel.String='k (-)';
ax1.YLabel.FontSize=16;
ax1.XLabel.FontSize=16;
ax1.FontSize=14;


plot3=plot(T_grafica, k_paraGrafia,'Parent',ax1);
plot3.LineWidth  =2;
plot3.DisplayName=strcat('k Predichos - R^2: ',num2str(round(R2,4)), ' \newline Ea (kJ/mol):
',num2str(round(parametros.Ea,1)),...
    '; ln(k): ',num2str(round(parametros.Ln_k0,3)));
plot3.Marker='none';
plot3.LineStyle='-';
plot3.Color='b';

lg=legend([plot2 plot3],{plot2.DisplayName,plot3.DisplayName});
lg.Location='best';
lg.Box='off';
```

6.3.3. Funciones empleadas en el generador que han de desarrollar los estudiantes y comprobar usando Matlab Grader.

- Función *generador_Valores*:

```
function [T,k,Ea,Ln_K]=generador_Valores(varargin)

    % Se puede introducir como parámetro el número de puntos; si no se define se generan 10
puntos por defecto

    numArgumentos = nargin;
    if numArgumentos > 1
        error('La función tiene más parámetros de los permitidos: número de puntos');
    end

    % Se definen los valores por defecto de los parámetros: 10 puntos
    optargs = {10};

    % Copia de los valores por defecto
    optargs(1:numArgumentos) = varargin;

    n_ptos = optargs{:};

    [Ea,Ln_K] = generador_parametros();
```

```
    % Valor mínimo de la temperatura entre 0 y 20 ∫C (1 decimal)
    T_min=numeroAleatorio(0,20,1);
    % Valor máximo de la temperatura entre 50 y 100 ∫C (1 decimal)
    T_max=numeroAleatorio(50,100,1);

    % iniciar como ceros vector k y vector temperaturas equiespaciadas
    k=zeros(n_ptos,1);
    T = linspace(T_min,T_max, n_ptos);


    for i=1:size(T,2)
        k(i)=Arrhenius(T(i),Ea,Ln_K);
    end
    T=T';

end
```

- Función *generador_parametros*:

```
function [Ea,Ln_K]=generador_parametros()
    % generación de parámetros del modelo cinético potencial
    % Ea entre 30 y 150 kJ/mol (1 cifra decimal), Ln(K) entre 1 y 20 (3 cifras decimales)

    Ea=numeroAleatorio(50,150,1);
    Ln_K =numeroAleatorio(10,40,3);


end
```

- Función *Arrhenius*:

```
function k=Arrhenius(T,Ea,Ln_K)

    k=exp(Ln_K+numeroAleatorio(-0.05, 0.05,4))*exp(-((Ea+numeroAleatorio(-0.5,
0.5,3))*1000)/(8.31*(T+273.15)));

end
```

- Función *AjusteArrhenius_lineal*:

```
function [y_predicha,k_paraGrafia,T_grafica,parametros,R2]=AjusteArrhenius_lineal(T,k)

    % como parámetros se han de introducir los valores de x e y
    % experimentales
    numArgumentos = nargin;
    if numArgumentos > 2
        error('La funciÛn tiene m·s par·metros de los permitidos: x e y experimentales');
```

```
    elseif numArgumentos < 2
        error('La funciÛn tiene menos par·metros de los permitidos: x e y experimentales');

    end
    Ln_k=log(k);
    inv_T=1./(T+273.15);
    [x,y]=prepareCurveData(inv_T,Ln_k);

       % definir función de ajuste y el método de cálculo de los parámetros.
    ft = fittype( 'poly1' ); % Ec. Arrhenius
    opts = fitoptions( 'Method', 'LinearLeastSquares' );

    % Ajuste del modelo a los datos

    fitresult = fit( x,y, ft, opts );
    parametros=struct();
    parametros.Ln_k0=fitresult.p2;
    parametros.Ea=(fitresult.p1)*8.31/1000;



    y_predicha=feval(fitresult,inv_T);
    R2=Calculo_R2( y_predicha, Ln_k);
    T_grafica = linspace(T(1), T(end), 200);
    Ln_k_paraGrafia=feval(fitresult,1./(T_grafica+273.15));
    k_paraGrafia=exp(Ln_k_paraGrafia);

end
```

6.4. Cinética química.

6.4.1. Enunciado del problema.

Determine los valores de los parámetros de un modelo cinético potencial, constante cinética y orden de reacción, empleando para ello los métodos diferencial e integral partiendo de los valores experimentales obtenidos para la concentración del reactivo frente al tiempo de reacción.

6.4.2. Generador del problema (parámetros al azar y soluciones para ese caso).

```
clc
clearvars
n_ptos = input('¿Cuántos pares de datos concentración vs. Tiempo quieres generar? ');


[k,Ea,Ln_K0,n,T,Ca0]=generador_parametros();
[t_final,Ca_final,tiempo,Ca,Ca_tabla,tiempo_tabla]=Calculo_CaVstiempo(Ca0, n, k,n_ptos);
tabla=table(tiempo_tabla,Ca_tabla,'VariableNames',{'tiempo','Concentracion'});

close all
Figure_1=figure();

Figure_1.Position =[200 800 600 400];
ax=axes('Parent',Figure_1);
```

```
hold(ax,'on');

plot1=scatter(tiempo_tabla,Ca_tabla,'Parent',ax);
plot1.LineWidth  =2;

ax.Title.String='Evolución de la concentración';
ax.Title.FontSize=16;
ax.XLabel.String='tiempo (s)';
ax.YLabel.String='Concentración (mol/L)';
ax.YLabel.FontSize=16;
ax.XLabel.FontSize=16;
ax.FontSize=14;

%% Método diferencial

r=diferencial(Ca_tabla,tiempo_tabla);
Ca_medio=media(Ca_tabla);

Ln_r=log(r);
Ln_CaMedio=log(Ca_medio);

% Ajuste de del modelo lineal a los datos experimentales
[y_predicha,Lnr_paraGrafia,LnCa_grafica,parametros,R2]=AjusteLineal(Ln_CaMedio,Ln_r);


Figure_2=figure();
Figure_2.Position =[900 800 600 400];

ax1=axes('Parent',Figure_2);
hold(ax1,'on');

plot2=scatter(Ln_CaMedio,Ln_r,'Parent',ax1);
plot2.LineWidth  =2;
plot2.DisplayName='Experimentales';
plot2.MarkerEdgeColor ='r';

ax1.Title.String='Método diferencial';
ax1.Title.FontSize=16;
ax1.XLabel.String='Ln Ca medio (-)';
ax1.YLabel.String='Ln r (-)';
ax1.YLabel.FontSize=16;
ax1.XLabel.FontSize=16;
ax1.FontSize=14;


plot3=plot(LnCa_grafica, Lnr_paraGrafia,'Parent',ax1);
plot3.LineWidth  =2;
plot3.DisplayName=strcat('Predichos - R^2: ',num2str(round(R2,4)), ' \newline n:
',num2str(round(parametros.n,3)),...
    '; ln(k): ',num2str(round(parametros.Ln_k,3)));
plot3.Marker='none';
plot3.LineStyle='-';
plot3.Color='b';

lg=legend([plot2 plot3],{plot2.DisplayName,plot3.DisplayName});
lg.Location='best';
```

```
lg.Box='off';

%% %% Método integral


% Modelo potencia con n=1

[LnCa_predicha,LnCa_paraGrafia,tiempo_grafica_int1,parametros_int1,R2_int1]=AjusteLineal_int
1(tiempo_tabla,log(Ca_tabla),Ca0);

Ca_predicha_int1=exp(LnCa_paraGrafia);
Figure_3=figure();
Figure_3.Position =[200 0 600 400];

ax2=axes('Parent',Figure_3);
hold(ax2,'on');

plot4=scatter(tiempo_tabla,Ca_tabla,'Parent',ax2);
plot4.LineWidth  =2;
plot4.DisplayName='Experimentales';
plot4.MarkerEdgeColor ='g';

ax2.Title.String='Método integral - n=1';
ax2.Title.FontSize=16;
ax2.XLabel.String='tiempo (s)';
ax2.YLabel.String='Concentración (mol/L)';
ax2.YLabel.FontSize=16;
ax2.XLabel.FontSize=16;
ax2.FontSize=14;


plot5=plot(tiempo_grafica_int1, Ca_predicha_int1,'Parent',ax2);
plot5.LineWidth  =2;
plot5.DisplayName=strcat('Predichos - R^2: ',num2str(round(R2_int1,4)), ' \newline n: 1',...
    '; ln(k): ',num2str(round(parametros_int1.Ln_k,3)));
plot5.Marker='none';
plot5.LineStyle='-';
plot5.Color='k';

lg1=legend([plot4 plot5],{plot4.DisplayName,plot5.DisplayName});
lg1.Location='best';
lg1.Box='off';

% Modelo potencia con n distinto de 1

% Ajuste no lineal de n y k mediante integración numérica
parametros_inicialesIntegral(1)=parametros.n;
parametros_inicialesIntegral(2)=parametros.Ln_k;
options=optimset('Display','off','MaxFunEval',1000,'MaxIter',200,'TolFun',1e-6);
[x,resnorm,residual,exitflag,output]=lsqcurvefit(@modeloIntegral,real(parametros_inicialesIntegral)
,...
    tiempo_tabla,Ca_tabla,[0.001 -20],[5 40 ],options,Ca0);

Ca_predicha_int2=modeloIntegral(x,tiempo_tabla,Ca0);
Ca_predichaGrafica_int2=modeloIntegral(x,tiempo_grafica_int1,Ca0);
R2_int2=Calculo_R2( Ca_predicha_int2, Ca_tabla);
```

```
Figure_4=figure();
Figure_4.Position =[900 0 600 400];

ax3=axes('Parent',Figure_4);
hold(ax3,'on');

plot6=scatter(tiempo_tabla,Ca_tabla,'Parent',ax3);
plot6.LineWidth  =2;
plot6.DisplayName='Experimentales';
plot6.MarkerEdgeColor ='k';

ax3.Title.String='Método integral - n distinto de 1';
ax3.Title.FontSize=16;
ax3.XLabel.String='tiempo (s)';
ax3.YLabel.String='Concentración (mol/L)';
ax3.YLabel.FontSize=16;
ax3.XLabel.FontSize=16;
ax3.FontSize=14;


plot7=plot(tiempo_grafica_int1, Ca_predichaGrafica_int2,'Parent',ax3);
plot7.LineWidth  =2;
plot7.DisplayName=strcat('Predichos - R^2: ',num2str(round(R2_int2,4)), ' \newline n:
',num2str(round(x(1),3)),...
    '; ln(k): ',num2str(round(x(2),3)));
plot7.Marker='none';
plot7.LineStyle='-';
plot7.Color='r';

lg2=legend([plot6 plot7],{plot6.DisplayName,plot7.DisplayName});
lg2.Location='best';
lg2.Box='off';
```

6.4.3. Funciones empleadas en el generador que han de desarrollar los estudiantes y comprobar usando Matlab Grader.

- Función *generador_parametros*:

```
function [k,Ea,Ln_K0,n,T,Ca0]=generador_parametros()

    % generación de parámetros de la ecuación cinética y Temperatura de trabajo

    Ea=numeroAleatorio(40,80); % Energía de activación en kJ/mol
    Ln_K0 = numeroAleatorio(15,25,3); % Ln del parámetro preexponencial de la Ec. Arrhenius
    T= numeroAleatorio(40,80,1); %Temperatura en ºC
    k = Arrhenius(T,Ea,Ln_K0); % Contante cinética (valor, las unidades dependen de n)
    n=numeroAleatorio(0.5,2.05,2); % orden de reacción
    Ca0=numeroAleatorio(10,50,1); % Concentración inicial (mol/L)


end
```

- Función *Calculo_CaVstiempo*:

```
function [t_final,Ca_final,tiempo,Ca,Ca_tabla,tiempo_tabla]=Calculo_CaVstiempo(Ca0, n,
k,n_ptos)

    t_final= 10;
    Ca_final=0;

    while  (Ca_final>=0.5*Ca0 || Ca_final<=0.01*Ca0 )

        t=linspace(0,t_final,n_ptos*20);

        options=odeset('RelTol',1e-5,'AbsTol',1e-5);

        [t,y]=ode45(@ModeloPotencial,t,Ca0,options,n,k);
        Ca_final=y(end);
        Ca=y;
        tiempo =t;

        if Ca_final>=0.5*Ca0
            t_final=t_final+0.2*t_final;
        elseif Ca_final<=0.1*Ca0
            t_final=t_final-0.25*t_final;
        end

    end
    indices=linspace(1,20*n_ptos,n_ptos);
    indices=round(indices);
    Ca_tabla=Ca(indices);
    tiempo_tabla=tiempo(indices);

    for i=1:size(Ca_tabla,1)
        Ca_tabla(i)= Ca_tabla(i)*(1+numeroAleatorio(-0.0023, 0.002,3));
    end
    Ca_tabla=round(Ca_tabla,3);

end
```

- Función *ModeloPotencial*:

```
function ConcentracionFinal=ModeloPotencial(~,C,n,k)
    ConcentracionFinal=-k*C^n;
end
```

- Función *diferencial*:

```
function dy=diferencial(C,t)

    % Tanto C como t son vectores con el mismo tamaño. Comprobación.

    if size(C)~=size(t)
```

```
        error('Los vectores introducidos como argumentos tienen distinto tamaño');
    end

    dy=zeros(size(C,1)-1,1);
    for i=1:(size(C,1)-1)
        dy(i)=(-(C(i+1)-C(i))/(t(i+1)-t(i)));
    end

end
```

- Función *media*:

```
function C_medio=media(C)


    C_medio=zeros(size(C,1)-1,1);
    for i=1:(size(C,1)-1)
        C_medio(i)=((C(i+1)+C(i))/2);
    end

end
```

- Función *AjusteLineal*:

```
function
[y_predicha,Lnr_paraGrafia,LnCa_grafica,parametros,R2]=AjusteLineal(Ln_CaMedio,Ln_r)

    % como parámetros se han de introducir los valores de x e y
    % experimentales
    numArgumentos = nargin;
    if numArgumentos > 2
        error('La función tiene más parámetros de los permitidos: x e y experimentales');
    elseif numArgumentos < 2
         error('La función tiene menos parámetros de los permitidos: x e y experimentales');

    end


    % definir función de ajuste y el método de cálculo de los parámetros.
    ft = fittype( 'poly1' ); % Ec. Arrhenius
    opts = fitoptions( 'Method', 'LinearLeastSquares' );

    % Ajuste del modelo a los datos
    fitresult = fit( Ln_CaMedio, Ln_r, ft, opts );
    parametros=struct();
    parametros.Ln_k=fitresult.p2;
    parametros.n=fitresult.p1;


    y_predicha=feval(fitresult,Ln_CaMedio);
```

```
    R2=Calculo_R2( y_predicha, Ln_r);
    LnCa_grafica = linspace(Ln_CaMedio(1), Ln_CaMedio(end), 200);
    Lnr_paraGrafia=feval(fitresult,LnCa_grafica);



end
```

- Función *AjusteLineal_int1*:

```
function
[y_predicha,Lnr_paraGrafia,LnCa_grafica,parametros,R2]=AjusteLineal_int1(Ln_CaMedio,Ln_r,Ca
0)

    % como parámetros se han de introducir los valores de x e y
    % experimentales
    numArgumentos = nargin;
    if numArgumentos > 3
        error('La función tiene más parámetros de los permitidos: x e y experimentales');
    elseif numArgumentos < 3
         error('La funciÛn tiene menos par·metros de los permitidos: x e y experimentales');

    end



    % definir función de ajuste y el método de cálculo de los parámetros.
    ft = fittype( 'poly1' ); % Ec. Arrhenius
    opts = fitoptions( 'Method', 'LinearLeastSquares' );
    opts.Lower = [-Inf log(Ca0)];
    opts.Upper = [Inf log(Ca0)];

    % Ajuste del modelo a los datos
    fitresult = fit( Ln_CaMedio, Ln_r, ft, opts );
    parametros=struct();
    parametros.Ln_k=log(-fitresult.p1);

    y_predicha=feval(fitresult,Ln_CaMedio);
    R2=Calculo_R2( y_predicha, Ln_r);
    LnCa_grafica = linspace(Ln_CaMedio(1), Ln_CaMedio(end), 200);
    Lnr_paraGrafia=feval(fitresult,LnCa_grafica);

end
```

- Función *modeloIntegral*:

```
function [Ca]=modeloIntegral(x,t,Ca0)
        n=x(1);
        k=exp(x(2));
        options=odeset('RelTol',1e-5,'AbsTol',1e-5);
        [~,Ca]=ode45(@ModeloPotencial,t,Ca0,options,n,k);

end
```

6.5. Encuestas de opinión para los estudiantes.

Encuesta de Opinión

1. La descarga e instalación del software Matlab desde la web de la UCM fue sencilla.
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.
    - ◯ No sabe/no contesta.
    - ◯ En desacuerdo.
    - ◯ Completamente en desacuerdo.

2. El manejo del software Matlab y de la aplicación Matlab Grader fue sencillo.
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.
    - ◯ No sabe/no contesta.
    - ◯ En desacuerdo.
    - ◯ Completamente en desacuerdo.

3. Las explicaciones del profesor y los materiales alojados en el Campus Virtual fueron suficientes comprender los problemas planteados y el objetivo de las funciones que se han de programar..
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.
    - ◯ No sabe/no contesta.
    - ◯ En desacuerdo.
    - ◯ Completamente en desacuerdo.

4. Considero importante la metodología propuesta para la comprensión del temario de las clases teóricas.
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.
    - ◯ No sabe/no contesta.
    - ◯ En desacuerdo.
    - ◯ Completamente en desacuerdo.

5. El trabajo realizado durante la programación de las funciones que se emplearán en los generadores de problemas mejoró mis habilidades y facilitó mi comprensión de la asignatura Informática Aplicada.
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.
    - ◯ No sabe/no contesta.
    - ◯ En desacuerdo.
    - ◯ Completamente en desacuerdo.

6. El trabajo realizado durante la programación de las funciones que se emplearán en los generadores de problemas mejoró mis habilidades y facilitó mi comprensión de la asignatura Fundamentos de Ingeniería Química.
    - ◯ Completamente de acuerdo.
    - ◯ De acuerdo.

◯ No sabe/no contesta.
◯ En desacuerdo.
◯ Completamente en desacuerdo.

7. Considero que el proyecto llevado a cabo y las herramientas resultantes de dicho proyecto suponen una mejora docente significativa.
   ◯ Completamente de acuerdo.
   ◯ De acuerdo.
   ◯ No sabe/no contesta.
   ◯ En desacuerdo.
   ◯ Completamente en desacuerdo.

8. Considero que lo aprendido a lo largo de este proyecto será de utilidad durante el resto de mis estudios en el Grado en Ingeniería Química.
   ◯ Completamente de acuerdo.
   ◯ De acuerdo.
   ◯ No sabe/no contesta.
   ◯ En desacuerdo.
   ◯ Completamente en desacuerdo.

9. Me gustaría disponer de más herramientas similares en un futuro en otras asignaturas similares.
   ◯ Completamente de acuerdo.
   ◯ De acuerdo.
   ◯ No sabe/no contesta.
   ◯ En desacuerdo.
   ◯ Completamente en desacuerdo.

10. Me gustaría poder realizar proyectos docentes similares en futuras asignaturas, empleando la metodología *learn to program/program to learn.*
    ◯ Completamente de acuerdo.
    ◯ De acuerdo.
    ◯ No sabe/no contesta.
    ◯ En desacuerdo.
    ◯ Completamente en desacuerdo.