



TRABAJO FIN DE GRADO

GRADO EN MATEMÁTICAS

Introducción a los procesos de interacción de fluido-estructura (F.S.I.)

Trabajo realizado por

Ricardo Massoxi Brandão Domingos António

Pasaporte: N2356286

Tutelado por

Eliseo Chacón Vera

Departamento de Matemáticas, Universidad de Murcia

FACULTAD DE MATEMÁTICAS, UNIVERSIDAD DE MURCIA

—
Año académico 2018-2019

D. Ricardo Massoxi Brandão Domingos António, estudiante matriculado en el Grado en Matemáticas en la Facultad de Matemáticas de la Universidad de Murcia, **DECLARO:**

Que el Trabajo de Fin de Grado que presento para su exposición y defensa titulado

Introducción a los procesos de interacción de fluido-estructura (F.S.I.).

y cuyo tutor es

D. Eliseo Chacón Vera

es original y que todas las fuentes utilizadas para su realización han sido debidamente citadas en el mismo.

Murcia, a 23 de Enero de 2019.

Resumen

El objetivo de este trabajo es introducir los conceptos básicos necesarios para la resolución de un problema de interacción de fluido y estructura (FSI), a partir de un modelo matemático presentado por Emmanuel Lefrançois y Jean-Paul Boufflet (*An Introduction to Fluid-Structure Interaction: Application to the Piston Problem*) [15]. En este trabajo se buscó introducir el problema (FSI) de una forma básica, teniendo en cuenta que tanto la estructura como el fluido necesitan ser combinados o acoplados al tratar con problemas de interacción de fluido estructura (FSI). En primer lugar, presentamos el problema a partir del cual fue construido el modelo para fluido estructura, así como algunas nociones de física para fluidos y luego evaluamos el modelo del acoplamiento; teniendo en cuenta los resultados presentados por [15] y las posibles mejoras que pueden ser alcanzadas al ser utilizado un método numérico de orden superior, diferente del modelo numérico utilizado por los autores que tiene limitación de orden (orden 2 el máximo que puede alcanzar), sabiendo que es un modelo desarrollado para cálculo de dinámica estructural en un momento en que los mayores obstáculos de las aproximaciones numéricas computacionales era el tiempo de procesamiento de los ordenadores, que actualmente está superado. En este trabajo también se estudiaron los métodos de Runge-Kutta explícito con detalles hasta el orden 4 como parte de un capítulo y se hizo una comparación del método de Runge-Kutta clásico de orden 4 con el método de Newmark utilizado por [15].

El enfoque que presentamos examina la interacción entre un gas contenido en una cámara unidimensional cerrado por un pistón móvil, fijado a un punto externo con un muelle. Se utiliza un modelo para la estructura y otro para el flujo del fluido. En la versión más simples de este problema podemos utilizar un método numérico de Runge-Kutta explícito de orden 4 para aproximar las respectivas soluciones numéricas. Versiones más completas de este problema quedan fuera de el alcance de este trabajo de fin de grado y se pueden ver en [15]. Al final (en el apéndice) de este trabajo presentamos una introducción a MATLAB con contenidos suficientes para entender el método utilizando y familiarizarse con este software. El código utilizado en este trabajo se presenta y se explica.

En resumen, podemos decir que se ha trabajado y profundizado en las siguientes temáticas:

1. Se ha presentado y estudiado el problema básico de FSI introducido en [15].
2. Se han estudiado los métodos de Runge-Kutta explícitos de forma detallada y se ha aplicado el método de Runge-Kutta clásico de cuarto orden (RKE4) para la resolución numérica del problema FSI modelo.

-
3. Se ha considerado el método de Newmark usado en [15] y comparado de forma elemental con el método RKE4 en términos de orden y de estabilidad.
 4. Se han reproducido los resultados numéricos expuestos en el trabajo [15] mediante RKE4 con absoluta concordancia. Derivándose las consecuencias físicas razonables.
 5. Se ha aprendido el manejo del software MATLAB y se ha generado un pequeño manual introductorio al mismo.
 6. Se ha aprendido el manejo del software LaTeX desconocido por el autor de este trabajo anteriormente.

Palabra clave: Interacción-Fluido-Estructura, métodos numéricos, masa-resorte, presión, MATLAB, Runge-Kutta clásico de orden 4.

Resumo

O objetivo deste trabalho é introduzir os conceitos básicos necessários para a resolução de um problema de interação de fluido estrutura (FSI), a partir de um modelo matemático apresentado por Emmanuel Lefrançois e Jean-Paul Boufflet (*An Introduction to Fluid-Structure Interaction: Application to the Piston Problem*) [15]. Neste trabalho se buscou introduzir o problema (FSI) de uma forma básica, levando em conta que tanto a estrutura quanto o fluido precisam ser combinados ou acoplados ao lidar com problemas de interação de fluido estrutura (FSI). Primeiro, apresentamos o problema a partir do qual foi construído o modelo para fluido estrutura, assim como algumas noções de física para fluidos e depois avaliamos o modelo do acoplamento; tendo em consideração os resultados apresentados por [15] e as possíveis melhorias que podem ser alcançadas ao ser utilizado um método numérico de ordem superior, diferente do modelo numérico utilizado pelos autores que possui limitação de ordem (ordem 2 o máximo que pode alcançar), sabendo que é um modelo desenvolvido para cálculo de dinâmica estrutural numa altura em que os maiores obstáculos das aproximações numéricas computacionais era o tempo de processamento dos computadores, que atualmente está ultrapassado. Neste trabalho também estudou-se os métodos de Runge-Kutta explícito com detalhes até a ordem 4 como parte de um capítulo e fez-se uma comparação do método de Runge-Kutta clássico de ordem 4 com o método de Newmark utilizado por [15].

A abordagem apresentada examina a interação entre um gás contido em uma câmara unidimensional fechado por um pistão móvel, fixado a um ponto externo com uma mola. Se utiliza um modelo para a estrutura e outro para resolver o fluxo do fluido. Na versão mais simples deste problema podemos utilizar um método numérico de Runge-kutta explícito de ordem 4 para aproximar as respectivas soluções numéricas. Versões mais completas deste problema estão fora do alcance deste trabalho de fim de curso e pode ser consultado em [15]. No final (Anexo) deste trabalho apresentamos uma introdução ao MATLAB com conteúdos suficientes para entender o método utilizado e familiarizar-se com este software. O código utilizado neste trabalho é apresentado e explicado.

De uma forma resumida, podemos dizer que trabalhou-se e aprofundou-se os seguintes tópicos:

1. Apresentou-se e estudou-se o problema básico de FSI introduzido em [15].
2. Estudou-se os métodos de Runge-Kutta explícitos de forma detalhada e aplicou-se o método de Runge-Kutta clássico de quarta ordem (RKE4) para a resolução numérica do problema modelo de FSI.

-
3. Considerou-se o método de Newmark usado em [15] e comparou-se de forma elementar com o método RKE4 em relação a ordem e a estabilidade.
 4. Reproduziu-se os resultados numéricos apresentados no trabalho [15] usando RKE4 com absoluta concordância. Derivando as consequências físicas razoáveis.
 5. Aprendeu-se a manejar o software MATLAB e criou-se um pequeno manual introdutório ao MATLAB.
 6. Aprendeu-se a manejar o software LaTeX, anteriormente desconhecido pelo autor deste trabalho.

Palavras chaves: Interação Fluido-Estrutura, métodos numéricos, massa-mola, pressão, MATLAB, Runge-Kutta clássico de ordem 4.

Índice general

1	Introducción	1
1.1	Fluido-Estructura	1
1.2	Métodos Numéricos	2
2	Modelo FSI	6
2.1	Parte estructura: Movimiento armónico	9
2.2	Parte del fluido	14
3	Modelo Numérico	18
3.1	Métodos de Runge-Kutta Explícito	18
3.1.1	Convergencia	24
3.1.2	Estudio de la estabilidad: 0-estabilidad	27
3.1.3	Estudio del error local	28
3.1.4	Extensión a sistemas	32
3.1.5	Estabilidad absoluta: A-estabilidad	33
3.2	Método de Newmark	36
3.2.1	A-estabilidad para Newmark	39
3.3	Método de Newmark vs RK orden 4	40
4	Resolución Numérica	41
4.1	Resultados prácticos	43
4.2	Aplicación práctica	48
5	Conclusión	52
A	Apéndice	54
A.1	Algunas Definiciones	54
A.2	Ecuaciones diferenciales ordinarias	55
A.3	Introducción a MATLAB	60
A.3.1	Aspectos básicos	61
A.3.2	Documentación y ayuda on-line	65

A.3.3	Script y funciones: El editor integrado	66
A.3.4	Matrices	68
A.3.5	Dibujo de curvas	70
A.3.6	Programación con MATLAB	74
A.3.7	Operaciones de lectura y escritura	76
A.3.8	Aproximación numérica	78
A.4	Explicación (código usado)	84
Bibliografía.		88

1 Introducción

1.1 Fluido-Estructura

Los problemas de FSI (FSI siglas en inglés para Fluid-Structure Interaction) desempeñan un papel prominente en muchos campos científicos y de ingeniería y una amplia gama de estos problemas sigue siendo un desafío debido a su fuerte carácter no lineal y multidisciplinario. Al hablar de FSI nos estamos refiriendo al acoplamiento entre leyes de diferentes medios físicos, en especial las de la dinámica del fluido y las de mecánica estructural. Este acoplamiento toma en consideración el campo de presión o térmico de un análisis de dinámica de fluidos computacionales (CFD Computational Fluid-Dynamic) y las consecuencias directas de esta carga en el análisis estructural.

Existen innumerables componentes que pueden ser estudiados bajo la óptica FSI, como las válvulas industriales, las tuberías que transportan fluido a alta temperatura, rotores de máquinas rotativas, turbinas eólicas, válvulas sanguíneas, antenas parabólicas, el rendimiento de una embarcación como el resultado de la interacción entre la hidrodinámica del agua y la dinámica estructural, sedimentación y ensamblaje de partículas, aerodinámica, turbulencia, flujos complejos en dominios irregulares, electro-hidrodinámica, flujos magneto-hidrodinámicos, biofluidos y biomecánicos (como la agregación y deformación celular, interacción con el corazón, dinámica del fluido del oído interno, natación de medusas, motilidad de los espermatozoides, etc.). Básicamente, la mayoría de los productos envuelve múltiples físicas y gracias a la explosión de la era tecnológica hoy en día este tipo de estudio es aún más viable y se utiliza ampliamente.

La matemática está fuertemente presente en la construcción de los modelos que describen los movimientos de fluido estructura. Una de las etapas del proceso de modelado consiste en expresar las leyes físicas que gobiernan la dinámica de los movimientos del fluido en términos de ecuaciones matemáticas, representadas básicamente por ecuaciones diferenciales. Necesitamos también aproximar las soluciones de estas ecuaciones, que en el caso de las ecuaciones completas sólo es posible a través de métodos de aproximación numérica, dada su complejidad. De hecho, el modelo más completo que describe la dinámica de un fluido (líquido o gaseoso)

so) son las llamadas ecuaciones de Navier-Stokes que gobiernan la dinámica de los fluidos (fluidos newtonianos) constituyen quizá el principal objetivo matemático del milenio, "que consiste en demostrar la existencia de una solución fuerte de esas ecuaciones", nombre dado en homenaje al físico e ingeniero francés Claude-Louis Navier (1785-1836) y al matemático y físico británico Geoge Gabriel Stokes (1819-1903). Estas ecuaciones son bastante complicadas y no son lineales. En el esfuerzo para obtener soluciones y analizar las propiedades, muchas simplificaciones diferentes se pueden hacer, y hay un número de ecuaciones menos complicadas que describen diferentes tipos de fluidos con comportamiento especial. Cualquier fluido tiene una cierta viscosidad, es decir, existe algún grado de resistencia interna a las fuerzas de flujo, por ejemplo: *el aceite tiene alta viscosidad y el agua tiene baja viscosidad, para el aire, la viscosidad es aún menor* y, a menudo, es una suposición tan precisa que ningún error se comete desde el punto de vista práctico. Y, en cierto sentido, simplifica considerablemente el análisis matemático/numérico. Un fluido es siempre compresible, lo que significa que la densidad del fluido cambia como resultado de variaciones de presión. Para gases, los efectos de la compresibilidad pueden ser considerables. Por otro lado, ciertos fluidos pueden ser casi incompresibles, como por ejemplo el agua en condiciones normales.

En este trabajo vamos a abordar de forma introductoria la interacción fluido-estructura (FSI), a través de un modelo mecánico presentado por Emmanuel Lefrançois y Jean-Paul Boufflet (*An Introduction to Fluid-Structure Interaction: Application to the Piston Problem*)[15] en ecuación diferencial ordinaria de compresión de un gas por un péndulo cuyo movimiento depende del muelle acoplado a sí, cuya la ecuación general resulta de la aplicación del principio fundamental de la dinámica, (la segunda ley de Newton), para sistemas mecánicos (1.1), donde \vec{F} son las fuerzas aplicadas (gravedad, aerodinámicas), m la masa y \vec{a} el vector aceleración.

$$\sum_{\text{fuerzas}} \vec{F} = m\vec{a} \tag{1.1}$$

1.2 Métodos Numéricos

Históricamente, los modelos numéricos se utilizaron por primera vez para estudiar el comportamiento elástico de las estructuras flexibles, entretanto la dinámica de fluidos es un área que ha impulsado el desarrollo de métodos numéricos, probablemente más que cualquier otra área. Las simulaciones de computadora ya se hacían en 1940, pero más de 60 años después, todavía hay muchos retos que están fuera de alcance, incluso con los más potentes computadores que están disponibles hoy. Los modelos matemáticos se conocen desde hace más de 150 años, pero todavía faltan

métodos computacionales suficientemente eficientes, porque los cálculos de flujo de fluidos son más complejos, debido a los efectos de convección y turbulencia, por ejemplo. En los problemas computacionales se pueden utilizar diferentes niveles de aproximación, según el nivel de precisión requerido. En todos los casos, es necesario tener cuidado al acoplar la estructura y los solucionadores de fluidos. La conservación de la masa, el impulso y la energía debe respetarse entre los términos de (1.1); este no es automáticamente el caso cuando se utilizan distintos métodos numéricos para aproximar el término izquierdo y derecho. Estos criterios proporcionan la base principal para verificar la calidad de los cálculos del FSI.

Sin embargo para la mayoría de los problemas de FSI, las soluciones analíticas para las ecuaciones del modelo son imposibles de obtener, mientras que los experimentos de laboratorio tienen un alcance limitado; por lo tanto, para investigar la física fundamental involucrada en la interacción compleja entre fluidos y sólidos, se pueden emplear simulaciones numéricas. Los procedimientos numéricos para resolver los problemas de FSI pueden clasificarse en general en dos enfoques: el enfoque monolítico y el enfoque dividido, ver por ejemplo (G. Hou, J. Wang and A. Layton)[14]. El enfoque monolítico trata la dinámica de fluidos y estructuras en el mismo marco matemático para formar una única ecuación de sistema para todo el problema. Este problema se resuelve simultáneamente mediante un algoritmo unificado. La interfaz está implícita en el procedimiento de soluciones. Este enfoque puede potencialmente lograr una mayor precisión para un problema multidisciplinario, pero puede requerir substancialmente más recursos y experiencia para desarrollar y mantener un código tan especializado. En el enfoque dividido se trata el fluido y la estructura como dos campos computacionales que pueden resolverse por separado con su respectiva discretización de malla y algoritmo numérico. Las condiciones interfaciales se utilizan explícitamente para comunicar información entre las soluciones de fluidos y estructuras. Un aspecto positivo de este enfoque es integrar algoritmos disciplinarios disponibles (es decir, fluidicos y estructurales) y reducir el tiempo de desarrollo del código aprovechando los códigos heredados o algoritmos numéricos que han sido validados y utilizados para resolver muchos problemas complicados de fluidos o estructurales. Otra clasificación general de los procedimientos de la solución FSI se basa en el tratamiento de las mallas: los métodos de *malla conformes* y los métodos de *malla no conformes* [14]. Los métodos de malla conformes consideran las condiciones de la interfaz como condiciones de límites físicos, que tratan la ubicación de la interfaz como parte de la solución y requieren mallas que se ajustan a la interfaz. Debido al movimiento y/o deformación de la estructura sólida, es necesario volver a redefinir (o actualizar la malla) a medida que la solución avanza. Por otro lado, los métodos de malla no conforme tratan las ubicaciones de los límites y las condiciones de la interfaz relacionadas como restricciones impuestas en las ecuaciones del modelo, de modo que se pueden emplear mallas no conformes. Como resultado, las ecuaciones fluida y de estructura

pueden ser convenientemente resueltas independientemente unas de otras con sus respectivas mallas, y el remallado no es necesaria. La distinción entre estos dos tipos de mallas se puede observar en la Figura 1.1 , donde un cuerpo sólido (una esfera) se está moviendo en un dominio fluido.

En este trabajo presentamos un modelo básico en una dimensión (1D) [15], que resolveremos utilizando Runge-kutta explícito de orden 4, sin la necesidad de recurrir al estudio de las mallas, pero buscando en todo momento respetar las condiciones físicas , relacionadas con el fluido que se introduce en la sección "modelo físico" y con ello transmitir la idea básica, de lo que se debe tener en cuenta para la resolución matemática de un problema de fluido estructura, aplicando conceptos fundamentales básicos de la física de fluidos, como la ley clásica de gases en proceso adiabático.

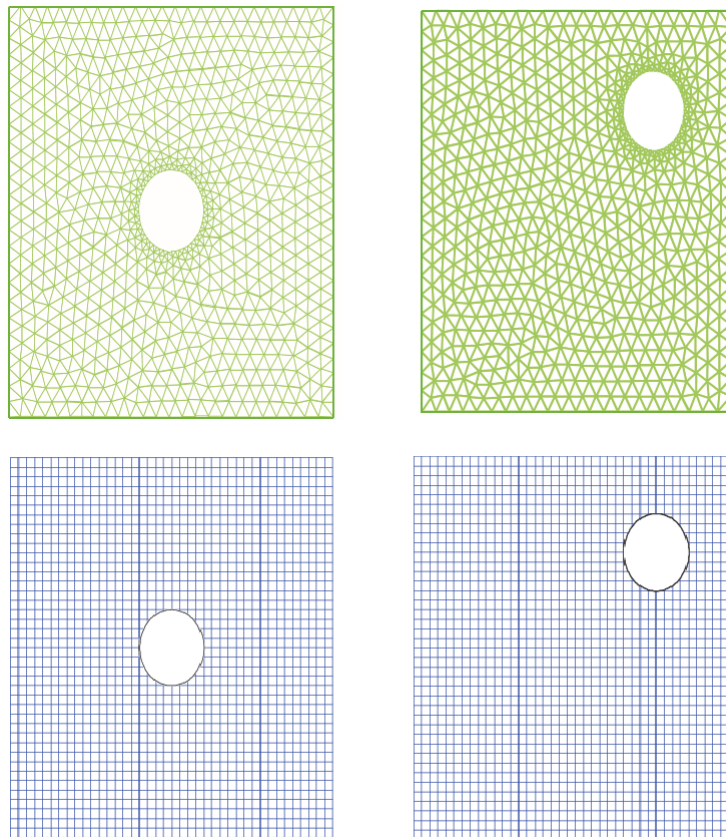


Figura 1.1: Ejemplo de malla conforme (arriba) y malla no conforme (abajo)

Este trabajo se compone de cuatro capítulos, el primer capítulo presenta el modelo de FSI donde está incluido la descripción del modelo para la estructura y los conceptos básicos para la física del fluido; el segundo capítulo presenta de forma

detallada el método de Runge-Kutta explícito, así como un resumen del método de Newmark y una pequeña comparación entre el método de Runge-Kutta y el método de Newmark; el tercer capítulo presenta los resultados numéricos obtenidos con el método de Runge-Kutta de orden 4, mediante el software MATLAB, así como su aplicación práctica utilizando la medida adimensional de Deborah [20]; el cuarto capítulo se presenta la conclusión concerniente a los resultados obtenidos en este trabajo en comparación con los obtenidos por [15] con el método de Newmark de orden 2. En el apéndice constan algunas definiciones a ser consideradas en el estudio de métodos numéricos para ecuaciones diferenciales. También hemos considerado interesante construir un pequeño manual introductorio sobre MATLAB. Finalmente se presenta el código que hemos utilizado y lo explicamos.

2 Modelo FSI

Como se ilustra en la Figura (2.1) consideramos un gas (por ejemplo aire) contenido en una cámara 1D, cerrado en su lado derecho por un pistón en movimiento y en su izquierda una pared fija. El pistón es de masa m_p y está unido a un punto fijo externo con una muelle (de rigidez k_p). El muelle se define por tres longitudes diferentes, a saber, sin estirar (L_{so}), en reposo bajo presión (L_{se}) y en un dado momento t durante el proceso de interacción de la estructura del fluido (FSI) ($L_s(t)$). Desde una posición inicial el desplazamiento del pistón se describe por $u(t)$, la velocidad del pistón por $\dot{u}(t)$ y la aceleración del pistón por $\ddot{u}(t)$, con respecto a su posición inicial.

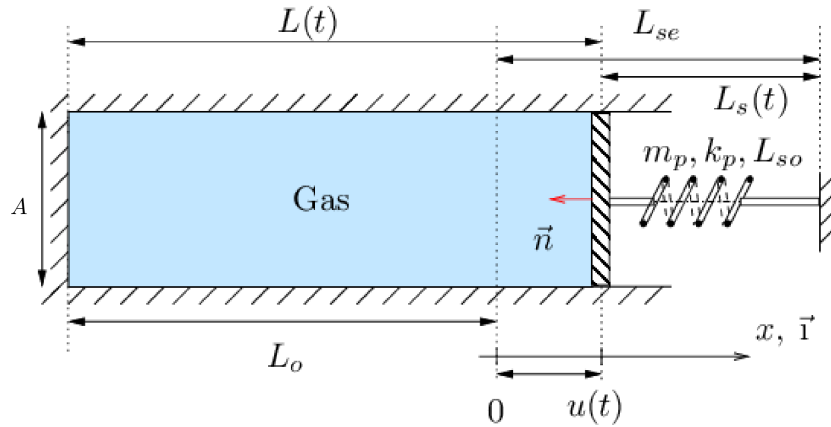


Figura 2.1: Un gas encerrado en una cámara con un pistón móvil.

Vamos a describir los parámetros que aparecen

1. si solo estuviese el muelle, ver Figura (2.2), y su posición de reposo fuese L_{so} , entonces cualquier desplazamiento representado por $L_s(t)$ generaría una fuerza dada por la ley de Hooke con valor $-k_p(L_s(t) - L_{so})$

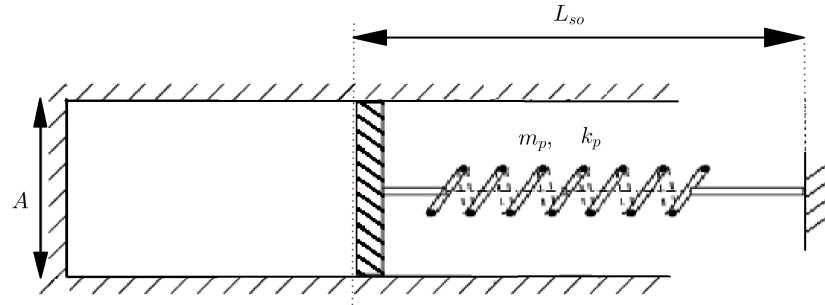


Figura 2.2: Reposo sin gas

- Al añadirse gas en la cámara, se genera una presión inicial p_o y una fuerza de fluido dada por Ap_o , donde A es la sección de la cámara como muestra la Figura (2.3). Se alcanza una posición de equilibrio cuando la fuerza de presión se equilibra con la fuerza de Hooke donde

$$k_p(L_{se} - L_{so}) + Ap_o = 0 \quad \text{donde } p_o = \text{la presión en reposo en la cámara}$$

$$- Ap_o = k_p(L_{se} - L_{so}) \quad (2.1)$$

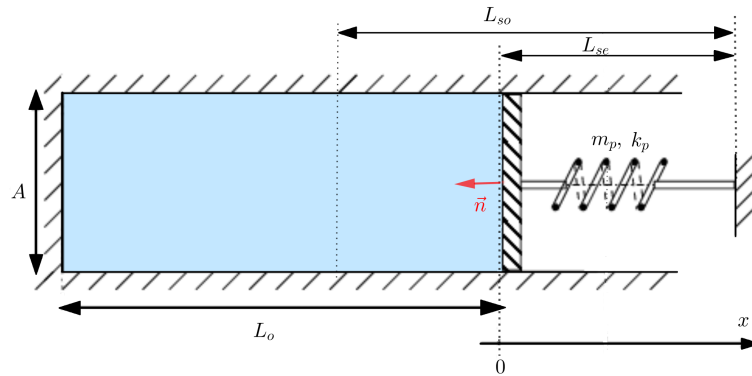


Figura 2.3: Reposo con gas

- A partir de la posición de equilibrio desplazamos el pistón hasta una posición inicial $u(0)$ donde comenzará el proceso FSI a partir del cual obtenemos una nueva posición del muelle $L_s(t) \neq L_{so}$, tal que $L_s(t) = L_{se} - u(t)$ la longitud actual como muestra la figura (2.1). A partir de (1.1) y por la ley de Hooke tenemos que

$$m_p \ddot{u}(t) = -k_p(L_s(t) - L_{so})\vec{n} \cdot \vec{1} + Ap(t) \quad (2.2)$$

donde A es la sección del pistón, $\vec{1}$ el vector de unidad en el eje x , y $\vec{n} = -\vec{1}$ el vector normal del pistón. La posición estática en reposo está definida por $u = \dot{u} = \ddot{u} = 0$ para $p = p_o$.

$$m_p \ddot{u}(t) + k_p u(t) = k_p (L_{se} - L_{so}) + Ap(t) \quad (2.3)$$

sustituyendo (2.1) en (2.3), obtenemos

$$m_p \ddot{u}(t) + k_p u(t) = A(p(t) - p_o) \quad (2.4)$$

que es la forma clásica del sistema masa-resorte forzado sin amortiguación, con las condiciones iniciales $u(0) = u^0$ y $\dot{u}(0) = 0$, pero la presión $p(t)$ ejercida sobre el pistón es desconocida, porque este término solo se puede calcular mediante un modelo de flujo de fluido que se debe acoplar al modelo de pistón, y para ello tres modelos de fluido fueron propuestos por [15]:

1. **Modelo A: Ley del gas ideal con condición de adiabaticidad** el gas en la cámara (aire), está regido por la ley del gas ideal con condiciones adiabáticas, no habiendo intercambios con el medio externo (y PV -constantes)

$$p(t)V(t)^\gamma = p_o V_o^\gamma \quad \text{con} \quad p(t) = \rho(t)\mathcal{R}T(t). \quad (2.5)$$

donde $V(t) = V_o + Au(t)$ es el volumen actual del gas, $T(t)$ es a temperatura, ρ es la masa volúmica del gas, \mathcal{R} es la constante individual del gas y $\gamma = 1.4$ es la relación del calor específico del gas. Los términos con un subíndice o se refieren a condiciones en reposo.

2. **Modelo B: Modelo de Analogía de Pistón.** Este modelo proporciona una relación analítica para la variación de la presión resultante del desplazamiento de un pistón en una cámara semiinfinita. La presión exacta ejercida sobre los pistones móviles viene dada por

$$p(t) = p_o \left(1 + \frac{\gamma - 1}{2} \left(\frac{|\dot{u}(t)| \cdot \vec{n}}{c_o} \right) \right)^{\frac{2\gamma}{\gamma - 1}}, \quad (2.6)$$

donde p_o y c_o son, respectivamente, la presión y la velocidad del sonido de las condiciones ambientales frente a la onda generada por el pistón y $\gamma = 1.4$ es la relación de calor específica del gas. Sin embargo, el modelo B solo se puede utilizar para calcular la presión en el pistón, y no en ningún otro punto de la cámara.

3. **Modelo C: Evolución del flujo de fluido compresible 1D** Este modelo se basa en un conjunto de tres ecuaciones acopladas que rigen la evolución

2.1. Parte estructura: Movimiento armónico

no estacionaria de un flujo 1D compresible. Para un dominio fijo de longitud constante L , estas ecuaciones corresponden a las leyes de conservación

$$\begin{aligned} \text{masa:} & \quad \frac{\partial \rho}{\partial t} + \frac{\partial \rho v}{\partial x} = 0, \\ \text{impulso:} & \quad \frac{\partial \rho v}{\partial t} + \frac{\partial(\rho v^2 + p)}{\partial x} = 0 \quad \text{paratodo } x \in [0, L], t \geq 0, \\ \text{energía total:} & \quad \frac{\partial \rho e}{\partial t} + \frac{\partial(\rho e + p)v}{\partial x} = 0. \end{aligned}$$

todas las variables son dependientes de (x, t) , pero esta notación se ha suprimido para mayor claridad. La energía volúmica total $e(x, t)$ viene dada por

$$e = C_v T + \frac{v^2}{2}, \quad \text{con } C_v = \frac{\mathcal{R}}{\gamma - 1}, \quad \gamma = 1.4, \quad \text{y } \mathcal{R} = 287 \text{ m}^2 \text{s}^{-2} \text{k}^{-1}.$$

C_v es la capacidad calorífica específica del gas en un proceso de volumen constante, y \mathcal{R} es la constante de gas individual. La presión local $p(x, t)$ está relacionada con la temperatura $T(x, t)$ de acuerdo con la ley del gas ideal:

$$p = \rho \mathcal{R} T = (\gamma - 1) \left(\rho e - \frac{1}{2} \rho v^2 \right).$$

no se consideran efectos viscosos.

Sin embargo en este trabajo vamos a analizar el modelo A y utilizar el modelo B para calcular la velocidad del gas en la cámara y la presión en el pistón.

Por lo tanto, nuestra ecuación para $u(t)$ y la presión es

$$\begin{cases} m_p \ddot{u}(t) + k_p u(t) = A(p_o \left(\frac{V_o}{V_o + Au(t)} \right)^\gamma - p_o) \\ u(0) = u_0 \quad \dot{u}(0) = 0 \end{cases} \quad (2.7)$$

2.1 Parte estructura: Movimiento armónico

Tenemos que (2.7) es una ecuación no lineal de segundo orden no homogénea para $u(t)$. Vamos a necesitar métodos numéricos para analizar el comportamiento del gas en la cámara en relación al movimiento del pistón. En todo caso vamos a recapitular el comportamiento de un movimiento armónico simple no amortiguado, o sea analizar la ecuación (2.4) sin la fuerza externa (en nuestro caso ejercida por la presión del gas en la cámara).

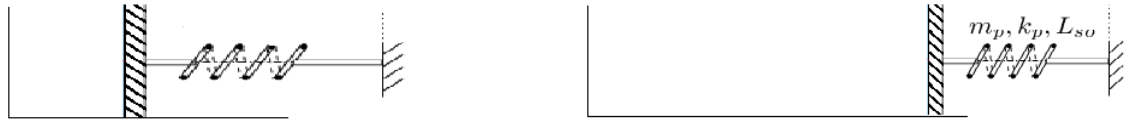


Figura 2.4: Resorte no amortiguado

Tenemos

$$m_p \ddot{u}(t) + k_p u(t) = 0 \quad (2.8)$$

y $m_p r^2 + k_p = 0$ es su ecuación característica. Siendo k_p una constante positiva cuya magnitud mide la rigidez del resorte tenemos que $r_{1,2} = \pm \sqrt{\frac{k_p}{m_p}} i$.

La solución general tiene la forma

$$u(t) = e^{\alpha t} (c_1 \cos \beta t + c_2 \sin \beta t), \quad \text{donde } r_{1,2} = \alpha \pm i\beta = 0 \pm \sqrt{\frac{k_p}{m_p}} i$$

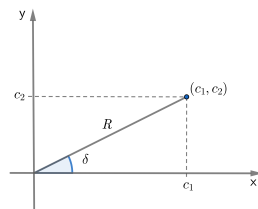
lo que se puede escribir como

$$u(t) = c_1 \cos \left(\sqrt{\frac{k_p}{m_p}} t \right) + c_2 \sin \left(\sqrt{\frac{k_p}{m_p}} t \right) \quad (2.9)$$

Sea $\omega_0 = \sqrt{\frac{k_p}{m_p}}$ entonces la ecuación (2.9) se puede escribir en términos de ω_0

$$u(t) = c_1 \cos \omega_0 t + c_2 \sin \omega_0 t \quad (2.10)$$

Marcando (c_1, c_2) en el plano y escribiendo en coordenadas polares tenemos que



$$\begin{cases} c_1 = R \cos \delta \\ c_2 = R \sin \delta \end{cases}$$

Sustituyendo los valores de c_1 y c_2 obtenemos

$$\begin{aligned} u(t) &= R \cos \delta \cos \omega_0 t + R \sin \delta \sin \omega_0 t \\ &= R (\cos \delta \cos(\omega_0 t) + \sin \delta \sin(\omega_0 t)) \\ &= R \cos(\omega_0 t - \delta) \end{aligned}$$

2.1. Parte estructura: Movimiento armónico

donde $f = \frac{1}{2\pi} \sqrt{\frac{k_p}{m_p}}$ es la frecuencia natural del sistema (o sea es el numero de ciclos por unidad de tiempo), δ es el ángulo de fase y $R = \sqrt{c_1^2 + c_2^2}$ es la amplitud. En este caso la solución de la ecuación es periódica de período $T = f^{-1}$ (T es el tiempo requerido para completar un ciclo).

Observación 1. Si crece la razón k_p/m_p , entonces crece f .

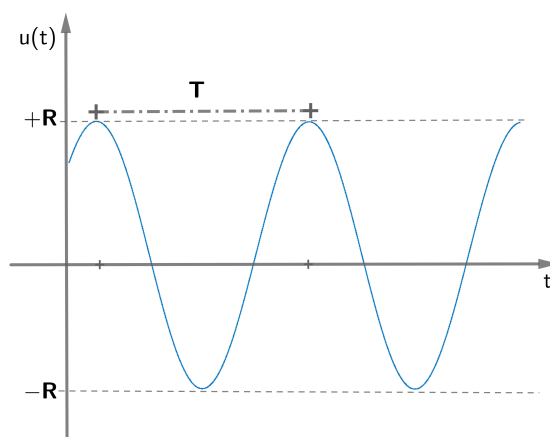


Figura 2.5: Gráfico del sistema masa-resorte no amortiguada

Supongamos ahora que a ese resorte añadimos una fuerza externa, pero periódica del tipo $F_0 \cos(\omega t)$ con $\omega > 0$ se aplica al pistón, en este caso la ecuación (2.8) para el movimiento del resorte es

$$m_p \ddot{u} + k_p u(t) = F_0 \cos(\omega t) \quad (2.11)$$

Como es una ecuación diferencial de segundo orden no homogénea su solución será de la forma

$$u(t) = u_h + u_p$$

donde $u_h(t)$ es la solución homogénea (2.10) y $u_p(t)$ es la solución particular que vamos a determinar, usando el método de los coeficientes indeterminados,

$$u_p(t) = t^s [A \cos(\omega t) + \sin(\omega t)]$$

es una solución particular y s es el menor entero no negativo que garantice que ninguno de los términos de $u_p(t)$ sea solución de la ecuación homogénea correspondiente y A y B son coeficientes a ser determinados reemplazando $u_p(t)$ en la ecuación diferencial. Vamos a considerar lo siguiente:

- (a) Recordemos que $\omega_0 = \sqrt{\frac{k_p}{m_p}}$, entonces si $\omega_0 \neq \omega$, en este caso $s = 0$ ya que ninguno de los elementos de $u_p(t)$ es la solución de la ecuación homogénea, entonces la ecuación particular es de la forma

$$u_p = A \cos(\omega t) + B \sin(\omega t)$$

substituyendo en la ecuación (2.11) obtenemos

$$A = \frac{F_0}{m_p(\omega_0^2 - \omega^2)} \quad \text{y} \quad B = 0$$

luego

$$u_p = \left[\frac{F_0}{m_p(\omega_0^2 - \omega^2)} \right] \cos(\omega t)$$

y la solución general de (2.11) es

$$u(t) = c_1 \cos(\omega_0 t) + c_2 \sin(\omega_0 t) + \left[\frac{F_0}{m_p(\omega_0^2 - \omega^2)} \right] \cos(\omega t) \quad (2.12)$$

En este caso la solución es oscilatoria y limitada. El movimiento resultante es, en general, la suma de dos movimientos periódicos de frecuencias (ω_0 y ω) y amplitudes diferentes. Si en principio consideramos que la masa está inicialmente en reposo, de modo que las condiciones iniciales sean $u(0) = 0$ y $\dot{u}(0) = 0$. Entonces la energía que alimenta el sistema viene completamente de la fuerza externa, sin aporte de las condiciones iniciales. En este caso las constantes c_1 y c_2 en la ecuación (2.12) son iguales a:

$$c_1 = -\frac{F_0}{m_p(\omega_0^2 - \omega^2)}, \quad \text{y} \quad c_2 = 0$$

y, por consiguiente,

$$u(t) = \frac{F_0}{m_p(\omega_0^2 - \omega^2)} [\cos(\omega t) - \cos(\omega_0 t)] \quad (2.13)$$

que es la suma de dos funciones periódicas con períodos diferentes pero con la misma amplitud. Usando las identidades trigonométricas para $\cos(A \pm B)$ con $A = (\omega_0 + \omega)t/2$ y $B = (\omega_0 - \omega)t/2$, entonces tenemos

$$u(t) = \left[\frac{F_0}{m_p(\omega_0^2 - \omega^2)} \sin\left(\frac{(\omega_0 - \omega)t}{2}\right) \right] \sin\left(\frac{(\omega_0 + \omega)t}{2}\right) \quad (2.14)$$

Si $|\omega_0 - \omega|$ es pequeño, entonces $\omega_0 + \omega$ es mucho mayor que $|\omega_0 - \omega|$, luego $\sin((\omega_0 + \omega)t/2)$ es una función que oscila mucho más rápidamente de que

2.1. Parte estructura: Movimiento armónico

$\sin((\omega_0 - \omega)t/2)$, entonces el movimiento consiste en una oscilación rápida con frecuencia $(\omega_0 + \omega)t/2$ pero con una amplitud senoidal que varía despacio e igual a

$$\frac{F_0}{m_p|\omega_0^2 - \omega^2|} \left| \sin\left(\frac{(\omega_0 - \omega)t}{2}\right) \right|$$

Este tipo de movimiento que tiene variación periódica de amplitud, muestra lo que se conoce como una **pulsación**.

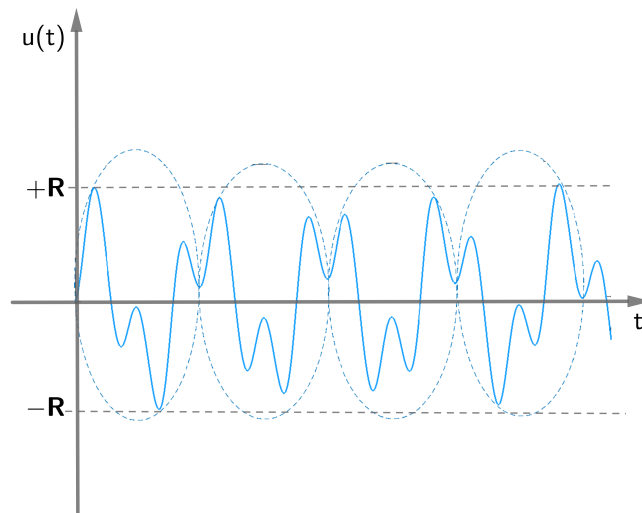


Figura 2.6: Pulsación: variación periódica de amplitud

- (b) $\omega_0 = \omega$, en este caso $s = 1$ ya que si $s = 0$ los elementos de $u_p(t)$ son solución de la ecuación homogénea, entonces la ecuación particular es de la forma

$$u_p = t[A \cos(\omega t) + B \sin(\omega t)]$$

substituyendo en la ecuación (2.11) obtenemos

$$A = 0 \quad \text{y} \quad B = \frac{F_0}{2m_p\omega_0}$$

luego

$$u_p = \left(\frac{F_0}{2m_p\omega_0}\right) t \sin(\omega_0 t)$$

y la solución general de (2.11) es

$$u(t) = c_1 \cos(\omega_0 t) + c_2 \sin(\omega_0 t) + \left(\frac{F_0}{2m_p\omega_0}\right) t \sin(\omega_0 t). \quad (2.15)$$

En este caso debido al término $t \sin(\omega_0 t)$, la solución prevé que el movimiento se vuelve ilimitado cuando $t \rightarrow \infty$ independientemente de los valores de c_1 y de c_2 . Por supuesto, en el mundo real, las oscilaciones ilimitadas no pueden ocurrir. Cuando $u(t)$ se vuelve muy grande, el modelo matemático en el cual la ecuación (2.11) se basa no es más válido, ya que la hipótesis de que la fuerza del muelle depende linealmente del desplazamiento precisa que $u(t)$ sea pequeño. Este fenómeno se conoce como **resonancia** (la frecuencia de forzamiento es igual a la frecuencia natural del sistema) y la frecuencia $\omega = \omega_0$ se llama **frecuencia de resonancia**.

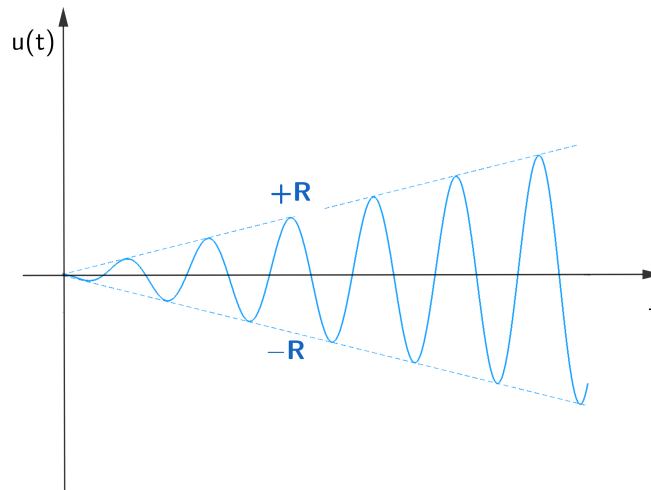


Figura 2.7: Resonancia

2.2 Parte del fluido

Definición 2.2.1. *fluidos* son todas las sustancias que se deforman continuamente cuando están sometidas a una tensión de cizallamiento o tensión tangencial ¹, no importando cuánto pequeña puede ser esa tensión (ejemplo de fluidos: gases, líquidos, plasmas y también los sólidos plásticos).

Las propiedades comunes de los fluidos son que toman casi inmediatamente la forma del recipiente sólido donde se les introduce y se une a si mismos casi instantáneamente (la *fluidomecánica*).

¹Es la tensión generada por fuerzas aplicadas en sentidos iguales u opuestos, en direcciones similares, pero con intensidades diferentes en el material analizado.

2.2. Parte del fluido

Teniendo en cuenta que el gas (aire) contenido en la cámara de la figura (2.1) es un **gas ideal** o **perfecto**, que por definición es un **fluido no viscoso**, no hay desplazamiento relativo entre sus moléculas estando en reposo o en movimiento y las ecuaciones de estado son dados por:

$$PV = \mathcal{R}T, \quad P = \rho\mathcal{R}T, \quad PV_m = \mathcal{R}T. \quad PV = m\mathcal{R}T \quad \text{y} \quad PV = N\mathcal{R}T \quad (2.16)$$

A través de experiencias hechas en expansión libre adiabática de un gas a baja presión, Joule observó que, a bajas presiones, la temperatura del gas no cambiaba. Era una expansión libre ($W = 0$, expansión contra el vacío) adiabática ($Q = 0$) irreversible (no estático) en sistema cerrado. Aplicando el principio de la termodinámica $\Delta U = Q - W = 0$, luego es un proceso isoenergético (la energía interna se mantiene constante). Es decir, en un gas a bajas presiones la energía interna no depende del volumen; por tanto, sólo es función de la temperatura, así como el calor específico a volumen constante (c_v), el calor específico a presión constante (c_p) y el cociente de los calor específicos (γ) son funciones de la temperatura.

$$\gamma = \frac{c_p}{c_v} \quad (2.17)$$

c_v y c_p se expresan en función de γ :

$$c_v = \frac{\mathcal{R}}{\gamma - 1} \quad c_p = \frac{\gamma\mathcal{R}}{\gamma - 1}. \quad (2.18)$$

Para un gas ideal perfecto su calor específico es constante, es decir, la diferencia de energía interna y de entalpía es proporcional a la diferencia de temperatura entre dos estados. Lo que supone una simplificación aún mayor del modelo para un gas ideal. Experimentalmente se observa que, a las temperaturas habituales de trabajo ($200K < T < 1000K$):

- **Gases monoatómicos**($He, Ne, Ar, etc.$): $c_v = 3R/2 \Rightarrow c_p = 5R/2 \Rightarrow \gamma = 5/3 = 1.667$
- **Gases biatómicos**($O_2, H_2, CO, aire, etc.$): $c_v = 5R/2 \Rightarrow c_p = 7R/2 \Rightarrow \gamma = 7/5 = 1.40$
- **Gases poliatómicos**(H_2O, CH_4, SO_2, etc): $k = 1.1 - 1.35$ (*variable*).

Para un gas ideal con calores específicos constantes, la energía interna y la entalpía son

$$\Delta U = mc_v\Delta T \quad \text{y} \quad \Delta h = mc_p\Delta T. \quad (2.19)$$

A partir de las ecuaciones de estado (2.16) y de las expresiones para calcular la energía interna y la entalpía del gas ideal se pueden obtener expresiones para

el trabajo y el calor de procesos cuasiestáticos en sistema cerrado. La expresión matemática que relaciona los estados termodinámicos intermedios de un proceso se denomina **ecuación de la línea de estados** (ELE). Es evidente que sólo está definida cuando los estados intermedios del proceso son estados de equilibrio, es decir, en procesos cuasiestáticos.

Para un sistema simple el trabajo en un proceso cuasiestático se puede calcular integrando

$$W = \int_1^2 PdW + W_d \quad (\text{con } W_d \leq 0) \quad (2.20)$$

Si el trabajo disipativo es nulo, el proceso será además reversible (cuasiestático sin disipación). Supondremos que el trabajo disipativo es nulo; en caso contrario habría que añadirlo al trabajo asociado al cambio de volumen para tener el trabajo total.

Tendremos que para ecuación de la línea de estados de un proceso adiabático $\delta Q = 0$, y la Primera Ley en forma diferencial es

$$dU = -\delta W \quad (2.21)$$

y para el proceso cuasiestático

$$mc_v dT = -PdV$$

Empleando las ecuaciones de estado térmica del gas ideal (2.16), se pueden obtener expresiones de la ELE de un proceso adiabático cuasiestático y sin disipación en gas ideal

$$\frac{P_2}{P_1} = \left(\frac{V_1}{V_2}\right)^\gamma \quad PV^\gamma = cte$$

y

$$\frac{T_2}{T_1} = \left(\frac{P_2}{P_1}\right)^{\frac{\gamma-1}{\gamma}} \quad \frac{P^{\frac{\gamma-1}{\gamma}}}{T} = cte.$$

Como el calor es igual a cero, el trabajo en un proceso adiabático es

$$W = -\Delta U. \quad (2.22)$$

Por tanto, a partir de las ecuaciones (2.20) y (2.16), es posible determinar el trabajo de un proceso adiabático para un gas ideal en función de los estados inicial y final:

$$W = \frac{mRT}{\gamma - 1} \left[1 - \left(\frac{P_2}{P_1}\right)^{\frac{\gamma-1}{\gamma}} \right]. \quad (2.23)$$

Sin embargo, durante el proceso FSI en la figura (2.1) se crían perturbaciones sobre el fluido, las perturbaciones creadas en el fluido por un cuerpo en movimiento

2.2. Parte del fluido

se propagan o se comunican a otras partes del fluido. El movimiento de las perturbaciones relativas al fluido se llama *movimiento de las ondas*, y la velocidad de propagación se llama *velocidad de la onda*.

Cuando se mueve el pistón dentro del fluido se produce una onda de compresión, y cuando el pistón se retira, se produce una onda de expansión. Si suponemos que el movimiento comienza en una onda de expansión el pistón comienza impulsivamente, con velocidad $|\dot{u}_p|$ la distribución de la velocidad de la partícula en el primer instante es un "paso". Pero en una onda de expansión, ese paso no puede ser mantenido; así que la onda comienza a propagarse, ella comienza a "aplanar". Posteriormente, t_1 la velocidad de la partícula tiene distribución lineal y la presión tiene una distribución correspondiente. La onda se propaga con velocidad c_1 (velocidad del sonido o velocidad acústica) al fluido no perturbado, es decir, en la dirección opuesta al pistón y al movimiento del fluido y las propiedades del fluido tienen el valor uniforme P_1, ρ_1, c_1 , que para un gas perfecto, se dan en términos de las condiciones

$$\frac{\rho_2}{\rho_1} = \left(1 - \frac{\gamma - 1}{2} \frac{|\dot{u}_p|}{c_1}\right)^{2/(\gamma-1)} \quad (2.24a)$$

$$\frac{P_2}{P_1} = \left(1 - \frac{\gamma - 1}{2} \frac{|\dot{u}_p|}{c_1}\right)^{2\gamma/(\gamma-1)}. \quad (2.24b)$$

La relación de presión P_2/P_1 define la fuerza de la onda de expansión.

3 Modelo Numérico

La simulación del movimiento del pistón se reduce a resolver esta ecuación

$$\begin{cases} m_p \ddot{u}(t) + k_p u(t) = A(p_o \left(\frac{V_o}{V_o + Au(t)} \right)^\gamma - p_o) \\ u(0) = u_0 \quad \dot{u}(0) = 0 \end{cases} \quad (3.1)$$

La solución de esta ecuación no se puede expresar en términos de funciones elementales, puesto que es una ecuación no lineal, por tanto tenemos que recurrir a un método numérico para aproximar la solución. En [15] se utilizó el método de Newmark que es de orden 2, de una convergencia cuadrática y en este trabajo vamos a utilizar el método de Runge-Kutta que tiene mejores propiedades de convergencia.

3.1 Métodos de Runge-Kutta Explícito

Los métodos de Runge-Kutta forman una familia importante de métodos iterativos implícitos y explícitos para la resolución numérica (aproximación) de soluciones de ecuaciones diferenciales ordinarias. Se pueden interpretar geoméricamente como una extensión del método de Euler progresivo. Ajustando mediante un promedio la mejor pendiente por la que avanzar desde un punto t al punto $t + h$. Tienen la característica de que se generan cálculos no lineales puesto que se anidan evaluaciones de la función pendiente.

Estos métodos fueron desarrollados en torno a 1900 por los matemáticos alemanes Carl David Tomé Runge y Martin Wilhelm Kutta. Conceptualmente, esta es una mejora muy notable puesto que:

- Tiene en cuenta el campo de soluciones y las distintas curvas vecinas a la buscada, no sólo la solución que se busca.
- Por otro lado, la no linealidad es una propiedad intrínseca del problema. Por lo que es muy razonable que aparezca en el método numérico.

3.1. Métodos de Runge-Kutta Explícito

La forma más general de un **método de Runge-Kutta** consiste en tomar S muestras de pendientes k_1, k_2, \dots, k_S y promediarlas para generar una nueva dirección de avance. En esta nueva dirección se da el paso definitivo.

Sea el problema de Cauchy

$$\begin{cases} y' = f(t, y), & t \in [t_0, t_f] \\ y(t_0) = y_0 \end{cases} \quad (3.2)$$

Así, una aproximación numérica de la solución (3.2) consiste en una sucesión de valores de la variable independiente:

$$t_0 < t_1 < \dots < t_n = t_f$$

y una sucesión de valores y_0, y_1, \dots, y_n , tales que

$$y_n \approx y(t_n), \quad n = 0, 1, \dots, n,$$

es decir, y_n es una aproximación del valor en t_n de la solución del problema (3.2).

$$y_n \approx y(t_n), \quad n = 0, 1, \dots$$

Tenemos que para el método de Runge-Kutta en general es

1. Dado $y_0 \approx y(t_0)$, para $0 \leq n \leq N - 1$
2. Construir las pendientes k_1, k_2, \dots, k_S y obtener la pendiente promedio

$$k = \sum_{i=1}^S b_i k_i$$

siendo $b_1 + b_2 + \dots + b_S = 1$.

3. Dar el paso definitivo en esta dirección promedio generada

$$y_{n+1} = y_n + hk.$$

Al número de muestra S obtenidas se le llama etapas del método, y la forma de construir los valores k_i sigue la misma idea de promediar pendientes.

Definición 3.1.1. (*Método explícito e implícito*) Un método se llama explícito si el valor y_{k+1} se da en términos de valores calculados previamente y_j , $j \leq k$. De lo contrario, se llama implícito.

Idea en la construcción

Supongamos que partimos del valor $k_1 = f(t_n, y_n)$. En la etapa $i = 2, 3, \dots, S$ construimos la pendiente k_i usando las pendientes k_1, k_2, \dots, k_{i-1} como sigue: Tomamos una pendiente promedio dada por

$$\tilde{k}_i = \sum_{j=1}^{i-1} \alpha_{i,j} k_j$$

donde tomamos $\alpha_{i,1} + \alpha_{i,2} + \dots + \alpha_{i,i-1} = 1$ arbitrarios. Por ejemplo, si $i = 2$ entonces simplemente $\tilde{k}_2 = k_1$ si $i = 3$ entonces $\tilde{k}_3 = \alpha_{3,1}k_1 + \alpha_{3,2}k_2$ para $\alpha_{3,1} + \alpha_{3,2} = 1$ arbitrarios, etc...

Entonces avanzamos por la dirección \tilde{k}_i y desde t_n una longitud $c_i h$, esto es, usamos la recta

$$r_i(x) = y_n + (x - t_n)\tilde{k}_i$$

y tomamos muestra en $x = t_n + c_i h$:

$$y^i = r_i(t_n + c_i h) = y_n + c_i h \tilde{k}_i = y_n + h \sum_{j=1}^{i-1} c_i \alpha_{i,j} k_j.$$

Usando la notación $a_{i,j} = \alpha_{i,j} c_i$, $j = 1, 2, 3, \dots, i-1$ lo podemos dejar como

$$y^i = y_n + h \sum_{j=1}^{i-1} a_{i,j} k_j$$

siendo evidente que

$$a_{i,1} + a_{i,2} + \dots + a_{i,i-1} = c_i(\alpha_{i,1} + \alpha_{i,2} + \dots + \alpha_{i,i-1}) = c_i, \quad i = 2, 3, \dots, S-1$$

Entonces calculamos la pendiente en este nuevo punto

$$k_i = f(t_n + c_i h, y^i),$$

o lo que es lo mismo,

$$k_i = f(t_n + c_i h, y_n + h(a_{i,1}k_1 + a_{i,2}k_2 + \dots + a_{i,i-1}k_{i-1})), \quad i = 1, 2, \dots, S.$$

Tablero de Butcher

Los coeficientes $\{a_{i,j}\}$, $\{c_i\}$, $\{a_i\}$ determinan completamente el método de Runge-Kutta y normalmente se recolectan en lo que se conoce como **matriz o tablero de Butcher** introducido por John Butcher en los años 60 [5].

3.1. Métodos de Runge-Kutta Explícito

En el caso explícito el tablero queda como

$$\begin{array}{c|cccccc}
 0 & 0 & 0 & \cdots & 0 & 0 \\
 c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\
 c_3 & a_{3,1} & a_{3,2} & 0 & \cdots & 0 \\
 \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\
 c_S & a_{S,1} & a_{S,2} & \cdots & a_{S,S-1} & 0 \\
 \hline
 & b_1 & b_2 & \cdots & b_{S-1} & b_S
 \end{array}$$

es decir en la forma

$$\frac{c}{b} \left| \begin{array}{c} A \\ b \end{array} \right.$$

donde A es la matriz cuadrada $S \times S$ con ceros en la diagonal y en la parte superior a la diagonal.

Algunos ejemplos de métodos de tipo Runge-Kutta

1. **Método de Heun de orden 2:** Tomamos la pendiente inicial

$$k_1 = f(t_n, y_n)$$

y luego avanzamos en esa dirección hasta llegar al punto final del intervalo donde volvemos a tomar una nueva muestra para la pendiente:

$$k_2 = f(t_n + h, y_n + hk_1)$$

con los dos valores k_1 y k_2 terminamos haciendo un promedio

$$k = \frac{1}{2}k_1 + \frac{1}{2}k_2$$

siendo esta la dirección de avance definitiva

$$y_{n+1} = y_n + hk \quad \text{o solamente} \quad y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$$

y su tablero de Butcher es

$$\begin{array}{c|cc}
 0 & 0 & 0 \\
 1 & 1 & 0 \\
 \hline
 & 1/2 & 1/2
 \end{array}$$

2. **Regla del punto medio:** Tomamos la pendiente inicial

$$k_1 = f(t_n, y_n)$$

y luego avanzamos en esa dirección hasta llegar al punto medio del intervalo donde volvemos a evaluar la pendiente como muestra

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right).$$

Con los dos valores k_1 y k_2 terminamos haciendo un promedio que simplemente consiste en usar el valor k_2

$$k = 0k_1 + 1k_2 = k_2$$

siendo esta la dirección de avance definitiva

$$y_{n+1} = y_n + hk.$$

También se puede interpretar como que usamos el punto medio del intervalo para aproximar: tomamos

$$y_* = y_n + \frac{h}{2}f(t_n, y_n)$$

y escribimos

$$y_{n+1} = y_n + hf\left(t_n + \frac{h}{2}, y_*\right).$$

y su tablero Butcher es

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ \hline & 0 & 1 \end{array}$$

Este método es de orden 2 con respecto a h .

3. **Método de Heun de orden 3:** Se avanza desde t_n hasta $t_n + h/3$ con la pendiente k_1 y se genera k_2 . Luego desde t_n hasta $t_n + 2h/3$ con la pendiente k_2 y se genera k_3 . Finalmente se toma el promedio para avanzar definitivamente a $t_n + h$

$$\begin{aligned} k_1 &= f(t_n, y_n) && \text{(primera pendiente)} \\ k_2 &= f\left(t_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1\right) && \text{(segunda pendiente)} \\ k_3 &= f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_2\right) && \text{(tercera pendiente)} \\ y_{n+1} &= y_n + h\left(\frac{1}{4}k_1 + \frac{3}{4}k_3\right) && \text{(pendiente promedio)} \end{aligned}$$

3.1. Métodos de Runge-Kutta Explícito

su tablero de Butcher es

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 \\ 2/3 & 0 & 2/3 & 0 \\ \hline & 1/4 & 0 & 3/4 \end{array}$$

Este método es de orden 3 con respecto a h

4. **Runge-Kutta clásico de orden 4:** Se avanza desde t_n hasta $t_n + h/2$ con la pendiente k_1 y desde t_n también hasta el mismo punto $t_n + h/2$ pero ahora con la pendiente k_2 . Luego avanzamos hasta $t_n + h$ con k_3 y finalmente se toma el promedio para avanzar definitivamente a $t_n + h$

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

su tablero de Butcher es

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 2/6 & 2/6 & 1/6 \end{array}$$

Este método es de orden 4 con respecto a h , y su popularidad proviene de la era pre-computacional (antes de los años 50) puesto que los coeficientes son sencillos y además tiene orden 4. Esto fue decisivo para su extensión como el más usado ya que los cálculos se hacían a mano.

Estos esquemas son los métodos clásicos dentro de la familia general de métodos de Runge-Kutta. Como se avanza desde t_n a t_{n+1} se dice que son **métodos de un paso**. Siempre son **métodos de un paso explícitos** con respecto al valor buscado y_{n+1} . Pueden no serlo con respecto a las pendientes k_1, k_2, \dots, k_S .

3.1.1 Convergencia

Para poder demostrar la convergencia de estos métodos vamos a introducir las nociones fundamentales de consistencia y estabilidad.

Definimos el **error de truncatura local en** $(t, y(t))$ como el error que se comete al aproximar la ecuación diferencial por el esquema en diferencias cuando se supone que se aplica en el valor exacto $(t, y(t))$. Esto es

$$\tau(y(t); h) = \frac{y(t+h) - y(t)}{h} - \sum_{i=1}^S b_i k_i(t, y(t); h).$$

Tomemos el **error de truncatura global sobre** $y(t)$ como

$$\tau(y; h) = \max_t |\tau(y(t); h)|.$$

Definición 3.1.2. Consistencia: *Un método de aproximación se dice consistente con la ecuación diferencial si el error de truncatura global $\tau(y; h)$ tiende a cero a medida que el paso de tiempo h tiende a cero. Si $\tau(y; h) = \mathcal{O}(h^p)$ entonces se dice que es consistente con orden p*

Teorema 3.1.1. *Todos los métodos de Runge Kutta explícitos son consistentes con orden de consistencia $p \geq 1$.*

Demostración. Gracias a la restricción

$$b_1 + b_2 + \dots + b_S = 1,$$

es fácil ver que el error local o de consistencia es siempre al menos $O(h^2)$. Usando el desarrollo de Taylor en torno al punto $(t, y(t))$

$$k_i = f(t + c_i h, y(t) + h(a_{i1}k_1 + \dots + a_{iS}k_S)) = f(t, y(t)) + O(h) = y'(t) + O(h)$$

luego

$$\begin{aligned} h\tau(y(t); h) &= y(t+h) - y(t) - h \sum_{i=1}^S b_i k_i(t, y(t); h) \\ &= hy'(t) + O(h^2) - h \sum_{i=1}^S b_i y'(t) - O(h^2) \\ &= hy'(t) - hy'(t) + O(h^2) = O(h^2). \end{aligned}$$

luego

$$\tau(y; h) = O(h).$$

□

3.1. Métodos de Runge-Kutta Explícito

Definición 3.1.3. Decimos que el **método es 0-estable** (leer cero-estable) cuando fijado el intervalo de cálculo $[t_0, t_0 + T]$ y particiones de talla h con $h = T/N$, si la perturbación cumple $|\delta_n^{(h)}| \leq \epsilon$ para $\epsilon > 0$, entonces existe una constante $C(T, f)$ y $h_0 > 0$ tal que para todo $h < h_0$ se cumple

$$|z_n^{(h)} - y_n^{(h)}| \leq C\{\epsilon + |z_0 - y_0|\}, \quad 1 \leq n \leq N = T/h.$$

donde

$$z_{n+1} = z_n + h \left[\sum_{i=1}^S b_i k_i(t_n, z_n; h) + \delta_n^{(h)} \right], \quad 0 \leq n \leq N^{(h)} - 1$$

$$y_{n+1} = y_n + h \sum_{i=1}^S b_i k_i(t_n, y_n; h), \quad 0 \leq n \leq N^{(h)} - 1.$$

Pongamos

$$\Phi_f(t_n, y_n; h) = \sum_{i=1}^S b_i k_i(t_n, y_n; h)$$

Tener 0-estabilidad está garantizado si Φ_f es Lipschitz con respecto al segundo argumento.

Teorema 3.1.2. Para un esquema numérico de la forma

$$y_{n+1} = u_n + h\Phi_f(t_n, y_n; h)$$

si Φ_f es Lipschitz con respecto al segundo argumento entonces es 0-estable.

Demostración. Ponemos $w_n^{(h)} = z_n^{(h)} - y_n^{(h)}$ para $n = 0, 1, \dots, N^{(h)}$. Entonces

$$w_{n+1}^{(h)} = w_n^{(h)} + h\{\Phi_f(t_n, z_n^{(h)}; h) - \Phi_f(t_n, y_n^{(h)}; h)\} + h\delta_{n+1}^{(h)}.$$

Supongamos que $|\delta_{n+1}^{(h)}| \leq \epsilon$, entonces aplicando la propiedad de Lipschitz que cumple Φ_f tenemos

$$|w_{n+1}^{(h)}| \leq |w_n^{(h)}| + hM_{\Phi_f}|w_n^{(h)}| + h\epsilon = (1 + hM_{\Phi_f})|w_n^{(h)}| + h\epsilon, \quad n \geq 0.$$

Una recursión nos da

$$\begin{aligned} |w_{n+1}^{(h)}| &\leq (1 + hM_{\Phi_f})|w_n^{(h)}| + h\epsilon \leq (1 + hM_{\Phi_f})^2|w_{n-1}^{(h)}| + (1 + hM_{\Phi_f})h\epsilon + h\epsilon \\ &\leq (1 + hM_{\Phi_f})^3|w_{n-2}^{(h)}| + (1 + hM_{\Phi_f})^2h\epsilon + (1 + hM_{\Phi_f})h\epsilon + h\epsilon \\ &\vdots \\ &\leq (1 + hM_{\Phi_f})^{n+1}|w_0^{(h)}| + h\epsilon \sum_{i=0}^n (1 + hM_{\Phi_f})^i \end{aligned}$$

usando ahora que $1 + hM_{\Phi_f} \leq e^{hM_{\Phi_f}}$ y la expresión para una suma geométrica de razón $1 + hM_{\Phi_f}$ obtenemos que

$$\begin{aligned} |w_{n+1}^{(h)}| &\leq e^{h(n+1)M_{\Phi_f}} |w_0^{(h)}| + h\epsilon \frac{(1 + hM_{\Phi_f})^{n+1} - 1}{1 + hM_{\Phi_f} - 1} \\ &\leq e^{TM_{\Phi_f}} |w_0^{(h)}| + \epsilon \frac{e^{TM_{\Phi_f}} - 1}{M_{\Phi_f}}, \quad n = 0, 1, 2, \dots, N - 1 \\ &\leq C(|w_0| + \epsilon) \end{aligned}$$

Si además también se tiene $|w_0^{(h)}| \leq \epsilon$ entonces para $n = 0, 1, 2, \dots, N - 1$

$$|w_{n+1}^{(h)}| \leq C\epsilon$$

siendo $C = e^{TM_{\Phi_f}} + (e^{TM_{\Phi_f}} - 1)M_{\Phi_f}^{-1}$ y ϵ el parámetro que acota a las perturbaciones y al error inicial. □

Finalmente, podemos concluir con el resultado principal de convergencia

Teorema 3.1.3. *Todos los métodos de Runge-Kutta explícitos son convergentes con orden al menos uno. Además, si un método de Runge-Kutta es consistente con orden $\mathcal{O}(h^p)$ entonces converge con orden p siempre que el error inicial también sea $\mathcal{O}(h^p)$.*

Demostración. Sabemos que $\Phi_f(t, y; h) = \sum_j b_j k_j$ y veremos después que cumple la condición de ser Lipschitz global para $h < h_0$. Tenemos también un error local de truncatura al menos $\mathcal{O}(h)$ (podremos tener ordenes mayor dependiendo de la elección del número de etapas S y de los coeficientes). Consideremos la solución exacta dentro del esquema numérico

$$y(t_{n+1}) = y(t_n) + h \left[\sum_{i=1}^S b_i k_i(t_n, y(t_n); h) + \tau(y(t_n); h) \right], \quad 0 \leq n \leq N^{(h)} - 1.$$

junto con el esquema de cálculo

$$y_{n+1} = y_n + h \left[\sum_{i=1}^S b_i k_i(t_n, y_n; h) \right], \quad 0 \leq n \leq N^{(h)} - 1.$$

Entonces, el error global $e_n = y(t_n) - y_n$ cumple

$$e_{n+1} = e_n + h\{\Phi_f(t_n, y(t_n); h) - \Phi_f(t_n, y_n; h)\} + h\tau(y(t_n); h).$$

3.1. Métodos de Runge-Kutta Explícito

Aplicando la propiedad de Lipschitz que cumple Φ_f tenemos

$$|e_{n+1}| \leq |e_n| + hM_{\Phi_f}|e_n| + h\tau(y; h) = (1 + hM_{\Phi_f})|e_n| + h\tau(y; h), \quad n \geq 0.$$

Una recursión nos da

$$\begin{aligned} |e_{n+1}| &\leq (1 + hM_{\Phi_f})|e_n| + h\tau(y; h) \leq (1 + hM_{\Phi_f})^2|e_{n-1}| + (1 + hM_{\Phi_f})h\tau(y; h) + h\tau(y; h) \\ &\leq (1 + hM_{\Phi_f})^3|e_{n-2}| + (1 + hM_{\Phi_f})^2h\tau(y; h) + (1 + hM_{\Phi_f})h\tau(y; h) + h\tau(y; h) \\ &\vdots \\ &\leq (1 + hM_{\Phi_f})^{n+1}|e_0| + h\tau(y; h) \sum_{i=0}^n (1 + hM_{\Phi_f})^i \end{aligned}$$

usando ahora que $1 + hM_{\Phi_f} \leq e^{hM_{\Phi_f}}$ y la expresión para una suma geométrica de razón $1 + hM_{\Phi_f}$ obtenemos que

$$\begin{aligned} |w_{n+1}^{(h)}| &\leq e^{h(n+1)M_{\Phi_f}}|w_0^{(h)}| + h\epsilon \frac{(1 + hM_{\Phi_f})^{n+1} - 1}{1 + hM_{\Phi_f} - 1} \\ &\leq e^{TM_{\Phi_f}}|w_0^{(h)}| + \epsilon \frac{e^{TM_{\Phi_f}} - 1}{M_{\Phi_f}}, \quad n = 0, 1, 2, \dots, N - 1 \\ &\leq C(|w_0| + \epsilon) \end{aligned}$$

Si además también se tiene $|w_0^{(h)}| \leq \epsilon$ entonces para $n = 0, 1, 2, \dots, N - 1$

$$|w_{n+1}^{(h)}| \leq C\epsilon$$

siendo $C = e^{TM_{\Phi_f}} + (e^{TM_{\Phi_f}} - 1)M_{\Phi_f}^{-1}$ y ϵ el parámetro que acota a las perturbaciones y al error inicial. \square

Una clasificación más cercana a la práctica la ofrece el concepto de **estabilidad absoluta** o **A-estabilidad**.

3.1.2 Estudio de la estabilidad: 0-estabilidad

Teorema 3.1.4. *Todos los métodos de Runge-Kutta explícitos son cero estables.*

Demostración. Para esto tenemos que garantizar que la expresión

$$\Phi_f(t, y; h) = \sum_{i=1}^S b_i k_i(t, y; h)$$

cumple una condición de Lipschitz con respecto a su segundo argumento de manera uniforme para $h < h_0$ si es preciso. Esto es, para todo $h < h_0$ (para algún $h_0 > 0$) se tiene M_{Φ_f} independiente de h tal que

$$|\Phi_f(t, z; h) - \Phi_f(t, y; h)| \leq M_{\Phi_f}|z - y|.$$

que se deduce de la condición de Lipschitz satisfecha por $f(t, y)$ de forma trivial.

Ejemplo 3.1.1. *Método de Heun, tomamos*

$$\Phi(t, y; h) = \frac{1}{2}\{f(t, y) + f(t + h, y + hf(t, y))\}$$

entonces, si $|f(t, y) - f(t, z)| \leq L_f|y - z|$

$$\begin{aligned} |\Phi(t, y; h) - \Phi(t, z; h)| &\leq \frac{1}{2}|f(t, y) - f(t, z)| \\ &+ \frac{1}{2}|f(t + h, y + hf(t, y)) - f(t + h, z + hf(t, z))| \\ &\leq \frac{L_f}{2}|y - z| + \frac{L_f}{2}|y + hf(t, y) - z - hf(t, z)| \\ &\leq L_f|y - z| + \frac{hL_f^2}{2}|y - z| = (L_f + \frac{hL_f^2}{2})|y - z|. \end{aligned}$$

Entonces, para cualquier $h_0 > 0$ fijamos $M_{\Phi_f} = L_f + h_0L_f^2/2$ y tenemos que $M_{\Phi_f} \geq L_f + hL_f^2/2$ por lo que

$$|\Phi(t, y; h) - \Phi(t, z; h)| \geq (L_f + \frac{hL_f^2}{2})|y - z| \leq M_{\Phi_f}|y - z|, \quad h \leq h_0.$$

□

Observación 2. *Las constantes que la estabilidad dependen de T y de M_{Φ_f} de manera exponencial. Por lo que pueden empeorar de manera muy importante. Este resultado cubre un gran espectro de situaciones, por lo tanto es razonable que la estimación sea muy general y tengamos las constantes $e^{TM_{\Phi_f}}$. Estas estimaciones podrían mejorar con información sobre las derivadas de Φ_f y el uso del **teorema del valor medio** pero desde el punto de vista práctico sólo interesa el de convergencia.*

3.1.3 Estudio del error local

La idea ahora es usar los parámetros y el número de etapas para aumentar el orden local o de consistencia del método.

Familias de métodos de acuerdo al orden

Métodos de Runge Kutta explícitos de una etapa En el caso $S = 1$ reduce la matriz de Butcher a la situación trivial

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

que representa el método de Euler explícito.

Métodos de Runge Kutta explícitos de dos etapas En este caso tenemos como matriz de Butcher

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c_2 & a_{2,1} & 0 \\ \hline & b_1 & b_2 \end{array}$$

con $c_2 = a_{2,1}$ y $b_1 + b_2 = 1$. Por lo tanto tenemos

$$\begin{aligned} k(t, y; h) &= b_1 k_1 + b_2 k_2, \\ k_1 &= k_1(t, y(t)) = f(t, y(t)) = y'(t), \\ k_2 &= k_1(t, y(t); h) = f(t + c_2 h, y(t) + h c_2 k_1). \end{aligned}$$

El error local resulta ser sobre una curva cualquier $y(t)$

$$l(y(t); h) = y(t + h) - y(t) - h b_1 y'(t) - h b_2 k_2$$

Usando el desarrollo de Taylor para $y(t + h)$ respecto a t obtenemos

$$y(t + h) - y(t) = h y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{3} y'''(t) + \mathcal{O}(h^4)$$

de donde

$$l(y(t); h) = h y'(t) + \frac{h^2}{2} y''(t) + \frac{h^3}{3} y'''(t) + \mathcal{O}(h^4) - h b_1 y'(t) - h b_2 k_2$$

Recordemos que desarrollo de Taylor de una función de dos variables se puede describir de forma simbólica como

$$f(t + h, y + k) = \sum_{n \geq 0} \frac{1}{n!} \{h \partial_t + k \partial_y\}^n f(t, y).$$

Vamos a desarrollar por Taylor $k_2 = f(t + h c_2, y(t) + h a_{2,1} f(t, y(t)))$ en el punto $(t, y(t))$. Abreviaremos usando $f = f(t, y(t))$, $\partial_t f = f_t$, $\partial_{tt} f = f_{tt}, \dots$ y también

que $c_2 = a_{2,1}$. Entonces tenemos

$$\begin{aligned} k_2 &= f(t, y(t)) + hc_2(f_t + y'(t)f_y) \\ &+ \frac{1}{2!}\{h^2c_2^2f_{tt} + 2hc_2ha_{21}ff_{ty} + h^2a_{21}^2f^2f_{yy}\} + \mathcal{O}(h^3) \\ &= y'(t) + hc_2y''(t) + \frac{h^2c_2^2}{2}(f_{tt} + 2ff_{ty} + f^2f_{yy}) + \mathcal{O}(h^3) \end{aligned}$$

usando que $y'(t) = f(t, y(t))$ y entonces $y''(t) = f_t + ff_y$. Por lo tanto, tenemos

$$\begin{aligned} l(y(t); h) &= hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{3!}y'''(t) + \mathcal{O}(h^4) \\ &- b_1hy'(t) - hb_2y'(t) - h^2b_2c_2y''(t) \\ &- \frac{h^3b_2c_2^2}{2}(f_{tt} + 2ff_{ty} + f^2f_{yy}) + \mathcal{O}(h^4) \end{aligned}$$

usando que $b_1 + b_2 = 1$ el término $hy'(t)$ se anula y si forzamos $c_2b_2 = 1/2$ el término $hy''(t)$ también y nos queda

$$l(y(t); h) = h^3 \overbrace{\left[\frac{1}{6}y'''(t) - \frac{c_2}{4}(f_{tt} + 2ff_{ty} + f^2f_{yy}) \right]}^{(*)} + \mathcal{O}(h^4) \quad h \rightarrow 0$$

luego tenemos

$$l(y(t); h) = \mathcal{O}(h^3) \quad h \rightarrow 0.$$

Esto es, para cualquier solución lo suficientemente regular se garantiza que el error global de este método decae como $\mathcal{O}(h^2)$ cuando $h \rightarrow 0$ si tomamos $y(t)$, $f(t, y(t))$ tal que $(*) \neq 0$, $l(y(t); h) = \mathcal{O}(h^3)$; evidentemente si tomamos $(*) = 0$ en este caso se mejora el orden. Luego el mayor orden posible con $S = 2$ del error local es 3, lo que nos da un error global 2.

Nos encontramos entonces con una completa familia de métodos explícitos de Runge-Kutta de orden 2 caracterizada por las restricciones sobre los parámetros

$$b_1 + b_2 = 1, \quad c_2b_2 = 1/2, \quad (c_2 = a_{2,1}).$$

entonces

$$\begin{aligned} c_2b_2 &= 1/2 \Rightarrow \text{orden 2} \\ c_2b_2 &\neq 1/2 \Rightarrow \text{orden 1} \end{aligned}$$

si usamos $\gamma = c_2$ (sabemos que $0 < \gamma \leq 1$ por ser γh la longitud que avanzamos dentro de $[t_n, t_{n+1}]$) encontramos que básicamente sólo hay un parámetro y el resto está en función de γ . Por ejemplo,

$$b_2 = 1/(2\gamma), \quad b_1 = 1 - 1/(2\gamma), \quad a_{2,1} = \gamma$$

3.1. Métodos de Runge-Kutta Explícito

Por lo tanto, aparece **una familia** infinita de métodos de Runge-Kutta explícitos de orden 2 que depende de un sólo parámetro. El tablero de Butcher es

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \gamma & \gamma & 0 \\ \hline & 1 - 1/(2\gamma) & 1/(2\gamma) \end{array}$$

Métodos de Runge Kutta explícitos de tres etapas Cuando $S = 3$ seguimos las mismas ideas aunque es más técnico. Para el tablero de Butcher

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 \\ c_3 & a_{31} & a_{32} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

las restricciones que se obtienen sobre los parámetros son

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, \\ b_2 c_2 + b_3 c_3 &= 1/2, \\ b_2 c_2^2 + b_3 c_3^2 &= 1/3, \\ b_3 c_2 a_{3,2} &= 1/6. \end{aligned}$$

Tenemos cuatro ecuaciones y seis incógnitas, por lo tanto aparecen **dos familias** infinitas de métodos de RK explícitos de orden 3. se puede observar también por los desarrollos de Taylor que este es el mayor orden posible con $S = 3$. Ejemplos típicos son el **método de Heun** de tercer orden

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1\right) \\ k_3 &= f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hk_1\right) \\ y_{n+1} &= y_n + h\left(\frac{1}{4}k_1 + \frac{3}{4}k_3\right). \end{aligned}$$

Métodos de Runge Kutta explícitos de cuatro etapas La técnica estándar para derivar estos métodos consiste en hacer desarrollos de Taylor y forzar el mayor número de coeficientes de potencias de h cero en el desarrollo de Taylor de $y(t+h)$ en torno a t . Un desarrollo similar nos lleva a las condiciones para obtener métodos de cuarto orden.

El ejemplo clásico de Runge-Kutta explícito de orden 4 con 4 etapas es

$$\begin{aligned}
 k_1 &= f(t_n, y_n) \\
 k_2 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\
 k_3 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\
 k_4 &= f(t_n + h, y_n + hk_3) \\
 y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
 \end{aligned}$$

y la matriz de Butcher en este caso es

$$\begin{array}{c|cccc}
 0 & 0 & 0 & 0 & 0 \\
 1/2 & 1/2 & 0 & 0 & 0 \\
 1/2 & 0 & 1/2 & 0 & 0 \\
 1 & 0 & 0 & 1 & 0 \\
 \hline
 & 1/6 & 2/6 & 2/6 & 1/6
 \end{array}$$

Tener coeficientes sencillos y alto orden fue decisivo para su extensión como el más usado en la era pre-computacional, esto es, cuando había que hacer los cálculos a mano.

3.1.4 Extensión a sistemas

Desde el punto de vista práctico la extensión de los esquemas de Runge-Kutta explícitos a sistemas es un aspecto trivial consistente en **repetir la ecuaciones para cada incógnita del sistema** de manera ordenada. En la obtención de estos métodos siempre se supuso además que el problema era escalar y que nada importante ocurre cuando pasamos al caso de sistemas.

La dificultad intrínseca en estos objetos matemáticos fue simplificada gracias a la introducción de la **Teoría de árboles** por parte de John C. Butcher en torno a 1960 [5]. Esta teoría relaciona la forma en la que se van construyendo las sucesivas derivadas con aspectos de la **teoría de grafos** que simplifican el estudio.

Se esperaba que cuando se obtuviese un método de orden p para un problema escalar, este orden se mantuviese en el caso vectorial. Pero de la Teoría de Butcher se deduce que no es así, siendo este un resultado que sorprendió a la comunidad científica en este campo cuando se dio a conocer.

Teorema 3.1.5. *Todo método de Runge-Kutta de orden $p = 1, 2, 3, 4$ en el caso $m = 1$ se extiende a un método del mismo orden en el caso de sistemas, $m > 1$.*

Teorema 3.1.6. *Consideremos las siguientes hipótesis:*

3.1. Métodos de Runge-Kutta Explícito

- **A** el método RK tiene orden p para $y'(t) = f(t, y(t))$ con $m > 1$.
- **B** el método RK tiene orden p para $y'(t) = f(t, y(t))$ con $m = 1$.
- **C** el método RK tiene orden p para $y'(t) = f(y(t))$ con $m = 1$.

Tenemos entonces

- Si $1 \leq p \leq 3$ entonces $\mathbf{A} \iff \mathbf{B} \iff \mathbf{C}$
- Si $p = 4$ entonces $\mathbf{A} \iff \mathbf{B} \Rightarrow \mathbf{C}$ pero $\mathbf{C} \not\Rightarrow \mathbf{B}$.
- Si $p \geq 5$ entonces $\mathbf{A} \Rightarrow \mathbf{B} \Rightarrow \mathbf{C}$ pero $\mathbf{C} \not\Rightarrow \mathbf{B}$ y $\mathbf{B} \not\Rightarrow \mathbf{A}$

Por ejemplo, si consideramos un problema de Cauchy de segundo orden

$$\begin{cases} y''(t) = f(t, y(t), y'(t)) & t \in (t, T] \\ y(t_0) = \alpha_0, \quad y'(t_0) = \beta_0 \end{cases} \quad (3.3)$$

Para aplicar cualquiera de los métodos de Runge-Kutta explícito aquí visto, sólo tendríamos que reducir la ecuación (3.3) a un sistema de ecuación diferencial de primer orden. Podemos hacer esto usando un par de incógnitas $(y(t), v(t))$, donde $v(t) = y'(t)$ y escribiendo entonces el problema como

$$\begin{cases} y'(t) = v(t) & t \in (t, T] \\ v'(t) = f(t, y(t), v(t)) \\ y(t_0) = \alpha_0, \quad v(t_0) = \beta_0 \end{cases} \quad (3.4)$$

Para su solución podríamos aplicar a cada ecuación individual un de los métodos de Runge-Kutta previamente introducidos para un problema escalar.

Observación 3. *De esta forma siempre podemos aproximar la solución de una ecuación diferencial de orden $m > 1$ recurriendo al sistema de m ecuaciones equivalente de primer orden, y luego aplicar a este sistema un método de discretización conveniente.*

3.1.5 Estabilidad absoluta: A-estabilidad

Los esquemas numéricos se clasifican no sólo por su orden de convergencia sino también sus propiedad de estabilidad.

Es evidente que la 0-estabilidad al ser un concepto que cuando se satisface sólo necesita $h \rightarrow 0$ no se preocupa de lo que ocurre para un valor de h fijo. La **estabilidad absoluta indica la restricción necesaria para reproducir un comportamiento cualitativo de la solución.**

Un campo continuo tiene comportamiento localmente distintos; para simplificar hacemos un estudio local del problema continuo y para ello linealizamos un campo cualquiera usando desarrollos de Taylor. Los puntos interesante a estudiar son los puntos de equilibrio. Entonce reemplazamos nuestro campo original por el aproximado y observamos como se comporta el esquema numérico en el problema aproximado, esto nos sirve para clasificar la estabilidad del esquema.

Supongamos tenemos el campo autónomo dado por $f(y)$ usando desarrollo de Taylor cerca de un punto de equilibrio $z_* = 0$ (para simplificar)

$$f(y) \sim -ay$$

donde $a = -f_y(0)$. Entonces, localmente, podemos trabajar con los problemas lineales

$$y'(t) = -ay(t), \quad t > 0$$

El concepto de estabilidad numérica es importante sólo en caso donde hay decaimiento de la solución o de parte de ella, y principalmente cuando este decaimiento es brusco (problemas rígidos). Por lo tanto vamos a fijar en el caso $a > 0$. Aquí sabemos que la solución exacta es $y(t) = e^{-at}$ y decae exponencialmente para $a > 0$: se dice que **es una edo estable**. En el caso $a < 0$ la solución crece exponencialmente: se dice que **es una edo inestable**.

El concepto de **A-estabilidad o estabilidad absoluta** nos permite clasificar los métodos determinando el intervalo de valores de h donde el método no se desestabiliza y converge para el problema modelo $y' = -\lambda y(t)$. Usamos

$$\begin{cases} y'(t) = -ay(t), & t > 0 \\ y(0) = 1 \end{cases}$$

y se observa cual es la restricción que el esquema numérico impone sobre h para poder reproducir el comportamiento $|y(t)| \rightarrow 0$ de la solución continua. Es decir, queremos asegurarnos de que si la solución continua decae a cero cuando t crece también lo hace la discreta.

Ejemplo 3.1.2. Recordemos que el **método de Heun de orden 2** es

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f(t_n + h, y_n + hk_1) \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2) \end{aligned}$$

La aplicación al **problema test** ($a > 0$)

$$\begin{cases} y'(t) = -ay(t), & 0 < t < T, \\ y(0) = 1 \end{cases}$$

3.1. Métodos de Runge-Kutta Explícito

genera

$$\begin{aligned}k_1 &= -ay_n. \\k_2 &= -a(y_n + h(-ay_n)) \\ &= -ay_n + ha^2y_n = y_n(-a + ha^2)\end{aligned}$$

y como

$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$$

entonces

$$y_{n+1} = y_n + \frac{h}{2}(-ay_n + y_n(-a + ha^2))$$

de donde

$$y_{n+1} = (1 - ha + \frac{h^2}{2}a^2)y_n = (1 - ha + \frac{h^2}{2}a^2)^{n+1}.$$

Conseguir que sea

$$|1 - ha + \frac{h^2}{2}a^2| < 1$$

nos lleva a observar el polinomio $p(z) = 1 - z + z^2/2$ que tiene el mínimo en $z = 1$ con valor $p(1) = 1/2$ y cumple $p(0) = p(2) = 1$. Por lo tanto, no tiene ceros en la recta real y $0 < p(z) < 1$ sólo se obtiene si $0 < z < 2$. Luego es $ha < 2$ y $h < 2/a$ la restricción de estabilidad para Heun. Además, tenemos

$$1/2 < 1 - ha + \frac{h^2}{2}a^2 < 1, \quad \forall h < 2/a.$$

Definición 3.1.4. Diremos que el **esquema es incondicionalmente A-estable (incondicionalmente absolutamente estable)**, cuando al aplicarlo al problema modelo

$$\begin{cases} y'(t) &= -ay(t), & 0 < t < T, \\ y(0) &= 1 \end{cases}$$

para cualquier $a > 0$ y en cualquier intervalo de tiempo $T > 0$, la solución discreta generada decae en valor absoluto. En caso contrario, el **esquema es condicionalmente A-estable**, a la restricción sobre h se le llama **restricción de estabilidad absoluta**. Al intervalo donde se cumple **intervalo de estabilidad absoluta**.

Ejemplo 3.1.3. Recordemos que el **método de Runge-Kutta de orden 4 clásico**

co es

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \\ y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{aligned}$$

La aplicación al **problema test** ($a > 0$)

$$\begin{cases} y'(t) = -ay(t), & 0 < t < T, \\ y(0) = 1 \end{cases}$$

genera siguiendo la misma idea que con el método de Heun de orden 2,

$$y_{n+1} = \left(1 - ha + \frac{h^2}{2}a^2 - \frac{h^3}{3!}a^3 + \frac{h^4}{4!}a^4\right)^{n+1}.$$

conseguir que sea

$$\left|1 - ha + \frac{h^2}{2}a^2 - \frac{h^3}{3!}a^3 + \frac{h^4}{4!}a^4\right| < 1$$

nos lleva a observar el polinomio $p(z) = 1 - z + z^2/2 - z^3/6 + z^4/24$ para $z = ha$. Igual que con Heun, vemos que $0 < p(z) < 1$ sólo si $z < z_* = 2.785\dots$ aproximadamente. Luego también aquí, a pesar de haber aumentado el orden, tenemos una restricción de estabilidad que cumplir:

$$ha < 2.785\dots, \Rightarrow h < 2.785\dots/a.$$

La restricción por A-estabilidad es la que necesita el método numérico para empezar a generar resultados mínimos razonables.

3.2 Método de Newmark

A veces, las aproximaciones numéricas se pueden derivar directamente en la ecuación de orden superior sin pasar por el sistema de primer orden equivalente. La ecuación de segundo orden general

$$y''(t) = f(t, y(t), y'(t))$$

3.2. Método de Newmark

se puede transformar en un sistema de primer orden y aplicar técnicas estándar. Pero también es posible desarrollar técnicas de aproximación más concretas y ajustadas a la forma de la ecuación. Esto no quiere decir tampoco que sean mejores que las previas. En particular, cuando la ecuación no contiene a la primera derivada

$$y''(t) = f(t, y(t))$$

puede pensarse que no es natural la introducción de $y'(t)$ para convertir la ecuación en un sistema de primer orden. También si que es natural buscar un esquema donde se pueda trabajar exclusivamente con las posiciones y las aceleraciones en distintos tiempos. Esta es la idea subyacente en el método de Newmark que pasamos a describir.

Consideremos entonces el problema de Cauchy de segundo orden (3.3). Dos secuencias u_n y v_n se pueden usar para aproximar a $y(t_n)$ e $y'(t_n)$, respectivamente. Usando un desarrollo de Taylor en $y(t+h)$ obtenemos

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{3!}y'''(t) + \frac{h^4}{4!}y^{(4)}(t) + \dots$$

ahora hacemos otro desarrollo de Taylor para $y''(t+h)$

$$y''(t+h) = y''(t) + hy'''(t) + \frac{h^2}{2}y^{(4)}(t) + \frac{h^3}{3!}y^{(5)}(t) + \dots$$

promediando $y''(t+h)$ e $y''(t)$ con $\alpha + \beta = 1$ tenemos

$$\begin{aligned} \alpha y''(t+h) + \beta y''(t) &= (\alpha + \beta)y''(t) + \alpha hy'''(t) + \alpha \frac{h^2}{2}y^{(4)}(t) + \alpha \frac{h^3}{3!}y^{(5)}(t) + \dots \\ &= y''(t) + \alpha hy'''(t) + \alpha \frac{h^2}{2}y^{(4)}(t) + \alpha \frac{h^3}{3!}y^{(5)}(t) + \dots \end{aligned}$$

entonces

$$y''(t) = [\alpha y''(t+h) + \beta y''(t)] - \alpha hy'''(t) - \alpha \frac{h^2}{2}y^{(4)}(t) - \alpha \frac{h^3}{3!}y^{(5)}(t) + \dots$$

y

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}[\alpha y''(t+h) + \beta y''(t)] + [\frac{h^3}{3!} - \alpha \frac{h^3}{2}]y'''(t) + [\frac{h^4}{4!} - \alpha \frac{h^4}{4}]y^{(4)}(t) + \dots$$

o lo que es lo mismo

$$\begin{aligned} y(t+h) &= y(t) + hy'(t) + \frac{h^2}{2}[\alpha f(t+h, y(t+h), y'(t+h)) + \beta f(t, y(t), y'(t))] \\ &+ h^3(\frac{1}{6} - \frac{\alpha}{2})y'''(t) + \frac{h^4}{4}(\frac{1}{6} - \alpha)y^{(4)}(t) + \dots \end{aligned}$$

Ademas

$$y'(t+h) = y'(t) + hy''(t) + \frac{h^2}{2}y'''(t) + \frac{h^3}{3!}y^{(4)}(t) + \frac{h^4}{4!}y^{(5)}(t) + \dots$$

reemplazando $y''(t)$ otra vez, usando parámetros distintos $\tilde{\alpha}, \tilde{\beta}$ con $\tilde{\alpha} + \tilde{\beta} = 1$

$$y'(t+h) = y'(t) + h[\tilde{\alpha}y''(t+h) + \tilde{\beta}y''(t)] + h^2\left(\frac{1}{2} - \tilde{\alpha}\right)y'''(t) + h^3\left(\frac{1}{6} - \frac{\tilde{\alpha}}{2}\right)y^{(4)}(t) + \dots$$

llamando $\alpha = \zeta, \beta = 1 - \zeta, \tilde{\alpha} = \theta$ y $\tilde{\beta} = 1 - \theta$ tenemos:

$$\begin{aligned} y(t+h) = y(t) &+ hy'(t) + \frac{h^2}{2}[\zeta f(t+h, y(t+h), y'(t+h)) + (1-\zeta)f(t, y(t), y'(t))] \\ &+ \frac{h^3}{2}\left(\frac{1}{3} - \zeta\right)y'''(t) + \frac{h^4}{4}\left(\frac{1}{6} - \zeta\right)y^{(4)}(t) + \dots \end{aligned} \quad (3.5)$$

$$\begin{aligned} y'(t+h) = y'(t) &+ h[\theta f(t+h, y(t+h), y'(t+h)) + (1-\theta)f(t, y(t), y'(t))] \\ &+ h^2\left(\frac{1}{2} - \theta\right)y'''(t) + \frac{h^3}{2}\left(\frac{1}{3} - \theta\right)y^{(4)}(t) + \dots \end{aligned} \quad (3.6)$$

Si ponemos $v_n \sim y'(t_n)$ y $u_n \sim y(t_n)$ podemos construir los esquemas:

$$\begin{cases} v_{n+1} = v_n + h[\theta f(t_{n+1}, u_{n+1}, v_{n+1}) + (1-\theta)f(t_n, u_n, v_n)] & v_0 \text{ dado} \\ u_{n+1} = u_n + hv_n + \frac{h^2}{2}[\zeta f(t_{n+1}, u_{n+1}, v_{n+1}) + (1-\zeta)f(t_n, u_n, v_n)] & u_0 \text{ dado} \end{cases}$$

donde se observa que sólo usamos los valores de $f(t_n, u_n, v_n)$ y de $f(t_{n+1}, u_{n+1}, v_{n+1})$ que son los naturales a considera en la dinámica del problema, en contraste con los valores necesarios a usar con un método de Runge-Kutta donde las evaluaciones de f se deben de anidar de forma no lineal.

- Entonces, si $\theta = \zeta = 0$ obtenemos un esquema de cálculo explícito

$$\begin{cases} v_{n+1} = v_n + hf(t_n, u_n, v_n) & v_0 \text{ dado} \\ u_{n+1} = u_n + hv_n + \frac{h^2}{2}f(t_n, u_n, v_n) & u_0 \text{ dado} \end{cases}$$

siendo el error global de orden $\mathcal{O}(h)$.

- Por otro lado, si ponemos $\zeta = \frac{1}{3}, \theta = \frac{1}{2}$ los errores locales son $\mathcal{O}(h^4)$ y $\mathcal{O}(h^3)$ y se obtiene:

$$\begin{cases} v_{n+1} = v_n + \frac{h}{2}[f(t_{n+1}, u_{n+1}, v_{n+1}) + f(t_n, u_n, v_n)] & v_0 \text{ dado} \\ u_{n+1} = u_n + hv_n + \frac{h^2}{6}[f(t_{n+1}, u_{n+1}, v_{n+1}) + 2f(t_n, u_n, v_n)] & u_0 \text{ dado} \end{cases}$$

siendo el error global de orden $\mathcal{O}(h^2)$.

3.2. Método de Newmark

Siendo $u''(t) = f(t, u(t), u'(t))$

$$\Rightarrow \frac{d}{dt} \begin{pmatrix} u(t) \\ v(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ f(t, u(t), v(t)) \end{pmatrix}$$

y el esquema general es

$$\begin{cases} v_{n+1} = v_n + h[\theta f(t_{n+1}, u_{n+1}, v_{n+1}) + (1 - \theta)f(t_n, u_n, v_n)] \\ u_{n+1} = u_n + hv_n + h^2[\beta f(t_{n+1}, u_{n+1}, v_{n+1}) + (\frac{1}{2} - \beta)f(t_n, u_n, v_n)] \end{cases}$$

que es conocido como el método de **Newmark**, con u_0, v_0 dados. Este método fue propuesto por el profesor Nathan M. Newmark en 1959 [18] y es uno de los esquemas de integración de tiempo implícitos más ampliamente utilizados en la dinámica estructural. A pesar de los numerosos desarrollos en esquemas de integración de tiempo para dinámicas estructurales desde 1959, este método todavía es utilizado por muchos ingenieros.

3.2.1 A-estabilidad para Newmark

La estabilidad absoluta que se estudia cuando se trabaja con EDOs de segundo orden es relativa a la ecuación test $d''(t) + \omega^2 d(t) = 0$, que representa un sistema físico oscilatorio no amortiguado con frecuencia natural $f = \omega/(2\pi)$. Transformando esta ecuación en su sistema equivalente de dos EDO de primer orden obtenemos

$$\begin{pmatrix} d \\ d' \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ -\omega & 0 \end{pmatrix} \begin{pmatrix} d \\ d' \end{pmatrix}$$

que se puede desacoplar en dos ecuaciones independiente en la forma:

$$y'(t) = \pm i\omega y(t)$$

y la aplicación de la teoría de A-estabilidad al problema de segundo orden $d''(t) + \omega^2 d(t) = 0$ es equivalente a aplicar esta teoría a la ecuación de prueba de primer orden $y' = -ay$, pero ahora teniendo en cuenta que a toma el valor específico $a = i\omega$ con a es un número complejo y $|a| = 1$, mientras que para las edos el problema test es $y' = -ay$ pero con a un número real y $a \gg 1$.

En mecánica computacional, la EDO lineal de segundo orden canónica toma la forma:

$$ma + cv + kd = f(t) \tag{3.7}$$

que corresponde a un sistema masa-resorte amortiguado, donde m es la masa, k la constante del resorte, c el coeficiente de amortiguamiento, $v = d'(t)$ la velocidad

y $a = d''(t)$ la aceleración. Convirtiendo la ecuación de segundo orden (3.7) en un sistema de primer orden:

$$\begin{pmatrix} d \\ v \end{pmatrix}' = \begin{pmatrix} v \\ a \end{pmatrix}$$

donde a verifica (3.7), y aplicándole el método Newmark obtenemos [6]:

$$\begin{cases} d_{n+1} = d_n + hv_n + \frac{h^2}{2}[(1 - 2\beta)a_n + 2\beta a_{n+1}] \\ v_{n+1} = v_n + h[(1 - \theta)a_n + \theta a_{n+1}] \\ ma_{n+1} + cv_{n+1} + kd_{n+1} = f_{n+1} \end{cases} \quad (3.8)$$

donde β y θ son parámetros libres que gobiernan la precisión, estabilidad y disipación numérica del método de Newmark.

Como ya hemos dicho, para estudiar la estabilidad y la amortiguación numérica del método, se aplica el método a la ecuación de prueba $d''(t) + \omega^2 d(t) = 0$, donde $\omega = \sqrt{k/m}$. En este caso, (3.8) puede escribirse de manera sucinta

$$X_{n+1} = AX_n \quad (3.9)$$

donde $X_{n+i} = (d_{n+i}, hv_{n+i}, h^2 a_{n+i})^T$ para $i = 0, 1$ y A es dado por $A = A_1^{-1}A_2$, y

$$A_1 = \begin{pmatrix} 1 & 0 & -\beta \\ 0 & \frac{1}{h} & -\frac{1}{h}\theta \\ \omega^2 & 0 & \frac{1}{h^2} \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 & \frac{1}{2} - \beta \\ 0 & \frac{1}{h} & \frac{1}{h}(1 - \theta) \\ 0 & 0 & 0 \end{pmatrix}.$$

Lo que se requiere en este caso es que todos los autovalores de la matriz A tengan módulo menor que uno.

El orden del método se puede calcular aplicando las condiciones de orden para los métodos lineales multi pasos [6]. El método resulta de segundo orden preciso cuando $\theta = 1/2$. El método es inestable cuando $\theta < 1/2$ y es incondicionalmente estable cuando $\frac{1}{2} \leq \theta \leq 2\beta$. La disipación de alta frecuencia se consigue cuando

$$\beta = \frac{(\theta + 1/2)^2}{4}.$$

En el método Newmark de segundo orden $\theta = 1/2$, $\beta \geq 1/4$ conserva la estabilidad incondicional [6][19][18]. Si además se requiere una disipación de alta frecuencia, $\beta = 1/4$ debe ser verificado. En este caso, Newmark se convierte en un método trapezoidal, y aunque verifica las condiciones de disipación de alta frecuencia.

3.3 Método de Newmark vs RK orden 4

En la versión óptima del método de Newmark tenemos orden 2 de convergencia y estabilidad absoluta pero el esquema es implícito. Por otro lado, con RKE4 tenemos orden 4 y un trabajo explícito aunque es condicionalmente estable. Buscando un equilibrio entre el trabajo realizado y los resultados obtenidos puede ser razonable trabajar con RKE4.

4 Resolución Numérica

Con base en los conocimientos abordados anteriormente podemos asegurar la resolución temporal de la ecuación

$$\begin{cases} m_p \ddot{u}(t) + k_p u(t) = A(p(t) - p_o) \\ u(0) = u_0 \quad \dot{u}(t) = 0 \end{cases} \quad (4.1)$$

usando el método Runge-Kutta explícito de orden 4 (cuatro etapas). Donde $p(t)$ es la presión durante el proceso de fluido estructura, calculado a partir de la ecuación

$$p(t) = p_o \left(\frac{V_o}{V_o + Au(t)} \right)^\gamma, \quad (4.2)$$

que es la ecuación de la línea de estados para la ley del gas ideal para un proceso adiabático cuasiestático.

La ecuación (4.1) es una ecuación diferencial ordinaria de segundo orden que se puede escribir como un sistema de dos ecuaciones diferenciales de primer orden en la forma

$$\begin{cases} u' = v & u(0) = u_0 \\ v' = -\frac{k_p}{m_p} u + \frac{A}{m_p} (p(t) - p_o) & v(0) = 0; \end{cases} \quad (4.3)$$

O lo que es lo mismo, en notación más compacta

$$\begin{cases} u' = f(u, v) & u(0) = u_0, \\ v' = g(u, v) & v(0) = 0, \end{cases} \quad (4.4)$$

para f y g funciones de u y v dadas por

$$\begin{cases} f(u, v) = v \\ g(u, v) = -\frac{k}{m} u + \frac{A}{m} (p(t) - p_o). \end{cases}$$

donde $p(t)$ es una función de $u(t)$ mediante (4.2). Entonces, la aplicación de RKE4 a este sistema queda como

$$\begin{cases} u_{n+1} = u_n + \frac{h}{6}(k_1 u + 2k_2 u + 2k_3 u + k_4 u) \\ v_{n+1} = v_n + \frac{h}{6}(k_1 v + 2k_2 v + 2k_3 v + k_4 v) \end{cases}$$

donde

$$\begin{cases} k_1 u = f(t_n, u_n, v_n) \\ k_1 v = g(t_n, u_n, v_n), \\ \left\{ \begin{array}{l} k_2 u = f(t_n + \frac{h}{2}, u_n + \frac{h}{2} k_1 u, v_n + \frac{h}{2} k_1 v) \\ k_2 v = g(t_n + \frac{h}{2}, u_n + \frac{h}{2} k_1 u, v_n + \frac{h}{2} k_1 v), \end{array} \right. \\ \left\{ \begin{array}{l} k_3 u = f(t_n + \frac{h}{2}, u_n + \frac{h}{2} k_2 u, v_n + \frac{h}{2} k_2 v) \\ k_3 v = g(t_n + \frac{h}{2}, u_n + \frac{h}{2} k_2 u, v_n + \frac{h}{2} k_2 v), \end{array} \right. \\ \left\{ \begin{array}{l} k_4 u = f(t_n + h, u_n + h k_3 u, v_n + h k_3 v) \\ k_4 v = g(t_n + h, u_n + h k_3 u, v_n + h k_3 v). \end{array} \right. \end{cases}$$

Esto se puede escribir de forma compacta para el caso vectorial $\mathbf{U}'(t) = \mathbf{F}(t, \mathbf{U}(t))$, haciendo $\mathbf{U}_n \sim \mathbf{U}(t_n)$ donde

$$\mathbf{U}_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$$

y tenemos entonces

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \frac{h}{6} (\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4) \quad (4.5)$$

donde

$$\begin{aligned} \mathbf{K}_1 &= \mathbf{F}(t_n, \mathbf{U}_n) \\ \mathbf{K}_2 &= \mathbf{F}(t_n + \frac{1}{2}h, \mathbf{U}_n + \frac{1}{2}h\mathbf{K}_1) \\ \mathbf{K}_3 &= \mathbf{F}(t_n + \frac{1}{2}h, \mathbf{U}_n + \frac{1}{2}h\mathbf{K}_2) \\ \mathbf{K}_4 &= \mathbf{F}(t_n + h, \mathbf{U}_n + h\mathbf{K}_3) \end{aligned} \quad (4.6)$$

para

$$\mathbf{K}_1 = \begin{pmatrix} k_1 u \\ k_1 v \end{pmatrix}, \quad \mathbf{K}_2 = \begin{pmatrix} k_2 u \\ k_2 v \end{pmatrix}, \quad \mathbf{K}_3 = \begin{pmatrix} k_3 u \\ k_3 v \end{pmatrix} \quad \text{y} \quad \mathbf{K}_4 = \begin{pmatrix} k_4 u \\ k_4 v \end{pmatrix}$$

$$\mathbf{F} = \begin{pmatrix} f(t_n, u_n, v_n) \\ g(t_n, u_n, v_n) \end{pmatrix}$$

Esta relación de recurrencia permitirá que la nueva posición del pistón \mathbf{U}_{n+1} sea calculada a partir de \mathbf{U}_n . Para el primer paso es necesario que las condiciones iniciales denotadas por u_0 y v_0 en (4.4) sean tenidas en cuenta para el cálculo de \mathbf{U}_1 .

4.1 Resultados prácticos

Ahora vamos a presentar los resultados para este modelo, con los valores de los parámetros generales dados por:

$$L_{so} = 1.2m, L_o = 1m, K_p = 10^7 N/m, m_p = 10kg, 20kg, 100kg \text{ y } 1000kg, u^0 = 0.2m,$$

$$p_o = 10^5 Pa, T_o = 300K, c_o = \sqrt{\gamma RT_o} = 347.1887m/s, N = 1000.$$

Observación 4. Se eligió el número de paso para el método de Runge-Kutta, $N = 1000$ (fijo), para tener una mejor aproximación numérica a la solución exacta de este problema, respetando la región de estabilidad impuesta por el propio método para el tamaño de h en el intervalo $[t_0; \tau_0]$ (donde τ_0 es el periodo de oscilaciones). Vamos tomar que la frecuencia natural del pistón es una función de m_p (donde k_p se mantiene constante).

Podemos ver por el gráfico (4.1) para la presión y el gráfico (4.2) para el pistón, que a medida que el tiempo crece hay variación de la amplitud tanto del pistón y de la fuerza externa que en nuestro caso es la presión del aire haciendo que el movimiento se haga limitado, decreciendo hasta la posición de reposo. Sin embargo vamos analizar el comportamiento del pistón en relación a la presión del gas en un período. El cuadro 4.1 muestra el valor de la frecuencia para las diferentes masa del pistón.

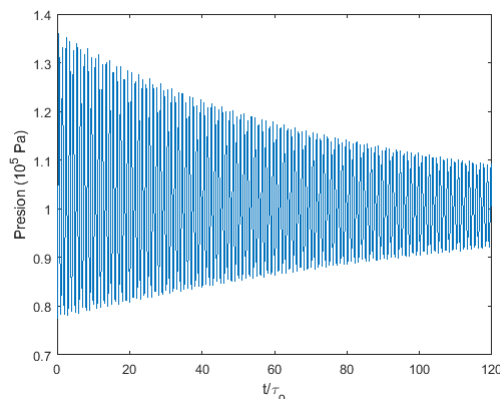


Figura 4.1: Presión del gas en la cámara ($t \rightarrow \infty$)

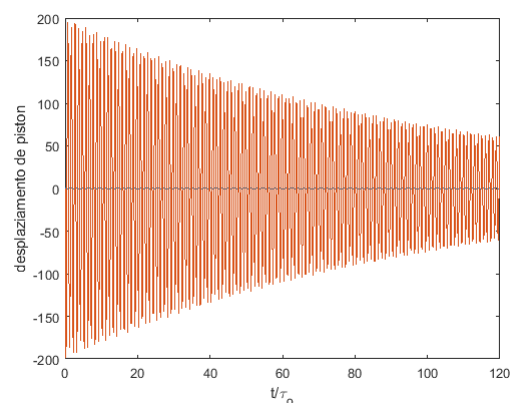


Figura 4.2: Desplazamiento del pistón cuando $t \rightarrow \infty$

m_p (kg)	10	20	100	1000
f_o (Hz)	159.16	112.54	50.33	15.92
τ_o (s)	$6.28 \cdot 10^{-3}$	$8.89 \cdot 10^{-3}$	$1.99 \cdot 10^{-2}$	$6.28 \cdot 10^{-2}$

Cuadro 4.1: Frecuencia y período natural del pistón

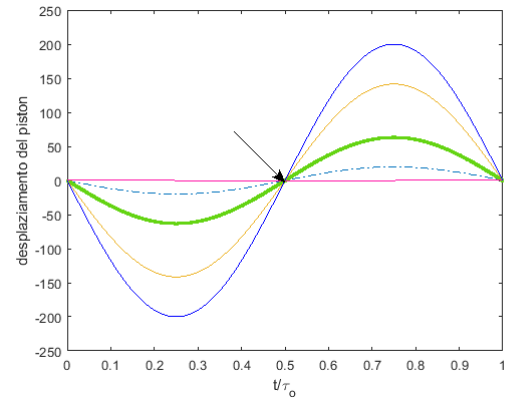
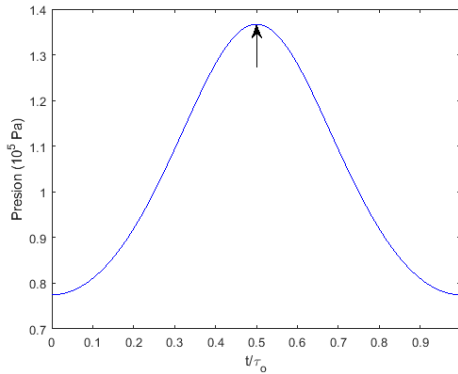


Figura 4.3: Presión del gas en la cámara Figura 4.4: Desplazamiento y la velocidad

A partir de estos datos obtenidos por el método numérico de Runge-Kutta de orden 4 tenemos la idea clara que en el momento inicial del proceso de fluido estructura el gas ocupaba todo el volumen de la cámara (estaba en una fase expansiva), mientras que el muelle que gobierna el movimiento del pistón estaba comprimido, entonces, tenemos que el gráfico de la Figura (4.3) presenta el comportamiento de la presión del gas para las diferentes masas y el gráfico de la Figura (4.4) presenta el movimiento del pistón sometido a una fuerza externa (la presión del gas) cuando el mismo tiene una masa de 10kg (línea azul), 20kg (línea amarilla), 100kg (línea verde) y con 1000kg (línea discontinua). La flecha en el gráfico de la Figura (4.4) indica el período en que el pistón está el máximo de la compresión y la flecha en el gráfico de la Figura (4.3) indica el punto en que la presión cambia su comportamiento en relación a su crecimiento (cuando el gas alcanza el máximo de su presión en la cámara al ser comprimido). Con base en estos datos podemos ver que cuando el pistón posee una masa de 10kg transporta mucho más energía que cuando posee una masa superior (según el modelo de Planck para distribución de energía, deducido por Albert Einstein¹) su longitud de onda es menor que las demás masas,

¹Albert Einstein (1879 - 1955) En su artículo "Sobre un punto de vista heurístico relativo a la producción y transformación de la luz" ("Über einen die Erzeugung und Verwandlung des Lichtes betreffenden heuristischen Gesichtspunkt"), postuló que la luz en sí consiste de partículas

4.1. Resultados prácticos

sin embargo posee una mayor amplitud y también el pistón se desplaza en menor velocidad que las demás masa (o sea para este modelo, cuanto menor sea la masa menor será la velocidad de las oscilaciones).

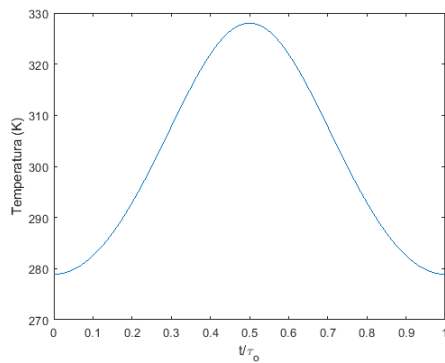


Figura 4.5: La temperatura del gas

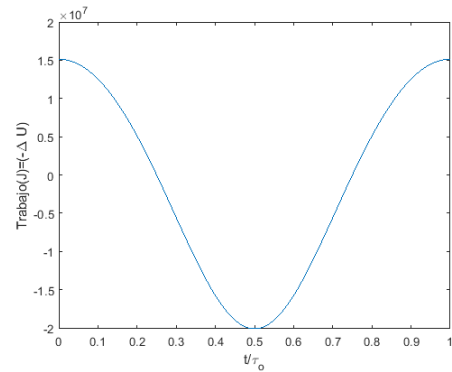


Figura 4.6: El trabajo del gas

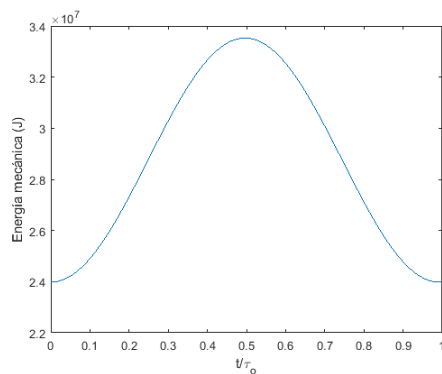


Figura 4.7: Energía Mecánica

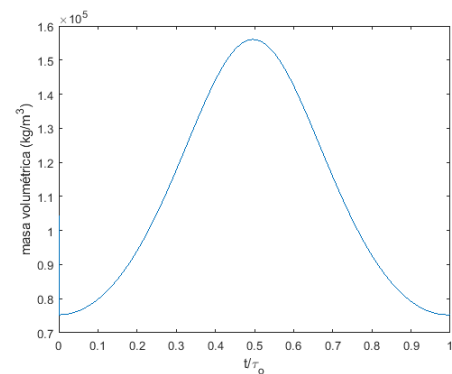


Figura 4.8: Masa Volumétrica

localizadas (cuánta). Los cuantos de luz de Einstein fueron casi universalmente rechazados por todos los físicos, incluyendo Max Planck y Niels Bohr. Esta idea sólo se volvió universalmente aceptada en 1919, con los experimentos detallados de Robert Millikan sobre el efecto fotoeléctrico. Einstein concluyó que cada onda de frecuencia f se asocia con un conjunto de fotones con una *energía* hf cada, donde h ($= 6.626 \cdot 10^{-34} \text{ J}\cdot\text{s}$) es la constante de Planck).

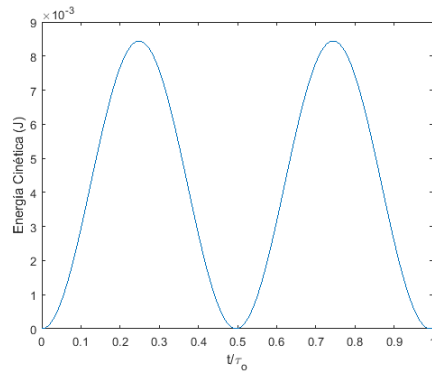


Figura 4.9: Energía Cinética

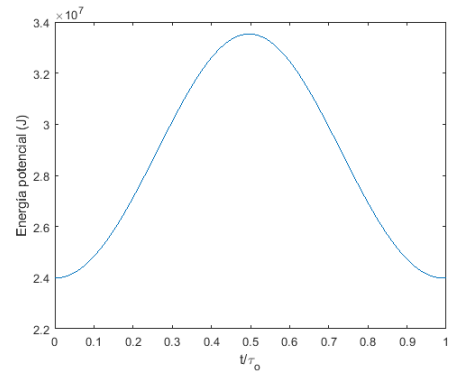


Figura 4.10: Energía Potencial

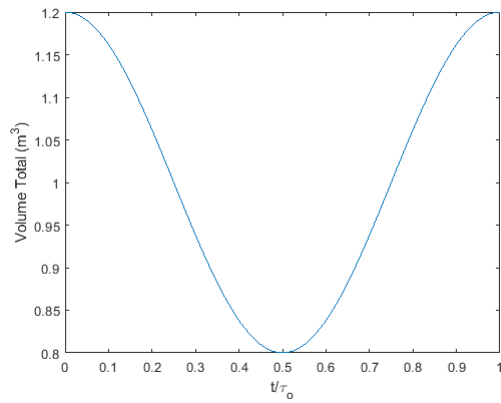


Figura 4.11: Volumen de la cámara

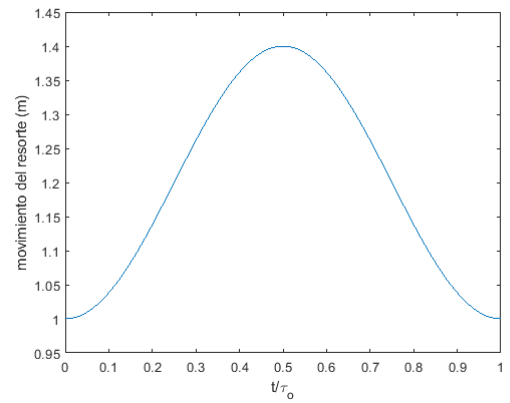


Figura 4.12: Movimiento del resorte

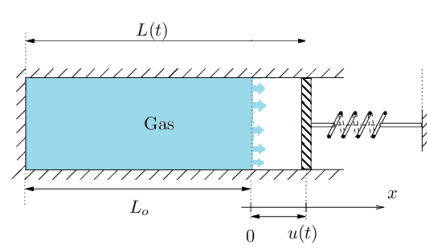


Figura 4.13: La expansión del gas

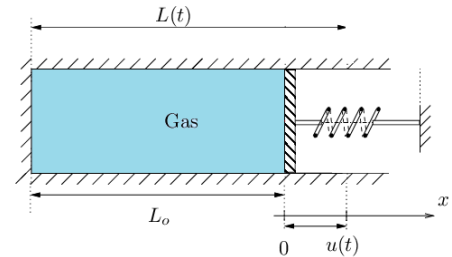


Figura 4.14: La compresión del gas

En cuanto al fluido, podemos observar conforme la Figura (4.13) que a medida

que el resorte comprime el fluido empieza a llenar el espacio vacío dejado por el pistón en la cámara, comienza a expandirse lo que es una característica de los gases que presentan la propiedad de expandirse libremente cuando no comprimidos, no formando por lo tanto una superficie libre (fluido compresible), pero disminuyendo su presión, y en el instante que el resorte del pistón alcanza el máximo de su compresión corresponde también al máximo de expansión del volumen del gas y por consiguiente en la disminución de la presión.

El aumento de la presión del gas durante el momento de su compresión Figura 4.3, debido a la elongación del resorte, es fruto del trabajo realizado por el sistema correspondiente a la variación de la energía interna durante el proceso y por la variación de la temperatura, aumentando su trabajo mecánico durante los instantes iniciales de la compresión del gas y disminuyendo ese trabajo en las fases intermedia del proceso de expansión y compresión del gas, así como la temperatura que va aumentando durante la compresión del gas y disminuyendo durante su expansión. La medida que el resorte realiza el movimiento de elongación, resulta en la compresión del gas y en la reducción del volumen como muestra la Figura (4.14).

Las Figuras (4.5) y (4.6), muestran ese aumento de temperatura y también el trabajo interno realizado por el fluido, lo que nos da una percepción clara que durante el proceso de fluido estructura el gas en la cámara tiende verdaderamente a cambiar su comportamiento interno a la presión ejercida por la fuerza del pistón, o sea durante el proceso de compresión fruto de la mayor agitación de las moléculas del gas lleva a un aumento de la temperatura lo que corresponde a un aumento de la presión del gas y una disminución del volumen ocupado por el mismo, pero por otro lado el gas deja de realizar trabajo, o sea durante el proceso de compresión el gas no realiza esfuerzo para ser comprimido, ya que este esfuerzo recae para la estructura mecánica del pistón como muestra la Figura (4.7), que durante la compresión del gas la energía mecánica del sistema es mayor que durante el proceso de compresión del muelle (expansión del gas). Los datos referentes al trabajo ($W(t)$), la energía mecánica ($\mathcal{E}(t)$) y la temperatura ($T(t)$) del gas durante el proceso, se obtienen a partir de las siguientes ecuaciones:

$$W(t) = \frac{m\mathcal{R}T}{\gamma - 1} \left[1 - \left(\frac{p(t)}{p_o} \right)^{\frac{\gamma-1}{\gamma}} \right], \quad \mathcal{E}(t) = \frac{1}{2}m_p\dot{u}(t)^2 + \frac{k_p}{2}(L_{se} - u(t) + L_{so})^2$$

y

$$T(t) = T_o \left(\frac{p(t)}{p_o} \right)^{\frac{\gamma-1}{\gamma}}.$$

la energía mecánica $\mathcal{E}(t)$ está compuesta por la parte cinética $\mathcal{E}_c(t)$ y por la parte potencial $\mathcal{E}_p(t)$.

Además, hay que señalar que, independientemente de la masa que posea el pistón, el comportamiento del gas es el mismo Figura (4.3), manteniendo constantes su comportamiento para los diferentes tipos de masa del pistón, creciendo en

la misma proporción de presión cuando es comprimido por el pistón en el momento de elongación del resorte y volviendo a su posición inicial durante la compresión del resorte.

Los resultados abajo, Figuras (4.11) y (4.12) presentan los cambios que ocurren en cuanto a la ocupación volumétrica del gas en la cámara y de la expansión (compresión) del muelle respectivamente durante el proceso de fluido estructura, siendo que el volumen ocupado por el gas en la cámara se regula por la expresión $L(t) = L_o + u(t)$ que es equivalente a la dimensión volumétrica total de la cámara y la ecuación que nos permite describir los momentos de compresión y expansión del muelle es dada por $L_s(t) = L_o - u(t)$, donde $L_{si} = -\frac{Ap_o}{k_p} + L_{so}$ es cuando el resorte está en reposo sufriendo presión por parte del gas.

4.2 Aplicación práctica

Con los resultados numéricos tenemos la idea del instante en que el gas está en expansión y compresión así como la posición inicial y final del muelle al completar un ciclo, dados esos, que nos muestran que para las masas probadas y sus respectivas frecuencias hay estabilidad del modelo, sin embargo para su aplicación práctica es necesario tener en cuenta el tiempo característico de deformación del fluido y para ello vamos a utilizar el número de Deborah²: es un número adimensional usado en reología para caracterizar la fluidez de un material bajo condiciones específicas de flujo. Formalmente el número se define como el cociente entre el tiempo de relajación, que caracteriza la fluidez intrínseca de un material, y la escala temporal característica de un experimento (o simulación por ordenador). Cuanto más pequeño sea el número, el material es más fluido. Su ecuación se puede escribir como

$$D_e = \frac{\tau_{char}^f}{\tau_{char}^{est}} \quad (4.7)$$

donde τ_{char}^f es el tiempo característico requerido para la onda del fluido cruzar la cámara desde un lado al otro, es igual $L(t)/c_o \approx 3.5 \cdot 10^{-3} s$, y τ_{char}^{est} es el tiempo característico para la estructura es igual al periodo del pistón τ_o . De este modo tenemos para frecuencia del pistón igual a:

- $f_o = 159.16$

$$D_e = \frac{3.5 \cdot 10^{-3}}{6.28 \cdot 10^{-3}} = 0.5501$$

²El número fue originalmente propuesto por Markus Reiner, profesor de Technion, inspirado en un versículo bíblico de la jueza Deborah en la Biblia "Las montañas fluyeron delante del Señor" (Libro de Jueces 5:5).

4.2. Aplicación práctica

- $f_o = 112.54$

$$D_e = \frac{3.5 \cdot 10^{-3}}{8.89 \cdot 10^{-3}} = 0.3890$$

- $f_o = 50.33$

$$D_e = \frac{3.5 \cdot 10^{-3}}{1.99 \cdot 10^{-2}} = 0.1740$$

- $f_o = 15.92$

$$D_e = \frac{3.5 \cdot 10^{-3}}{6.28 \cdot 10^{-2}} = 0.0550$$

En base al número de Deborah podemos afirmar que ese modelo tiene un mejor acoplamiento (suave) al pistón con la masa $m = 1000kg$, cuya frecuencia es menor de todas probada. En este caso, el tiempo característico del fluido es varias órdenes de magnitud menor que el período natural del pistón, lo que nos lleva a creer que el fluido se adapta instantáneamente al movimiento del pistón en un régimen de fluidos casi constante para la menor frecuencia, Figura (4.21). En general, esta suposición de estado casi estable (también llamada suposición de pseudo estado estacionario) se utiliza cuando una parte del sistema reacciona mucho más rápidamente (el fluido tiene una escala de tiempo característica más corta/equilibra más rápido) que otra parte. Cuando una parte del sistema se equilibra más rápido que otra parte, podemos decir que esta parte del sistema está básicamente en estado estacionario con el sistema de movimiento más lento (aproximadamente). Entonces podemos concluir que verdaderamente ese modelo tiene mejor utilidad para bajas frecuencias porque hay un mayor equilibrio por parte del fluido con el movimiento del pistón cuando las frecuencias tienen un menor valor.

Los gráficos siguientes presentan las diferencias entre la onda de presión del gas (línea azul de los gráficos a la izquierda) con relación al movimiento del pistón en la cámara durante el proceso del fluido estructura para las diferentes masas, mientras que los gráficos a la derecha presentan la diferencia entre el modelo de onda de presión (línea marrón) en relación al modelo del gas utilizado, sabiendo que cuando se mueve el pistón dentro del fluido se produce una onda de compresión, y cuando el pistón se retira, se produce una onda de expansión. Sabiendo que el movimiento comienza en una onda de expansión el pistón comienza impulsivamente, con velocidad $|u_p|$ la distribución de la velocidad de la partícula en el primer instante es un "paso". Pero en una onda de expansión, ese paso no puede ser mantenido; así que la onda comienza a propagarse, ella comienza a "aplanar". La onda se propaga con velocidad c_o (velocidad del sonido) al fluido no perturbado, es decir, en la dirección opuesta al pistón y al movimiento del fluido y las propiedades del fluido tienen el valor uniforme P_o, c_o , que para el nuestro caso (gas perfecto), se dan en términos de la ecuación

$$p(t) = p_o \left(1 - \frac{\gamma - 1}{2} \frac{|\dot{u}_p|}{c_o} \right)^{2\gamma/(\gamma-1)}$$

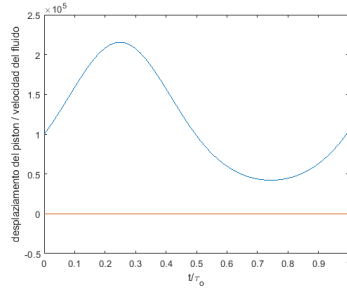


Figura 4.15: Velocidad vs movimiento del pistón $m = 10 \text{ kg}$

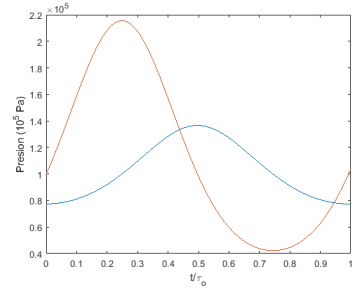


Figura 4.16: Onda de presión del gas $m = 10 \text{ Kg}$

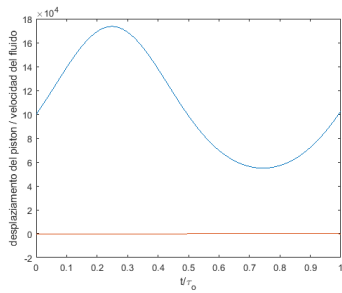


Figura 4.17: Velocidad vs movimiento del pistón $m = 20 \text{ kg}$

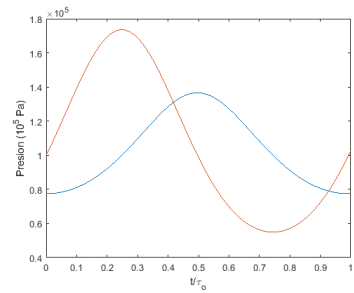


Figura 4.18: Onda de presión del gas $m = 20 \text{ Kg}$

4.2. Aplicación práctica

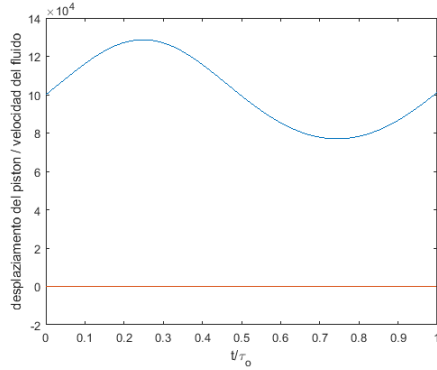


Figura 4.19: Velocidad vs movimiento del pistón $m = 100 \text{ kg}$

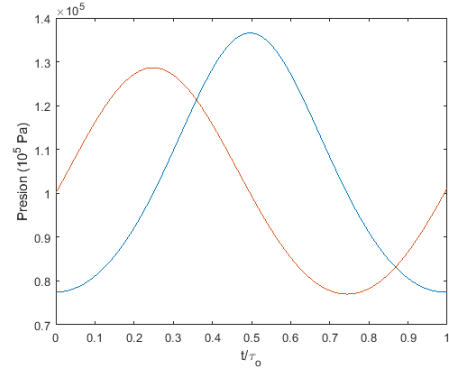


Figura 4.20: Onda de presión del gas $m = 100 \text{ Kg}$

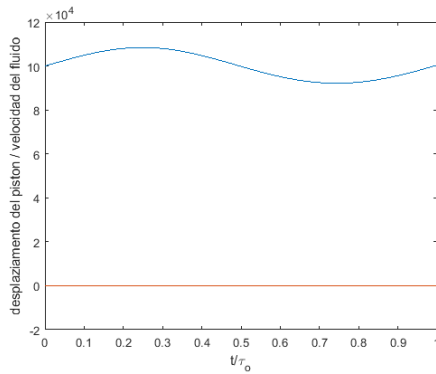


Figura 4.21: Velocidad vs movimiento del pistón $m = 1000 \text{ kg}$

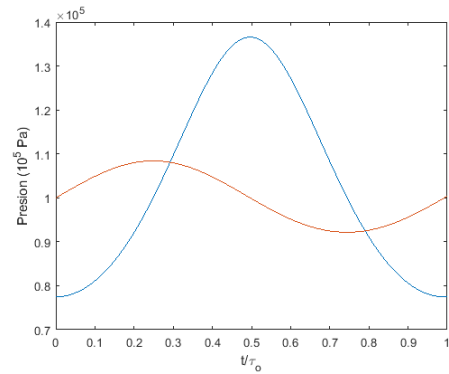


Figura 4.22: Onda de presión del gas $m = 1000 \text{ Kg}$

5 Conclusión

Este trabajo de introducción a los problemas de interacción de fluido estructura (FSI) consistió en la resolución del ejemplo de un gas contenido en una cámara (1D) cerrada por un pistón móvil. Este ejemplo presentado por Emmanuel Lefrançois y Jean-Paul Boufflet [15], utilizando el método numérico de Runge-Kutta clásico de orden 4, que se aplicó a un modelo de fluido analítico estacionario.

Se observó que con el método de Runge-Kutta el modelo se mantuvo estable para las diferentes masas del pistón, sin necesidad de condiciones de reajustes auxiliares para garantizar esta estabilidad, por otro lado la presión en la cámara se mantiene la misma para las las diferentes masas (así como para el modelo de Newmark utilizado en [15]), a través del concepto de tiempo característico, calculado utilizando el número adimensional de Deborah fue posible constatar que cuando el pistón posee una masa $m = 1000 \text{ Kg}$ hay una mayor fluidez por parte del fluido que con las demás masas, o sea el período de oscilaciones realizadas por el pistón, permiten que el fluido se adapte instantáneamente a su movimiento ya que el tiempo característico del fluido es mucho menor que el período natural del pistón produciendo así un acoplamiento suave y el proceso puede ser visto como casi-estático; pero cuando el pistón oscila en altas frecuencias, el acoplamiento entre las dos físicas se realiza de manera abrupta lo que puede crear fallas o irregularidades en su aplicación práctica.

En este orden de idea, podemos concluir que, independientemente del orden del método numérico a ser utilizado para la resolución de ese modelo, las mejoras nunca serán satisfactoria hasta el punto de permitir que el mismo sea funcional para altas frecuencias, ya que el tiempo característico del pistón casi o, no difiere de los calculados por el método numérico de Newmark y por el método numérico de Runge-Kutta clásico de orden 4 (en el anexo A.4 tenemos un ejemplo de ese modelo usando un método mejor que el RK clásico de orden 4, que es el método de RK adaptativo de Dormand-Prince 5(4) presentando en el MATLAB por la función `ode45` que comprueba esa característica del modelo).

Como solución para evitar este hecho es necesario introducir un modelo para el fluido a partir de las ecuaciones de Navier-Stokes como se sugiere en [15] o la ecuación de Euler para fluidos ya que se pueden aplicar técnicas de capturas de *shock* para lo hace aplicable a altas frecuencias sin correr el riesgo de perder la

estabilidad en el método numérico.

A Apéndice

A.1 Algunas Definiciones

Definición A.1.1. (Condición de Lipschitz) una función f satisface la condición de Lipschitz en la variable y , $\forall t, \forall y_1, y_2 \in \mathbb{R}^n$, si y sólo si existe una constante $L > 0$ tal que

$$\|f(t, y_1) - f(t, y_2)\| \leq L\|y_1 - y_2\|$$

Teorema A.1.1. (Teorema fundamental de la integración) Si f es una función continua en $[a, b]$, entonces

$$F(x) = \int_a^x f(t) dt \quad \forall x \in [a, b],$$

es una función diferenciable, llamada primitiva de f , que satisface,

$$F'(x) = f(x) \quad \forall x \in [a, b].$$

Teorema A.1.2 (Teorema de Taylor). Sea $f : [a, b] \rightarrow \mathbb{R}$ una función con $n + 1$ derivadas continua en $[a, b]$ para algunos $n \geq 0$ y $x, x_0 \in [a, b]$, entonces

$$f(x) = p_n(x) + R_n(x)$$

para

$$p_n(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k$$

y

$$R_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!} (x - x_0)^{n+1}$$

para un punto ξ_x entre x y x_0 .

El punto x_0 generalmente se elige a discreción del usuario y, a menudo, se toma como 0.

El teorema de Taylor es importante porque nos permite representar, exactamente, funciones bastante generales en términos de polinomios con un error conocido, especificado y limitable. Esto nos permite reemplazar, en un entorno de cómputo, estas mismas funciones generales por algo que es mucho más simple, un polinomio, pero al mismo tiempo podemos vincular el error cometido. Ninguna otra herramienta es tan importante para métodos numéricos como el Teorema de Taylor.

Teorema A.1.3. (*Teorema del valor medio 'para integrales'*) si f es una función continua en $[a, b]$ y $x_1, x_2 \in [a, b]$ con $x_1 < x_2$, entonces $\exists \xi \in (x_1, x_2)$ tal que

$$f(\xi) = \frac{1}{x_2 - x_1} \int_{x_1}^{x_2} f(t) dt.$$

Teorema A.1.4. (*Teorema del valor medio*) Sea f una función dada, continua en $[a, b]$ y diferenciable en (a, b) . Entonces existe un punto $\xi \in [a, b]$, tal que

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}.$$

A.2 Ecuaciones diferenciales ordinarias

Una ecuación diferencial es una ecuación en que la incógnita es una función y que además, involucra también las derivadas de la función hasta un cierto orden. La incógnita no es el valor de la función en uno o varios puntos, sino la función en sí misma.

Cuando la incógnita es una función de una variable se dice que la ecuación es ordinaria, debido a que la o las derivadas que aparecen en ella son derivadas ordinarias (por contraposición a las derivadas parciales de las funciones de varias variables).

Una ecuación diferencial ordinaria

$$y' = f(t, y), \quad f : \Omega \subset \mathbb{R}^2 \rightarrow \mathbb{R}$$

admite, en general infinitas soluciones. Si, por ejemplo, $f \in C^1(\Omega; \mathbb{R})^1$, por cada punto $(t_0, y_0) \in \Omega$ pasa una única solución $\varphi : I \rightarrow \mathbb{R}$, definida en un cierto intervalo $I \subset \mathbb{R}$, que contiene a t_0 .

Se denomina Problema de Cauchy o problema de valor inicial (PVI)

$$\begin{cases} y' = f(t, y), \\ y(t_0) = y_0 \end{cases} \quad (\text{A.1})$$

al problema de determinar, de entre todas las soluciones de ecuación diferencial $y' = f(t, y)$, aquella que pasa por el punto (t_0, y_0) , es decir, que verifica $y(t_0) = y_0$.

Definición A.2.1. *Se llama solución general de una ecuación diferencial de primer orden a la función*

$$y = \phi(t, C),$$

que depende de una constante arbitraria C y satisface las condiciones siguientes:

- a) *satisface la ecuación diferencial para cualquier valor de la constante C ;*
- b) *cualquiera que sea la condición inicial $y = y_0$, para $t = t_0$, es decir $(y)_{t=t_0} = y_0$, se puede encontrar un valor $C = C_0$ tal que la función $y = \phi(t, C_0)$ satisfaga la condición inicial dada.*

Teorema A.2.1 (Existencia e unicidad de la solución). *Si en la ecuación*

$$y'(t) = f(t, y)$$

la función $f(t, y)$ y su derivada parcial $\partial_y f$ respecto a y son continuas en un cierto dominio D del plano Ot y y si (t_0, y_0) es un punto de este dominio, entonces existe una solución única de esta ecuación, $y(t) = \phi(t)$, satisface la condición $y = y_0$, para $t = t_0$

La condición de que la función y debe tomar el valor dado y_0 para $t = t_0$ se llama condición inicial.

Demostración. (**Existencia**) Definimos la secuencia de funciones $y_n(t)$ por

$$y_0(t) = y_0, \quad y_n(t) = y_0 + \int_{t_0}^t f(s, y_{n-1}(s)) ds, \quad \text{para } n = 1, 2, \dots$$

Como $f(t, y)$ es continua en D , entonces existe una constante positiva b tal que

$$|f(t, y)| \leq b \quad \text{para } (t, y) \in D$$

entonces

$$|y_1(t) - y_0| \leq b|t - t_0|, \quad \text{para } \alpha < t < \beta$$

Como $\frac{\partial f}{\partial y}$ es continua en D , entonces también existe una constante positiva a tal que

$$|f(t, y) - f(t, z)| \leq a|y - z|, \quad \text{para } \alpha < t < \beta \quad \text{e} \quad \delta < y, z < \gamma$$

Así

$$\begin{aligned} |y_2(t) - y_1(t)| &\leq \int_{t_0}^t |f(s, y_1(s)) - f(s, y_0(s))| ds \leq a \int_{t_0}^t |y_2(s) - y_1(s)| ds \\ &\leq ab \int_{t_0}^t |s - t_0| ds = ab \frac{|t - t_0|^2}{2} \end{aligned}$$

y

$$\begin{aligned} |y_3(t) - y_2(t)| &\leq \int_{t_0}^t |f(s, y_2(s)) - f(s, y_1(s))| ds \leq a \int_{t_0}^t |y_2(s) - y_1(s)| ds \\ &\leq a^2 b \int_{t_0}^t \frac{|s - t_0|^2}{2} ds = a^2 b \frac{|t - t_0|^3}{6} \end{aligned}$$

vamos a suponer por inducción, que

$$|y_{n-1}(t) - y_{n-2}(t)| \leq a^{n-2} b \frac{|t - t_0|^{n-1}}{(n-1)!} \quad \text{para } n = 2, 3, \dots$$

entonces

$$\begin{aligned} |y_n(t) - y_{n-1}(t)| &\leq \int_{t_0}^t |f(s, y_{n-1}(s)) - f(s, y_{n-2}(s))| ds \leq a \int_{t_0}^t |y_{n-1}(s) - y_{n-2}(s)| ds \\ &\leq a \int_{t_0}^t a^{n-2} b \frac{|t - t_0|^{n-1}}{(n-1)!} ds = a^{n-1} b \frac{|t - t_0|^n}{n!} \end{aligned}$$

Estas desigualdades son válidas para $\alpha \leq \alpha' \leq t < \beta' \leq \beta$ en que α' y β' son tales que $\delta < y_n(t) < \gamma$ siempre que $\alpha' < t < \beta'$.

Entonces tenemos que

$$\sum_{n=1}^{\infty} |y_n(s) - y_{n-1}(s)| \leq b \sum_{n=1}^{\infty} a^{n-1} \frac{(\beta - \alpha)^n}{n!}$$

que es convergente, y como

$$y_n(t) = y_0 + \sum_{k=1}^n (y_k(t) - y_{k-1}(t))$$

entonces $y_n(t)$ es convergente. Sea

$$y(t) = \lim_{n \rightarrow \infty} y_n(t)$$

como

$$|y_m(t) - y_n(t)| \leq \sum_{k=n+1}^m |y_k(t) - y_{k-1}(t)| \leq b \sum_{k=n+1}^m a^{k-1} \frac{(\beta - \alpha)^k}{k!}$$

entonces pasando al límite cuando m tiende a infinito obtenemos que

$$|y(t) - y_n(t)| \leq b \sum_{k=n+1}^m a^{k-1} \frac{(\beta - \alpha)^k}{k!} \quad (\text{A.2})$$

tomando un $\epsilon > 0$, para un n suficientemente grande, $|y(t) - y_n(t)| < \epsilon/3$, para $\alpha' < t < \beta'$. $y(t)$ es continua, ya que para s suficientemente cercano de t , $|y_n(t) - y_n(s)| < \epsilon/3$ y para n suficientemente grande $|y(s) - y_n(s)| < \epsilon/3$, lo que implica, que

$$|y(t) - y(s)| \leq |y(t) - y_n(t)| + |y_n(t) - y_n(s)| + |y(s) - y_n(s)| < \epsilon.$$

Y además para $\alpha' < t < \beta'$, tenemos que

$$\lim_{n \rightarrow \infty} \int_{t_0}^t f(s, y_n(s)) ds = \int_{t_0}^t f(s, \lim_{n \rightarrow \infty} y_n(s)) ds = \int_{t_0}^t f(s, y(s)) ds$$

y pela desigualdad (A.2) tenemos que

$$\begin{aligned} \left| \int_{t_0}^t f(s, y_n(s)) ds - \int_{t_0}^t f(s, y(s)) ds \right| &\leq \int_{t_0}^t |f(s, y_n(s)) - f(s, y(s))| ds \\ &\leq a \int_{t_0}^t |y_n(s) - y(s)| ds \\ &\leq ab(t - t_0) \sum_{k=n+1}^m a^{k-1} \frac{(\beta - \alpha)^k}{k!} \end{aligned}$$

que tiende a cero cuando n tiende al infinito. Por tanto

$$\begin{aligned} y(t) = \lim_{n \rightarrow \infty} y_n(t) &= y_0 + \lim_{n \rightarrow \infty} \int_{t_0}^t f(s, y_{n-1}(s)) ds = \int_{t_0}^t f(s, \lim_{n \rightarrow \infty} y_{n-1}(s)) ds \\ &= y_0 + \int_{t_0}^t f(s, y(s)) ds \end{aligned}$$

Derivando en relación a t esta ecuación vemos que $y(t)$ es solución del problema de valor inicial (teorema fundamental del cálculo)¹, o sea

$$\frac{d}{dt}y(t) = f(y(t), t).$$

□

Demostración. (Unicidad) Supongamos que $y(t)$ y $z(t)$ sean soluciones del problema de valor inicial. sea

$$u(t) = \int_{t_0}^t |y(s) - z(s)| ds$$

así como,

$$y(t) = \int_{t_0}^t y'(s) ds = \int_{t_0}^t f(s, y(s)) ds, \quad z(t) = \int_{t_0}^t z'(s) ds = \int_{t_0}^t f(s, z(s)) ds$$

entonces

$$\begin{aligned} u'(t) = |y(t) - z(t)| &\leq \int_{t_0}^t |y'(s) - z'(s)| ds \\ &\leq \int_{t_0}^t |f(s, y(s)) - f(s, z(s))| ds \leq a \int_{t_0}^t |y(s) - z(s)| ds \end{aligned}$$

Sustraer $au(t)$ y multiplicándose por e^{-at} obtenemos (lemma de Gronwall)²

$$\frac{d}{dt}(e^{-at}u(t)) \leq 0 \quad \text{con} \quad u(t_0) = 0$$

Luego $e^{-at}u(t) = 0$ (ya que $u(t) \geq 0$) y por lo tanto $u(t) = 0$, para todo t . Así $y(t) = z(t)$, para todo t . □

¹El teorema afirma que si I es un intervalo de \mathbf{R} con más de un punto y si f es una función continua de I en \mathbf{R} , entonces, para cada $a \in I$ la función F de I en \mathbf{R} definida por

$$F(x) = \int_a^x f(t) dt$$

es derivable y su derivada es precisamente la función f (F es una primitiva de f)

²sea $u(t)$ una función no negativa y diferenciable en $[0, T]$ que satisfice: $u'(t) \leq f(t)u(t) + g(t)$, donde $f(t)$ y $g(t)$ son funciones integrable en $[0, T]$, entonces:

$$u(t) \leq u(0)e^{\int_0^t f(\psi)d\psi} + \int_0^t g(s)ds, \quad \forall t \in [0, T]$$

A.3 Introducción a MATLAB

MATLAB es un potente paquete de software para computación científica, orientado al cálculo numérico, a las operaciones matriciales y especialmente a las aplicaciones científicas y de ingeniería; que ofrece un entorno de desarrollo integrado (IDE) con una lenguaje de programación propia (**lenguaje M**). Está disponible para las plataformas *Unix*, *Windows*, *Mac OS X* y *GNU/Linux*. La lenguaje de programación del MATLAB es más tolerante en su sintaxis, que permite al usuario escribir sus propios *scripts* (conjunto de comando escritos en un fichero, que se pueden ejecutar con una única orden) para resolver un problema concreto y también escribir nuevas funciones con, por ejemplo, sus propios algoritmos, o para modularizar la resolución de un problema complejo. MATLAB dispone, además, de numerosas *Toolboxes*, que le añaden funcionalidades especializadas.

Numerosas contribuciones de sus miles de usuarios en todo el mundo pueden encontrarse en la web de The MathWorks: <https://es.mathworks.com/>

Comenzando

Dependiendo del sistema operativo y de la version del MATLAB, al iniciar nos aparecerá una ventana más o menos como la de la Figura A.1

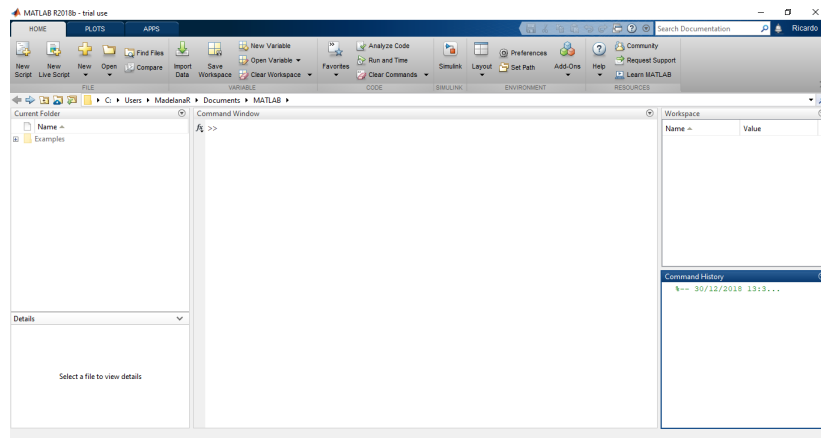


Figura A.1: La ventana del MATLAB

si $f(t)$ y $g(t)$ son no negativo, entonces la expresión se simplifica a:

$$u(t) \leq e^{\int_0^t f(\psi) d\psi} \left[u(0) + \int_0^t g(s) ds \right], \quad \forall t \in [0, T]$$

Si la ubicación de las ventanas integradas es diferente, se puede volver a ésta mediante: Menú Desktop → Desktop Layout → Default ...

Se puede experimentar con otras disposiciones. Si hay alguna que nos gusta, se puede salvar con: Menú Desktop → Desktop Layout → Save Layout ... dándole un nombre, para usarla en otras ocasiones. A través de la ventana principal de MATLAB Command Window (ventana de comandos) es posible comunicarse con el MATLAB, escribiendo las órdenes en la línea de comandos. El símbolo >> al comienzo de una línea de la ventana de comandos se denomina *prompt* y indica que MATLAB está desocupado, disponible para ejecutar nuestras órdenes.

A.3.1 Aspectos básicos

La mayoría de las funciones tienen más y/o distintas funcionalidades que las que se exponen aquí. Para una descripción exacta y exhaustiva es preciso consultar la ayuda on-line.

Los tipos básicos de datos que maneja MATLAB son números reales, booleanos (valores lógicos) y cadena de caracteres (string). También puede manipular distintos tipos de números enteros, aunque sólo suele ser necesario en circunstancias específicas.

En MATLAB, por defecto, los números son codificados como números reales en coma flotante en doble precisión. La precisión, esto es, el número de bits dedicados a representar la mantisa y el exponente depende de cada (tipo de) máquina.

MATLAB manipula también otros objetos, compuestos a partir de los anteriores: números complejos, matrices, *cells*, estructuras definidas por el usuario, clases Java, etc.

El objeto básico de trabajo de MATLAB es una matriz bidimensional cuyos elementos son números reales o complejos. Escalares y vectores son considerados casos particulares de matrices. También se pueden manipular matrices de cadenas de caracteres, booleanas y enteras.

Algunas constantes numéricas están predefinidas

<code>i, j</code>	Unidad imaginaria: $2 + 3i, -1 - 2j$
<code>pi</code>	Número π
<code>Inf</code>	<i>Infinito</i> , Número mayor que el más grande que se puede almacenar. Se produce con operaciones como $x/0$, con $x \neq 0$
<code>NaN</code>	<i>Not a Number</i> , magnitud numérica resultado de cálculos indefinidos. Se produce con cálculos del tipo $0/0$ o ∞/∞ . $(0+2i)/0$ da como resultado <code>NaN + Inf i</code>
<code>eps, intmax intmin, realmax, realmin</code>	Otras constantes. Consultar la ayuda on-line.

Cuadro A.1: Algunas constantes pre-definidas en MATLAB

MATLAB distingue entre mayúscula y minúsculas: `pi` no es lo mismo que `Pi`; también conserva un historial de las instrucciones escritas en la línea de comandos, se pueden recuperar instrucciones anteriores, usando las teclas de flechas arriba y abajo. Con las flechas izquierdas y derecha nos podemos desplazar sobre la línea de comando y modificarlo.

Se pueden salvar todas las instrucciones y la salida de resultados de una sesión de trabajo de MATLAB a un fichero:

```

>> diary nombre_fichero
>> diary off           % suspende la salvaguarda
    
```

Constantes y operadores

Números reales	8.01 -5.2 .056 1.4e+5 0.23E-2 -.567d-21 8.003D-12
Números complejos	1+2i -pi-3j
Booleanos	true false
Caracteres	Entre apóstrofos: 'esto es una cadena de caracteres (string)'
Operadores aritméticos	+ - * / ^
Operadores de comparación	== ~= (ó <>) < > <= >=
Operadores lógicos (los dos últimos sólo para escalares)	& ~ && && no evalúan el operando de la derecha si no es necesario.

Cuadro A.2: Constantes y operadores

Funciones elementales

Los nombres de las funciones elementales son los habituales. Los argumentos pueden ser, siempre que tenga sentido, reales o complejos y el resultado se devuelve en el mismo tipo del argumento.

La lista de todas las funciones matemáticas elementales se puede consultar en: `Help` → `MATLAB` → `Fuctions:By Category` → `Mathematics` → `Elementary Math`. Algunas de las más habituales se muestran en el cuadro A.3.

Uso como calculadora

Se puede utilizar MATLAB como simple calculadora, escribiendo expresiones aritmético y terminando por `RETURN(<R>)`. Se obtiene el resultado inmediatamente a través de la variable del sistema `ans` (de *answer*). Si no se desea que MATLAB escriba el resultado en el terminal, debe terminarse la orden por punto y coma (útil, sobre todo en programación).

<code>sqrt(x)</code>	raiz cuadrada	<code>sin(x)</code>	seno (radianes)
<code>abs(x)</code>	módulo	<code>cos(x)</code>	coseno (radianes)
<code>conj(z)</code>	complejo conjugado	<code>tan(z)</code>	tangente (radianes)
<code>real(z)</code>	parte real	<code>cotg(x)</code>	cotangente (radianes)
<code>imag(z)</code>	parte imaginaria	<code>asin(x)</code>	arcoseno
<code>exp(x)</code>	exponencial	<code>acos(x)</code>	arcocoseno
<code>log(x)</code>	logaritmo natural	<code>atan(x)</code>	arcotangente
<code>log10(x)</code>	logaritmo decimal	<code>cosh(x)</code>	cos. hiperbólico
<code>rat(x)</code>	aprox. racional	<code>sinh(x)</code>	seno hiperbólico
<code>mod(x,y)</code>	resto de dividir x por y	<code>tanh(x)</code>	tangente hiperbólica
<code>rem(x,y)</code>	Ver help para definición exacta		
<code>fix(x)</code>	Redondeo hacia 0	<code>acosh(x)</code>	arcocoseno hiperb.
<code>ceil(x)</code>	Redondeo hacia $+\infty$	<code>asin(x)</code>	arcoseno hiperb.
<code>floor(x)</code>	Redondeo hacia $-\infty$	<code>atanh(x)</code>	arcotangente hiperb.
<code>round(x)</code>	Redondeo al entero más próximo		

Cuadro A.3: Algunas funciones matemáticas elementales

Ejemplo A.3.1.

```
Command Window
>> sqrt(34*exp(2))/(cos(23.7)+12)

ans =

    1.3059

>> 7*exp(5/4)+3.54

ans =

    27.9724

>> exp(1+3i)

ans =

-2.6911 + 0.3836i
```

Variables

Ejemplo A.3.2.

```
Command Window
>> a=10

a =

    10

>> numerico=exp(2.4/3)

numerico =

    2.2255

>> numerico=a+numerico*(4-0.5i)

numerico =

    18.9022 - 1.1128i

>> clear numerico
```

En MATLAB las variables no son nunca declaradas: su tipo y su tamaño cambian de forma dinámica de acuerdo con los valores que le son asignados. Así, una misma variable puede ser utilizada, por ejemplo, para almacenar un número complejo,

A.3. Introducción a MATLAB

a continuación una matriz 25×40 de números enteros y luego para almacenar un texto. Las variables se crean automáticamente al asignarles un contenido. (Atención: recuérdese que las variables **AB**, **ab**, **Ab** y **aB** son **distintas**, ya que MATLAB distingue entre mayúsculas y minúscula).

Formatos

Por defecto, MATLAB muestra los números en formato de punto fijo con 5 dígitos. se puede modificar este comportamiento mediante el comando **format**

<code>format</code>	Cambia el formato de salida a su valor por defecto, <code>short</code>
<code>format short</code>	El formato por defecto
<code>format long</code>	Muestra 15 dígitos
<code>format short e</code>	Formato short, en coma flotante
<code>format long e</code>	Formato long, en coma flotante
<code>format rat</code>	Muestra los números como cociente de enteros

Cuadro A.4: Formatos en el MATLAB

Algunos comandos utilitarios del sistema operativo

Están disponibles algunos comandos utilitarios, como

<code>ls</code>	Lista de ficheros del directorio de trabajo
<code>dir</code>	
<code>pwd</code>	Devuelve el nombre y la ruta (<i>path</i>) del directorio de trabajo
<code>cd</code>	Para cambiar de directorio
<code>clc</code>	Limpia la ventana de comandos Command Window
<code>date</code>	Fecha actual

Cuadro A.5: Comandos del sistema operativo

A.3.2 Documentación y ayuda on-line

- Ayuda on-line en la ventana de comandos

```
>> help nombre_de_comando
```

La información se obtiene en la misma ventana de comandos.

- Ayuda on-line con ventana de navegador

`>> helpwin` ó bien Menú Help ó bien Botón Start → Help.

Además, a través del navegador del Help se pueden descargar, desde The MathWorks, guías detalladas, en formato pdf, de cada capítulo.

A.3.3 Script y funciones: El editor integrado

Scripts

En términos generales, en informática, un *script* (guión o archivo por lotes) es un conjunto de instrucciones (programas), usualmente simple, guardadas en un fichero (usualmente de texto plano) que son ejecutadas normalmente mediante un intérprete. Son útiles para automatizar pequeñas tareas. También puede hacer las veces de un "programa principal" para ejecutar una aplicación.

Así, para llevar a cabo una tarea, en vez de escribir las instrucciones una por una en la línea de comandos de MATLAB, se pueden escribir las órdenes una detrás de otra en un fichero. Para ello se puede utilizar el Editor integrado de MATLAB. **File** → **New** → **Script** Un *script* de MATLAB debe guardarse en un fichero con sufijo `.m` para ser reconocido. El nombre del fichero puede ser cualquiera *razonable*, es decir, sin acentos, sin espacios en blanco y sin caracteres "extraños".

M-Funciones

Una función (habitualmente denominadas M-funciones en MATLAB), es un programa con una «interfaz» de comunicación con el exterior mediante argumentos de entrada y de salida. Las funciones MATLAB responden al siguiente formato de escritura:

```
function [argumentos de salida] = nombre(argumentos de entrada)
%
% comentarios
%
....
instrucciones (normalmente terminadas por ; para evitar eco en
%                                     pantalla)
....
```

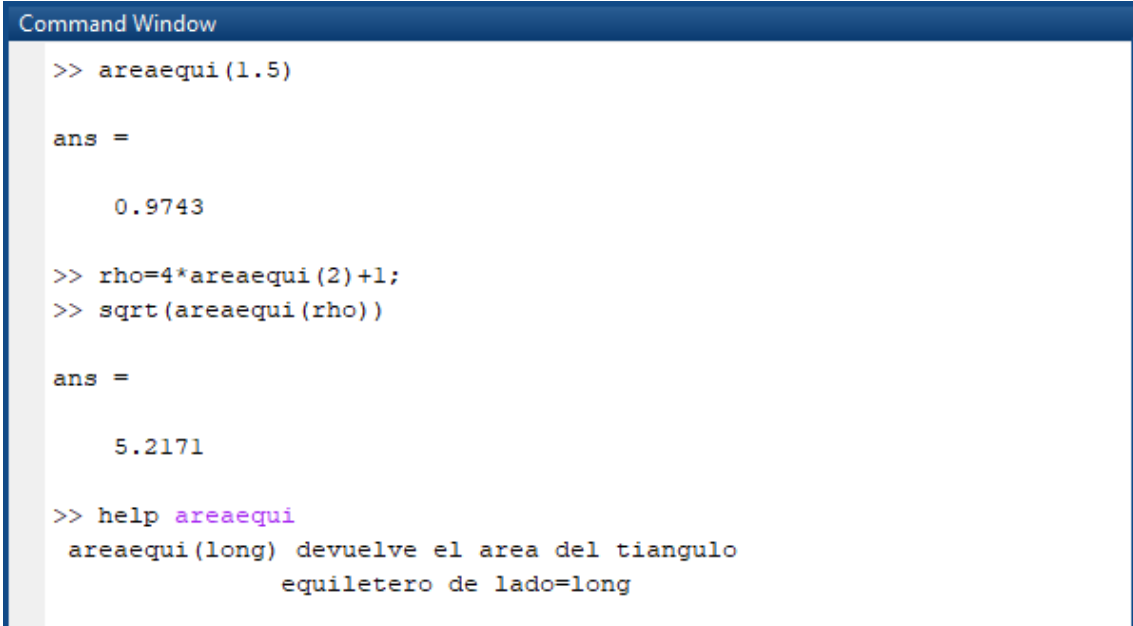
Las funciones deben guardarse en un fichero con el mismo nombre que la función y sufijo `.m`. Lo que se escribe en cualquier línea detrás de `%` es considerado como comentario (en los casos de función debe explicar, brevemente, el funcionamiento y uso

A.3. Introducción a MATLAB

de la función, constituyen la ayuda on-line de la función”). La primera línea de una M-función siempre debe comenzar con la cláusula (palabra reservada) `function`. El fichero que contiene la función debe estar en un sitio en el que MATLAB lo pueda encontrar, normalmente en la carpeta de trabajo.

Ejemplo A.3.3. *El siguiente código debe guardarse en un fichero de nombre `areaequi.m`*

```
function [sup] = areaequi(long)
%
% areaequi(long) devuelve el area del triangulo
%           equilatero de lado = long
%
sup = sqrt(3)*long ^ 2/4;
```



```
Command Window
>> areaequi(1.5)

ans =

    0.9743

>> rho=4*areaequi(2)+1;
>> sqrt(areaequi(rho))

ans =

    5.2171

>> help areaequi
areaequi(long) devuelve el area del triangulo
equilatero de lado=long
```

Funciones anónimas

Algunas funciones sencillas, que devuelvan el resultado de una expresión, se pueden definir mediante una sola instrucción, en mitad de un programa (script o función) o en la línea de comandos. Se llaman funciones anónimas. La sintaxis para definir las es: `nombre_funcion = @(argumentos) expresion_funcion`

Ejemplo A.3.4. *Función anónima para calcular el área de un círculo*

```
Command Window
>> area_circulo=@(r)pi*r.^2;
>> area_circulo(1)

ans =

    3.1416

>> semicirc=area_circulo(3)/2

semicirc =

    14.1372
```

Las funciones anónimas pueden tener varios argumentos y hacer uso de variables previamente definidas:

Ejemplo A.3.5. *Función anónima de dos variables*

```
Command Window
>> a=2;
>> mifun=@(x,t) sin(a*x).*cos(t/a);
>> mifun(pi/4,1)

ans =

    0.8776
```

A.3.4 Matrices

Como ya se ha dicho, las matrices bidimensionales de números reales o complejos son los objetos básicos con los que trabaja MATLAB. Los vectores y escalares son casos particulares de matrices.

Construcciones de matrices

La forma más elemental de introducir matrices es describir sus elementos de forma exhaustiva (por fila y entre corchetes rectos `[]`): elementos de una fila se separan unos de otros por comas y una fila de la siguiente por punto y coma.

A.3. Introducción a MATLAB

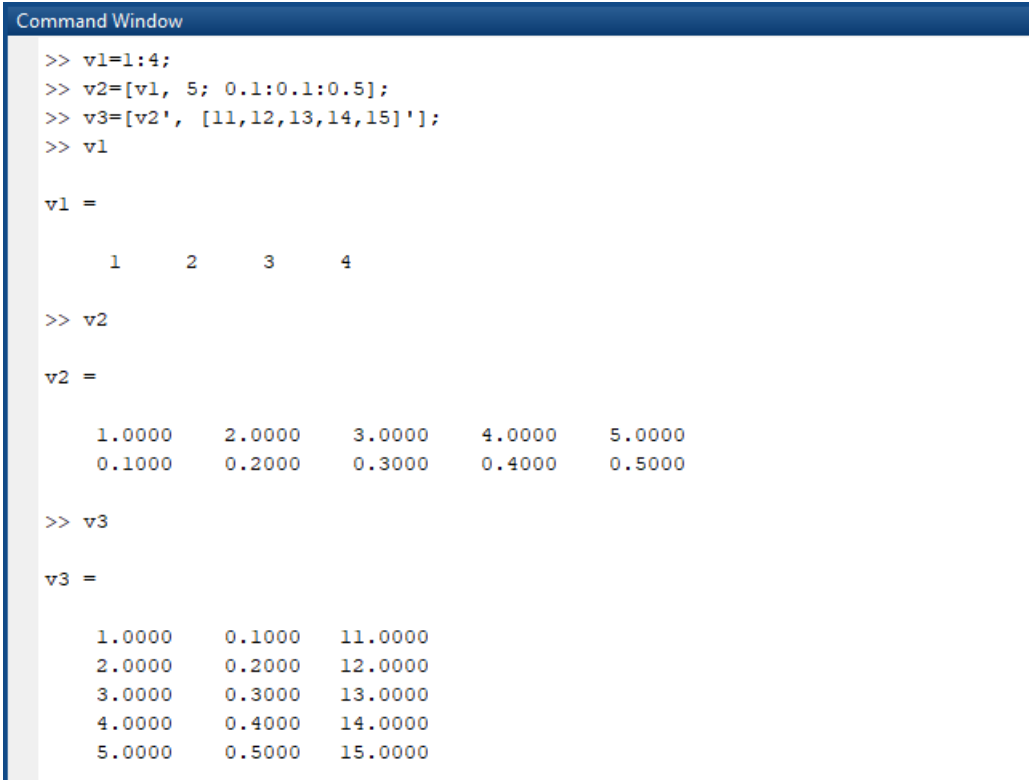
```
>> v = [1, -1, 0, sin(2.88)]           % vector fila longitud 4
>> w = [0; 1.003; 2; 3; 4; 5*pi]      % vector columna longitud 6
>> a = [1,2,3,4; 5,6,7,8; 9,10,11,12] % matriz 3 x 4
```

Cuadro A.6: Construcciones elementales de matrices

```
>> v1 = a:h:b % Crea un vector fila de números regularmente
              % espaciados a a+h a+2h ... hasta c <= b < c+h
>> v2 = a:b   % con el anterior, con paso h=1
>> v3 =v2'   % matriz traspuesta (conjugada si es compleja)
>> v4 =v2.'  % matriz traspuesta sin conjugar
```

Cuadro A.7: Otras órdenes para crear matrices

Ejemplo A.3.6. (*Matrices construidas con bloques*)



```
Command Window
>> v1=1:4;
>> v2=[v1, 5; 0.1:0.1:0.5];
>> v3=[v2', [11,12,13,14,15]'];
>> v1

v1 =

     1     2     3     4

>> v2

v2 =

     1.0000     2.0000     3.0000     4.0000     5.0000
     0.1000     0.2000     0.3000     0.4000     0.5000

>> v3

v3 =

     1.0000     0.1000    11.0000
     2.0000     0.2000    12.0000
     3.0000     0.3000    13.0000
     4.0000     0.4000    14.0000
     5.0000     0.5000    15.0000
```

Las siguientes funciones generan vectores de elementos regularmente espaciados, útiles en muchas circunstancias, especialmente para creación de gráficas.

<code>linspace(a,b,n)</code>	Si a y b son números reales y n un número entero, genera una partición regular del intervalo $[a,b]$ con n nodos ($n-1$ subintervalos)
<code>linspace(a,b)</code>	como el anterior, con $n=100$

Las siguientes funciones generan algunas matrices especiales que serán de utilidad.

<code>zeros(n,m)</code>	matriz $n \times m$ con todas sus componentes iguales a cero
<code>ones(n,m)</code>	matriz $n \times m$ con todas sus componentes iguales a uno
<code>eye(n,m)</code>	matriz unidad $n \times m$: diagonal principal = 1 y el resto de las componentes = 0
<code>diag(v)</code>	Si v es un vector, es una matriz cuadrada de ceros con diagonal principal = v
<code>diag(A)</code>	Si A es una matriz, es su diagonal principal

MATLAB posee, además, decenas de funciones útiles para generar distintos tipos de matrices. Para ver una lista exhaustiva consultar:

Help \rightarrow MATLAB \rightarrow Functions: By Category \rightarrow Mathematics \rightarrow Arrays and Matrices

Operaciones con vectores y matrices

Los operadores aritméticos representan las correspondientes operaciones matriciales siempre que tengan sentido.

Sean A y B dos matrices de elementos respectivos a_{ij} y b_{ij} y sean k un escalar.	
$A+B$, $A-B$	matrices de elementos respectivos $a_{ij} + b_{ij}$, $a_{ij} - b_{ij}$ (si las dimensiones son iguales)
$A+k$, $A-k$	matrices de elementos respectivos $a_{ij} + k$, $a_{ij} - k$
$k*A$, A/k	matrices de elementos respectivos $k a_{ij}$ $\frac{1}{k} a_{ij}$
$A*B$	producto matricial de A y B (si las dimensiones son adecuadas)
$A \wedge n$	Si n es un entero positivo, $A*A*...*A$

Además de estos operadores MATLAB dispone de ciertos operadores aritméticos que operan elemento a elemento. Son los operadores `.*`, `./` y `.^`, muy útiles para aprovechar las funcionalidades vectoriales de MATLAB.

A.3.5 Dibujo de curvas

La representación gráfica de una curva en un ordenador es una línea poligonal construida uniendo mediante segmentos rectos un conjunto discretos y ordenado de puntos: $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. La línea así obtenida tendrá mayor apariencia de "suave" cuanto más puntos se utilicen para construirla, ya que los segmentos serán imperceptibles.

Para dibujar una curva plana en MATLAB se usa el comando `plot(x,y)`, siendo `x` y `y` dos vectores de las mismas dimensiones conteniendo respectivamente, las abscisas y las ordenadas de los puntos de la gráfica.

Ejemplo A.3.7. Dibujar la curva $y = \frac{x^2+2}{x+5}$ para $x \in [-2, 3]$

```
Command Window
>> f=@(x) (x.^2+2) ./ (x+5);
>> x=linspace(-2,3);
>> plot(x,f(x))
```

Ejemplo A.3.8. Dibujar la curva $y = 2 \sin^3(x)$ para $x \in [-1, 1]$

```
Command Window
>> y = 2*sin(x).^3;
>> x=-1:0.01:1;
>> plot(x,y)
```

Se pueden dibujar dos o más curvas de una sola vez, proporcionando al comando `plot` varios pares de vectores abscisas-ordenadas, como en el ejemplo siguiente.

Ejemplo A.3.9. Dibujar la curva $y = 2 \sin^3(x) \cos^2(x)$ y $y = e^x - 2x - 3$ para $x \in [-1.5, 1.5]$

```
Command Window
>> f1=@(x) 2*sin(x).^3.*cos(x).^2;
>> f2=@(x) exp(x)-2*x-3;
>> x=linspace(-1.5,1.5);
>> plot(x,f1(x),x,f2(x))
```

A cada par de vectores abscisas-ordenadas en el comando `plot` se puede añadir un argumento opcional de tipo cadena de caracteres, que modifica el aspecto con que se dibuja la curva. Por ejemplo, la orden siguiente dibuja la curva `f1` en color negro (`k`, de `black`), con marcadores `*` y con línea punteada, y la curva `f2` en color azul (`b`, `blue`) y con marcadores `+`: `plot(x,f1(x),'k *:',x,f2(x),'b+')`.

Además, mediante argumentos opcionales, es posible modificar otras propiedades de las curvas. Esto se hace siempre mediante un par de argumentos en la orden de dibujo que indican `Nombre de la propiedad`, `Valor de la propiedad`. Para más información hojear el `help` (`help plot`).

Se pueden añadir elementos a la gráfica, para ayudar a su comprensión. Para añadir una leyenda que identifique cada curva se usa el comando siguiente, que asigna las leyendas a las curvas en el orden en que han sido dibujadas.

```
legend('Leyenda1', 'Leyenda2')
```

Para añadir etiquetas a los ejes que aclaren el significado de las variables se usan los comandos

```
xlabel('Étiqueta del eje OX')  
ylabel('Étiqueta del eje OY')
```

Se puede añadir una cuadrícula mediante la orden

```
grid on
```

También es muy útil la orden, siguiente, que define las coordenadas mínimas y máximas del rectángulo del plano OXY que se visualiza en la gráfica.

```
axis([xmin, xmax, ymin, ymax])
```

Cada nueva orden de dibujo borra el contenido previo de la ventana gráfica, si existe. Para evitar esto existen la órdenes

```
hold on  
hold off
```

La orden `hold on` permanece activa hasta que se cierre la ventana gráfica o bien se dé la orden `hold off`

Ejemplo A.3.10. *Las siguientes órdenes dará como resultado la gráfica de la figura A.2*

```

Command Window
>> x=linspace(0,pi,30);
>> axis([-0.5,pi+0.5,-0.5,1.5])
>> hold on
>> plot(x,sin(x).^3,'g',x,cos(x).^2,'b+', 'LineWidth',1.1)
>> x=linspace(-0.95,pi);
>> plot(x,log(x+1)/2,'r','LineWidth',1.1)
>> plot([-5,5],[0,0],'k','LineWidth',1)
>> plot([0,0],[-5,5],'k','LineWidth',1)
>> legend('Leyenda1','Leyenda2','Leyenda3')
>> xlabel('Etiqueta del eje OX')
>> ylabel('Etiqueta del eje OY')
>> hold off
>> shg
    
```

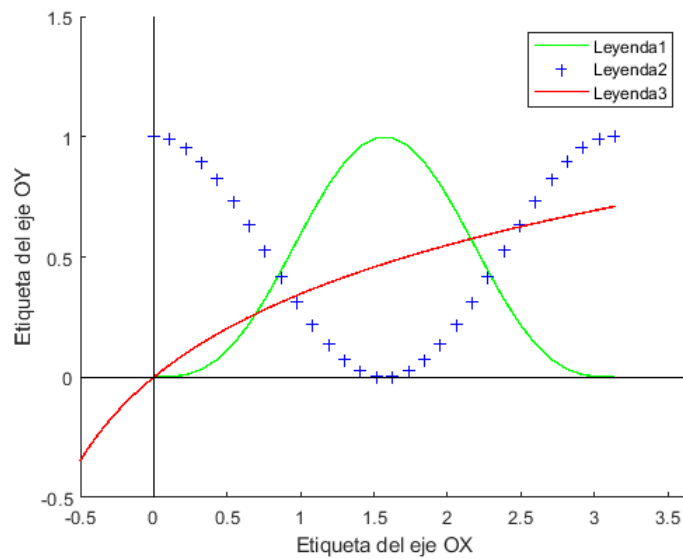


Figura A.2: Varias curvas con leyenda y etiquetas

A.3.6 Programación con MATLAB

Estructuras condicionales: if

En casi todos los lenguajes de programación se utilizan estructuras condicionales, este tipo de estructuras se implementan mediante una instrucción (ó super-instrucción) denominada `if`, cuya sintaxis puede variar ligeramente de unos lenguajes a otros. En MATLAB, concretamente, su forma es la siguiente:

```

instruccion-anterior
if expresion
    bloque-de-instrucciones
end
instruccion-siguiente
    
```

Se evalúa `expresión`. Si el resultado es `true`, se ejecuta el `bloque-de-instrucciones` y cuando termina, se continúa por `instruccion-siguiente`. Si el resultado no es `true`, se va directamente a `instruccion-siguiente`.

```

instruccion-anterior
if expresion
    bloque-de-instrucciones-1
    bloque-de-instrucciones-2
end
instruccion-siguiente
    
```

Se evalúa `expresión`. Si el resultado es `true`, se ejecuta el `bloque-de-instrucciones-1` y cuando se termina, se continúa por `instruccion-siguiente`. Si el resultado no es `true`, se ejecuta el `bloque-de-instrucciones-2` y cuando se termina se va a la `instruccion-siguiente`.

Ejemplo A.3.11. (Uso de un condicional simple) Escribir una *M-función* que, dado $x \in \mathbb{R}$, devuelva el valor en x de la función a trozos

$$f(x) = \begin{cases} x + 1 & \text{si } x < -1, \\ 1 - x^2 & \text{si } x \geq -1. \end{cases}$$

```

function [fx] = mifun(x)
%
% v=mifun(x) devuelve el valor en x de la función
%          f(x)= x+1 si x<-1
%          f(x)=1-x^2 si no
%
fx=x+1;
if x>-1
    fx=1-x^2;
end
    
```

Estructura de repetición o bucles: while

Este mecanismo de programación permite un grupo de instrucciones mientras que se verifique una cierta condición. Su sintaxis en MATLAB es la siguiente:

```
instruccion-anterior
while expresion
    bloque-de-instrucciones
end
instruccion-siguiente
```

Al comienzo se evalúa expresión. Si el resultado **no es true**, se va directamente a la **instrucción-siguiente**. En este caso, no se ejecuta el **bloque-de-instrucciones**.

Si, por el contrario, el resultado de expresión es true, se ejecuta el **bloque-de-instrucciones**. Cuando se termina se vuelve a evaluar la expresión y se vuelve a decidir.

Naturalmente este mecanismo precisa que, dentro del **bloque-de-instrucciones** se modifique, en alguna de las repeticiones, el resultado de evaluar expresión. En caso contrario, el programa entraría en un *bucle infinito*. Llegado este caso, se puede detener el proceso pulsando la combinación de teclas CTRL+C.

Ejemplo A.3.12. (*Uso de un while*) *Escribir una M-función que, dado un número natural n , calcule y devuelva la suma de todos los números naturales entre 1 y n*

```
function [suma] = SumaNat(n)
%
% suma= SumaNat(n) es la suma de los n primeros números naturales
%
suma = 0;
k = 1;
while k<=n
    suma = suma + k;
    k = k+1;
end
```

La orden de MATLAB `sum(1:n)` tiene el mismo efecto que `SumaNat(n)`.

Estructuras de repetición o bucles indexados: for

En muchas ocasiones, las repeticiones de un bucle dependen en realidad de una variable entera cuyo valor se va incrementando hasta llegar a uno dado, momento en que se detienen las repeticiones. Esto sucede, especialmente, con los algoritmos que manipulan vectores y matrices, que es lo más habitual cuando se programan métodos numéricos. En estas ocasiones, para implementar el bucle es preferible utilizar el **bucle indexado**. En MATLAB este tipo de mecanismos se implementan

mediante una instrucción denominada `for`.

```

instruccion-anterior
for k= 1 : n
    bloque-de-instrucciones
end
instruccion-siguiente

```

Para cada valor de la variable `k` desde 1 hasta `n`, se ejecuta una vez el `bloque-de-instrucciones`. Cuando se termina, el programa se continua ejecutando por la `instrucción-siguiente`.

Ejemplo A.3.13. (*Uso de for*) Escribir una *M-función* que, dado un vector v , calcule el valor máximo entre todos sus componentes.

```

function [vmax] = Maximo(v)
%
% Maximo(v) es el máximo de las componentes del vector v
%
vmax = v(1);
for k = 2:length(v)
    if v(k) > vmax
        vmax = v(k);
    end
end
end

```

Con esta instrucción, no hay que ocuparse ni de inicializar ni de incrementar dentro del bucle la variable-índice, k , basta con indicar, junto a la cláusula `for`, el conjunto de valores que debe tomar; puesto que es la propia instrucción `for` que gestiona la variable-índice k .

Siempre que en un bucle sea posible determinar a priori el número de veces que se va a repetir el bloque de instrucciones, es preferible utilizar la instrucción `for`, ya que la instrucción `while` es más lenta.

A.3.7 Operaciones de lectura y escritura

Instrucción básica de lectura: `input`

La instrucción `input` permite almacenar en una variable un dato que se introduce a través del teclado, imprime mensaje en la pantalla y se queda esperando hasta que el usuario teclea algo en el teclado, terminando por la tecla `return`. La orden

```
var = input('Mensaje')
```

Instrucción básica de impresión en pantalla: `disp`

La instrucción `disp` permite imprimir en la pantalla el valor de una (matriz) constante o variable, sin imprimir el nombre de la variable.

```
disp(algo)
```

Instrucción de impresión en pantalla con formato: `fprintf`

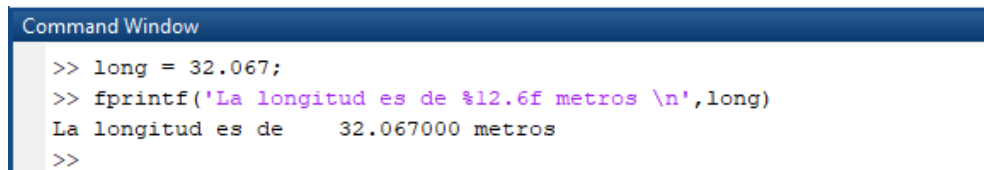
Esta orden permite controlar la forma en que se imprimen los datos. Su sintaxis para imprimir en la pantalla es

```
fprintf(formato, lista_de_datos)
```

donde `lista_de_datos` son los datos a imprimir, pueden ser constantes y/o variables, separados por comas y `formato` es una cadena de caracteres que describe la forma en que se deben imprimir los datos. Puede contener combinaciones de los siguientes elementos:

- Códigos de conversión: formados por el símbolo `%`, una letra (como `f`, `e`, `i`, `s`) y eventualmente unos números para indicar el número de espacios que ocupará el dato a imprimir.
- Texto literal a imprimir
- Caracteres de escape, como `\n`.

Ejemplo A.3.14. (Uso de `fprintf`)



```
Command Window
>> long = 32.067;
>> fprintf('La longitud es de %12.6f metros \n',long)
La longitud es de      32.067000 metros
>>
```

En este ejemplo el formato se compone de:

- el texto literal `'La longitud es de'` (incluye los espacios en blanco).

- el código `%12.6f` que indica que se escriba un número (en este caso el valor de la variable `long`) ocupando un total de 12 espacios, de los cuales 6 son para las cifras decimales,
- el texto literal `'metros'` (también incluyendo los blancos),
- el carácter de escape `\n` que provoca un salto de línea.

A.3.8 Aproximación numérica

Solo para algunos (pocos) tipos muy especiales de ecuaciones diferenciales es posible encontrar la expresión de sus soluciones en términos de funciones elementales. En la inmensa mayoría de los casos prácticos sólo es posible encontrar aproximaciones numéricas de los valores de una solución en algunos puntos. Así, una aproximación numérica de la solución del problema de Cauchy

$$\begin{cases} y' = f(t, y), & t \in [t_0, t_f] \\ y(t_0) = y_0 \end{cases} \quad (\text{A.3})$$

consiste en una sucesión de valores de la variable independiente:

$$t_0 < t_1 < \dots < t_n = t_f$$

y una sucesión de valores y_0, y_1, \dots, y_n , tales que

$$y_k \approx y(t_k), \quad k = 0, 1, \dots, n,$$

es decir, y_k es una aproximación del valor en t_k de la solución del problema A.3.

$$y_k \approx y(t_k), \quad k = 0, 1, \dots$$

Resolución con MATLAB

MATLAB dispone de toda una familia de funciones para resolver (numéricamente) ecuaciones diferenciales:

```
ode45, ode23, ode113
ode15s, ode23s, ode23t, . . .
```

Cada una de ellas implementa un método numérico diferente, siendo adecuado usar unas u otras en función de las dificultades de cada problema en concreto. Para problemas no demasiado "difíciles" será suficiente con la función `ode45` ó bien `ode23`

A.3. Introducción a MATLAB

Exponemos aquí la utilización de la función `ode45`, aunque la utilización de todas ellas es similar, al menos en lo más básico.

Para dibujar la gráfica de la solución (numérica) de A.3 se usará la orden:

```
ode45(odefun, [t0,tf], y0)
```

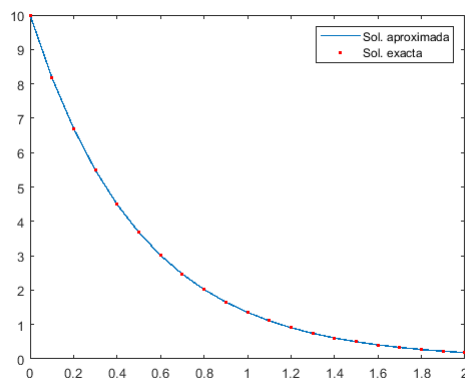
donde `odefun` es un manejador de la función que evalúa el segundo miembro de la ecuación, $f(t, y)$. `[t0, tf]` es el intervalo en el que se quiere resolver la ecuación, i.e. $t_0=t_0$, $t_f = t_f$ y y_0 es el valor de la condición inicial, $y_0=y_0$.

Ejemplo A.3.15. *Calcular (aproximaciones de) los valores de la solución del problema*

$$\begin{cases} y' = -2y \\ y(0) = 10 \end{cases}$$

en los puntos: $0, 0.1, 0.2, \dots, 1.9, 2$. Comparar (gráficamente) con la solución exacta $y = 10e^{-2t}$.

```
Command Window
>> f=@(t,y)-2*y;
>> t=0:0.1:2;
>> [t,y]=ode45(f,t,10);
>> plot(t,y,'LineWidth',1.1)
>> hold on
>> plot(t,10*exp(-2*t),'r.')
>> legend('Sol. aproximada','Sol. exacta')
>> shg
>>
>>
```



Sistemas de primer orden

Nos planteamos ahora la resolución de sistemas diferencial ordinarios:

$$\begin{cases} y_1' = f_1(t, y_1, y_2, \dots, y_n), \\ y_1' = f_2(t, y_1, y_2, \dots, y_n), \\ \dots \\ y_1' = f_n(t, y_1, y_2, \dots, y_n). \end{cases} \quad (\text{A.4})$$

y concretamente, de Problemas de Cauchy asociados a ellos, es decir, problemas consistentes en calcular la solución de (A.3) que verifica las condiciones iniciales:

$$\begin{cases} y_1(t_0) = y_1^0, \\ y_2(t_0) = y_2^0, \\ \dots \\ y_n(t_0) = y_n^0. \end{cases} \quad (\text{A.5})$$

Gracias a las características vectoriales del lenguaje de MATLAB, estos problemas se resuelven con las mismas funciones (`ode45`, `ode23`, etc.) que los problemas escalares, basta escribir en el problema en forma vectorial. Para ello, denotamos:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix}, \quad \mathbf{F}(t, \mathbf{Y}) = \begin{pmatrix} f_1(t, y_1, y_2, \dots, y_n) \\ f_2(t, y_1, y_2, \dots, y_n) \\ \dots \\ f_n(t, y_1, y_2, \dots, y_n) \end{pmatrix}, \quad \mathbf{Y}_0 = \begin{pmatrix} y_1^0 \\ y_2^0 \\ \dots \\ y_n^0 \end{pmatrix}, \quad (\text{A.6})$$

con esta notación el problema planteado se escribe:

$$\begin{cases} \mathbf{Y}' = \mathbf{F}(t, \mathbf{Y}) \\ \mathbf{Y}(t_0) = \mathbf{Y}_0 \end{cases} \quad (\text{A.7})$$

y se puede resolver (numéricamente) con la función `ode45`, de forma similar al caso escalar, con las adaptaciones oportunas, como en el ejemplo siguiente.

Ejemplo A.3.16. *Calcular (una aproximación de) la solución del problema*

$$\begin{cases} \begin{cases} y_1' = y_2 y_3, \\ y_2' = -0.7 y_1 y_3, \\ y_3' = -0.51 y_1 y_2, \end{cases} & t \in [0, 5\pi] \\ \begin{cases} y_1(0) = 0 \\ y_2(0) = 1 \\ y_3(0) = 1 \end{cases} \end{cases}$$

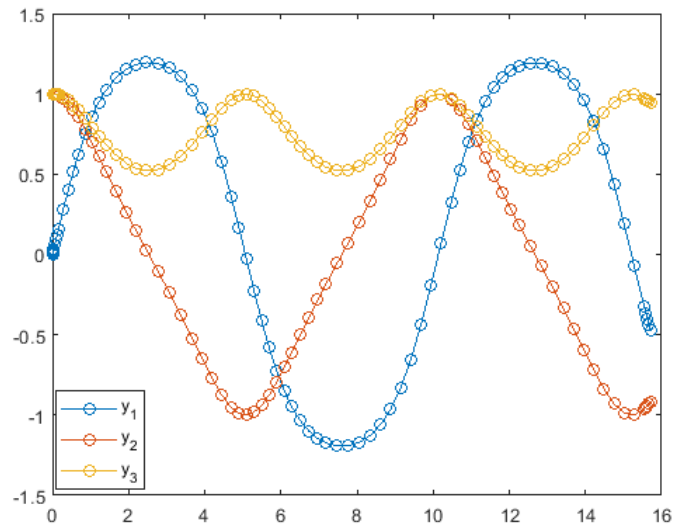
Escribimos el sistema en notación vectorial:

$$\mathbf{Y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \quad \mathbf{F}(t, \mathbf{Y}) = \begin{pmatrix} y_2 y_3 \\ -0.7 y_1 y_3 \\ -0.51 y_1 y_2 \end{pmatrix}, \quad \mathbf{Y}_0 = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \quad (\text{A.8})$$

y escribimos una función anónima para $\mathbf{F}(t, \mathbf{Y})$ y el dato inicial (ambos son vectores columna con tres componentes):

A.3. Introducción a MATLAB

```
Command Window
>> f=@(t,y)[y(2)*y(3); -0.7*y(1)*y(3); -0.51*y(1)*y(2)];
>> y0=[0; 1; 1];
>> ode45(f,[0,5*pi],y0)
>> legend('y_1','y_2','y_3')
>> shg
```

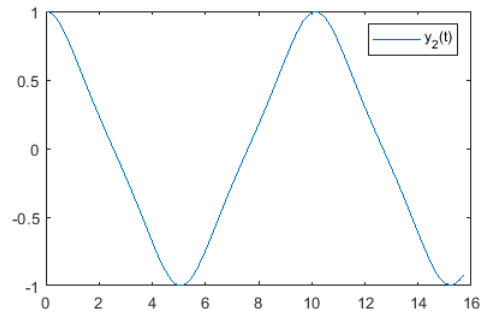


Vamos ahora a utilizar `ode45` recuperando las variables de salida de la aproximación numérica: $[t, y] = \text{ode45}(f, [0, 5\pi], y_0)$; Obsérvese que t es un vector columna y que y es una matriz con tantas filas como t y tres columnas: cada columna es una de las componentes de la solución \mathbf{Y} . es decir

$$y(k, i) \approx y_i(t_k)$$

Si, por ejemplo, sólo quisiéramos dibujar la gráfica de la segunda componente $y_2(t)$ usaríamos la orden `plot` con la segunda columna de y :

```
Command Window
>> [t,y]=ode45(f,[0,5*pi],y0);
>> plot(t,y(:,2))
>> legend('y_2(t)')
>> shg
>>
```



Ecuaciones de orden superior

Plantear la resolución numérica de problemas de Cauchy para ecuaciones diferenciales ordinarias de orden superior a uno, es decir, problemas como:

$$\begin{cases} y^{(n)} = f(t, y, y', \dots, y^{(n-1)}) \\ y(t_0) = y_0 \\ y'(t_0) = y_1 \\ \dots \\ y^{(n-1)}(t_0) = y_{n-1} \end{cases} \quad (\text{A.9})$$

La resolución de este problema se puede reducir a la resolución de un problema de Cauchy para un sistema diferencial de primer orden mediante el cambio de variables:

$$z_1(t) = y(t), \quad z_2(t) = y'(t), \quad \dots \quad z_n(t) = y^{(n-1)}(t).$$

En efecto se tiene:

$$\begin{cases} z_1' = y' = z_2 \\ z_2' = y'' = z_3 \\ \dots \\ z_n' = y^{(n)} = f(t, y, y', \dots, y^{(n-1)}) = f(t, z_1, z_2, \dots, z_{n-1}) \end{cases}$$

$$\begin{cases} z_1(t_0) = y(t_0) = y_0 \\ z_2(t_0) = y'(t_0) = y_1 \\ \dots \\ z_n(t_0) = y^{(n-1)}(t_0) = y_{n-1} \end{cases}$$

Para escribir este sistema en notación vectorial, denotamos:

$$\mathbf{Z} = \begin{pmatrix} z_1 \\ z_2 \\ \dots \\ z_n \end{pmatrix}, \quad \mathbf{F}(t, \mathbf{Z}) = \begin{pmatrix} z_2 \\ z_3 \\ \dots \\ f_n(t, z_1, z_2, \dots, z_{n-1}) \end{pmatrix}, \quad \mathbf{Z}_0 = \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{pmatrix},$$

Con esta notación el problema planteado se escribe:

$$\begin{cases} \mathbf{Z}' = \mathbf{F}(t, \mathbf{Z}) \\ \mathbf{Z}(t_0) = \mathbf{Z}_0 \end{cases} \quad (\text{A.10})$$

y puede ser resuelto usando `ode45` como se ha mostrado anteriormente.

Ejemplo A.3.17. *Calcular y dibujar la solución del problema*

$$\begin{cases} y'' + 7 \sin y + 0.1 \cos t = 0, & t \in [0, 5] \\ y(0) = 0 \\ y'(0) = 1 \end{cases}$$

Siguiendo los pasos antes indicados, este problema se puede reducir al siguiente:

$$\begin{cases} \mathbf{Z}' = \mathbf{F}(t, \mathbf{Z}) \\ \mathbf{Z}(0) = \mathbf{Z}_0 \end{cases}$$

con

$$\mathbf{F}(t, \mathbf{Z}) = \begin{pmatrix} z_2 \\ -7 \sin z_1 - 0.1 \cos t \end{pmatrix}, \quad \mathbf{Z}_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Vamos escribir la función $\mathbf{F}(t, \mathbf{Z})$ como una M-función. También se podría hacer como una función anónima, como en el ejercicio anterior

```
function [dz] = tfg(t,z)
dz=zeros(2,1);
dz(1)=z(2);
dz(2)=-7*z(1)-0.1*cos(t);
```

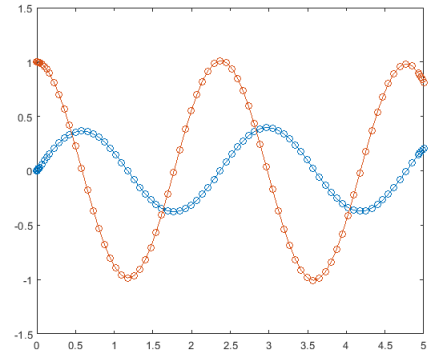
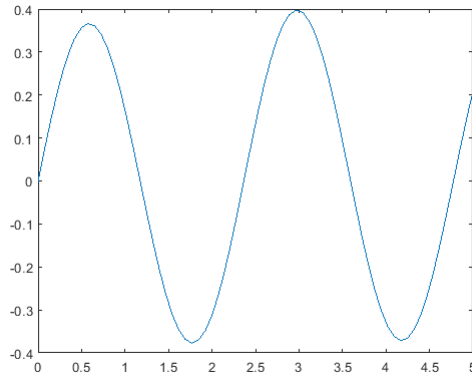
Observación 5. 1. Esta función debe ser guardada en un fichero de nombre `tfg.m` que debe estar en la carpeta de trabajo de MATLAB.

2. El nombre `tfg` es sólo un ejemplo.

3. La orden `dz = zeros(2;1)`; en la función tiene el objetivo de crear la variable `dz` desde un principio con las dimensiones adecuadas: un vector columna con dos componentes.

calculando la solución obtenemos: `ode45(@tfg, [0,5], [0;1])` (figura a la derecha). Pero, a nosotros sólo nos interesa la primera de las componentes de $\mathbf{Z}(t)$, ya que es $y(t) = z_1(t)$. Por lo tanto usaremos `ode45` para recuperar los valores de la solución y dibujaremos únicamente la primera componente ($y = z(:,1)$) (figura a la izquierda):

```
Command Window
>> [t, z]=ode45(@tfg, [0,5], [0;1]);
>> y=z(:,1);
>> plot(t,y)    % o bien, directamente, plot(t, z(:,1))
```



A.4 Explicación (código usado)

Para los resultados numéricos presentados en este trabajo, se utilizó el método clásico de Runge-Kutta de orden 4, cuya fórmula general para el caso escalar es:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad \text{con } h = t_{n+1} - t_n$$

k_1 , k_2 , k_3 y k_4 se expresan por las siguientes ecuaciones

$$\begin{aligned} k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1\right) \\ k_3 &= f\left(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_2\right) \\ k_4 &= f(t_n + h, y_n + hk_3) \end{aligned}$$

Como el modelo matemático utilizado es gobernado por una ecuación diferencial de segundo orden y sabiendo que el método de Runge-Kutta de orden 4 no resuelve una EDO de segundo orden directamente fue necesario escribir el modelo matemático como dos EDOs de primer orden, para hacerse una aproximación por el método de RK.

Usando el MATLAB se escribió a dos funciones como dos M-funciones f1 y f2

```
function valor = f1(u,v,k,p0,A,V0,gamma,m)
```

```
valor = v;
```

y

```
function valor = f2(u,v,k,p0,A,V0,gamma,m)
```

```
valor = -k*u/m + p0*A*(V0^gamma/(V0+A*u)^gamma -1)/m;
```

Que se guardaron en una carpeta de archivo con los nombres f1.m y f2.m respectivamente.

A.4. Explicación (código usado)

En la carpeta principal del programa archivada con un nombre arbitrario `TFG.m` fue donde se insertó los datos necesarios para calcular las aproximaciones por el método de RK 4, así como los componentes de salida de las funciones `f1.m` y `f2.m`, se utilizó la orden `u=zeros(N+1,1)` y `v=zeros(N+1,1)` para crear un vector columna con $N+1$ componentes, como muestra la figura (A.4) del archivo `TFG` que contiene la resolución por el método de RK 4

Para calcular las aproximaciones se utilizó el bucle indexado `for` con la siguiente estructura

```
for i=1:N
    k1u=f1(u(i),v(i),k,p0,A,V0,gamma,m);
    k1v=f2(u(i),v(i),k,p0,A,V0,gamma,m);

    k2u=f1(u(i)+h*k1u/2,v(i)+h*k1v/2,k,p0,A,V0,gamma,m);
    k2v=f2(u(i)+h*k1u/2,v(i)+h*k1v/2,k,p0,A,V0,gamma,m);

    k3u=f1(u(i)+h*k2u/2,v(i)+h*k2v/2,k,p0,A,V0,gamma,m);
    k3v=f2(u(i)+h*k2u/2,v(i)+h*k2v/2,k,p0,A,V0,gamma,m);

    k4u=f1(u(i)+h*k3u,v(i)+h*k3v,k,p0,A,V0,gamma,m);
    k4v=f2(u(i)+h*k3u,v(i)+h*k3v,k,p0,A,V0,gamma,m);

    u(i+1)=u(i)+h*(k1u+2*k2u+2*k3u+k4u)/6;
    v(i+1)=v(i)+h*(k1v+2*k2v+2*k3v+k4v)/6;
    p(i+1)=p0*V0^gamma/(V0+A*u(i+1))^gamma;
end
```

Observación 6. Al comparar el método utilizado en este trabajo (Runge-Kutta clásico de orden 4) con la función `ode45` obtenemos una buena aproximación para los valores de presión, el desplazamiento y la velocidad del pistón, recordar que la función `ode45` se basa en un par de métodos explícitos de Runge-Kutta llamado par de Dormand-Prince, que corresponde a un método de 7 pasos de orden 5 que contiene uno de orden 4. En estos métodos el paso de integración varía para garantizar que el error permanezca por debajo de una tolerancia dada (la tolerancia de error relativa escalar predeterminada `RelTol` es igual a 10^{-3})

```

A=1;           % Área de la camara del gás
m=10;         % [10,20,100,1000] Kg, masa del piston
R=287;        % Constante individual del gas
gamma=1.4;    % Calor específico del gas
V0=1;         % Volume inicial del gás
k=1e7;        % constante elástico del resorte (se mantiene fijo)
f0=sqrt(k/m)/(2*pi); % Frecuencia natural
tau0=1/f0;    % Período natural
disp(['Frecuencia ',num2str(f0),' tau0 ',num2str(tau0)]); %presenta la frecuencia e el periodo natural
%% Condições Iniciais
T=tau0;
N=1000;
u0=0.2;
p0=1e5;
v0=0;
h=T/N;
t=0:h:T;
u=zeros(N+1,1);
v=zeros(N+1,1);
p=zeros(N+1,1);
u(1)=u0;
v(1)=v0;
p(1)=p0*V0^gamma/(V0+A*u(1))^gamma; % Presion inicial del gas

%% Método de Runge-Kutta Orden 4
for i=1:N

%% Salidas (Los resultados)
figure(1); %plot el despalaziamento del piston
plot(t/tau0,u,t/tau0,v)
ylabel('desplaziamento del piston')
xlabel('t/\tau_o')

figure(2); %plot el resultado de la presión del gas
plot(t/tau0,p)
ylabel('Presion (10^5 Pa)')

```

Figura A.3: Carpeta TFG.m

A.4. Explicación (código usado)

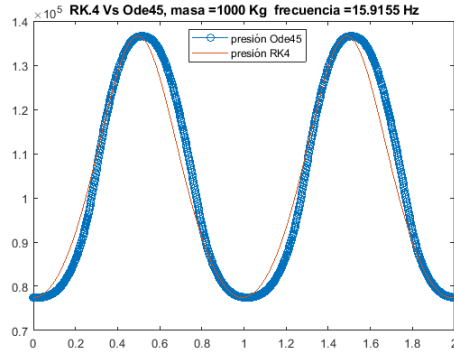


Figura A.4: Presión del gas, $m = 1000$

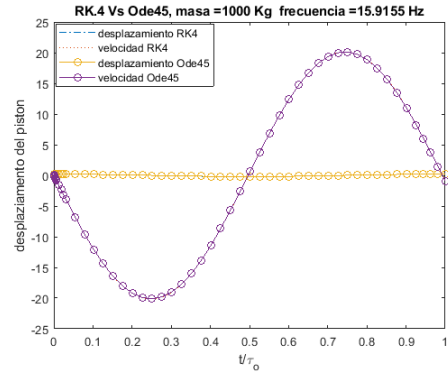


Figura A.5: Desplazamiento del pistón con $m = 1000$

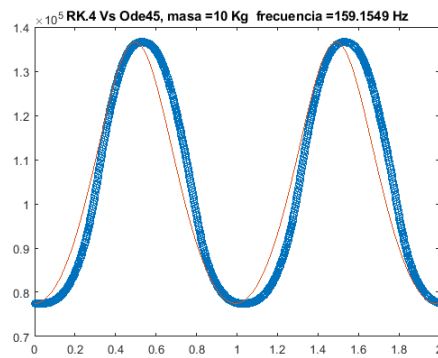


Figura A.6: Presión del gas, $m = 10$

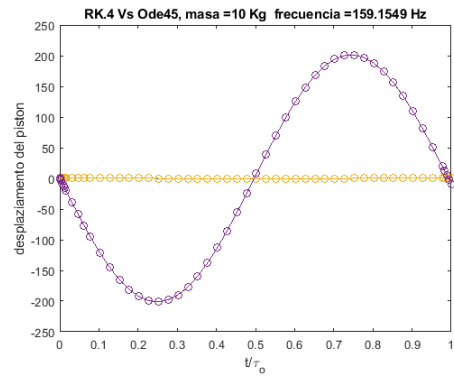


Figura A.7: Desplazamiento del pistón con $m = 10$

Bibliografía

- [1] G. K. BATCHELOR, *An introduction to fluid dynamics*, Cambridge University press, 2000.
- [2] N. BELLAMO and L. PREZIOSE, *Modelling Mathematical Methods and Scientific computation*, CRC Press. Inc, 1995.
- [3] D. M. G. BOGDANOVIĆ, L. IVANOVIĆ, V. GEROSKI, M. RAFAILOVIĆ, *Comparison of Numerical Integration Methods in the Linear Dynamic Analysis*, University of Banja Luka - Faculty of Mechanical Engineering, DEMI 2017.
- [4] W. E. BOYCE and R. C. DIPRIMA, *Equações Diferenciais Elementares e Problemas de Valores de Contorno*, LTC editora, Oitava edição, 2006.
- [5] J. C. BUTCHER *numerical methods for ordinary differential equations*, John Wiley and Sons, Ltd, 2008.
- [6] E. A. CELAYA and J. J. ANZA, *BDF- α : A Multistep Method with Numerical Damping Control*, Universal Journal of Computational Mathematics, 1(3), (2013), 96-108.
- [7] E. A. CELAYA and J. J. A. AGUIRREZABALA, *Solution of the Wave-Type PDE by Numerical Damping Control Multistep Methods*, Elsevier B.V., 29, (2014), 779-789.
- [8] S. E. CHAPRA and R. P. CANALE, *Métodos numéricos para ingenieros con aplicaciones en computadores personales*, McGraw-Hill/InterAmerican de Mexico, 1991.
- [9] R. ECHEVERIA, *Apuntes de MATLAB orientados a métodos numéricos elementales*, Dpto. de Ecuaciones Diferenciales y Análisis Numérico, Universidad de Sevilla, 2017.
- [10] L. EDSBERG, *Introduction to computation and modeling for differential equations*, John Wiley & sons, second edition, 2016.

- [11] R. W. FOX, A. T. McDONALD and P. J. PRITCHARD, *Introdução à Mecânica dos Fluidos*, gen-grupo editorial nacional, Oitava edição, 2014.
- [12] A. GILANT, *Matlab: una introducción con ejemplos prácticos*, Barcelona: Reverté, D.L., 2011.
- [13] T. GÓMEZ-ACEBO, *Física y Química-Termodinámica básica*, CAMPUS TECNOLÓGICO DE LA UNIVERSIDAD DE NAVARRA, 2001.
- [14] G. HOU, J. WANG and A. LAYTON, *Numerical Methods for Fluid-Structure Interaction-A Review*, Global-Science Press, 12, (2012) 337-377.
- [15] E. LEFRANÇOIS and J. P. BOUFFLET, *An introduction to Fluid-Structure Interaction: Application to the Piston Problem*, SIAM REVIEW, 52, (2010) 747-767.
- [16] H. W. LIEPMANN and A. ROSHKO, *Elements of Gas Dynamics*, Courier Corporation, 2013.
- [17] R. L. K. MOORTHY, A. KAKODKAR, H. R. SRIRANGARAJAN and S. SURYANARAYAN *An assessment of the Newmark Method for solving Chaotic Vibrations of Impacting Oscillators*, Computers & Structures, Vol. 49, 4, 597-603, 1993.
- [18] N. M. NEWMARK and S. P. CHAN, *A comparison of numerical methods for analyzing the Dynamic Response of Structures*, UNIVERSITY OF ILLINOIS, 1952.
- [19] A. QUARTERONI, F. SALERI and P. GERVASIO, *Scientific Computing with MATLAB and Octave*, Springer, third edition, 2010.
- [20] R. M. RODRIGUEZ, *Números Adimensionales*, academia.edu, 2011.
- [21] R. J. SANTOS, *Introdução às equações diferenciais ordinárias*, Universidade Federal de Minas Gerais, 2011.
- [22] N. G. TAPIA, *Ingeniería fluidomecánica*, Universidad de Valladolid, Secretariado de Publicaciones e Intercambio Editorial, 2ª edición, 2002.
- [23] E. CHACON VERA, *Métodos Numéricos para las ecuaciones diferenciales. Apuntes de clase*, 2018.