Informática

1^o Grao Matemáticas
Guía Docente

1. Datos descriptivos de la materia

1.1. Descripción de la materia

La materia "Informática" pertenece al Grado en Matemáticas, definido por la Memoria de Grado[1] aprobada por la ANECA el 22 de abril de 2008, e impartido desde el curso 2008/2009 en la Facultad de Matemáticas de la Universidad de Santiago de Compostela (USC). Según esta Memoria, la materia tiene un carácter obligatorio, pertenece al 1^{er} cuatrimestre del 1^{er} curso del Grado, y está enmarcada en el módulo de "Formación básica transversal", con un contenido de 6 créditos ECTS. Su código en la Universidad de Santiago es G10111039. Los **Contenidos** fijados por la Memoria de Grado para esta asignatura son los siguientes:

- Introducción a un paquete de **cálculo simbólico** de uso en el centro: elementos básicos, ejemplos sencillos en Matemáticas, representación gráfica de curvas y superficies.
- Introducción a un paquete de **cálculo numérico** de uso en el centro: elementos básicos, ejemplos en Matemáticas (operaciones con polinomios, cálculo matricial, representación de funciones, integración, ...).
- Sistema operativo del entorno de programación de uso en el centro.
- Lenguaje de **programación estructurada** de uso en el centro: elementos básicos, bucles, instrucciones de control, programación modular. Representación de números en el ordenador.
- Programación e implementación de algoritmos de resolución de problemas matemáticos básicos en análisis, álgebra, combinatoria ....

No se especifican en la Memoria de Grado requisitos previos (sólo los generales para entrar a cursar esta titulación), ni tampoco hay indicaciones metodológicas propias para la asignatura. La evaluación debe seguir el criterio general (establecido para todas las asignaturas) con el añadido de que el examen final debe realizarse en el ordenador. La memoria especifica también las actividades formativas asociadas a esta asignatura, con su contenido en horas presenciales y de trabajo del alumno, que se indican en el cuadro 1. La materia está siendo impartida actualmente por el Área de Ciencias de la Computación e Inteligencia Artificial, perteneciente al Departamento de Electrónica y Computación. La página web de la asignatura es:

`www-gsi.dec.usc.es/~delgado/informatica`

[1] `www.usc.es/mate/02documentos/documentos/Memoria_Final_Corregida_Grado_Mate_USC.pdf`

TRABAJO PRESENCIAL EN EL AULA	Horas	TRABAJO PERSONAL DEL ALUMNO	Horas
Clases expositivas en grupo grande	15	Estudio autónomo individual o en grupo	30
Clases expositivas en grupo reducido	–	Escritura de ejercicios, conclusiones u otros trabajos	10
Clases con ordenador/laboratorio en grupo reducido (interactivas)	30	Programación/experimentación u otros trabajos en ordenador/laboratorio	50
Tutorías en grupo reducido sin ordenador/laboratorio	–	Lecturas recomendadas, actividades en biblioteca o similar	–
Tutorías en grupo reducido con ordenador/laboratorio (interactivas)	13	Preparación de presentaciones orales, debates o similar	–
Tutorías en grupos muy reducidos o individualizadas	2	Asistencia a charlas, exposiciones u otras actividades recomendadas	–
Otras sesiones con profesor Especificar:	–	Otras tareas propuestas por el profesor Especificar:	–
Total horas trabajo presencial en el aula	60	Total horas trabajo personal del alumno	90

Cuadro 1: Actividades presenciales y personales del alumno fijadas por la Memoria de Grado para la materia "Informática".

Al tratarse de una materia del primer cuatrimestre del primer curso, los alumnos acaban de ingresar en la Universidad y sus conocimientos informáticos son básicos. De hecho, sólo una pequeña parte de ellos cursaron asignaturas (opcionales) de Informática en el Bachillerato, y tan sólo uno o dos alumnos en cada curso poseen conocimientos de programación en algún lenguaje, con frecuencia Visual Basic o C. Por lo tanto, su visión de la Informática es la de usuarios familiarizados con el entorno Windows y las aplicaciones típicas (ofimática, Internet, multimedia, etc.), sin conocimientos de programación ni de Informática aplicada a las Matemáticas. El número de alumnos es de 80 aproximadamente, distribuidos en dos grupos de clases expositivas (40 alumnos por grupo) y 4 grupos de clases interactivas (20 alumnos por grupo). Las clases interactivas se realizan en las aulas de Informática de la Facultad de Matemáticas.

1.2. Pre-requisitos

Como se indica en la Memoria de Grado, no existen pre-requisitos esenciales para cursar esta materia. No obstante, resulta recomendable una mínima destreza en el manejo del ordenador (teclado, ratón) y del entorno Windows, así como los conocimientos matemáticos básicos impartidos en el Bachillerato, cuya utilización será constante porque gran parte de la asignatura consistirá en realizar en el ordenador cálculos que hasta ahora los alumnos realizaban en el papel (por ejemplo, derivadas e integrales, ecuaciones de segundo grado, sistemas de ecuaciones, cálculos de geometría euclídea, vectoriales y matriciales, entre otros).

2. Sentido de la materia en el perfil de la titulación

Esta asignatura tiene un carácter instrumental, ya que proporciona las competencias para que el alumno realice en el ordenador cálculos matemáticos necesarios para las asignaturas posteriores de la titulación. En este apartado se contextualiza la materia dentro del Grado en Matemáticas.

2.1. Materias relacionadas

La Memoria de Grado sitúa la materia "Informática" en el módulo "Formación básica transversal", compuesto por 30 créditos repartidos en las siguientes asignaturas:

- **Informática**, 6 créditos, 1^{er} curso, 1^{er} cuatrimestre.
- **Lenguaje matemático, conjuntos y números**, 6 créditos, 1^{o} curso, 1^{er} cuatrimestre.
- **Química básica**, 6 créditos, 1^{er} curso, 1^{er} cuatrimestre.
- **Biología básica**, 6 créditos, 1^{er} curso, 2^{o} cuatrimestre.
- **Física básica**, 6 créditos, 2^{o} curso, 1^{er} cuatrimestre.

Sin embargo, no es con las materias de este módulo con las que existe una mayor vinculación. Obviamente, no con "Química básica", "Biología básica" ni con "Física básica". Tampoco con "Lenguaje matemático, conjuntos y números", materia orientada a proporcionar una base matemática teórica de tipo algebráica (sus contenidos incluyen técnicas de demostracion, lógica, conjuntos, relaciones, inducción matemática, entre otros) en la cual el uso del ordenador no es fundamental. En realidad las asignaturas más estrechamente relacionadas con "Informática" son las pertenecientes a los módulos "**Métodos numéricos**" y, en menor medida, "**Análisis matemático en una variable**". En las materias de métodos numéricos, el ordenador es una herramienta básica para la resolución de problemas que no admiten una solución analítica (ecuaciones y sistemas de ecuaciones no lineales, ecuaciones diferenciales, cálculo de integrales, etc.). Este módulo, que representa 18 créditos, está compuesto por las tres asignaturas siguientes (todas ellas obligatorias con 6 créditos):

- **Cálculo numérico en una variable**, del 1^{er} cuatrimestre del 2^{o} curso. En esta asignatura se analizan los errores en el cálculo numérico, métodos numéricos para la aproximación iterativa de las raíces de una ecuación numérica (algoritmos de dicotomía, iteración funcional y Newton-Raphson). También estudia la interpolación polinómica (fórmula de Lagrange), y realiza una introducción a la integración y derivación numéricas.
- **Análisis numérico matricial**, del 2^{o} cuatrimestre del 2^{o} curso. Esta materia estudia los cálculos con matrices: normas, radio espectral, cociente de Rayleigh; resolución directa de sistemas de ecuaciones lineales por los métodos de Gauss, descomposición LU, pivote parcial, factorizaciones de Cholesky y QR, método de Householder; aproximación numérica de autovalores y autovectores por los métodos de potencia iterada y potencia iterada inversa; resolución numérica iterativa de sistemas lineales y no lineales mediante los métodos del punto fijo, Jacobi, Gauss-Seidel, relajación y Newton-Raphson.
- **Métodos numéricos en optimización y ecuaciones diferenciales**, del 1^{er} cuatrimestre del 3^{er} curso. Estudia los métodos de optimización sin restricciones (método de gradiente y sus variantes) y con restricciones (métodos de Lagrange, penalización y gradiente con proyección); aproximación polinómica y trigonométrica de funciones por mínimos cuadrados; ajuste de datos mediante ecuaciones normales; resolución numérica de problemas de valor inicial mediante los métodos de Euler explícito e implícito, Runge-Kutta y multipaso; métodos de diferencias finitas para las ecuaciones de Poisson y del calor.

La Memoria de Grado incluye entre los pre-requisitos de estas tres asignaturas el conocimiento de un **lenguaje de programación estructurado** y de un **programa de cálculo numérico** general, material que se presupone impartido en la materia "Informática". En cuanto al módulo "Análisis matemático de una variable", existe una relación importante con la materia "**Introducción al análisis matemático**", básica de rama, con 6 créditos que se cursa simultáneamente con la asignatura Informática (1^{er} curso, 1^{er} cuatrimestre), en la cual se utiliza extensivamente un **programa de cálculo simbólico**, aunque no se incluye explícitamente entre los pre-requisitos.

2.2. Papel de la asignatura en el plan de estudios de Grado

A partir de las consideraciones del apartado anterior, la materia "Informática" debe proporcionar las bases informáticas y de programación que serán usadas en las cuatro materias anteriores:

1. Manejo del entorno proporcionado por el **sistema operativo** a nivel de usuario: aplicaciones básicas, gestión de archivos, navegación por la red, etc. La práctica totalidad de los alumnos domina ya estas competencias en el entorno Windows. No obstante, y dado que buena parte de las asignaturas relacionadas se usa **GNU/Linux**[2] como entorno de trabajo, en Informática introduciremos al alumno en este entorno, con el que no está familiarizado en general, aumentando así sus competencias en el manejo de sistemas informáticos, específicamente de software libre.

2. Manejo de un paquete informático para el **cálculo simbólico** en Matemáticas usado en el centro (Facultad de Matemáticas). Este tipo de programas permiten el manejo de números de distintos tipos (enteros, racionales, irracionales, reales y complejos), la resolución de ecuaciones y sistemas de ecuaciones lineales y no lineales, la simplificación y expansión de expresiones y, en general, el manejo de funciones de todo tipo: representación gráfica, derivación e integración, etc. En nuestro caso, el paquete de cálculo simbólico elegido es **Maple**[3], de uso muy extendido en este campo y, en concreto, en la Facultad de Matemáticas. Se trata de un paquete propietario, aunque existen otras alternativas de carácter libre como Maxima[4].

3. Conocimiento de un **lenguaje de programación estructurada** orientado a la resolución de problemas matemáticos, y de los conceptos necesarios para analizar problemas de este tipo y diseñar e implementar algoritmos con un cierto nivel de complejidad. El objetivo es que el alumno pueda programar de forma estructurada la resolución de algunos problemas ya tratados con las técnicas de cálculo simbólico del apartado anterior. Entre los distintos lenguajes de programación, se ha elegido **Fortran 90** por su amplio uso en el centro, y por ser el lenguaje más orientado al campo de las Matemáticas desde su aparición en los años 50. En este caso, existen compiladores de Fortran libres, como gfortran[5] y g95[6].

4. Manejo de un paquete para la realización de **cálculos numéricos** y científicos que aporte la base para las asignaturas de cálculo numérico de cursos posteriores y permita la resolución de problemas matemáticos. El paquete de cálculo numérico de uso más amplio en el

[2] `www.gnu.org, www.linux.org`
[3] `www.maplesoft.com`
[4] `maxima.sourceforge.net`
[5] `www.gfortran.org`
[6] `www.g95.org`

centro es **Matlab**[7], que además tiene gran popularidad como lenguaje de programación científico e ingenieril. Existen, no obstante, alternativas libres como Octave[8].

Al enmarcarse en un Grado en Matemáticas, la práctica totalidad de los ejercicios que se realizan resuelven problemas matemáticos: cálculo y representación de funciones, límites, derivadas, series numéricas, integrales, manipulación de vectores y matrices, resolución de ecuaciones y sistemas, manipulación algebráica de expresiones, etc. De este modo, la materia no sólo proporciona resultados de salida en forma de manejo de herramientas informáticas para otras materias del Grado, sino que también usa entradas (conocimientos) procedentes de otras ramas de las Matemáticas (Análisis Matemático y Álgebra), que el alumno conoce ya a partir del Bachillerato.

2.3. Interés de la materia para la profesión

Los fundamentos informáticos que se aportan en esta materia son de interés claro para el ejercicio profesional, ya que en casi cualquier trabajo se requiere el manejo cotidiano del ordenador. En el caso de un graduado en Matemáticas, debe ser capaz de realizar cálculos matemáticos avanzados de tipo simbólico o numérico, y de manejar algún lenguaje de programación estructurado y científico, que probablemente sea alguno de los estudiados en esta materia (Fortran y Matlab). Aún en el caso de usar otro lenguaje (C/C++, Java, Octave, Pascal, ...), el conocimiento de Fortran y Matlab facilitará en gran medida su aprendizaje, aportando una base para su actualización posterior. Además, el alumno se familiariza con el sistema operativo Linux, como alternativa al entorno Windows, que la mayoría de ellos ya conocen. Por otra parte, si el graduado opta por la carrera docente en la enseñanza secundaria, los profesores de Matemáticas asumen con frecuencia la docencia en Informática y Tecnología, en las cuales el manejo del ordenador y la programación ocupan un lugar relevante.

2.4. Materia en otras universidades

Seguidamente se revisan las materias equivalentes a la de Informática en las titulaciones de Matemáticas de otras universidades españolas y europeas.

- **Univ. Complutense de Madrid** (`www.ucm.es`). Grado en Matemáticas. Materia de *Informática* 1^{er} curso, anual con 7.5 créditos (1.5 expositivas, 2.8 problemas, 2.7 prácticas). Contenidos: Introducción a la Informática. Programación estructurada (en Pascal y Python): expresiones, condicionales, bucles. Recursión. Tipos estructurados. Entorno de trabajo: GNU/Linux.

- **Univ. Politécnica de Catalunya** (`www.upc.es`). Grado en Matemáticas. Materia de *Informática* con 7.5 créditos, 1^{er} curso, 1^{er} cuatrimestre. Contenidos: Estructura del ordenador, procesos e instrucciones. Variables e instrucciones elementales. Secuencias. Funciones. Datos no elementales. Tuplas y clases. Límites de la computación.

- **Univ. de Valencia** (`www.uv.es`). Licenciatura en Matemáticas. Hay una materia troncal anual (1^{er} curso) de *Informática* con 10.5 créditos (6T + 4.5P) que usa C++, con los siguientes temas: Estructura interna del ordenador. Lenguajes de programación. Sistemas operativos. Algoritmos y programas. Aritmética de computadores y representación interna de la información. Tipos compuestos y estruturas de datos. Archivos y bases de datos. Estudio de algoritmos y de su complejidad.

[7] `www.mathworks.com`

[8] `www.octave.org`

- **Univ. de Sevilla** (`www.us.es`). Licenciatura en Matemáticas. Materia de *Informática* (1^{er} curso, 1^{er} cuatrimestre) con 9 créditos (6T + 3P) que usa el lenguaje de programación Scheme e incluye los siguientes temas: datos y operadores; procedimientos; recursión; tipos abstractos de datos; listas; recursión sobre datos; entornos locales; abstracción de datos; procedimientos locales; abstracción de procedimientos; asignaciones; vectores; ordenación y búsqueda; programación interactiva; archivos; aplicaciones matemáticas.

- **Univ. Autónoma de Madrid** (`www.uam.es`). Grado en Matemáticas. No incluye una asignatura de Informática, sino que hay una materia de *Laboratorio* anual del 1^{er} curso con 6 créditos, que tiene tres bloques asociados con otras asignaturas (álgebra, análisis y estadística) y dos bloques asociados respectivamente a "Algoritmos y programación" (lógica, condicionales, bucles, recursión) y "Algoritmos avanzados" (listas, diccionarios y conjuntos; matrices densas y dispersas; análisis de algoritmos; algoritmos combinatoriales). Usa el lenguaje de programación Python y el paquete de cálculo simbólico Sage[9].

- **Univ. do Porto** (`www.uporto.pt`). Licenciatura en Matemáticas. Materia de *Introducción á programación* (1^{er} curso, 1^{er} cuatrimestre) con 7.5 créditos. Contenidos: Introducción a los computadores: hardware, software, sistemas operativos e Internet. Lenguajes de programación. Datos. Variables. Operaciones aritméticas. Expresiones. Operadores. Prioridades. Procedimientos. Módulos. Compiladores e intérpretes. Variables globales y locales. Ejecución condicional. Ciclos. Flujo de ejecución. Cadenas de caracteres. E/S por archivos. Estructuras de datos: listas, vectores, matrices. Librerías. Desarrollo, depuración y eficiencia de algoritmos. Resolución de problemas con el ordenador.

- **Univ. de Cambridge**. No hay una asignatura de Informática propiamente dicha, sino un curso de *Proyectos computacionales* (*Computer-Aided Teaching of All Mathematics* - CATAM[10]) con docencia en primer curso pero realizados durante el segundo curso, usando C y Matlab. El material está muy orientado hacia problemas físicos.

- **Univ. de Oxford** (`www.ox.ac.uk`). Ofrece varias licenciaturas de Matemáticas, entre ellas la de "Matemáticas y Computación", que incluye contenidos en Informática con un nivel obviamente superior al necesario en una titulación de Matemáticas. Así, por ejemplo, el primero curso incluye materias de programación funcional, análisis y diseño de algoritmos y programación imperativa, representando un 50 % del curso. En el segundo curso hay asignaturas de programación orientada a objetos, concurrencia y modelos de computación, entre otros.

- **Univ. de Manchester** (`www.maths.manchester.ac.uk`). Oferta licenciaturas de Matemáticas puras y Matemáticas combinadas con Gestión empresarial, Filosofía, Física, Estatística y Finanzas. En la licenciatura de Matemáticas no hay una materia específica de Informática, pero la materia *Mathematical Workshop II* (de 10 créditos) usa Matlab para resolver problemas mateḿaticos (ecuaciones, representación de funciones, manejo de matrices, programación básica), especialmente de tipo estadístico.

- **Universidad de Bolonia** (`www.unibo.it`). Ofrece un Grado en Matemáticas con una materia de Informática (1^{er} curso, 2^{er} cuatrimestre) con 8 créditos en la que se usa C como lenguaje de programación. Contenidos: Arquitectura de ordenadores. Algoritmos para cálculo. Tipos y estructuras de datos. Estructuras de control. Subprogramas. Estructuras de datos dinámicas. Recursividad. Sistemas operativos. Programación en C.

[9] `www.sagemath.org`

[10] `www.maths.cam.ac.uk/undergrad/catam`

Se puede apreciar un alto perfil "informático" (en oposición al perfil "matemático") en todos los casos estudiados, especialmente en las universidades españolas. Sólo en la Autónoma de Madrid se usa un paquete de cálculo simbólico (Sage), y en Manchester se usa Matlab, pero en el resto de los casos se usa únicamente un lenguaje de programación estructurado, usualmente C, pero también Pascal, Python y Scheme, un lenguaje de programación lógica derivada de Lisp. Se trata, por tanto, de lenguajes de propósito general (caso de C) o con un perfil claramente informático en el resto de los casos (más aún en el caso de Scheme). No son lenguajes matemáticos, ni están orientados al cálculo numérico, a diferencia de Fortran y Matlab, para los que existe una ingente cantidad de código de tipo matemático disponible. Por otra parte, las sintaxis de estos lenguajes (en especial C) utilizan gran cantidad de símbolos especiales (puntuación, asteriscos, llaves, dólar, etc.) y es por ello sensiblemente más compleja que la de Fortran o Matlab. Esto constituye un inconveniente importante para alumnos recién ingresados en la Universidad, sin conocimientos de programación, que deben dedicar más esfuerzos a asimilar las convenciones sintácticas que a trabajar en el diseño de algoritmos para resolver problemas. En este sentido, las sintaxis de Fortran y Matlab figuran entre las más simples, en comparación con otros lenguajes empleados en los primeros cursos de Ingeniería Informática (Pascal, Modula, entre otros) y, por lo tanto, resultan muy aptas para no informáticos en una primera aproximación a la programación.

Las diferencias también se manifiestan en los contenidos. En la mayoría de las universidades analizadas la práctica totalidad del tiempo se dedica a la algorítmica y programación (variables, expresiones, control de flujo, subprogramas, recursividad), sin dedicar recursos al cálculo simbólico o numérico. También aparecen contenidos de estrutura de ordenadores, hardware y software, sistemas operativos, redes y representación interna da información (Politécnica de Cataluña, Valencia, Porto y Bolonia). Finalmente, es frecuente la presencia de apartados sobre estructuras de datos (Complutense, Politécnica de Cataluña, Valencia, Sevilla, Autónoma de Madrid, Porto y Bolonia). Sin embargo, en muchos casos no hay mención alguna a las aplicaciones de tipo matemático (Complutense, Politécnica de Cataluña, Valencia y Autónoma de Madrid), y en otras se realiza una mención al final (Sevilla y Porto). Las excepciones son Manchester (muy orientada a temas estadísticos) y Cambridge (con un perfil muy físico, pero con pocos contenidos de tipo informático). Finalmente, en Oxford, aunque no hay muchos detalles disponibles, al tratarse de una titulación mixta Matemáticas - Computación se introducen conceptos específicamente informáticos (programación orientada a objetos y programación funcional, entre otros), que no son de gran utilidad para un matemático, y menos en una materia básica de Informática orientada al cálculo simbólico y numérico.

3. Objetivos y competencias

El objetivo de esta asignatura es formar al alumno para realizar cálculos matemáticos complejos con el ordenador en las restantes materias del Grado, en especial aquellas con las que mantiene una relación importante (apartado 2.1). Con esta idea en mente, podríamos formular los siguientes **objetivos**:

Objetivo 1. Realizar la mayoría de las operaciones matemáticas conocidas a partir del Bachillerato (y algunas otras que conocerá en los primeros cursos del Grado en Matemáticas) utilizando una herramienta de cálculo simbólico (Maple). Concretamente, debe poder resolver (exactamente) cálculos básicos, ecuaciones de 1^o y 2^o grado y sistemas de ecuaciones lineales, cálculos con vectores y matrices, límites y series numéricas, derivadas, integrales, y representar gráficamente funciones de una y dos variables.

Objetivo 2. Desarrollar programas básicos para resolver (numéricamente) algunos de estos cálculos, usando un lenguaje de programación estructurada de nivel medio (Fortran). Es necesario, por tanto, conocer la sintaxis del lenguaje, pero también aprender a "pensar" un método (algoritmo) para la resolución del problema y ser capaz, llegado el caso, a usar otro lenguaje para resolver el mismo problema.

Objetivo 3. Usar una herramienta de cálculo numérico para realizar operaciones matemáticas, elaborar representaciones gráficas y escribir programas en un lenguaje de programación estructurado de alto nivel (Matlab) para resolver problemas matemáticos complejos.

Para alcanzar estos objetivos, el alumno debe adquirir una serie de **competencias**, consistentes en saber usar el ordenador (a través de los programas de aplicación correspondientes) para:

Competencia 1. Realizar cálculos matemáticos básicos con números, vectores y matrices, y representar gráficamente funciones. Esta competencia está asociada a los objetivos 1, 2 y 3.

Competencia 2. Manipular algebráicamente funciones, especialmente funciones polinómicas y racionales (fracciones de polinomios), realizando simplificaciones, expansiones, composiciones, etc. Esta competencia está asociada al objetivo 1.

Competencia 3. Realizar simbólica y numéricamente operaciones de análisis matemático con funciones de una y varias variables: límites, derivación, integración definida e indefinida, suma de series y productos infinitos. Esta competencia está asociada a los objetivos 1, 2 y 3.

Competencia 4. Resolver numérica y simbólicamente ecuaciones y sistemas de ecuaciones lineales y no lineales, en concreto encontrar raíces de polinomios. Esta competencia está asociada a los objetivos 1 y 3.

Competencia 5. Diseñar algoritmos que resuelvan problemas de análisis matemático, álgebra y cálculo numérico básicos: aproximación de raíces de ecuaciones no lineales, integrales definidas e indefinidas, derivadas, cálculo vectorial y matricial, y suma de series numéricas finitas e infinitas. Esta competencia está asociada a los objetivos 2 y 3.

Competencia 6. Desarrollar un programa de tamaño medio (es decir, con varios subprogramas) que resuelva un problema matemático en un lenguaje de programación estructurado. En este contexto, "desarrollar" significa escribir, compilar y depurar el programa, que además debe tener una eficiencia aceptable en términos de consumo de memoria RAM y tiempo de computación. Esta competencia está asociada a los objetivos 2 y 3.

Competencia 7. Almacenar datos en, y leer datos a partir de, archivos de disco desde un programa, y configurar el formato de estos datos. Esta competencia está asociada a los objetivos 2 y 3.

La figura 1 muestra estos objetivos y competencias, junto con las relaciones entre ellos. Además de estas competencias, directamente asociadas a los contenidos de la materia, se desarrollan otras de tipo **instrumental**, como son:

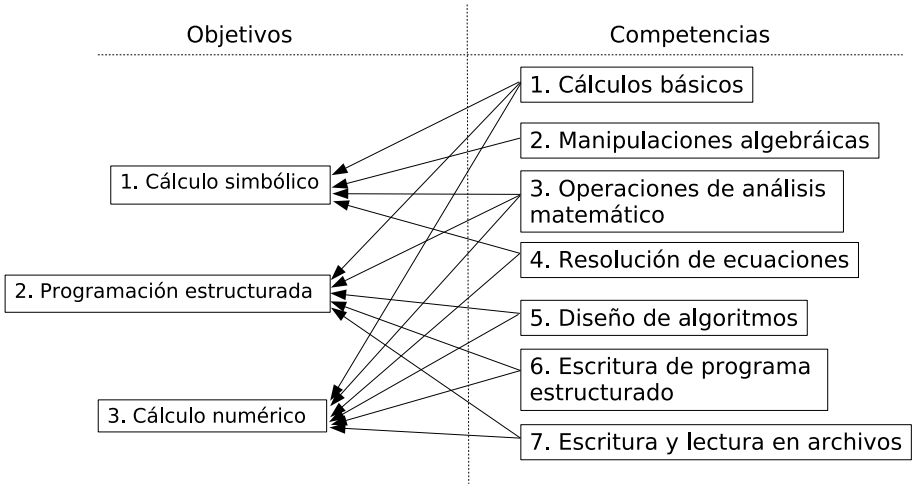

Figura 1: Relaciones entre objetivos y competencias.

1. **Competencia Instrumental 1**. Manejo del ordenador y del sistema operativo a nivel de usuario y programador. A nivel de usuario, el alumno debe familiarizarse con la gestión de archivos, edición de textos, navegación y búsqueda de documentación por la red, entre otras. A nivel de programador, el alumno debe ser capaz de detectar y corregir errores de compilación, rastrear errores de ejecución y, en general, manejar herramientas para la depuración de programas.

2. **Competencia Instrumental 2**. Capacidad para la resolución de problemas y la toma de decisiones de diseño en el desarrollo de algoritmos y programas, ponderando aspectos del programa como la velocidad de ejecución y el consumo de recursos (memoria RAM, archivos de disco, etc.).

3. **Competencia Instrumental 3**. Capacidad para la actualización continua en el ámbito informático a los nuevos sistemas operativos y entornos de programación.

4. **Competencia Instrumental 4**. Manejo de idiomas para la consulta de documentación en inglés, usual en el campo informático.

Finalmente, otras competencias **sistémicas** que podrían considerarse asociadas al mundo de la programación, como son la **gestión de proyectos** informáticos o el **trabajo en grupo**, usual en la realización de estos proyectos, no tienen presencia en esta materia. La razón es que, al tratarse de una asignatura de primer curso, el nivel de complejidad exigible en los programas a desarrollar no puede ser excesivamente elevado, más aún si tenemos en cuenta que en la misma asignatura se manejan tres entornos de trabajo (y tres sintaxis) distintos. Por lo tanto, la realización de un proyecto de complejidad elevada que requiera trabajo en grupo no es asumible, y queda pospuesto para materias posteriores.

4. Programa de la materia

En este apartado se describen los programas asociados a las clases expositivas e interactivas, que reflejan los contenidos indicados por la Memoria de Grado. La principal diferencia entre ambos radica en el apartado de cálculo simbólico. En efecto, el programa de cálculo simbólico usado (Maple) es muy amigable y fácil de usar, y proporciona comandos simples pero muy potentes. Por esta razón, y dado su carácter eminentemente práctico, no resulta operativo dedicar clases expositivas a Maple, sino más bien trabajarlo durante las clases interactivas. De este modo, resulta posible dedicar la totalidad de las clases expositivas a las partes de programación estructurada (Fortran) y cálculo numérico (Matlab), donde la complejidad conceptual es mayor por introducir nociones de algorítmica y programación. De hecho, la experiencia revela que este campo es el más problemático para los alumnos, por su escasa o nula familiaridad con el tema. Por otra parte, se facilita así el desarrollo temporal de las clases interactivas de programación estructurada en Fortran, ya que en el momento en que éstas comienzan, las clases expositivas de este tema se encuentran ya avanzadas.

4.1. Programa de las clases expositivas

Las clases expositivas representan 15 horas presenciales, y están asociadas a 30 horas de trabajo del alumno, con un total de 1.8 créditos. Sus contenidos se estructuran en los siguientes temas:

1. Programación estructurada en Fortran (8 horas de clase, 16 horas de trabajo del alumno, 1 crédito)

   a) Metodología de la programación: análisis, diseño, codificación y depuración

   b) Estructura básica de un programa. Entrada y salida estándar

   c) Tipos de datos elementales

   d) Vectores y matrices estáticos y dinámicos

   e) Expresiones aritméticas. Sentencias de asignación

   f) Operadores relacionales y lógicos

   g) Sentencias de selección

   h) Sentencias de iteración

   i) Subprogramas: funciones externas y subrutinas

   j) Entrada y salida a archivos. Formatos

2. Cálculo numérico con Matlab (7 horas de clase, 14 horas de trabajo del alumno, 0.8 créditos)

   a) Archivos script. Variables. Entrada y salida básica

   b) Archivos de función

   c) Sentencias de selección e iteración definida e indefinida

   d) Entrada y salida de datos a archivos

   e) Manejo de vectores y matrices

   f) Comandos de cálculo numérico

g) Operaciones con polinomios

h) Representación gráfica

El hecho de impartir la programación estructurada en Fortran antes del cálculo numérico con Matlab responde al nivel de ambos lenguajes (medio en Fortran, alto en Matlab) y a los conceptos de algorítmica y programación que ambos utilizan. En este sentido, resulta más instructivo y fácil para el alumno conocer primero un lenguaje de nivel medio (Fortran) y después pasar a otro de nivel alto (Matlab). De este modo, se asimilan primero los conceptos de bajo nivel (por ejemplo, la existencia de distintos tipos de datos, la declaración de variables, la reserva de memoria o las estructuras iterativas), que en Fortran deben gestionarse directamente, y después el paso a Matlab es menos traumático (por ejemplo, no exige la declaración de variables, todos los datos numéricos son del mismo tipo, no hay reserva de memoria y permite sustituir algunos bucles por estructuras vectorizadas). La alternativa inversa, que consistiría en estudiar primero Matlab e luego pasar a Fortran, sería más problemática, ya que el alumno se acostumbraría a programar "fácilmente" en Matlab, y después tendría que pasar a Fortran, que es más complicado y requiere un mayor esfuerzo, agravado por tratarse de la última parte del cuatrimestre.

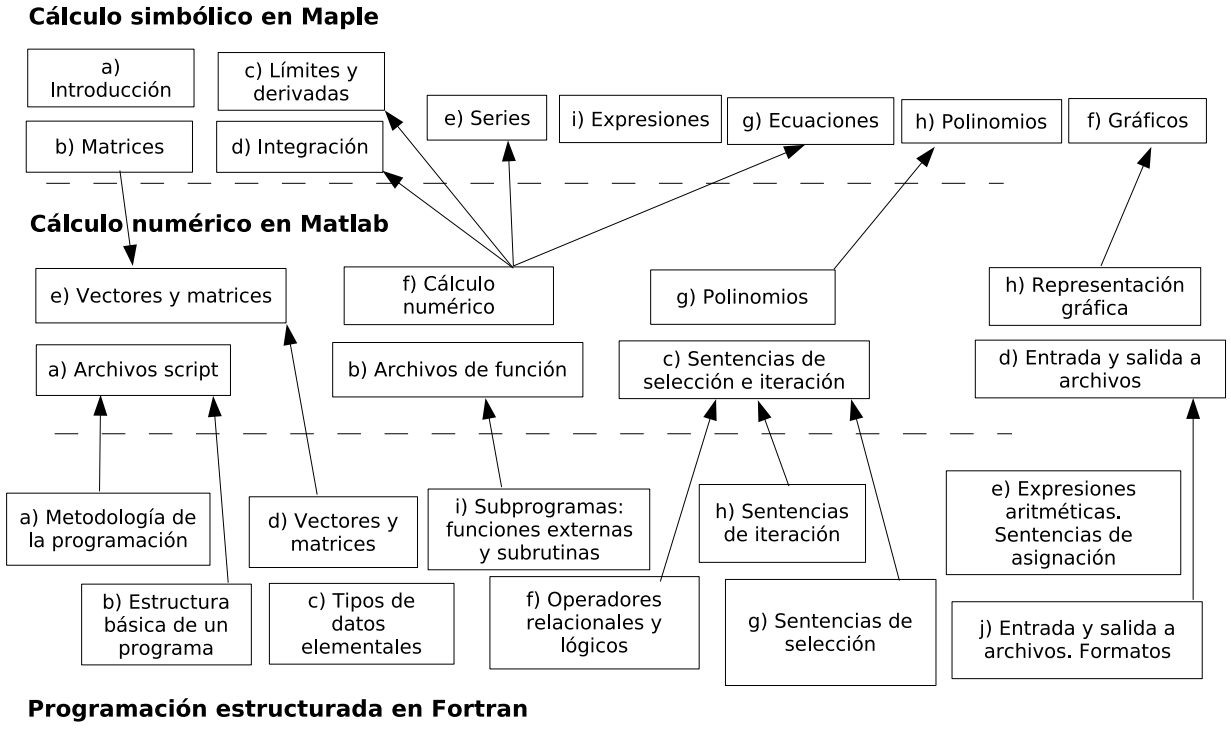

Figura 2: Diagrama de relaciones entre apartados del programa de clases expositivas.

La figura 2 muestra las relaciones entre los distintos apartados. Existen varias relaciones entre los apartados de programación en Fortran y sus correspondientes de Matlab, así como entre los apartados de cálculo numérico y simbólico y representación gráfica en Maple y Matlab, de modo que Matlab se puede valorar como una herramienta a medio camino entre Maple y Fortran.

Resulta interesante adelantar el **nivel de dificultad** asociado a cada apartado, con el fin de romper el "encefalograma plano" e indicarle al alumno cuáles son los más difíciles o más importantes para la materia. El tema de **Programación en Fortran** es el que plantea más dificultades, porque introduce al alumno en el mundo de la programación, completamente desconocido para la gran mayoría de ellos. Los apartados que más problemas plantean en este tema son los de **Estructuras de iteración** y **Subprogramas**. El primero de ellos, por las dificultades para la gestión de los índices de vectores y matrices en los bucles, que con frecuencia salen del rango del vector conduciendo a errores de segmentación. En el segundo caso, el paso de argumentos de tipo vector y matriz a subprogramas, así como el retorno de resultados correctos con las funciones externas, son las causas de la dificultad. Precisamente son éstos dos los apartados de mayor importancia, dado que tanto las estructuras iterativas, especialmente para la gestión de vectores y matrices, como los subprogramas, son elementos fundamentales para dominar la programación estructurada.

El tema de **Programación en Matlab** plantea menos dificultades que el de Fortran, que concentra los problemas derivados de "pensar" los algoritmos e implementarlos en un programa. El principal problema con Matlab es el cambio de sintaxis, que conduce a los alumnos a confundir palabras clave de ambos lenguajes. El apartado más difícil de este tema es el de **Funciones**, precisamente por este cambio de sintaxis con respecto a Fortran. Sin embargo, es un apartado que no se utiliza excesivamente porque las utilidades de Matlab para realizar cálculos reducen la extensión de los programas, reduciendo la necesidad de dividir el programa en funciones con respecto a Fortran. Otro apartado que provoca cierta confusión es el de **Manejo de vectores y matrices**, en especial las operaciones de concatenación, adición y supresión de filas y columnas usando los operadores "`[]`" y "`:`". Finalmente, los apartados de **Cálculo numérico**, **Polinomios** y **Representación gráfica** no suelen plantear una excesiva dificultad.

4.2. Planificación temporal de las clases expositivas

El cuadro 2 muestra la planificación semanal prevista para las clases expositivas.

4.3. Programa de las clases interactivas

Las clases interactivas ejemplifican sobre el ordenador los conceptos impartidos en las clases expositivas. Además, el tema 1 **Cálculo simbólico con Maple** se trata solamente en las clases interactivas, como se ha mencionado anteriormente. Los contenidos de estas clases, que representan 30 horas presenciales, asociadas a 50 horas de trabajo del alumno (en total 3.6 créditos) son los siguientes:

1. Introducción al entorno operativo Linux Fedora Core (2 horas de clase, 3 horas de trabajo del alumno, 0.2 créditos)

   a) Comandos básicos de manejo de archivos y directorios

   b) Revisión de aplicaciones

   c) Entornos de Maple, Fortran y Matlab

2. Cálculo simbólico con Maple (10 horas de clase, 18 horas de trabajo del alumno, 1.4 créditos)

   a) Introducción: números y variables; representación gráfica básica; definición de polinomios

Semana	Contenidos
1. Programación estructurada en Fortran	
1	1.a Metodología de la programación
2	1.b Estructura básica de un programa
3	1.c Tipos de datos elementales 1.d Vectores y matrices
4	1.e Expresiones aritméticas. Sentencias de asignación
5	1.f Operadores relacionales y lógicos 1.g Sentencias de selección
6	1.h Sentencias de iteración
7	1.i Subprogramas: funciones externas y subrutinas
8	1.j Entrada y salida a archivos. Formatos
2. Cálculo numérico con Matlab	
9	2.a Archivos script. Variables. Entrada y salida básicas 2.b Archivos de función
10	2.c Sentencias de selección e iteración
11	2.d Entrada y salida a archivos
12	2.e Manejo de vectores y matrices
13	2.f Comandos de cálculo numérico
14	2.g Operaciones con polinomios
15	2.h Representación gráfica

Cuadro 2: Planificación temporal por semanas de las clases expositivas.

b) Vectores y matrices

c) Límites y diferenciación

d) Integración definida e indefinida

e) Series finitas e infinitas. Series de potencias

f) Representación gráfica

g) Solución simbólica y numérica de ecuaciones y sistemas de ecuaciones

h) Manipulación de polinomios y funciones racionales

i) Simplificación y expansión de expresiones

3. Programación estructurada en Fortran (9 horas de clase, 15 horas de trabajo del alumno, 1.1 créditos)

a) Programa básico

b) Manejo de estructuras de selección e iteración

1) Ecuación de 2^o grado
2) Representación gráfica de funciones
3) Operaciones iterativas con vectores: acumuladores
4) Procesamiento de matrices
5) Productos matriciales
6) Cálculo numérico de límites
7) Cálculo numérico de derivadas

8) Cálculo numérico de integrales definidas e indefinidas.
9) Suma de series numéricas

c) Subprogramas

1) Cálculo del determinante de una matriz cuadrada
2) Resolución numérica de ecuaciones no lineales por el método de Newton

d) Lectura y escritura en archivos de texto

4. Cálculo numérico con Matlab (9 horas de clase, 15 horas de trabajo del alumno, 1.1 créditos)

a) Programación estructurada

1) Cálculos con vectores
2) Procesamiento de matrices
3) Resolución numérica de ecuaciones no lineales por el método de bisección

b) Funciones. Acceso a archivos

c) Cálculo numérico y simbólico

1) Resolución de ecuaciones y sistemas de ecuaciones
2) Mínimo de una función
3) Integración numérica
4) Límites
5) Derivación
6) Integración
7) Series numéricas

d) Polinomios

1) Operaciones básicas
2) Ajuste a funciones polinómicas
3) Interpolación

e) Representación 2D y 3D de curvas y superficies

Podemos considerar que el tema **Cálculo simbólico en Maple** es el que menos dificultades plantea a los alumnos, porque en él se manejan comandos simples que realizan operaciones matemáticas conocidas por ellos desde el Bachillerato. Dentro de este tema, los apartados más difíciles son el **Series finitas e infinitas. Series de potencias** y **Representación gráfica**. El primero de ellos, por manejar los conceptos de serie numérica, serie de potencias y desarrollo en serie de una función en torno a un punto, que los alumnos desconocen completamente. La dificultad del apartado de representación gráfica se debe a la falta de conocimientos geométricos, en particular sobre las funciones de dos variables para su representación en 3-D, que les plantea dificultades a la hora de representar gráficamente curvas y superficies en forma no explícita (bien paramétrica o implícita).

Finalmente, las tutorías interactivas (13 horas de tutorías, 0.5 créditos) se organizan de la siguiente forma:

1. Cálculo simbólico con Maple (4 horas de clase, 0.15 créditos)
2. Programación estructurada en Fortran (5 horas de clase, 0.2 créditos)
3. Cálculo numérico con Matlab (4 horas de clase, 0.15 créditos)

4.4. Planificación temporal de las clases interactivas

Semana	Tema	Contenidos
1	Linux	Sistema operativo Linux Fedora Core. Comandos básicos. Aplicaciones Entornos de Maple, Fortran y Matlab
2	Maple	Introducción: números y variables; representación gráfica básica Definición de polinomios
3	Maple	Vectores y matrices. Límites y diferenciación Integración definida e indefinida
4	Maple	Series finitas e infinitas. Series de potencias. Representación gráfica
5	Maple	Solución simbólica y numérica de ecuaciones y sistemas de ecuaciones Manipulación de polinomios y funciones racionales
6	Maple	Simplificación y expansión de expresiones. Asistentes y tutoriales
7	Fortran	Programa básico. Ecuación de 2^o grado. Representación de funciones
8	Fortran	Operaciones iterativas con vectores: acumuladores Procesamiento de matrices. Productos matriciales
9	Fortran	Cálculo numérico de límites y derivadas Cálculo numérico de integrales definidas e indefinidas
10	Fortran	Suma de series numéricas Cálculo del determinante de una matriz cuadrada
11	Fortran	Solución numérica de ecuaciones por el método de Newton Lectura y escritura en archivos de texto
12	Matlab	Programación estructurada. Cálculos con vectores Procesamiento de matrices
13	Matlab	Método de bisección. Funciones en Matlab
14	Matlab	Lectura y escritura de datos en archivos Cálculo numérico y simbólico.
15	Matlab	Polinomios. Representación gráfica 2D y 3D

Cuadro 3: Planificación temporal por semanas de las clases interactivas.

El cuadro 3 muestra la planificación temporal prevista para las clases interactivas. En la sesión inicial (2 horas) los alumnos se familiarizan con el entorno operativo Linux, usando la distribución Fedora Core. Se dedican 14 horas (10 h. de clase + 4 h. de tutorías, que representa aproximadamente el primer mes de clase) al **cálculo simbólico** con Maple. Las siguientes 14 horas (9 h. de clase + 5 h. de tutorías) servirán para introducir al alumno a la **programación estructurada** en Fortran, divididas en 2 semanas para asimilar las estructuras de control, 2 semanas para subprogramas y 1 semana para manejo de archivos. La parte de **cálculo numérico** en Matlab consumirá 13 h. (9 h. de clase + 4 h. de tutorías), dedicadas a desarrollar programas y ejecutar comandos de cálculo numérico, manejo de vectores, matrices, polinomios y representación gráfica.

Es importante destacar que en el módulo de Fortran no sólo se aprende su sintaxis, sino también conceptos de algorítmica y programación (variables, sentencias de control, subprogramas, E/S a archivos, etc.). Por esta razón, aunque se dedican aproximadamente las mismas horas a Fortran que a Matlab, en Matlab las horas se reparten entre la programación y la ejecución de comandos de utilidades diversas (cálculo numérico, polinomios, gráficos, etc.), de modo que el número de horas dedicadas a la programación en Matlab es inferior al correspondiente en Fortran. La razón es que la programación en Matlab maneja los mismos conceptos que en Fortran, de modo que resulta más fácil de asimilar por el alumno, aunque su sintaxis sea distinta. Otro factor que facilita la programación en Matlab es el alto número de operaciones matemáticas que proporciona, que en Fortran deben ser programadas directamente.

5. Contenidos de las clases expositivas

En este apartado se describen con detalle los contenidos de los distintos temas y apartados de las clases expositivas.

5.1. Programación estructurada en Fortran 90

En este tema se estudia el paradigma de programación estructurada, ejemplificado en el lenguaje Fortran. Los apartados del tema son los siguientes:
a) Metodología de la programación: análisis, diseño, codificación y depuración
b) Estructura básica de un programa. Entrada y salida estándar
c) Tipos de datos elementales
d) Vectores y matrices estáticos y dinámicos
e) Expresiones aritméticas. Sentencias de asignación
f) Operadores relacionales y lógicos
g) Sentencias de selección
h) Sentencias de iteración
i) Subprogramas: funciones externas y subrutinas
j) Entrada y salida a archivos. Formatos

El material para la preparación de este tema se encuentra en la página web de la asignatura. Asimismo, utilizaremos como libros de referencia básicos los siguientes [5, 9]:

Programación estructurada con Fortran 90/95
J. Martínez Baena, I. Requena, N. Marín, Ed. Universidad de Granada, 2006
Curso básico de Fortran 90
S. Ventura Soto, J.L. Cruz Soto, C.R. Morales. Univ. de Córdoba. 2000

En las siguientes subsecciones se describe cada uno de estos apartados.

5.1.1. Metodología de la programación: análisis, diseño, codificación y depuración

Se exponen brevemente los conceptos de lenguaje de programación y programa, distinguiendo entre lenguajes compiladas e interpretadas. También se presenta el concepto de algoritmo, y se analizan las etapas de análisis del problema, diseño del algoritmo (usando el método de diseño descendente), codificación y depuración del programa, así como las etapas de prueba y mantenimiento de los programas elaborados, todo ello al margen de cualquier lenguaje de programación. El objetivo es que los alumnos perciban la complejidad del proceso de programación y asimilen algunas nociones básicas de ingeniería del software, acercándose por un momento al proceso de producción industrial de software. Aunque ellos sólo programarán en el campo matemático, este tema les aportará un cierto formalismo y metodología en la elaboración de sus programas.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos).

Materiales. Página web de la asignatura. Apéndices D y E de [5]. También se pueden usar los capítulos 1-3 y 5 del libro [6], un clásico de metodología de la programación con gran cantidad de ejercicios resueltos.

5.1.2. Estructura básica de un programa. Entrada y salida estándar

Se introduce al alumno a la programación en Fortran 90, y para ello se presenta inicialmente un programa elemental en Fortran: un esqueleto básico (sentencias de comienzo y final del programa principal) que iremos enriqueciendo con elementos simples. El programa contiene datos y acciones básicos: lectura de datos por teclado, visualización de información por pantalla y operaciones aritméticas elementales con números. Posteriormente, se revisan los procesos de compilación (con el compilador `g95`[11] de GNU), ejecución, y resolución de los errores que se producen en ambas fases (además de los errores lógicos), todo desde un punto de vista práctico como guía para las clases interactivas.

Competencias trabajadas. Este apartado está asociado a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulos 2 y 3 de [5]. Capítulo 2 de [9].

5.1.3. Tipos de datos elementales

Los datos manejados en el primer programa básico son numéricos, sin considerar su tipo y rango de valores. En este segundo apartado veremos cómo manejar distintos tipos de datos numéricos (enteros, reales, reales de doble precisión y complejos), datos lógicos y cadenas de caracteres, distinguiendo entre constantes y variables, y mostrando sus rangos y precisiones (y los errores que se producen cuando se sobrepasan estos rangos). Se presenta la declaración explícita de variables, y se indica que ésta no es necesaria para variables enteras y reales, siempre que el nombre de la variable comience con ciertas letras (*declaración implícita*).

Competencias trabajadas. Este apartado está asociado a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 2 de [5]. Capítulo 3 de [9].

5.1.4. Vectores y matrices estáticos y dinámicos

Las combinaciones más elementales de datos son las colecciones de datos del mismo tipo, formando así vectores y matrices, conceptos que los alumnos ya conocen a partir del Bachillerato. En este apartado se describe la declaración de vectores y matrices (para los que no existe declaración implícita), acceder a sus elementos para imprimirlos o leerlos por teclado, o para realizar operaciones aritméticas básicas. También se comenta la posibilidad de salir del rango de los índices del vector, produciéndose en tal caso un error de ejecución conocido como "fallo de segmentación", y se aborda el uso de un depurador (usaremos el comando `gdb`[12]) para encontrar la sentencia donde se produce el error y su causa. También se estudian algunas funciones intrínsecas proporcionadas por las librerías estándar de Fortran para el manejo de vectores y matrices. La distinción entre vectores estáticos y dinámicos es otro concepto clave, ya que las alternativas dinámicas aportan una mayor flexibilidad a los programas y permiten un mejor aprovechamiento de la memoria RAM (sólo se ocupa la memoria que va a ser realmente utilizada). También se revisan los errores de segmentación que se pueden producir con vectores dinámicos.

Es importante destacar ante el alumno que el manejo de vectores y matrices, que usualmente requiere de estructuras iterativas a estudiar posteriormente, debe ser dominado para la superación de la asignatura. Es decir, constituye un contenido fundamental en programación estructurada, ya que casi todos los programas requieren de algún modo el manejo de vectores y matrices con estructuras iterativas.

[11] `www.g95.org`

[12] `www.gnu.org/software/gdb`

Competencias trabajadas. Este apartado está asociado a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 6 de [5]. Capítulo 2 de [9].

5.1.5. Expresiones aritméticas. Sentencias de asignación

Una vez que se conocen los datos básicos se pueden realizar operaciones aritméticas, bien para mostrar su resultado por pantalla o para almacenarlo en una variable que se usará posteriormente. Se abordan las cuestiones relativas al tipo del resultado que devuelve una expresión cuyos operandos son de distintos tipos, así como las prioridades de los operadores aritméticos y el uso de paréntesis para modificarlas. La sentencia de asignación se presenta también como el modo normal de almacenar en memoria RAM el resultado de una operación para su uso posterior. Se discuten las cuestiones relativas a los tipos de una expresión y de la variable en la que se almacena su resultado, con el objetivo de que no se trunquen valores numéricos con la consiguiente pérdida de información (que probablemente conduzca a errores lógicos en el programa). Finalmente, como ejemplo de sentencia de asignación, se muestra el intercambio de dos variables, con el uso de una variable auxiliar. Se enfatiza el hecho de que una variable es un contenedor de datos cuyo valor cambia a lo largo de la ejecución del programa, y por tanto debemos distinguir entre la variable en sí y su valor en un momento dado. Esta distinción resulta con frecuencia difícil de asimilar por los alumnos, ya que están familiarizados con las ideas matemáticas de variable y de igualdad (ecuación), pero no con las ideas de variable y asignación (operador `=`) en programación.

Competencias trabajadas. Este apartado está asociado a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 2 de [5]. Capítulo 2 de [9].

5.1.6. Operadores relacionales y lógicos

Se presentan los operadores de relación, que permiten definir relaciones de orden e igualdad sobre operandos de tipo numérico, para dar un resultado lógico (la relación o se cumple o no se cumple), que se puede almacenar en una variable lógica. Una relación se puede interpretar como una condición, que se evalúa sobre operandos constantes o variables: dependiendo de sus valores, la condición se cumplirá o no. Los operadores lógicos permiten combinar condiciones simples para obtener condiciones compuestas, y las sentencias de selección permiten evaluar estas condiciones y ejecutar unas sentencias u otras dependiendo de su cumplimiento. Se estudian las prioridades de los operadores relacionales y lógicos entre sí y con los operadores aritméticos, analizando ejemplos que combinan los tres tipos.

Competencias trabajadas. Este apartado está asociado a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 2 de [5]. Capítulo 5 de [9].

5.1.7. Sentencias de selección

Las sentencias de selección se presentan como la primera forma de controlar la ejecución del programa en base a condiciones, rompiendo el flujo de ejecución secuencial que por defecto sigue el programa. La idea de construir condiciones a partir de variables, constantes y operadores relacionales y lógicos, ya presentada en el apartado anterior, se ejemplifica mostrando cómo el programa ejecuta sentencias distintas dependiendo del valor de estas variables, en el momento actual leidas por teclado. Se presentan las variantes de la sentencia `if` y la sentencia `select`, describiendo su comportamiento.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 4 de [5]. Capítulo 5 de [9].

5.1.8. Sentencias de iteración

Se describe en este apartado el modo de repetir bloques de sentencias un cierto número de veces usando la sentencia `do` (bucle o estructura iterativa). Se aborda su comportamiento, número de iteraciones, repetición hacia delante y hacia atrás, anidamiento de bucles y posibles errores de compilación. Como ejemplos, asociados al procesamiento de vectores, se realiza el cálculo del valor medio y mínimo de un vector. También se describen la sentencia `exit` para la ruptura prematura del bucle, como base de los bucles indefinidos, y sus distintos tipos: bucles infinitos, bucles híbridos (definido - indefinido) e implícitos. La sentencia `cycle`, que provoca un salto inmediato a la siguiente iteración, y el nombramiento de bucles como instrumento para terminar precipitadamente bucles anidados, terminan este apartado. Se destaca ante el alumno la importancia del correcto manejo de las estructuras iterativas, que representa un contenido fundamental para superar la parte de programación estructurada.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 5 de [5]. Capítulo 5 de [9]. Capítulos 3 y 4 de [1], también muy didácticos y con ejercicios propuestos. Además, el capítulo 4 de [7], aunque este libro es más árido y menos didáctico.

5.1.9. Subprogramas: funciones externas y subrutinas

Una vez conocidas las sentencias de control de flujo, los alumnos tienen los bloques necesarios para construir cualquier programa. La única limitación es que, con programas grandes, la existencia de un único módulo (el programa principal) se convierte en un problema para la interpretación, manipulación y depuración del código. Por esta razón resulta fundamental la división del programa en subprogramas. Este apartado estudia los dos tipos de subprogramas en Fortran: funciones externas y subrutinas. Se describe la sintaxis del cuerpo del subprograma (sentencias), la llamada desde el programa principal (u otro subprograma) y el paso de argumentos, que pueden ser datos a usar dentro del subprograma (argumentos de entrada), resultados a calcular por el subprograma (argumentos de salida), o ambas cosas (argumentos de entrada / salida). El paso de vectores y matrices como argumentos es un aspecto fundamental, dada la frecuencia con que se usan, y constituye uno de los aspectos más difíciles de asimilar por los alumnos. También se indica que las variables propias del subprograma (variables locales) no son accesibles desde el subprograma llamador, y viceversa. Se comenta que cada llamada a un subprograma tiene un coste computacional en términos de tiempo y memoria RAM, ya que la información de estado (variables locales, entre otras) debe almacenarse en memoria hasta que retorne el subprograma llamado.

Se estudia el concepto de recursividad (subprograma que se llama a sí mismo), y las particularidades que en Fortran deben tener los subprogramas para poder llamarse a sí mismos. Un subprograma recursivo no puede llamarse indefinidamente, porque esto conduciría a una secuencia infinita de llamadas y un error de ejecución (desbordamento de pila o *stack overflow*). Por lo tanto, se indica que dentro del subprograma debe existir una ruta de ejecución que no realice una llamada recursiva, y que el diseño del subprograma debe garantizar que en algún momento se ejecute dicha ruta no recursiva. Finalmente, se estudia el concepto de variable estática (que conserva su valor entre llamadas al subprograma) mediante la sentencia `save`, y

el paso de subprogramas como argumento, mediante la sentencia `external`.

El objetivo de este apartado es que el alumno sepa modularizar un programa, es decir, dividilo en subprogramas, decidir qué tipo de subprograma usar en cada caso y distribuir adecuadamente el código entre ellos. La correcta utilización de subprogramas es una competencia basica en el estudio de cualquier lenguaje de programación, y por ello constituye un contenido fundamental para superar la parte de programación estructurada.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulos 8 y 11 de [5]. Capítulo 6 de [9]. Capítulos 7 y 9 de [1]. Los capítulos 10 y 12 de [10] (éste último capítulo dedicado a las librerías matemáticas, estadísticas y gráficas disponibles en Fortran). También el capítulo 5 de [7].

Dificultades principales. La sintaxis del paso de argumentos de tipo vector y matriz a subprogramas y la llamada al subprograma son el principal problema en este apartado.

5.1.10. Entrada y salida a archivos. Formatos

Todos los programas vistos hasta el momento manejan exclusivamente variables en memoria RAM. Los datos de entrada se leen por teclado, y los resultados se muestran por pantalla. Obviamente, esto no es funcional cuando se necesita manipular una cantidad grande de datos o resultados (imaginemos el cálculo del determinante de una matriz 10×10, que requeriría introducir 100 números por teclado). Para resolver estas situaciones necesitamos archivos de disco desde los que leer los datos o en los que almacenar los resultados. En este tema se describe cómo acceder a un archivo en Fortran para leer o escribir datos en el mismo. En el caso de la escritura veremos cómo crear archivos a partir de cero, y también cómo añadir datos a archivos ya existentes. En el caso de la lectura, veremos cómo leer todos los datos del archivo (en este caso, el proceso de lectura debe detenerse cuando termina el archivo, para lo cual tenemos que detectar su final), y cómo leer datos situados en una cierta posición (posicionamento en el archivo). También se describe el manejo de errores en el acceso a un archivo (inexistencia de un archivo en el que se pretende leer, existencia de un archivo que se pretende crear, archivos inexistentes o inaccesibles, protegidos contra lectura o escritura, etc.). Finalmente, se presenta el formateo de los datos a transferir, especificando la anchura de los números o cadenas de caracteres, el número de decimales y el formato con/sin exponente de un dato real, etc.

Competencias trabajadas. Este apartado está asociado a la **competencia 7** (acceso a archivos).

Materiales. Página web de la asignatura. Capítulos 9 y 10 de [5]. Capítulos 5 y 14 de [1]. Capítulo 10 de [7].

5.2. Cálculo numérico con Matlab

Se estudia la resolución de problemas de cálculo numérico en Matlab ejecutando comandos y escribiendo programas. En un primer momento se aborda la programación, para aprovechar los conceptos de algorítmica y programación estructurada ya vistos en Fortran. Matlab es un lenguaje de alto nivel, lo cual evita la necesidad de gestionar muchos aspectos con gran importancia en Fortran (dimensión de los vectores y matrices, reserva dinámica de memoria y, sobre todo, los tipos de datos, que en Matlab siempre son reales de doble precisión). Además, sus librerías proporcionan una gran variedad de operaciones que en Fortran deben ser programadas directamente. Todo ello facilita el trabajo de programación, aunque usualmente a costa de una menor eficiencia computacional, en parte por tratarse Matlab de un lenguaje interpretado. El tema se estructura en los siguientes apartados:

a) Archivos script. Variables. Entrada y salida básica
b) Archivos de función
c) Sentencias de selección e iteración definida e indefinida
d) Entrada y salida de datos a archivos
e) Manejo de vectores y matrices
f) Comandos de cálculo numérico
g) Operaciones con polinomios
h) Representación gráfica

Materiales. Este tema debería prepararse usando la página web de la asignatura. Como libro de referencia resulta adecuado el siguiente [3]:

Matlab: Una introducción con ejemplos prácticos. Amos Gilat. Reverté. 2006

También resultan de utilidad los libros [2], concretamente los capítulos 4 (programación), 5 (operaciones con vectores y matrices), 10 (sentencias de control de flujo), 11 (funciones), 16 (álgebra matricial), 19 (polinomios), 25 (gráficas 2-D) y 26 (gráficas 3-D), y [8].

En los siguientes apartados se describen en detalle estos contenidos.

5.2.1. Archivos script. Variables. Entrada y salida básica

Se comienza por una introducción al entorno de trabajo de Matlab, con sus distintas partes y en especial las ventanas de ejecución de comandos y de edición de programas. Se aborda la escritura de programas (*scripts*): estructura general, variables y formas de ejecución del programa, haciendo hincapié en el carácter de lenguaje interpretado de Matlab, junto con la ausencia del proceso de compilación y de un programa ejecutable desde el sistema operativo. Se resalta la ausencia de declaraciones de variables, aunque éstas deben inicializarse (lo cual implícitamente significa, en el caso de vectores y matrices, que se reserva memoria) antes de ser usadas en cálculos. También se describe la entrada básica de datos por teclado y la visualización de mensajes y resultados por pantalla, realizando ejemplos de operaciones aritméticas básicas con variables y constantes. Se comentan las utilidades que ofrece el editor de Matlab para detectar errores sintácticos y de ejecución, y el depurador de Matlab para ejecutar los programas paso a paso y ver los valores de las variables.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 4 de [3].

5.2.2. Archivos de función

La función es el otro archivo de código en Matlab, que permite modularizar un programa de forma análoga a los subprogramas en Fortran. Se describe la sintaxis del cuerpo de una función en Matlab, el paso de argumentos y el retorno de valores, así como la sentencia de llamada a función. Se analiza el contexto de las variables en una función (variables locales) y la forma de hacer las variables accesibles desde fuera de la función (variables globales). También se mencionan aspectos más avanzados, como la función `feval`, para pasar subprogramas como argumentos, y las funciones de tipo `inline` .

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 6 de [3].

5.2.3. Sentencias de selección e iteración definida e indefinida

Los conceptos de estructuras de selección e iteración ya se asimilaron en la parte correspondiente de Fortran, y las sentencias correspondientes en Matlab son muy similares, razón por la cual este apartado se puede explicar de forma relativamente rápida. Inicialmente se describen los operadores relacionales y lógicos y su precedencia, haciendo hincapié en su operación sobre vectores y matrices, núcleo de la programación vectorial en Matlab, que constituye una diferencia importante respecto a Fortran. Posteriormente se describen los operadores lógicos y algunas funciones lógicas que permiten el procesamiento de vectores y matrices, tales como `find`, `any` y `all`, entre otras. Se explican las sentencias de selección `if`, `if/else` `if/else if` y `switch`, basadas en condiciones elaboradas con estos operadores y funciones. Las sentencias de iteración `for` (definida) y `while` (indefinida) se explican junto con las sentencias `break` y `continue`, análogas al `exit` y `cycle`, respectivamente, de Fortran. Como ejemplos de bucles definidos e indefinidos en Matemáticas se examina el código que calcula series numéricas.

Competencias trabajadas. Este apartado está asociado a la **competencia 5** (diseño de algoritmos) y a la **competencia 6** (programación estructurada).

Materiales. Página web de la asignatura. Capítulo 7 de [3].

5.2.4. Entrada y salida de datos a archivos

La sentencia `load` se verá como la alternativa más simple para la lectura de datos uniformes desde archivos. Para un procesamiento más avanzado, que permita la escritura en archivo, veremos las sentencias `fopen` y `fclose`, con sus diversos modos de apertura (lectura, escritura y adición), y las sentencias `fprintf`, ya conocida para la salida por pantalla, y `fscanf` para la escritura y lectura en archivo respectivamente en formato texto. Finalmente, se comentan las sentencias `xlsread/xlswrite` para la lectura y escritura en formato de hoja de cálculo `.xls` para Excel.

Competencias trabajadas. Este apartado está asociado a la **competencia 7** (acceso a archivos).

Materiales. Página web de la asignatura. Capítulo 4 de [3].

5.2.5. Manejo de vectores y matrices

Finalizada la parte de programación, se estudian con más profundidad las comandos de Matlab para el procesamiento de vectores y matrices: formas de definir vectores y matrices de distintos tipos (con valores constantes, pre-definidos, nulos, constantes, aleatorios, con valores lineal o logarítmicamente equiespaciados), el acceso, modificación, adición y borrado de elementos, producto escalar y matricial, matrices especiales (identidad, mágica), diagonal y triángulos superior e inferior de una matriz, suma, producto, máximo y mínimo, media, varianza, desviación y mediana de un vector, transposición, operaciones componente a componente, inversión, determinante y rango, matrices dispersas, autovalores y autovectores de una matriz, ordenamiento de vectores y resolución de sistemas de ecuaciones lineales por el método matricial.

Competencias trabajadas. Este apartado está asociado a la **competencia 1** (operaciones básicas con vectores y matrices).

Materiales. Página web de la asignatura. Capítulos 1-3 de [3].

5.2.6. Comandos de cálculo numérico

Las funciones de Matlab para el cálculo numérico y simbólico constituyen una parte fundamental de este tema. Se explican los comandos para la definición y sustitución de variables simbólicas en expresiones, resolución numérica de ecuaciones y sistemas de ecuaciones lineales y no lineales (funciones `fzero` y `solve`), optimización de funciones (`fminbnd`) e integración definida numérica (funciones `quad` y `trapz`) y simbólica (`int`), integración indefinida (`int`), cálculo simbólico de límites (`limit`), diferenciación simbólica en una y varias variables (`diff`) y cálculo simbólico de series numéricas finitas e infinitas (`symsum`).

Competencias trabajadas. Este apartado está asociado a la **competencia 3** (operaciones de análisis matemático) y a la **competencia 4** (resolución de ecuaciones).

Materiales. Página web de la asignatura. Capítulo 10 e 11 de [3].

5.2.7. Operaciones con polinomios

Se describen las utilidades de Matlab para la definición de polinomios como vectores de coeficientes, además de los comandos vistos en el apartado anterior para funciones genéricas, aplicables también a los polinomios. Concretamente, el cálculo de las raíces de un polinomio a partir de sus coeficientes y viceversa, operaciones básicas (suma y resta, producto y cociente, derivación e integración) y funciones racionales (descomposición en fracciones parciales). Otro aspecto de gran importancia son los comandos para ajustar datos a funciones polinómicas (función `polyfit`) y no polinómicas (exponencial, logarítmica e inversa), y para la interpolación de funciones (función `interp1`). Finalmente, veremos el asistente de Matlab para la interpolación, que permite ajustar puntos de forma fácil a polinomios de cualquier tipo y *splines*.

Competencias trabajadas. Este apartado está asociado a la **competencia 2** (manipulación de polinomios) y a la **competencia 4** (resolución de ecuaciones polinómicas).

Materiales. Página web de la asignatura. Capítulo 8 de [3].

5.2.8. Representación gráfica

La función `plot` es el comando básico para la representación de datos en el plano. Las distintas opciones de representación, su comportamiento con vectores y matrices y la visualización de varias curvas en la misma gráfica son temas importantes para representar vectores y matrices. También se explica la representación de funciones a partir de su expresión analítica (funciones `fplot` y `ezplot`), las curvas en coordenadas polares, los gráficos logarítmicos y la creación de una matriz de gráficas (`subplot`), junto con los gráficos especiales: barras verticales y horizontales, tartas, diagramas de troncos e histogramas. Finalmente, se estudian los comandos para la representación de curvas y superficies en el espacio. Las curvas se representan en forma paramétrica (`plot3`), y para las superficies utilizaremos la forma explícita $z = f(x, y)$ en coordenadas cartesianas con la función `mesh` y sus variantes (`surf`, `meshc` y `contour`). También se comentan algunas gráficas especiales como esferas, cilindros, diagramas de barras y tartas en tres dimensiones.

Competencias trabajadas. Este apartado está asociado a la **competencia 1** (cálculos básicos y representación gráfica).

Materiales. Página web de la asignatura. Capítulos 5 y 9 de [3].

6. Contenidos de las clases interactivas

Describiremos ahora las actividades a desarrollar en los distintos apartados de las clases interactivas.

6.1. Entorno operativo Linux Fedora Core

En esta primera sesión se introduce al alumno en el entorno de trabajo de Linux, en su distribución Fedora Core[13] y al entorno de escritorio KDE[14]. Se analizan brevemente las principales aplicaciones (editor, gestor de archivos, navegador, compresor de archivos, transferencia de archivos por red, correo electrónico, terminal de comandos, procesador de textos, hoja de cálculo, etc.). También ejecutaremos comandos básicos de Linux para la gestión de archivos desde una terminal: listado, navegación por los directorios, copia, borrado, visualización, etc. Entraremos en los entornos de Maple, Fortran y Matlab, y visitaremos las páginas web de las principales distribuciones de Linux. Los alumnos están acostumbrados al entorno Windows, y la necesidad de memorizar comandos para ejecutar desde una terminal les resulta algo problemático. Sin embargo, el aprendizaje de estos comandos resulta de gran interés para manejar el entorno Linux, tanto para la programación en Fortran como para las materias de cursos posteriores (sección 2.1).

Competencias trabajadas. Este apartado está asociado a la **competencia instrumental 1** (manejo del ordenador).

6.2. Cálculo simbólico en Maple

6.2.1. Introducción: números y variables. Representación gráfica básica. Definición de polinomios

Se presenta al alumno el entorno de Maple, ejecutando algunos comandos para definir variables simbólicas y realizar operaciones aritméticas básicas con números y variables. Se hará hincapié en el carácter simbólico de Maple, que manipula valores simbólicos (exactos) en sus operaciones y sólo proporciona valores reales (en punto flotante) cuando se le pide expresamente con la función `evalf`. De este modo se evita la acumulación de errores derivados de aproximaciones numéricas en la ejecución secuencial de operaciones. Se explica cómo maneja Maple los distintos tipos de números (enteros, racionales, irracionales, reales y complejos), algunas operaciones simples con enteros, aspectos de precisión en la conversión a números en punto flotante, las constantes pre-definidas de Maple y las funciones aritméticas elementales. La representación gráfica básica de funciones, usando el comando `plot`, es otro punto que conviene ver desde el principio, ya que a lo largo del curso nos permite comprobar la corrección de los resultados obtenidos. Finalmente, se aborda la definición de polinomios en Maple y hacer operaciones básicas (suma y resta, producto y cociente, y ordenamiento de monomios).

Competencias trabajadas. Este apartado está asociado a la **competencia 1** (cálculos básicos y representación gráfica).

Materiales. Este primer apartado deben prepararse usando los capítulos 2 y 3 del libro de referencia de Maple [4]:

Introduction to Maple. A. Heck. Springer. 2003

Como material auxiliar para el tema de Maple se recomiendan los siguientes libros:

[13] `fedoraproject.org`
[14] `www.kde.org`

Introduction to mathematics with Maple. P. Adams. World Scientific. 2004
Solving problems in scientific computing using Maple and Matlab. W. Gander. Springer. 2004

6.2.2. Vectores y matrices. Límites y diferenciación. Integración definida e indefinida

Una vez conocidos los datos numéricos básicos, se aborda la definición de vectores y matrices con valores numéricos y simbólicos, junto con las operaciones básicas con matrices (suma y resta, producto matricial, transposición e inversión), acceso y edición de elementos, y copia de matrices. También veremos operaciones de álgebra matricial, como el cálculo de la matriz y polinomio característico, autovalores y autovectores, determinante, triangularización y resolución de sistemas de ecuaciones lineales.

En el apartado **Límites y diferenciación** se introduce la definición de expresiones y funciones de una y varias variables en Maple, así como la transformación de expresión en función. Un caso particular son las funciones definidas por intervalos (comando `piecewise`). Asimismo, se describen las operaciones básicas con funciones: suma y resta, multiplicación y división, aplicación secuencial (composición) y paralela (componente a componente). Sobre esta base se introduce el cálculo del límite (y límite lateral) de una función en un punto (función `limit`), y el cálculo de derivadas y derivadas parciales (función `diff`) de primer orden y de órdenes superiores. Se continúa con los comandos para el cálculo de integrales definidas e indefinidas, y de integrales dobles (función `int`). Estos conceptos son familiares para el alumno a partir del Bachillerato, pero aquí se introduce como novedad su extensión a funciones de varias variables, antes de que se introduzcan en las asignaturas de Análisis Matemático (2^o curso).

Competencias trabajadas. Este apartado está asociado a la **competencia 1** (cálculos con vectores y matrices) y a la **competencia 3** (operaciones de análisis matemático).

Materiales. Capítulos 8, 9, 10, 12 y 18 del libro de referencia de Maple [4].

6.2.3. Series finitas e infinitas. Series de potencias. Representación gráfica

Un concepto nuevo para los alumnos es el de serie numérica y serie de funciones, que se presenta aquí brevemente relacionándolo con el de límite de una sucesión, que sí conocen desde el Bachillerato. Se estudian los comandos de Maple (`sum`) para el cálculo de series numéricas finitas e infinitas, y de productorios (comando `product`). Se introduce la noción de serie de potencias, como generalización de las series numéricas, y de aproximación de una función como una serie de potencias en torno a un punto, con el desarrollo en serie de Taylor como ejemplo. Se estudian los comandos de Maple para desarrollar una función en serie de Taylor en torno a un punto, y convertir esta serie en un polinomio, comprobando gráficamente la precisión de la aproximación (mayor cuanto más cerca del punto y cuanto mayor el orden de la aproximación). Se comentan también las funciones de Maple para crear series de potencias y realizar operaciones con ellas (aritméticas, derivación, integración, logaritmos, exponenciales, etc.).

La **Representación gráfica** es un apartado de gran importancia en Maple, que proporciona una gran flexibilidad para representar curvas en 2D y 3D. En el plano, permite representar curvas en forma explícita y paramétrica (`plot`) e implícita (`implicitplot`), y también en coordenadas polares (`polarplot`), así como vectores de puntos y sistemas de inecuaciones lineales (`inequal`). En el espacio permite representar curvas en forma paramétrica (`spacecurve`), así como superficies en forma explícita y paramétrica (`plot3d`) e implícita (`implicitplot3d`) en coordenadas cartesianas, cilíndricas y esféricas. Finalmente, se introducen las animaciones (funciones dependientes del tiempo) en el plano y el espacio con los comandos `animate` y `animate3d`.

Competencias trabajadas. Este apartado está asociado a la **competencia 3** (operaciones de análisis matemático) y a la **competencia 1** (representación gráfica).

Materiales. Capítulos 11 y 15 del libro de referencia de Maple [4].

6.2.4. Solución simbólica y numérica de ecuaciones y sistemas de ecuaciones. Manipulación de polinomios y funciones racionales

Se explica el resolvedor simbólico general de ecuaciones y sistemas de ecuaciones de Maple (función `solve`), junto con su alternativa numérica (`fsolve`). También se introducen otros resolvedores más específicos, como `isolve`, para ecuaciones con soluciones enteras, y `rsolve`, para resolver ecuaciones recursivas, cuya solución es una sucesión x_n donde desconocemos $x_n = f(n)$ pero conocemos $x_n = f(x_{n-1}, \ldots, x_{n-k}), k \geq 1$, junto con unas condiciones iniciales $\{x_{n_1} = X_1, \ldots, x_{n_k} = X_k\}$.

En el apartado **Manipulación de polinomios y funciones racionales** se explica la mayoría de las operaciones con polinomios con coeficientes enteros de una variable: definición, cociente y resto de la división de dos polinomios, máximo común divisor y mínimo común múltiplo, raíces exactas de un polinomio (función `roots`) y descomposición en factores primos. También se definen polinomios de varias variables, y se estudia la función `sort` para agrupar los monomios según su grado o usando criterios alfabéticos. Finalmente, se aborda la definición de funciones racionales (cocientes de polinomios), su simplificación (eliminación de factores comunes al numerador y denominador) y su descomposición en fracciones parciales.

Competencias trabajadas. Este apartado está asociado a la **competencia 2** (manipulación de polinomios).

Materiales. Capítulos 5, 7 y 16 del libro de referencia de Maple [4].

6.2.5. Simplificación y expansión de expresiones. Asistentes y tutoriales

Para finalizar este tema, se estudian los comandos generales de manipulación de expresiones de Maple, que permiten su expansión o reducción, simplificación o, en el caso de funciones polinómicas o racionales, su descomposición en factores. Para llevar a cabo estas manipulaciones, Maple utiliza las reglas algebráicas usuales con polinomios pero también las reglas de manipulación de funciones trigonométricas, exponenciales y logarítmicas. También se revisan los **Asistentes y tutoriales** de Maple como herramienta auxiliar para el alumno en su trayectoria por diferentes asignaturas de la titulación, haciendo hincapié en los tutoriales de: representación de funciones estándar, ajuste a curvas con polinomios y *splines*, gráficos, integración indefinida y aproximada, derivadas, cálculo de límites, método de Newton, polinomios y raíces, composición de funciones, cálculo de autovalores y autovectores, matrices inversas, funciones racionales y resolución de sistemas de ecuaciones lineales.

Competencias trabajadas. Este apartado está asociado a la **competencia 2** (manipulación de expresiones).

Materiales. Capítulo 14 del libro de referencia de Maple [4].

6.3. Programación estructurada en Fortran

En este apartado usaremos como material básico el proporcionado por la página web de la asignatura.

6.3.1. Programa básico. Ecuación de 2^o grado. Representación de funciones

Se comienza explicando el entorno de programación que usaremos para la programación en Fortran 90, integrado por un editor de texto, una terminal de comandos y el compilador de Fortran `g95` de GNU. Se le proporcionará al alumno el código de un programa, para que lo escriba en el editor y lo compile. El programa es muy simple, ya que sólo lee un número por teclado, realiza unas operaciones aritméticas básicas y finalmente muestra sus resultados por pantalla. El objetivo es que el alumno se familiarice con el proceso de escritura del programa, detección y corrección de errores (de compilación, ejecución y lógicos), sin tener que preocuparse todavía de pensar el algoritmo y el programa. Aún así, la adaptación al ciclo de programación resulta difícil para los alumnos, especialmente la detección de errores, a pesar de que inicialmente son escasos al tratarse de un programa muy simple.

Se continúa elaborando, de forma incremental y guiada por el profesor, un programa, cuyo código no se entrega ya al alumno, que lee por teclado los coeficientes de una ecuación de segundo grado y calcula sus raíces, distinguiendo mediante sentencias de selección entre los tres casos posibles: dos soluciones reales distintas, dos soluciones complejas conjugadas y una solución real doble. Posteriormente, se realiza un segundo programa que calcule los valores de una función definida por intervalos. Se usan estructuras de selección para distinguir entre los distintos intervalos, y una sentencia de iteración para repetir el cálculo de la función en los distintos puntos de su dominio de definición. El programa debe mostrar el resultado por pantalla. Además, este resultado se almacenará en un archivo redirigiendo los mensajes de salida del programa, y se representará gráficamente usando los programas `gnuplot`[15] y `octave`[16]), para que el alumno sepa cómo representar gráficamente funciones usando programas en Fortran. De ese modo, además, el alumno se familiariza con `gnuplot` y `octave`, dos herramientas matemáticas de software libre muy populares.

6.3.2. Operaciones iterativas con vectores: acumuladores. Procesamiento de matrices. Productos matriciales

Se comienza escribiendo y depurando programas con condiciones y sentencias de selección, usualmente combinadas con iteraciones. Concretamente, se implementan operaciones que involucran a dos vectores (declarados estáticamente al principio y dinámicamente después, de modo que se ensaye la reserva dinámica de memoria) con los que se realizan operaciones de tipo sumatorio, usando una sentencia de iteración y un acumulador para calcular operaciones con un número variable de operandos.

Una vez visto el manejo de vectores, se propone la realización de un programa que procese una matriz en Fortran. Para ello, se declara una matriz dinámica y se leen sus elementos por teclado, realizando diferentes operaciones que implican el recorrido de la matriz y el cálculo de algún tipo de valor (traza, suma de los elementos del triángulo superior o inferior, evaluación de la simetría de la matriz, etc.). Aprovecharemos la evaluación de la simetría para ejemplificar el uso de variables lógicas.

Otra operación de gran importancia con vectores y matrices es el producto matricial, y por esta razón se realiza también un programa que calcule el producto de un vector fila por una matriz y por un vector columna. En este caso, hay que usar también reserva dinámica de memoria, junto con estructuras iterativas y acumuladores.

[15] `www.gnuplot.info`
[16] `www.octave.org`

6.3.3. Cálculo numérico de límites y derivadas. Cálculo numérico de integrales definidas e indefinidas

Se comienza esta sesión escribiendo un programa que determine el límite de una función en un punto, calculando los valores de la función en un entorno de dicho punto. Para esto usaremos una estructura iterativa definida, y otra indefinida, que repita el cálculo de los valores de la función en ditintos puntos del entorno. Luego usaremos la representación gráfica con `gnuplot` u `octave` para comprobar el valor de este límite. Aprovechando esta forma de calcular el límite de una función, escribiremos otro programa que aproxime numéricamente su derivada en un intervalo, usando la definición de derivada como límite. La función y la derivada se representan gráficamente para comprobar que la aproximación es correcta. Es importante destacarle ante el alumno que la derivación realizada es numérica, y por lo tanto vale para cualquier función, independientemente de la complejidad de su expresión analítica, incluso si no conocemos su expresión analítica, sino sólo sus valores en un intervalo.

Invirtiendo la definición de derivada se obtiene un procedimiento simple para aproximar numéricamente la integral indefinida de una función en un intervalo (que también se representa gráficamente para comprobar la calidad de la aproximación realizada). Igualmente, se realiza un programa que aproxime numéricamente la integral definida de una función como suma de las áreas de los trapecios, comparando el valor obtenido con el proporcionado por Maple y modulando su precisión a través del número de trapecios y del uso de variables reales de doble precisión.

Usualmente el principal problema en este apartado suele ser asimilar la implementación en un lenguaje de programación de las definiciones matemáticas relacionadas con el paso al límite (por ejemplo, para el cálculo numérico de derivadas e integrales). Por otra parte, los alumnos también suelen tener problemas con la utilización de acumuladores.

6.3.4. Suma de series numéricas. Cálculo del determinante de una matriz cuadrada

En este apartado usaremos subprogramas (funciones externas y subrutinas) para modularizar los programas. El primer ejemplo, continuando con los del apartado anterior, será un programa para aproximar la suma de una serie numérica infinita, ejecutándolo con varios casos de series convergentes y divergentes, comparando los resultados con los obtenidos por métodos simbólicos usando Maple, y comprobando las diferencias en la precisión. En este caso usaremos funciones externas para calcular las sumas parciales de una serie numérica.

El segundo ejemplo será el cálculo del determinante de una matriz cuadrada desarrollando por adjuntos de la primera fila. Para ello se necesita una subrutina que calcule la matriz adjunta a un elemento dado. En este ejemplo es necesario el paso de vectores y matrices como argumentos de entrada y de salida. Por otra parte, se necesita una función recursiva que calcule el determinante de una matriz, lo cual permite ejemplificar el concepto de subprograma recursivo. Por todo ello, este programa resulta el más largo y complicado conceptualmente que realizaremos a lo largo de la asignatura.

6.3.5. Solución numérica de ecuaciones por el método de Newton. Lectura y escritura en archivos de texto

El tercer ejemplo del uso de subprogramas será un programa para implementar el método de Newton, que calcula las raíces de una ecuación no lineal. En este caso, se necesitan funciones externas para definir la función cuyas raíces se buscan y su derivada. Se verá un ejemplo con una función polinómica, para la cual se puede automatizar el cálculo de la expresión analítica de

la función derivada, y otro ejemplo con una función no polinómica, cuya derivada es necesario definir manualmente.

Para leer o escribir datos en archivos se añade una parte adicional a los programas realizados anteriormente. Concretamente, se lee la matriz cuadrada desde un archivo en el ejemplo del determinante, y se escriben en un archivo las raíces calculadas usando el método de Newton. También se ensayarán programas, escritos desde el principio, para leer, escribir y añadir datos en archivos, usando formatos por defecto y formatos definidos por el usuario (sentencia `format`). Se prueban los distintos modos de acceso, y los errores de ejecución que se pueden producir (por ejemplo, lectura desde un archivo inexistente o sobre-escritura de un archivo existente).

6.4. Cálculo numérico en Matlab

En este apartado, al igual que en el correspondiente a programación estructurada en Fortran, usaremos como material el disponible en la página web de la asignatura.

6.4.1. Programación estructurada. Cálculos con vectores. Procesamiento de matrices

Se realizan ejemplos similares a los mencionados en Fortran para ensayar las variables, operadores y estructuras de selección e iteración de Matlab. Concretamente, se escribe un programa que opera con vectores calculando sumatorios de sus componentes, para que el alumno aprenda el uso de estructuras iterativas y acumuladores en Matlab. Una vez finalizado, se introducen las utilidades de programación vectorial de Matlab (funciones lógicas, operadores relacionales y operaciones componente a componente con vectores y matrices) para reducir este código y mostrar el estilo de programación en Matlab. Seguidamente se escribe un programa que realice manipulaciones simples en matrices, para mostrar el uso de las funciones de Matlab (suma, producto, traza, diagonal, triángulo superior/inferior de una matriz, etc.) en la gestión de vectores y matrices. También se prueban los comandos de Matlab para la edición de vectores y matrices, especialmente la concatenación y adición/supresión de filas y columnas.

La principal dificultad de esta sesión es el aprendizaje de una sintaxis diferente para los mismos conceptos de programación vistos ya en Fortran, que puede desorientar al alumno. Sin embargo, la gestión de los datos (sin necesidad de declararlos), de vectores y matrices (que no se dimensionan) y la visualización de información (depuración) facilitan enormemente el trabajo de programación.

6.4.2. Método de bisección. Funciones en Matlab

Se realiza un programa que implementa el método de bisección para la aproximación numérica de las soluciones de una ecuación no lineal. Este método proporciona una alternativo, y con una complejidad similar, al de Newton, ya visto en los apartados de Fortran, y por lo tanto resulta fácil de entender por el alumno. Como ejemplo de función en Matlab definida por el usuario, se escribe el código para calcular y representar gráficamente una función matemática definida por intervalos. Este ejemplo se realiza en primer lugar usando sólo estructuras de selección e iteración, como lo haríamos en Fortran, para que el alumno perciba la facilidad de traducir código de Fortran a Matlab. Después, se propone una versión vectorizada.

6.4.3. Lectura y escritura en archivos. Cálculo numérico y simbólico

Como ejemplo del acceso a archivos en Matlab, se realiza un programa que lea datos desde un archivo a una matriz usando el comando `load`, para después realizar algún procesamiento

sobre esta matriz (búsqueda y recuento de elementos) y escriba los resultados (vectores y matrices, usando las funciones `fopen`, `fprintf` y `fclose`) en el mismo archivo, de modo que se ensaye el acceso para adición de datos en un archivo. Seguidamente, se ejecutan comandos de Matlab, ya introducidos durante las clases expositivas, para la manipulación de vectores y matrices (especialmente concatenación y supresión de filas y/o columnas en vectores y/o matrices), cálculo numérico y simbólico (resolución de ecuaciones y sistemas de ecuaciones, cálculo de mínimos de funciones, límites, derivadas, integrales numéricas y simbólicas definidas e indefinidas, y series numéricas)

6.4.4. Polinomios. Representación gráfica 2D y 3D

Las últimas sesiones de clases interactivas se dedican a la ejecución de comandos para la manipulación de polinomios (definición, cálculo de raíces, derivación e integración, producto y cociente, etc.), descomposición de funciones racionales en factores y ajuste de series de puntos a una función polinímica por mínimos cuadrados. También se prueba el comando `interp1` para la interpolación de funciones en Matlab, y los resultados obtenidos se comparan con el asistente para interpolación. Para finalizar, se representan gráficamente curvas y superficies en el plano y en el espacio usando las utilidades gráficas de Matlab.

7. Indicaciones metodológicas

Las **clases expositivas** se basarán en el uso de presentaciones de ordenador que cubrirán los distintos apartados del material (que se encuentra disponible en la página web de la asignatura). Durante estas clases se describirán de forma concisa estos contenidos que, dado su carácter eminentemente práctico, se desarrollarán con más profundidad en las clases interactivas. En las clases expositivas se hará más hincapié en aquellos contenidos que por su naturaleza no se amplíen en las clases prácticas (por ejemplo, variables estáticas o paso de subprogramas como argumentos en Fortran). Estas presentaciones incluirán ejemplos (y, en el caso de Fortran y Matlab, programas simples) que puedan usarse como referencia en las clases interactivas.

Las **clases interactivas** consisten en la ejecución de comandos (en el caso de Maple y Matlab) y en la elaboración y ejecución de programas (en el caso de Fortran y Matlab) para la resolución de problemas matemáticos, dirigidos por los profesores y siguiendo una planificación temporal predeterminada. Siempre que sea posible, se sigue un esquema "incremental", comenzando por exponer la solución más "fácil" o "inmediata", para discutir después sus deficiencias y aportar mejoras hasta alcanzar la solución final óptima. En los ejercicios de programación se hace especial hincapié en la eficiencia computacional, en términos de consumo de tiempo y memoria RAM, de los programas implementados. Además de los ejercicios resueltos durante las clases interactivas, cuyas soluciones están disponibles para los alumnos en la página web de la asignatura, ésta contiene también una colección de ejercicios propuestos para su realización en las horas de trabajo individual del alumno estipuladas en la Memoria de Grado. Las **tutorías interactivas** se dedicarán a la realización de ejercicios bajo la supervisión del profesor.

Por su parte, las **tutorías de grupo muy reducido** se dedicarán a la corrección de ejercicios propuestos que los alumnos deben realizar en sus horas de trabajo personal, resolución de dudas y seguimiento individualizado del aprendizaje del alumno.

8. Indicaciones sobre la evaluación

En la Memoria de Grado se establece un **criterio general** de evaluación, según el cual la calificación del alumno se realizará mediante evaluación continua y realización de un examen final. La evaluación continua se realizará por medio de controles escritos, trabajos entregados, participación del estudiante en el aula, tutorías u otros medios indicados en la programación de la materia. La cualificación del alumno no será inferior a la del examen final, ni a la obtenida ponderándola con la evaluación continua, dándole a esta última un peso no inferior al 25 %. El profesor fijará en la guía docente anual el peso que concederá a la evaluación continua y al examen final, respetando la regla anterior, así como la tipología, métodos y características del sistema de evaluación propuesto.

Además de este **criterio general**, existe un criterio específico para cada materia, que debe desarrollar el criterio general. En el caso de la materia "Informática", la evaluación continua se realizará mediante la resolución por el alumno de ejercicios propuestos durante las tutorías interactivas (su duración será por lo tanto de 1 hora) que serán evaluados por los profesores. La evaluación continua contará 3 puntos en la nota final de la asignatura (1 punto para cada tema: Maple, Fortran y Matlab). La realización de estos tres exámenes tiene también una finalidad didáctica: se pretende que los alumnos se familiaricen con la realización de pruebas escritas en la Universidad, dado que se trata de su primer cuatrimestre, contribuyendo a su preparación para el examen final de la asignatura. La nota de este exame final (entre 1 e 10) se sumará a la nota obtenida en la evaluación continua, si el alumno concurrió a la misma. De este modo, la nota máxima (10) podrá ser obtenida con o sin evaliación continua, aunque en el primer caso requerirá una menor nota en el examen final. La página web de la asignatura contiene también una colección de exámenes propuestos y resueltos correspondientes a las pruebas de años anteriores, para que el alumno tenga una idea clara del nivel que se exige en esta asignatura, además de los contenidos que con más frecuencia aparecen en las pruebas.

8.1. Recomendaciones para la evaluación

Para la superación de la materia, la primera recomendación es la asistencia a las clases expositivas e interactivas, que permitirá al alumno: 1) optimizar el tiempo de preparación de la asignatura; 2) tener una idea clara del material que entra en el examen; y 3) saber cuáles son los conceptos. También es recomendable la asistencia a las tutorías, donde resolveremos las dudas más frecuentes, ahorrando parte del tiempo que llevaría su resolución sin ayuda. El alumno debe realizar por sí mismo los problemas propuestos, especialmente los de programación, y los ejercicios de los exámenes de años anteriores, para tener una idea precisa del nivel de complejidad exigido en los mismos.

Aquellos alumnos que no tienen experiencia informática previa pueden tener problemas de "baja productividad" a la hora de escribir y depurar los programas en las clases interactivas. En estos casos, es importante practicar de forma autónoma su codificación para alcanzar la destreza en el manejo del ordenador que forma parte de las competencias instrumentales de la asignatura.

En cuanto a la programación estructurada, resulta importante escribir programas en Fortran y Matlab con varios subprogramas, pasando argumentos (especialmente vectores y matrices) de entrada y de salida, así como para retornar valores. Éste es uno de los aspectos fundamentales de la asignatura, y uno de los más difíciles para los alumnos. También es muy importante el manejo de vectores y matrices con estructuras iterativas.

8.2. Recomendaciones para la recuperación

En caso de no superar la asignatura, resulta fundamental la revisión del examen para analizar los fallos cometidos y corregirlos, y para saber qué aspectos necesitan más refuerzo. Los alumnos que ya cursaron la asignatura en años anteriores deberían acudir ocasionalmente a clase para actualizar los contenidos con respecto a años anteriores, o en su defecto asistir a las tutorías, y por supuesto revisar los exámenes más recientes.

Referencias

[1] S.J. Chapman. *Fortran 90/95 for scientists and engineers*. McGraw-Hill, 2004.

[2] B. Littlefield D. Hanselman. *Mastering Matlab 6*. Prentice-Hall, 2001.

[3] A. Gilat. *Matlab©: Una introducción con ejemplos prácticos*. Editorial Reverté, 2006.

[4] A. Heck. *Introduction to Maple*. Springer, 2003.

[5] I. Requena Ramos y N. Marín Ruiz J. Martínez Baena. *Programación estructurada con Fortran 90/95*. Editorial Universidad de Granada, 2006.

[6] L. Joyanes. *Problemas de metodología de la programación*. McGraw-Hill, 1990.

[7] M. Cohen M. Metcalf, J. Reid. *Fortran 95/2003 explained*. Oxford University Press, 2004.

[8] P. Quintela. *Introducción a MATLAB y sus aplicaciones*. Servizo de Publicacións da USC, 1997.

[9] C. Romero Morales S. Ventura Soto, J.L. Cruz Soto. *Curso básico de Fortran 90*. Universidad de Córdoba, 2000.

[10] I. M. Smith. *Programming in Fortran 90: a first course for engineers and scientists*. John Wiley & Sons, 2001.