porque pensamos mayoritariamente que la estética (la belleza) y no la utilidad es la principal justificación de las construcciones matemáticas y en parte porque no sabemos qué responder. Los vectores sirven para representar magnitudes, como las fuerzas en física, en las que no solo es importante el tamaño o intensidad sino también la dirección. Su introducción es relativamente tardía en la historia de las matemáticas, del siglo XIX, y el álgebra lineal no se asentó bien hasta principios del siglo XX, incluso a mediados de siglo no se impartía en todas las universidades, mientras que ahora forma parte de cualquier plan de estudios científico[3].

Si todo esto te parece muy lejano, quizá te acerque un poco a las matrices y vectores saber que en algunas aplicaciones de ingeniería, una señal se representa con un vector. Por ejemplo, sus coordenadas pueden ser los valores de los píxeles de una imagen o las amplitudes de una señal de audio que hemos muestreado. Una matriz procesa la señal de cierta forma cuando se multiplica por ella. En algunos contextos, las matrices se corresponden con filtros llamados lineales.

0.3. Usando el ordenador

Si estás cursando una ingeniería se da por hecho que no tienes alergia a los ordenadores. Entonces te alegrará saber que el álgebra lineal numérica está muy desarrollada. Aunque en este curso nadie te va a obligar a que uses el ordenador para cálculos con matrices, dedicaré esta sección a darte algunos rudimentos sobre cómo hacerlo. Seguro que en otras asignaturas no habrá opción, así que aprender lo básico ahora te puede ahorrar tiempo en el futuro.

De entre el *software* que se emplea en ingeniería con fines académicos el que ha alcanzado una mayor difusión es `matlab` cuyo nombre abrevia *matrix* y *laboratory*, lo que da una idea de que al menos originariamente, tiene casi 40 años, estaba orientado a cálculos con matrices. Actualmente es mucho más versátil, sobre todo cuando se instalan paquetes adicionales, sin embargo las matrices se siguen reflejando en la propia sintaxis y en mi experiencia, quizá sesgada, son los cálculos matriciales lo que mejor hace.

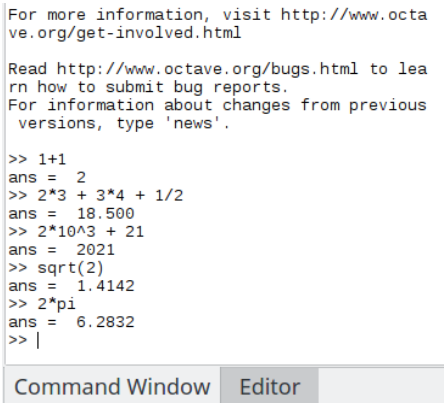
A pesar de que internamente esos cálculos se llevan a cabo con una adaptación de rutinas que hoy son de dominio público, `matlab` es *software* comercial propietario. Un punto a favor es que las licencias para estudiantes no son astronómicas y muchas universidades y departamentos tienen licencias generales (es el caso de la UAM, además lo puedes usar *online* registrándote con tu correo institucional). De todas formas, una alternativa libre y gratuita es (GNU) `octave`. Los comandos y programas básicos son intercambiables hasta el punto de que si no sales del ámbito básico te parecerá que el `octave` que estás usando en casa en tu portátil emula al `matlab`

[3]Una anécdota al respecto es que W. Heisenberg no conocía la multiplicación de matrices y con la ayuda de M. Born, que había seguido cursos de matemáticas en los que aparecía, llegó a una formulación adecuada de su principio de incertidumbre. Llama la atención que alguien que sabía tanta física y matemáticas como Heisenberg no dominara un tema más que obligatorio en primero de grado hoy en día, pero eso fue en 1925 cuando pocos físicos lo hacían.

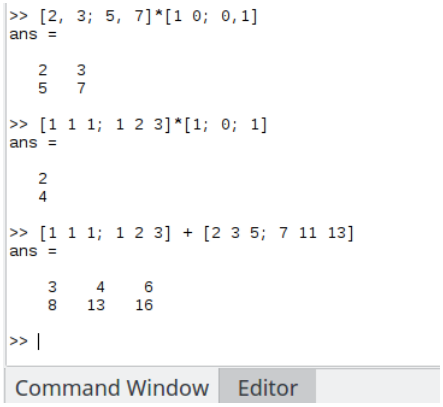
que usas cuando accedes *online* o estás en el laboratorio de la universidad. Aquí no pasaremos de este ámbito básico y me referiré en lo sucesivo a `matlab/octave`.

Una vez instalado, al ejecutarlo aparecerá una interfaz de usuario con una consola. Podemos usarlo como una calculadora tecleando la cuenta que queramos hacer y pulsando la tecla `Intro`. La sintaxis para las matrices consiste en escribir los elementos entre `[` y `]` separados con espacios o comas en cada fila e indicando el cambio de fila con punto y coma. Por ejemplo, la matriz identidad 2×2 sería `[1 0; 0 1]` o `[1, 0; 0, 1]`. Como veremos, hay abreviaturas para matrices especiales.

En la imagen de la izquierda hay algunos cálculos típicos de una calculadora. Tecleando `format long` obtenemos más de las cinco cifras significativas por defecto. A la derecha hay ejemplos de suma y producto de matrices.



Cálculos numéricos en la GUI de `octave`



Cálculos con matrices

En cuanto queramos hacer algo más que un cálculo breve, es mejor utilizar un fichero en vez de teclear en la consola. Por ejemplo, creemos con nuestro editor favorito (`matlab/octave` tiene uno incorporado si lo prefieres) el archivo `myprog.m`, donde la extensión `.m` es obligatoria, conteniendo:

```
1  % Mi primer programa
2  A = [1, 0, 1; 2, -1, 1];
3  B = [-2, 1, 0; 3, 1, 1];
4  % Calcula la suma de matrices
5  A+B
6  % La traspuesta de B es
7  B'
8  % El producto de A por la traspuesta de B es
9  A*B'
10 % Esto da A
11 A+zeros(2,3)
12 % y esto vuelve a dar A*B'
13 A*B'*eye(2)
```

Por cierto, indico los números de línea para hacer referencia a ellos y facilitar la copia, no hay que teclearlos. Al escribir en la consola del interfaz `myprog` se ejecutarán secuencialmente estas operaciones Es importante que hayamos arrancado `matlab/octave` en el mismo lugar donde está `myprog.m` o que hagamos que reconozca el *path*. Con las líneas 5 y 9 tendremos una comprobación del cálculo de suma y producto de matrices que se dio como ejemplo en la primera sección. Como es fácil adivinar, `%` es el prefijo

que indica un comentario. El apóstrofo corresponde a † y por tanto traspone las matrices reales. Con `zeros(n)` se tiene la matriz nula $n \times n$ y en general `zeros(m,n)` da la de dimensiones $m \times n$, mientras que `eye(n)` es la matriz identidad $n \times n$. Un punto y coma tras de una línea no muestra el resultado de la operación. Si en las línea 2 suprimimos este punto y como final, en la salida veríamos `A =` y la matriz indicada.

En `octave` nos puede resultar cansino que cuando la salida sea un poco larga haya que pasar línea a línea. Para desactivarlo basta escribir en la consola `more off`.

Terminaré refiriéndome al *software* libre para cálculo simbólico `sagemath`. Si solo vas a utilizarlo para álgebra lineal quizá no merezca la pena que pases por su instalación, la cual puede ser dura (¿o imposible?) si no tienes la distribución Linux adecuada. En ese caso todavía puedes ejecutar programas o cálculos cortos *online* en `https://sagecell.sagemath.org/`. Está basado en `python` y si conoces este lenguaje de programación podrás utilizarlo para hacer bucles y otras cosas. El constructor de matrices admite diversas formas. Con `matrix(m,n,L)` tenemos una matriz (simbólica) `m` × `n` cuyos elementos vienen dados por la lista `L`, la cual puede venir dada por los elementos leídos por filas o ser una lista de listas, una por cada fila.

La traducción a `sagemath` de las líneas 1–9 del programa anterior sería:

```
1  # Mi primer programa
2  A = matrix(2,3, [1, 0, 1, 2, -1, 1])
3  B = matrix(2,3, [[-2, 1, 0], [3, 1, 1]])
4  # Calcula la suma de matrices
5  print(A+B)
6  # La traspuesta de B es
7  print(B.transpose())
8  # El producto de A por la traspuesta de B es
9  print(A*B.transpose())
10 # Esto da A
11 print(A+zero_matrix(2,3))
12 # y esto vuelve a dar A*B^t
13 print(A*B.transpose()*identity_matrix(2))
```

El prefijo para comentario `#` y el comando `print` para mostrar en pantalla[4], están tomados de `python`. Si usas `sagemath` en consola, es decir, sin un interfaz como la versión *online*, esto debe ir en un archivo con extensión `.sage`, digamos `myprog.sage`, que se ejecuta con `%runfile myprog.sage`.

El hecho de que las matrices construidas de la forma anterior sean simbólicas significa en la práctica que los cálculos son exactos y que admiten variables. Por ejemplo, con `matrix(2,2,[sqrt(2),x,0,1])^2` calcularíamos el cuadrado, la multiplicación por sí misma, de una matriz 2×2 que contiene la variable `x` y en esta operación no se sustituye $\sqrt{2}$ por una aproximación con cierto número de cifras decimales, `sagemath` sabe que $(\sqrt{2})^2$ es exactamente 2.

[4]Disculpa futura: Yo uso habitualmente `sagemath` bajo `python2` en vez de bajo `python3` por tanto es posible que se me escape algún `print algo` que debiera ser `print(algo)`.