Tema 7

Ajustes

Dada una colección de puntos (x_i, y_i) con $i = 1, 2, \ldots, N$ (que podrían ser los resultados de un experimento) necesitamos generar una función que los ajuste para posteriormente realizar cálculos con ellos, como p. ej. derivadas, integrales, o cálculo de valores de y correspondientes a x no incluidos en el conjunto de datos.

Siempre que dispongamos de un conjunto de datos y de un modelo matemático o teoría que nos diga qué tipo de relación funcional, dependiente de una serie de parámetros, deberían cumplir esos datos lo mejor es realizar un ajuste por *mínimos cuadrados* para obtener los valores de los parámetros que mejor se ajustan a nuestros datos. Cuando no dispongamos de ninguna ley o teoría que nos diga qué tipo de dependencia funcional cumplen nuestros datos experimentales ya no podremos realizar un ajuste por mínimos cuadrados. En ese caso, si necesitamos calcular un dato que no está incluido en la serie de datos experimentales podemos hacerlo por medio de una función de interpolación. Una función de interpolación es una función matemática que pasa por todos nuestros datos experimentales. Existen diversas formas de generar interpolaciones, las más frecuentes son la interpolación lineal, la interpolación de Lagrange y la interpolación por *splines*, todas ellas tienen sus ventajas y sus inconvenientes. Con cualquiera de estos métodos de interpolación lo más frecuente es que la fórmula de interpolación proporcione valores más o menos aproximados para puntos que se encuentren intercalados entre dos puntos experimentales consecutivos, y que sean totalmente inexactas (produciendo errores inmensos) si pretendemos aplicarlas para calcular valores que estén fuera del conjunto de puntos experimentales, en cuyo caso estaríamos haciendo una *extrapolación*.

Para el estudio de este tema familiarícese con las funciones que el MAXIMA proporciona para realizar ajustes e interpolaciones

- Ajustes por mínimos cuadrados:

  - `lsquares_estimates(datos, variable, modelo, parámetros)`
    El mejor método para realizar ajustes es el de mínimos cuadrados, para ello en MAXIMA utilizamos el comando `lsquares_estimates()` al que suministramos los `datos` que queremos ajustar agrupados en una matriz (M : matrix([x1, y1], [x2, y2], ... [xN, yN])), la `variable` independiente (p. ej. x), como `modelo` suministramos la forma general de la función de x a la que queremos ajustar esos datos (p. ej. $ax + b$), esta forma general depende de una serie de `parámetros` (en nuestro ejemplo a y b) cuyos nombres debemos suministrar como último argumento de la función `lsquares_estimates()`.

- Interpolaciones:
  - Interpolación lineal (unimos los puntos por tramos rectos): `linearinterpol(datos)`
  - Interpolación lagrangiana (construimos un polinomio de Lagrange que pasa por todos nuestros puntos): `lagrange(datos)`
  - Interpolación por *splines cúbicos* (construimos una función polinomial a trozos que pasa por todos nuestros puntos de forma continua y diferenciable): `cspline(datos)`
- Consulte `describe()` para una descripción más extensa de la sintaxis y para ver ejemplos de uso de estos comandos.

7.1. De datos aislados a funciones

Un problema muy habitual en física computacional consiste en extraer de un conjunto de datos experimentales (del tipo (x_i, y_i), con $i = 1, 2, \ldots, N$) una función matemática que de alguna forma reproduzca dichos datos. El objetivo de esto es múltiple:

- por un lado se pretende conocer qué tipo de dependencia funcional siguen esos datos, es decir ¿existe alguna función $y = f(x)$ que nos reproduzca estos datos? si es así ¿cómo podemos encontrar esa función?
- por otro lado se pretende poder predecir valores de y correspondientes a x no incluidos en el conjunto de datos (p. ej, suponiendo que las x_i están ordenadas de manera creciente ¿qué valor de y corresponde a $x = (x_1 + x_2)/2$?)
- y también se pretende realizar operaciones de cálculo como integrales o derivadas sobre esos datos (p. ej. ¿cómo podemos calcular la derivada dy/dx (o en general $d^n y/dx^n$) a partir de los datos?)

Las técnicas para lograr estos objetivos se pueden clasificar en dos grandes familias complementarias: *interpolación* y *ajustes*, cada una de estas familias tiene sus ventajas y sus inconvenientes y es importante conocerlas. En este capítulo nos centraremos en los tres métodos de interpolación y ajustes más sencillos y habituales: interpolación lineal, splines cúbicos, polinomio interpolador de Lagrange y el método de ajuste por mínimos cuadrados.

7.1.1. Interpolación

Dado un conjunto de valores (x_i, y_i) (con $i = 1, 2, \ldots, N$) una función de interpolación es cualquier función $f(x)$ tal que $f(x_i) = y_i$ para todo el conjunto de valores. Existen diversas formas de construir funciones de interpolación, éstas se pueden clasificar en 2 grandes grupos:

- Funciones de interpolación locales: la función de interpolación se define como una función “a trozos”.
- Funciones de interpolación globales: una única función interpola todos los puntos (x_i, y_i).

El ejemplo más sencillo de función de interpolación local (o a trozos) consiste en unir todos los puntos (x_i, y_i) por trazos rectos, es decir, en cada intervalo $[x_i, x_{i+1}]$ construimos la aproximación lineal (dada por el polinomio de orden uno: $y = ax+b$) que pasa por los extremos del intervalo ((x_i, y_i) y (x_{i+1}, y_{i+1})). Esta fórmula de interpolación es muy sencilla y produce una función continua, pero con primera derivada discontinua en los nodos de interpolación x_i.

El siguiente ejemplo típico de función de interpolación local (o a trozos) son los *splines cúbicos*. En este caso, en lugar de unir los puntos por trazos rectos los unimos mediante polinomios de orden 3, imponiendo que la función de interpolación y sus dos primeras derivadas sean continuas en los nodos de interpolación, e imponiendo que la segunda derivada sea nula en el primer y el último nodo de interpolación para cerrar el sistema. Esta función es algo más complicada que la primera, pero produce una función continua y con sus dos primeras derivadas continuas en todo el intervalo.

El ejemplo clásico de función de interpolación global es el polinomio de interpolación de Lagrange: dados los N puntos (x_i, y_i) se puede construir un polinomio de orden $N-1$

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{N-1} x^{N-1},$$

de tal forma que imponiendo que esta función pase por los N nodos de interpolación ($f(x_i) = y_i$ para $i = 1, 2, \ldots, N$) obtenemos un conjunto de N ecuaciones que nos determina los valores de los N coeficientes del polinomio de interpolación.

Es muy sencillo escribir funciones que implementen estos métodos de interpolación usando cualquiera de los lenguajes de programación habituales, de todas formas estas funciones ya están programadas en el wxMaxima, para usarlas primero debemos cargar el paquete de interpolaciones:

```
load(interpol)$
```

posteriormente las funciones `linearinterpol` (para interpolación lineal), `cspline` (para esplines cúbicos) y `lagrange` (para interpolación por Lagrange), aplicadas sobre un conjunto de datos (x_i, y_i), nos devuelven la correspondiente fórmula de interpolación.

La primera parte de los ejercicios de este tema consiste en aplicar estas funciones sobre los conjuntos de datos que pueden encontrarse en la página web de la asignatura para generar, y posteriormente visualizar con `plot2d`, las correspondientes fórmulas de interpolación. Para cargar los conjuntos de datos lo que hay que hacer es:

```
data : read_nested_list(concat(path, filename))$
```

donde en la variable `path` tenemos el directorio donde está el archivo de datos y la variable `filename` es el nombre del correspondiente archivo de datos. Por supuesto, esto se podría hacer directamente "a mano", pero conviene aprender cómo se cargan archivos de datos de manera automática, ya que esto nos permitirá en el futuro realizar estas operaciones de manera automática operando sobre grandes cantidades de archivos. En otras palabras, imagine que en lugar de cargar un archivo de datos tuviera que cargar 10,000 archivos ¿lo haría a mano?

Al aplicar estas funciones sobre los conjuntos de datos disponibles en la página web de la asignatura vemos que el output que se obtiene al visualizar el resultado de `linearinterpol` o `cspline` es siempre lo que cabría esperar: los puntos unidos por trazos rectos en un caso y por una función suave en el segundo. Sin embargo, en el output del polinomio interpolador de Lagrange se observan unas oscilaciones de gran amplitud cerca de los extremos del conjunto de nodos de interpolación (para visualizar

los resultados hay que fijar manualmente la escala de ordenadas que se usa en `plot2d` mediante `[y, ymin, ymax]`), cuya presencia limita la aplicabilidad del polinomio interpolador de Lagrange. Para que el el polinomio interpolador de Lagrange proporcione una aproximación precisa a la función definida por el conjunto de datos es necesario que esta función sea suave (continua e infinitamente diferenciable) y que los valores de la variable independiente cumplan ciertas condiciones (los x_i deben corresponder a las abscisas de una fórmula de quadratura gaussiana), que no se verifican si el conjunto de datos experimentales se basa en valores x_i equiespaciados. Normalmente se da por hecho que el polinomio interpolador de Lagrange no sirve para realizar extrapolaciones (es decir, para predecir valores de y correspondientes a x fuera del intervalo de interpolación), pero si el conjunto de datos de partida no es suficientemente suave o los valores x_i en que se basa este conjunto no son los adecuados, la presencia de estas fuertes oscilaciones hará que el polinomio interpolador de Lagrange ni siquiera sea adecuado para predecir valores de y correspondientes a x dentro del intervalo de interpolación.

Por definición una función de interpolación pasa *exactamente* por los nodos de interpolación en que se basa, si el conjunto de valores no corresponde realmente a una función polinómica, y los nodos de interpolación son equiespaciados, es frecuente que el polinomio interpolador de Lagrange tenga estas oscilaciones cerca de los límites del intervalo de interpolación, limitando seriamente la utilidad de dicha función, tal y como sucede en todos ejemplos suministrados. Estas fuertes oscilaciones se podrían haber evitado si los nodos de interpolación, en lugar de estar equiespaciados, hubiesen estado distribuidos con una densidad creciente cerca de los límites del intervalo de interpolación. De todas formas, en lugar de generar los datos de la manera idónea para que el polinomio de interpolación de Lagrange funcione bien, hemos preferido generarlos tal y como se obtienen típicamente en un experimento, es decir, equiespaciados en la variable independiente. Con esto queremos mostrar que hay que tener precaución al manejar funciones de interpolación, ya que pueden inducirnos a errores, incluso si las usamos para predecir resultados dentro del intervalo de interpolación. Dado que las funciones de interpolación pasan *exactamente* por los puntos de interpolación en que se basan, sólo tiene sentido hacer una interpolación cuando el conjunto de valores está exento de error, pero no cuando los valores de que disponemos tienen errores apreciables. Esto es lo que sucede en los tres ejemplos propuestos, en los que los conjuntos de datos se han generado aplicando una cierta función sobre el conjunto de abscisas, añadiendo posteriormente un poco de ruido aleatorio.

De todas formas de esta discusión no debe extraerse la conclusión de que los polinomios interpoladores de Lagrange son poco prácticos. Si el conjunto de valores de que se dispone es suficientemente preciso las interpolaciones son extremadamente útiles. En particular, si los abscisas de interpolación corresponden a las abscisas de una fórmula de quadratura gaussiana (que ya se estudiará en la asignatura de cálculo numérico), y la función a la que responden estos valores es suficientemente suave, el polinomio interpolador de Lagrange es extremadamente preciso y útil.

7.1.2. Ajustes

Es evidente que ninguna de las funciones de interpolación construidas nos sirve para hacer predicciones fuera del intervalo de interpolación, si usamos cualquiera de estas funciones para predecir el valor de y correspondiente a una x fuera del intervalo

de interpolación el error será, en la inmensa mayoría de los casos, enorme. Por ese motivo, a menos que los valores (x_i, y_i) sean *exactos* (y que conozcamos la forma de la función a que corresponden) más que una interpolación lo que conviene hacer es un ajuste por mínimos cuadrados. Para ello suponemos que los datos suministrados corresponden a cierta forma funcional, como p. ej.

$$y = a + bx^c,$$

dependiente de algunos parámetros (en este caso a, b y c) cuyos valores se determinan imponiendo que el módulo al cuadrado de la distancia entre los datos y esta función sea mínimo. Este tipo de aproximación es un *ajuste por mínimos cuadrados* y, a diferencia de como sucedía en la interpolación, no pasa de forma exacta por ninguno de los datos, sino que pasa más o menos cerca de todos ellos.

Por supuesto, para que el ajuste por mínimos cuadrados funcione bien es necesario que los datos que queremos ajustar correspondan, al menos de manera aproximada, a la forma funcional que empleamos para hacer el ajuste. Es decir, necesitamos algo de *información adicional* para escoger una forma funcional adecuada. Existen dos formas de tener esta información adicional:

- Disponemos de un modelo físico que nos permite predecir esta forma funcional. Por ejemplo, si los datos corresponden a la posición de un oscilador armónico frente al tiempo sabemos que y debería estar dada por

  $$y = A\,\mathrm{sen}(\omega x + \delta)$$

  de modo que un ajuste por mínimos cuadrados nos permitiría obtener la amplitud A del oscilador, su frecuecia ω y la fase δ.

- Si no disponemos de un modelo físico en principio no hay ninguna manera sencilla de obtener la forma funcional apropiada, sin embargo, en algunos casos la mera inspección de los datos puede darnos algunas ideas, tal y como explicamos más abajo.

  - Supongamos que los datos suministrados corresponden (aproximadamente) a una relación lineal

    $$y = ax + b$$

    al representarlos gráficamente esto salta a la vista, y nos sugerirá hacer el ajuste basándonos en la sencilla relación lineal de arriba.

  - Otro caso sencillo es aquel en que los datos corresponden a una relación de tipo exponencial

    $$y = ae^{bx}$$

    entonces al representarlos gráficamente en una gráfica semi-logarítmica (es decir $\ln y$ frente a x) veremos una línea recta, que nos sugerirá probar la dependencia funcional de arriba.

  - Por último, si los datos responden a una relación tipo *ley de potencias*

    $$y = ax^b$$

    entonces al representarlos en una gráfica logarítmica ($\ln y$ frente a $\ln x$) veremos una línea recta, que nos indicará que debemos probar ese tipo de dependencia.

Estos tres tipos de dependencia son muy habituales en física, pero desde luego no son los únicos que se observan, por ejemplo también es habitual encontrar leyes de potencias desplazadas ($y = y_0 + ax^b$), exponenciales *estiradas* ($y = ae^{bx^c}$), logaritmos, funciones trigonométricas o combinaciones de varias de estas relaciones, en cuyo caso encontrar la forma funcional adecuada se complica. En última instancia para hacer ajustes en casos complicados es imprescindible tener un buen modelo físico y bastante experiencia.

Cada uno de los tres archivos de datos disponibles en la página web de la asignatura corresponde a uno de los tres casos sencillos que hemos indicado, y se pide identificar cada uno de ellos y obtener el correspondiente ajuste.

Al contrario a como sucedía con las interpolaciones, si un ajuste funciona bien normalmente sí puede usarse para hacer extrapolaciones (siempre con cierta precaución), es decir, para predecir valores de y correspondientes a x fuera del intervalo de interpolación.

7.2. Funciones del Maxima que debemos aplicar

La función del WXMAXIMA para realizar ajustes por mínimos cuadrados es `lsquares_estimate`; para usarla es necesario cargar antes el paquete de ajustes:

```
load(lsquares)$
```

Aparte de esto puede ser conveniente usar estas funciones:

- `data : read_nested_list(concat(path, filename))$` (para leer el archivo de datos)
- `apply(matrix, data)` (para convertir la lista de datos en una matriz para `lsquares_estimat`

En la ayuda del WXMAXIMA están todos los detalles sobre la sintaxis de estas funciones.

7.3. Problemas resueltos

Ejercicio 1

Dadas las colecciones de puntos experimentales con la forma (x_i, y_i) que puede encontrar en la página web de la asignatura, realice las correspondientes interpolaciones.

En primer lugar cargamos el paquete de interpolaciones y el primer archivo de datos:

```
load(interpol)$
data : read_nested_list(concat(path, "data-1.out"))$
```

A continuación hacemos las interpolaciones mediante las funciones `linearinterpol`, `lagrange` y `cspline`

```
aaa : linearinterpol(data)$
bbb : lagrange(data)$
ccc : cspline(data)$
```

Para visualizar los resultados (líneas continuas), junto con los datos de partida (puntos) empleamos la función `plot2d`

```
plot2d(
    [ [discrete, data], aaa, bbb, ccc ], xdom, ydom
    [ style, points, lines, lines, lines ]
);
```

donde previamente hemos definido el intervalo que queremos mostrar en la gráfica mediante:

```
xdom : [ x, apply(min, transpose(Mdata)[1]), apply(max, transpose(Mdata)
   [1]) ]
ydom : [ y, apply(min, transpose(Mdata)[2]), apply(max, transpose(Mdata)
   [2]) ]
```

siendo `Mdata` el conjunto de datos en forma matricial

```
Mdata : apply(matrix,data)$
```

de tal forma que `transpose(Mdata)[1]` es el conjunto de abscisas y `transpose(Mdata)[2]` el conjunto de ordenadas.

Para ver el eje x (respectivamente el eje y) en escala logarítmica hay que usar la opción `[logx]` (respectivamente `logy`) dentro de la función `plot2d`. Para ver ambos ejes en escala logarítmica hay que usar `[logx], [logy]`.

Análogamente haríamos lo mismo con los restantes archivos de datos.

Ejercicio 2

A la vista de las anteriores interpolaciones formule un modelo matemático aproximado y realice el correspondiente ajuste por mínimos cuadrados.

En primer lugar cargamos el paquete de ajustes por mínimos cuadrados:

```
load(lsquares)$
```

En principio vamos a operar con tres modelos sencillos, todos ellos dependientes sólo de dos parámetros que llamaremos A y B. Definimos los modelos que vamos a usar:

```
model1 : A + B * x$
model2 : A * exp( B * x )$
model3 : A * x^B$
```

Mediante inspección de los datos vemos que el primer archivo de datos corresponde al modelo 1, entonces para obtener el ajuste por mínimos cuadrados hacemos

```
model : model1$
aux : lsquares_estimates( Mdata, [x,y], y = model, [A, B] )$
f : float(subst(aux[1], model));
```

donde previamente hemos definido la matriz de datos `Mdata` tal y como se indica en el primer ejercicio.

Operamos de manera análoga con los archivos de datos restantes, lo único que hay que hacer es cargar el archivo de datos correspondiente, re-defininir `Mdata` y asignar a la variable `model` el modelo que consideremos apropiado (`model1`, `model2` o `model3`).

Ejercicio 3

Visualice los resultados usando `plot2d`.

Para visualizar el ajuste junto con los datos y las interpolaciones hacemos:

```
plot2d( [ [discrete, data], aaa, bbb, ccc, f ], xdom, ydom
       [ style, points, lines, lines, lines, lines ] );
```