Disclaimer: Este apunte no es autocontenido y fue pensado como un repaso de los conceptos, no para aprenderlos de aquí directamente.

1. Aritmética de la computadora

Representación en punto flotante. Los números en punto flotante se representan como sxb^y donde b es una constante dada, s es un bit de signo, x es llamada mantisa y esta entre 0 y $b-1$ y y es llamado exponente y es un entero. En la computadora usual se usa $b=2$ y la mantisa se representa solo la parte a partir del segundo dígito decimal, ya que por convención la mantisa empieza "0.1...". Esto se llama normalización. Esta representación implica que la distribución de los números no es del todo lineal, pero tampoco del todo logarítmica, sino un híbrido entre ambas (i.e., entre cada par de potencias de 2 consecutivas hay la misma cantidad de números distribuidos uniformemente).

Errores. El error absoluto de A representado como a se define como $|A-a|$ y el error relativo como $|A-a|/|A|$.
El error del cálculo de una función $f(x_1,...,x_n)$ se define como

$$\mathcal{E}(f(x_1,...,x_n)) = \mathcal{E}_f + \sum_{i=1}^{n} \frac{\frac{\partial f}{\partial x_i}}{f} x_i \mathcal{E}(x_i),$$

donde $\mathcal{E}_f$ es el error intrínseco de hacer la operación f y $\mathcal{E}(x_i)$ es el error de representación de x_i.

Condición y estabilidad.
Número de condición: términos que acompañan a los errores en las variables. (está bien condicionado si los números de condición son acotables por una constante).
Estabilidad del algoritmo: términos que acompañan a los errores de las operaciones (es estable si la estabilidad es acotable por una constante).
Para dos algoritmos distintos que calculan la misma expresión aritmética los números de condición son iguales. La estabilidad no necesariamente.

NOTA: Para las próximas secciones usaremos la siguiente notación: Los problemas lineales se identifican con una matriz A de $n \times n$ y un vector b de $n \times 1$. El problema entonces es buscar un vector x de $n \times 1$ tal que $Ax = b$.

2. Factorización LU

La factorización LU sirve para resolver varios problemas lineales. Si el problema está dado por $Ax = b$, la idea es encontrar matrices L triangular inferior y U triangular superior tales que $LU = A$, de manera de resolver el sistema $LUx = b$. Esto último es facil de hacer en 2 pasos: buscamos x_0 tal que $Lx_0 = b$, lo cual se puede hacer fácil porque es un sistema lineal en el que la matriz es triangular. Luego buscamos x tal que $Ux = x_0$, que es fácil por la misma razón. De este modo, $LUx = Lx_0 = b$, y el x es efectivamente la solución del sistema original. La idea de la factorización es que puede obtenerse en el mismo tiempo necesario para resolver un sistema lineal $\mathcal{O}(n^3)$, pero una vez que se tiene resolver un problema $Ax = b_i$ para cualquier b_i puede hacerse en $\mathcal{O}(n^2)$ como ya fue descripto. Esto quiere decir que si uno tiene que resolver k problemas $Ax = b_i (1 \leq i \leq k)$ haciendo k eliminaciones gaussianas tarda $\mathcal{O}(n^3 k)$, pero haciendo la factorización LU de A tarda $\mathcal{O}(n^3 + n^2 k)$ lo cual es siempre mejor o igual (en particular notemos que si $k = 1$, no estamos perdiendo nada de complejidad por hacer la factorización en lugar de la eliminación gaussiana directamente).

Para desambiguar a veces se pide que la diagonal de L sean solo unos (factorización de Doolittle) o la diagonal de U solo unos (factorización de Crout).

Eliminación Gaussiana. La eliminación gaussiana es un método para resolver problemas lineales. En particular también sirve para producir una factorización LU. Aquí describiremos la eliminación utilizada para hacer dicha factorización, asumiendo que no es necesario pivoteo. En particular produciremos la factorización de Doolittle (es decir, L con unos en la diagonal).
La idea de la eliminación es encontrar dos sucesiones de matrices $L^{(k)}$ y $U^{(k)}$ tal que para todo $L^{(n)}U^{(n)} = A$ y $L^{(n)}$ es triangular inferior y $U^{(n)}$ triangular superior. Como invariante adicional tomaremos que para todo k $L^{(k)}U^{(k)} = A$, y para $i < j < k$ se cumple $U^{(k)}_{i,j} = 0$, es decir, $U^{(k)}$ tiene las primeras k columnas sin 0s abajo de la diagonal, y $L^{(k)}$ es siempre triangular inferior con unos en la diagonal. Notemos que si $k = n$ el invariante implica la postcondición. Las sucesiones de matrices se construyen según el siguiente algoritmo:

Sean $L^{(1)} = I$ y $U^{(1)} = A$
para $k = 2$ hasta n
aquí asumiremos que $U^{(k-1)}_{k-1,k-1} \neq 0$ *(no es necesario pivoteo)*
Sea las matrices M y M^{-1} de $n \times n$ e inversas entre sí dadas por:

$$M_{i,j} = \begin{cases} 1 & i = j \\ -U^{(k-1)}_{i,k}/U^{(k-1)}_{k,k} & i > k \text{ y } j = k \\ 0 & \text{en otro caso} \end{cases} \qquad M^{-1}_{i,j} = \begin{cases} 1 & i = j \\ U^{(k-1)}_{i,k}/U^{(k-1)}_{k,k} & i > k \text{ y } j = k \\ 0 & \text{en otro caso} \end{cases}$$

Sean $L^{(k)} = L^{(k-1)}M^{-1}$ y $U^{(k)} = MU^{(k-1)}$

Notemos que $L^{(1)}U^{(1)} = A$ trivialmente y que $L^{(k)}U^{(k)} = L^{(k-1)}M^{-1}MU^{(k-1)} = L^{(k-1)}U^{(k-1)} = A$ por lo cual el invariante se conserva. El hecho de que $L^{(k)}$ es siempre triangular superior y $U^{(k)}$ lo es en sus primeras k columnas sale de observar la forma que toma la multiplicación por M^{-1} y M respectivamente (en el caso de L es producto de dos matrices triangular inferior, en el caso de U se ve como la multiplicación solo cambia filas por debajo de la $k -' esima$ y siempre manteniendo las columnas menores a k intactas y anulando la columna k).
Es importante notar que las matrices $L^{(k)}$ y $U^{(k)}$ pueden ir guardandose todas en el mismo espacio de $n \times n$. A su vez, la matriz M puede guardarse utilizando solo $\mathcal{O}(n)$ espacio y los productos de L o U por M o M^{-1} realizados consiguientemente en $\mathcal{O}(n^2)$. De este modo, todo el proceso utiliza solo $n^2 + \mathcal{O}(n)$ memoria y se realiza en $\mathcal{O}(n^3)$ tiempo.

Factorización $PA = LU$. Si la eliminación Gaussiana precisara de pivoteo, y lo conocemos de antemano (o lo calculamos), podemos factorizar $PA = LU$ con P la matriz de permutación que hace los intercambios de fila necesarios para pivotear la eliminación Gaussiana. Esto también conduce a una eficiente resolución de los sistemas $Ax = b_i$, ya que P, por ser de permutación, es trivialmente inversible.

Propiedad de existencia. Si las n submatrices principales de A son no singulares, entonces A tiene factorización LU.

3. Factorización QR

Una matriz Q de $n \times n$ es ortogonal si todas sus columnas tienen norma 1 y el producto interno de 2 distintas de ellas es 0. En particular cumplen $QQ^t = Q^tQ = I$ y por lo tanto se cumple lo mismo para las filas.

La factorización QR de una matriz consiste en, dada A, encontrar Q ortogonal y R triangular superior tal que $QR = A$. De esta manera, para resolver el sistema $Ax = b$ debemos resolver $QRx = b$ o equivalentemente $Rx = Q^{-1}b = Q^tb$, cosa que podemos hacer fácilmente (si es posible) ya que R es triangular. Para todo A existe la factorización QR.

La factorización es conveniente para resolver varios sistemas de la forma $Ax = b_i$ para varios b_i, ya que solo hace falta computar la factorización una vez (porque A está fijo) y luego se puede resolver cada sistema en $\mathcal{O}(n^2)$ en lugar del $\mathcal{O}(n^3)$ para resolver un sistema en general.

Para los métodos de obtener una factorización QR es importante notar que el producto de matrices ortogonales es ortogonal.

Transformación de Givens. La idea es ir multiplicando A sucesivamente por matrices de rotación (que son ortogonales) hasta que se convierta en R. El producto de las inversas de las rotaciones es, por lo tanto, Q. Para que Q^t sea una rotación debe ser, para ciertos p, q:

$$Q^t_{i,j} = \begin{cases} 1 & i = j \notin \{p, q\} \\ 0 & i \neq j \wedge \{i, j\} \neq \{p, q\} \\ \cos\theta & i = j \in \{p, q\} \\ \operatorname{sen}\theta & i = p, j = q \\ -\operatorname{sen}\theta & i = q, j = p \end{cases} \qquad \text{por ejemplo, para } p = 2 \text{ y } q = 5: \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \cos\theta & 0 & 0 & \operatorname{sen}\theta \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -\operatorname{sen}\theta & 0 & 0 & \cos\theta \end{pmatrix}$$

En particular queremos θ tal que provoque un cero particular en la matriz A. Miremos por ejemplo el caso de 2×2:

$$\begin{pmatrix} \cos\theta & \operatorname{sen}\theta \\ -\operatorname{sen}\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix} \Rightarrow \begin{cases} \cos\theta x_1 + \operatorname{sen}\theta x_2 &= & * \\ -\operatorname{sen}\theta x_1 + \cos\theta x_2 &= & 0 \end{cases}$$

y luego tomo como solución $\operatorname{sen}\theta = x_2/\left\|(x_1,x_2)\right\|$ y $\cos\theta = x_1/\left\|(x_1,x_2)\right\|$ resulta que

$$Q^t = \frac{1}{\|(x_1,x_2)\|}\begin{pmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{pmatrix}$$

produce el cero que queríamos. Podemos generalizar esto a Q de $n \times n$ reemplazando $\operatorname{sen}\theta = A_{q,p}$ y $\cos\theta = A_{p,p}$ en la definición original de Q^t. Haciendo eso iterativamente para cada p, q donde precisemos un cero nos da la factorización:

```
Sea Q* = I
Sea R = A
para q = 1 hasta n
    para p = q + 1 hasta n
        Sea Q^t como fue definido arriba para p, q.
        Q* = Q*Q
        R = Q^t R
```

El invariante es que $Q^*R = A$, que se mantiene porque en cada paso multiplico en medio de ambos por $QQ^t = I$. Q^* es siempre ortogonal por ser producto de ortogonales y R mantiene ceros en las posiciones p, q por las que ya se pasó. Al final del algoritmo, R tiene ceros en todas las posiciones debajo de la diagonal. Dado que las matrices Q y Q^t se pueden tienen solo 4 elementos distintos a la identidad y que los productos pueden hacerse por lo tanto en $\mathcal{O}(n)$ el tiempo total del algoritmo de factorización es $\mathcal{O}(n^3)$.

Transformación de Householder. La idea es usar reflexiones respecto de rectas que pasan por el 0, que también pueden verse como productos de matrices ortogonales. En particular, si queremos reflejar por la recta v: sea u perpendicular a v ($u^tv = 0$) de norma 1. Si la reflexión está dada por la matriz Q queremos: $Qu = -u$ y $Qv = v$. Observemos que tomando $Q = I - 2uu^t$ esto se cumple ya que $Qu = (I - 2uu^t)u = u - 2u(u^tu) = u - 2u\left\|u\right\| = -u$ y $Qv = (I - 2uu^t)v = v - 2u(u^tv) = v$.
La idea para factorizar es generar ceros en una columna por vez, generando matrices Q_p^t que al multiplicarlas por A dejan ceros en la columna p, dejando intactas las columnas anteriores a p. De esta manera, $R = \prod_{p=n}^{1} Q_p^t A$ y $Q = \prod_{p=1}^{n} Q_p$. Es claro que $QR = A$. Como utilizamos reflexiones, generar ceros en una columna debe modificar algún otro elemento, para mantener fija la norma de la columna. En particular, modificaremos la diagonal. Sea $x_1, ..., x_n$ la columna p a modificar. Queremos como resultado $y_i = x_i$ si $i < p$ y $y_i = 0$ si $i > p$, por lo cual resulta $y_p = \left\|(x_p, x_{p+1}, ..., x_n)\right\|$. Ahora que sabemos que queremos mandar el punto x al y, solo debemos escoger la recta v y luego una perpendicular u de norma 1. La recta que une x con y, $x - y$ es perpendicular al eje de reflexión por su definición, por lo tanto nos sirve. Para que sea de norma 1, simplemente lo normalizamos y usamos $u_p = (x - y)/\left\|x - y\right\|$. Definamos entonces Q_p^t según los siguientes bloques:

$$Q_p^t = \begin{pmatrix} I_{p-1\times p-1} & 0 \\ 0 & I_{n-p+1\times n-p+1} - 2u_pu_p^t \end{pmatrix}$$

donde u_p es como fue descripto anteriormente. Se ve que las matrices Q_p^t y Q_p pueden ser representada simplemente como el vector u_p de tamaño n y por lo tanto podemos resolver sus productos por otras matrices en $\mathcal{O}(n^2)$. Dado que tenemos $\mathcal{O}(n)$ de esos productos, la complejidad total resultante para hayar la factorización es de $\mathcal{O}(n^3)$.

Propiedad de existencia y unicidad. Si A es no singular entonces existen únicas matrices Q ortogonal y R triangular superior tal que la diagonal de R consta solo de elementos estrictamente positivos.

4. Resolución de sistemas con matrices especiales

Matriz estrictamente diagonal dominante. Llamamos así a una matriz A cuando $2A_{i,i} > \sum_{j=1}^{n} |A_{i,j}|$. En estos casos, A es no singular y no es necesario pivoteo al hacer Gauss, por lo cual tiene factorización LU.

Matrices banda. Llamamos matriz banda pq a A cuando $A_{i,j} = 0$ siempre que $i + p \leq j$ y $j + q \leq i$. En particular, si A tiene factorización LU, L tiene banda q, U tiene banda p y la eliminación gaussiana para encontrar la factorización se puede hacer en $\mathcal{O}(npq)$.

Matrices simétricas. Una matriz A es simétrica si $A = A^t$. En este caso la factorización LU de A (si es que existe) puede escribirse en términos de L y una matriz diagonal $A = LU = LD(L^t)$. Si aparte la matriz es definida positiva podemos estar seguros que tiene factorización LU y aparte la podemos descomponer como $L'(L')^t$ tomando $L' = LD^{1/2}$.

5. Inestabilidad numérica al resolver sistemas lineales

Al resolver sistemas lineales se pueden presentar problemas numéricos. Estos a veces pueden ser resueltos o minimizados mejorando la forma de resolverlo.

Eliminación Gaussiana. Durante la eliminación gaussiana a veces hay "necesidad" de pivotear por encontrar un 0 en la diagonal. Dado que podemos pivotear, a veces puede convenirnos numéricamente hacerlo, mas allá de que no sea matemáticamente necesario. En particular, intentaremos que el número sobre la diagonal (por el que vamos a dividir) sea lo mas grande posible en valor absoluto, ya que sabemos que dividir por números cercanos a cero es problemático numéricamente. Denominamos *pivoteo parcial* a buscar el mejor coeficiente dentro de la columna correspondiente. Denominamos *pivoteo total* a buscarlo en toda la submatriz que queda a la derecha y abajo del coeficiente actual (necesitando un intercambio de filas y uno de columnas).

Números de condición. Denominamos el número de condición de una matriz A a $K(A) = \|A\| \left\|A^{-1}\right\| \geq 1$. Este número esta relacionado con cuán problemático numéricamente es resolver un sistema con matriz A: mientras mas grande sea, mas problemático. Esto se ilustra con la siguiente propiedad:
Propiedad: Sea $\bar{x}$ solución aproximada de $Ax = b$ con A no singular y residuo $r = b - A\bar{x}$. Podemos acotar el error absoluto y relativo de la solución de la siguiente manera:

- $\|x - \bar{x}\| \leq \|r\| \left\|A^{-1}\right\|$
- $\|x - \bar{x}\| / \|x\| \leq K(A) \|r\| / \|b\|$

6. Métodos iterativos para sistemas lineales

Los métodos iterativos para solucionar sistemas lineales generan una sucesión de soluciones aproximadas x_n que bajo ciertas condiciones converge a la solución del sistema.

Autovalores y radio espectral. Un autovalor λ y su correspondiente autovector $v \neq 0$ de una matriz A son tales que $Av = \lambda v$. Los autovalores también son las raizes del polinomio $\det(A - \lambda I)$.
El radio espectral $\rho(A)$ se define como el máximo valor absoluto de un autovalor de la matriz A.
A es una matriz convergente si $\lim_{k\to\infty} A^k = 0$.
Propiedad: $\rho(A) \leq \|A\|$ para toda norma inducida.
Propiedad: A es matriz convergente $\Leftrightarrow \rho(A) < 1$.
Propiedad: Si $\|A\| < 1$ entonces $I - A$ es no singular y $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$.

Métodos iterativos en general. La idea es utilizar la iteración $x^{(k+1)} = Tx^{(k)} + c$ para hallar la solución del sistema $x = Tx + c$.
Propiedad: Dicha sucesión converge $\Leftrightarrow \rho(T) < 1$.
Propiedad: Si para una norma inducida $\|T\| < 1$ entonces $x^{(k+1)} = Tx^{(k)} + c$ converge y $\left\|x - x^k\right\| \leq \|T\|^k \left\|x^{(0)} - x\right\|$.

Jacobi. Dentro del esquema general, usaremos $T = D^{-1}(L+U)$ y $c = D^{-1}b$ donde $A = D - L - U$, D es diagonal y L y U son triangular inferior y superior respectivamente con ceros en la diagonal. La idea sale de: $Ax = b \Leftrightarrow (D - L - U)x = b \Leftrightarrow x = D^{-1}b + D^{-1}(L + U)x$.

Gauss Siedel. Sea $A = D - L - U$ igual que antes. Usaremos $T = (D - L)^{-1}U$ y $c = (D - L)^{-1}b$. Esto sale de $Ax = b \Leftrightarrow (D - L - U)x = b \Leftrightarrow x = (D - L)^{-1}Ux + (D - L)^{-1}b$.

Convergencias conocidas. Si A es estrictamente diagonal dominante, entonces Jacobi y Gauss Siedel convergen.

6.1. Direcciones conjugadas

Direcciones conjugadas. n vectores u_1,...,u_n se llaman A-conjugados sii $\forall i \neq j \; u_i^t A u_j = 0$. Si además $\forall i \; u_i^t A u_i = 1$ se llaman A-ortogonales.

Supongamos que tenemos que resolver $Ax = b$ donde A es simétrica y definida positiva. Sea $Q(x) = x^t Ax - 2x^t b$. Se ve que $\mathbf{D}Q(x) = 0$ y $\mathbf{H}Q(x) = 2A$ es definido positivo, por lo cual x es mínimo de Q, entonces resolveremos el sistema buscando dicho mínimo.

Para esto utilizaremos una iteración de la forma $x_k = x_{k-1} + \alpha_{k-1} d_{k-1}$, donde α indica cuánto me muevo y d indica la dirección del movimiento. Si tengo dada la dirección, puedo plantear cuanto moverme buscando que Q se minimice lo

máximo posible. Como esto es una ecuación en una variable, simplemente busco el cero de la derivada respecto de α, que al resolverlo queda $\alpha_{k-1} = d_{k-1}^t(b - Ax_{k-1})/d_{k-1}^t Ad_{k-1}$.
Propiedad: Si $d_0, ..., d_{n-1}$ son A-ortogonales y eligiendo α según se explicó, definiendo $x_i = x_{i-1} + d_{i-1}\alpha_{i-1}$ resulta $Ax_n = b$.
dibujito de direcciones conjugadas
La idea es irse moviendo por direcciones conjugadas, que son linealmente independientes. Se puede ver como moverse una vez en cada "coordenada".
Propiedad: Si elegimos $d_k = r_k + \beta_k d_{k-1}$ donde $r_k = b - Ax_k$ y $\beta_k = r_k^t Ad_{k-1}/(d_{k-1}^t Ad_{k-1})$ entonces las direcciones resultan A-ortogonales.

7. Sistemas no lineales

Los sistemas no lineales pueden verse como buscar ceros de funciones de varias variables. En particular, sea $F : \mathbb{R}^n \to \mathbb{R}^n$ (asumiendo que tenemos n variables y n ecuaciones) e intentemos buscar sus ceros. Sea x^* tal que $F(x^*) = 0$.

Newton. Podemos generalizar el método de Newton utilizando la ecuación $F(x_k) + J(x_k)(x - x_k) \sim F(x)$, lo cual deriva en la iteración $x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})F(x_k)$ donde J es el jacobiano o la matriz diferencial de F. Multiplicar por la inversa de dicha matriz es el análogo a dividir por la derivada de Newton para 1 variable.
Propiedad: Si $J(x^*)$ es no singular, $\left\|J^{-1}(x^*)\right\| \leq \beta$, J es Lipschitz F es C^1 en un intervalo alrededor de x^*, entonces el método converge cuadráticamente si x_0 está suficientemente cerca de x.

Criterio de Broyden. Broyden puede verse como una generalización de secante a muchas dimensiones o también como una simplificación de Newton, que en lugar de utilizar el Jacobiano, lo aproxima según la última diferencia finita que tiene (es decir, toma $J(x_n)(x - x_n) \sim F(x) - F(x_{n-1})$). El problema es que, al contrario que en una dimensión, esto resulta en un sistema subdeterminado. Lo que Broyden propone es tomar la solución que minimiza los cambios a la aproximación del Jacobiano, de manera de obtener un método mas eficiente. **me queda la duda de si está bien así o hay que agregarle formalismo (cuentitas)**

8. Ceros de funciones

El problema es encontrar los x^* tal que $f(x^*) = 0$ para una función dada f. Hay métodos particulares varias familias particulares de funciones, pero nos concentraremos en métodos "generales", para familias grandes.

Un algoritmo que busca ceros de funciones genera una sucesión de resultados x_n de manera que $\lim_{n\to\infty} x_n = x^*$.

Convergencia p. La sucesión x_n tiene convergencia p si existe un k tal que:

$$\lim_{n\to\infty} \ell_{n+1}/\ell_n = c \neq 0 \text{ donde } \ell_n = |x_n - x^*|, \text{ y } \ell_{n+1} \leq k\ {\ell_n}^p.$$

Si $p = 1$ decimos convergencia lineal, si $p = 2$, cuadrática. Si $\lim_{n\to\infty} \ell_{n+1}/\ell_n = 0$ la convergencia es superlineal (mejor que lineal pero peor que cuadrática).

Criterios de parada.
- Error absoluto $|x_n - x_{n-1}| < \varepsilon$.
- Error relativo $|x_n - x_{n-1}|/|x_n| < \varepsilon$.
- Error sobre la función $|f(x_n)| < \varepsilon$.
- Sobre diferencia de funciones $|f(x_n) - f(x_{n-1})| < \varepsilon$.
- Sobre diferencia de funciones con error relativo $|f(x_n) - f(x_{n-1})|/|f(x_n)| < \varepsilon$.

Bisección. Se basa en el teorema de Bolzano: Dada $f : [a, b] \to \mathbb{R}, f(a)f(b) < 0, \exists c \in (a, b)$ tq $f(c) = 0$. Algoritmo:

Sean a y b tal que $f(a)f(b) < 0$.
Mientras se desee mas precisión
    Si $f((a + b)/2)f(b) > 0$
        $a = (a + b)/2$
    sino
        $b = (a + b)/2$.
Devolver $(a + b)/2$

Propiedad: La convergencia del método de bisección es lineal.

Punto Fijo. Decimos punto fijo de una función g al punto x^* tal que $g(x^*) = x^*$. Si queremos buscar los ceros de f podemos definir $g(x) = f(x) + x$ de manera que los puntos fijos de g son los ceros de f. También podemos hacer $g(x) =$ cualquier forma de despejar x en la expresión $f(x) = 0$.
Propiedad: Si $g : [a, b] \to [a, b]$ es continua entonces tiene un punto fijo. Importante notar que la imagen de $[a, b]$ tiene que ser $[a, b]$ cerrado!. Si además $|g'(x)| \leq K < 1$ para todo $a \leq x \leq b$ entonces el punto fijo es único.
Algoritmo: Ir haciendo $x_n = g(x_{n-1})$.
Propiedad: Si $g : [a, b] \to [a, b]$ es continua y $|g'(x)| \leq K < 1$ entonces el algoritmo converge al único punto fijo.
Propiedad: Si g es como en la propiedad anterior y $g'(p) = ... = g^{(n-1)}(p) = 0$ entonces la convergencia del algoritmo es por lo menos n. **agregar grafiquito de punto fijo**

Algoritmo de Newton. En particular veamos el punto fijo de $g(x) = x - h(x)f(x)$. Si $g(x^*) = x^*$ entonces $h(x^*)f(x^*) = 0$ y si $h(x^*) \neq 0$, entonces x^* es cero de f. En particular, $g'(x) = 1 - h'(x)f(x) - h(x)f'(x)$. Como queremos una derivada igual a 0 (para asegurar convergencia cuadrática) precisaríamos $0 = g'(x^*) = 1 - h(x^*)f'(x^*)$, entonces tomemos $h(x) = 1/f'(x) \Rightarrow g(x) = x - f(x)/f'(x)$.
Propiedad: Sea $f \in C^2[a, b], x^* \in [a, b], f(x^*) = 0$ y $f'(x^*) \neq 0$. Existe un $\delta > 0$ tal que la sucesión $x_{n+1} = x_n - f(x_n)/f'(x_n)$ converge a x^* si $x_0 \in (x^* - \delta, x^* + \delta)$.
Propiedad: Si f además es creciente y convexa, entonces tiene un único cero y el algoritmo de newton converge a el para todo x_0.
También podemos interpretar Newton desde el polinomio de Taylor, ya que $0 = f(x) \sim f(x_0) + f'(x_0)(x - x_0) \Rightarrow x = f(x_0)/f'(x_0) + x_0$.
Observación: Si hay ceros múltiples la convergencia cuadrática puede perderse.

Método de la secante. La idea es aproximar la derivada de Newton por una diferencia finita dividida. También puede verse como tomar la recta secante a los últimos dos puntos encontrados (derivada estimada) y utilizar como nuevo punto su intersección con la función.

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

La convergencia de este método es $\varphi = (1 + \sqrt{5})/2$ si se eligen los puntos iniciales suficientemente cerca del buscado.

Método Regula Falsi. Es como bisección pero en lugar de elegir el punto siguiente como el promedio de los lados $c = (a + b)/2$, elige la intersección de la recta secante que une $(a, f(a))$ con $(b, f(b))$. Formalmente $c = (f(b)a - f(a)b)/(f(b) - f(a))$. El método recibe su nombre del hecho de que no necesariamente converge bien, aunque lo pareciera.

9. Interpolación

La interpolación es, dados pares de valores $(x_1, y_1), ..., (x_n, y_n)$ (mediciones), encontrar una función f tal que $\forall i\ f(x_i) = y_i$. En particular la interpolación polinómica es cuando las funciones f que podemos elegir se restringen a polinomios.

Polinomio de Lagrange. El Polinomio de Lagrange es el único polinomio de grado mínimo que interpola un conjunto de puntos (x_i, y_i). Está definido por

$$p(x) = \sum_i \left(y_i \prod_{j \neq i} \frac{x - x_j}{x_i - x_j} \right).$$

Propiedad: Si todo $x_i \in [a, b]$ y f es C^{n+1} entonces para todo x existe $\xi_x \in (a, b)$ tal que $f(x) = p(x) + f^{(n+1)}(\xi_x) \prod_i (x - x_i)/(n + 1)!$.

Diferencias divididas. Es una forma recursiva de escribir el polinomio de lagrange. Sirve para hacerlo incrementalmente. Supongamos que el polinomio es $p(x) = \sum_{i=0}^{n} a_i \prod_{j<i+1}(x - x_j)$. La idea es definir $f[x_i x_{i+1} ... x_{i+k}] = (f[x_{i+1} ... x_{i+k}] - f[x_i ... x_{i+k-1}])/(x_{i+k} - x_i)$ y $f[x_i] = f(x_i)$. Se puede ver que el coeficiente a_k debe ser $f[x_1 ... x_{k-1}]$.

Splines. La idea de un spline es interpolar con polinomios, pero éstos pueden ser distintos para cada pareja de mediciones consecutivas (asumimos que $x_1 < x_2 < ... < x_n$. Buscaremos entonces polinomios $p_1, ..., p_{n-1}$ tal que para todo i se cumpla $p_i(x_i) = y_i$ y $p_i(x_{i+1}) = y_{i+1}$ (1). Si los polinomios son de grado 1 (rectas) estas ecuaciones completamente los definen. Si vamos a grados mas grandes, necesitamos mas ecuaciones para definirlos. En particular, para que la curva sea suave podemos pedir $p_i'(x_{i+1}) = p_{i+1}'(x_{i+1})$ (2) o lo mismo para la derivada segunda, tercera, etcétera. En particular consideremos los splines cúbicos (con polinomios de grado 3). Cada polinomio introduce 4 variables (sus 4 coeficientes). Tenemos $2n - 2$ ecuaciones de tipo (1) y $n - 2$ ecuaciones de tipo (2). Si igualamos en los puntos intermedios tanto la derivada primera como la segunda, esto nos da $2n - 2 + n - 2 + n - 2 = 4n - 6$ ecuaciones para nuestras $4n - 4$ variables (4 para cada uno de los $n - 1$ polinomios). Las 2 ecuaciones que hacen falta se completan según el tipo de spline. Natural:

Las derivadas segundas en los extremos son cero. Sujeto: La derivada primera en cada extremo se iguala a la derivada de la función original. Se puede demostrar en ambos casos que el sistema planteado tiene siempre una única solución.

10. Cuadrados mínimos

El problema de cuadrados mínimos es encontrar, dadas mediciones $(x_1, y_1), ..., (x_n, y_n)$ una función f tal que minimice el error cuadrático medio $\sum_i (f(x_i) - y_i)^2$.

Cuadrados mínimos lineales. El problema de cuadrados mínimos lineales es cuando restringimos la función f a una función lineal. Puedo encontrar f planteado minimizar $\sum_i (ax_i + b - y_i)^2$ como función de a y b e igualando el gradiente de dicha expresión a 0.

Cuadrados mínimos polinomiales. Aquí restringimos f a los polinomios de cierto grado máximo. También se resuelve mediante un sistema lineal, que resulta escribir la expresión a minimizar en función de los coeficientes de f y luego igualar el gradiente a 0.

Ahora generalicemos el problema a muchas variables: Tenemos una tabla de valores

$$(x_1^{(1)}, x_2^{(1)}, ..., x_k^{(1)}, y^{(1)}), ..., (x_1^{(n)}, x_2^{(n)}, ..., x_k^{(n)}, y^{(n)})$$

y queremos la función f que minimiza $\sum_i (f(x_1^{(i)}, ..., x_k^{(i)}) - y_i)^2$.

Cuadrados mínimos lineales en varias variables. Si restringimos f a funciones lineales, podemos ver que el problema se trata de encontrar un x que minimiza $\|Ax - b\|^2$ donde A es la matriz de $n \times k$ donde $A_{i,j} = x_j^{(i)}$ y $b_i = y_i$.
Propiedad: El problema de cuadrados mínimos lineales siempre tiene solución. La solución es única $\Leftrightarrow$ no existe un vector v no nulo tal que $Av = 0$.
Propiedad: Si x es la solución de cuadrados mínimos lineales, entonces $A^t Ax = A^t b$.
Esta última propiedad da una forma de resolver el problema, resolviendo el sistema $A^t Ax = A^t b$ utilizando algún método ya visto (LU, QR, etc).
Observación: $A^t A$ es simétrica y semi definida positiva, lo cual trae problemas numéricos al resolver el sistema.

11. Cálculo de autovalores

Propiedad: Si A y B son similares (i.e., existe S tq $S^{-1}BS = B$) tienen los mismos autovalores.
Teorema: Sea $R_i = \{z| \ |z - A_{i,i}| \leq \sum_{j \neq i} A_{i,j}\}$. Si $\left(\bigcup_{i \in K} R_i\right) \cap \left(\bigcup_{i \notin K} R_i\right) = \emptyset \Rightarrow$ en $\bigcup_{i \in K} R_i$ hay $|K|$ autovalores de A.

Método de la potencia. Si tengo una base de autovectores $< v_1, ..., v_n >$ para todo x existen $\beta_1, ..., \beta_n$ tal que $x = \sum_i \beta_i v_i$. Fijo un x.

$$\begin{aligned} x &= \sum \beta_i v_i \\ Ax &= \sum \beta_i A v_i = \sum \lambda_i \beta_i v_i \\ A^k x &= \sum \lambda_i^k \beta_i v_i = \lambda_1^k \sum (\lambda_i/\lambda_1)^k \beta_i v_i = \lambda_1^k (\beta_1 v_i + \varepsilon^{(k)}), \end{aligned}$$

donde ε tiende a 0. (λ_1 es el autovalor mas grande en valor absoluto). Sea $x^{(k)} = A^k x$. Sea f continua tal que $f(\alpha x) = \alpha f(x)$ y $f(\beta_1 v_1) \neq 0$. Entonces

$$\lim_{k \to \infty} f(x^{(k+1)})/f(x^{(k)}) = \lambda_1 f(\beta_1 v_1 + \varepsilon^{(k+1)})/f(\beta_1 v_1 + \varepsilon^{(k+1)}) = \lambda_1.$$

Método de la potencia inversa. (Sirve si λ_n es el autovalor de menor valor absoluto y distinto de cero de A no singular). Si λ es autovalor de A, λ^{-1} lo es de A^{-1}. $Ax = \lambda x \Rightarrow A^{-1}Ax = \lambda A^{-1}x \Rightarrow x = \lambda A^{-1}x \Rightarrow \lambda^{-1}x = A^{-1}x$. Utilizando $x^{(k+1)} = A^{-1}x^k$ se puede demostrar que $f(x^{(k+1)})/f(x^{(k)})$ tiende a $1/\lambda_n$.

Encontrar todos los autovalores. Una vez que encontre λ_1 me busco Q ortonormal tal que QAQ^{-1}, que tiene los mismos autovalores que A, tenga todos ceros en la primer columna por debajo de la diagonal y λ_1 en la esquina superior izquierda (ver factorización QR). De este modo, la submatriz A' que resulta de tachar la primer fila y columna tiene todos los autovalores excepto λ_1, así puedo obtener el segundo más grande y repetir.