Math 385 Midterm Review

Mario L. Gutierrez Abed

Types of error

- Standard Error : Actual Value − Calculated Value
- Absolute Error: |Actual Value − Calculated Value|
- Relative Error: $\frac{\text{Actual Value}-\text{Calculated Value}}{\text{Actual Value}}$

Newton's Method

The tangent line at x_0 is given by $f'(x_0) = \frac{y - f(x_0)}{x - x_0}$. Now setting $y = 0$ and then solving for x we have

$$x = \frac{f'(x_0)\,x_0 - f(x_0)}{f'(x_0)}$$ **operative statement**

Secant Method

We pick two initial seeds (call them x_1 and x_2), one on each side of the root that we're looking for. Then we draw a secant line that passes through $f(x_1)$ and $f(x_2)$ (where $f(x_1) > 0$ and $f(x_2) < 0$), this line will intercept the x axis at a new seed x_{new}.

If $f(x_{new}) = 0$, we have found our root, hence we're done. Otherwise, if $f(x_{new})$ and $f(x_1)$ have the same sign (i.e. $f(x_{new}) > 0$), swap x_1 and x_{new}. Else swap x_2 and x_{new}.

Now the secant line has to satisfy the following :

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{y_{new} - f(x_1)}{x_{new} - x_1}$$

By making $y_{new} = 0$ we have

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = -\frac{f(x_1)}{x_{new} - x_1}.$$

Now we solve for x_{new} :

$$x_{new} = x_1 - \frac{f(x_1)\,(x_2 - x_1)}{f(x_2) - f(x_1)}$$ **operative statement**

Whereas Newton's method fails sometimes, the secant method never fails, even though it's computationally slower.

If we use FindRoot with two seeds instead of just one, *Mathematica* automatically uses the secant method.
For instance we use
FindRoot[$f[x]$, {x, 2.0, 4.0}] to find a root between the two roots $x_1 = 2$ and $x_2 = 4$.

Errors associated with different methods:

x = real root
$\hat{x}$ = Root found via Newton's method
$\tilde{x}$ = Root found via Secant method
Then we have the following inequalities:
- $|x - \hat{x}| \leq |\tilde{x} - \hat{x}|$
- $|x - \tilde{x}| \leq |\tilde{x} - \hat{x}|$

From these inequalities we can see that $|\tilde{x} - \hat{x}|$ is an upper bound for the error.

Linear Algebra Review

Row equivalence, elementary row operations, systems of equations

- Matrices A and B are row equivalent provided that there exist matrices E_1, E_2, ... E_n such that $B = \left(\prod_{i=1}^{n} E_i\right) A$.

- Multiplication on the left by an elementary matrix implements the corresponding row operations.

- There are three row operations :
i) The type-1 elementary row operation exchanges two rows of a matrix. It is denoted by $E_{(i,j)}$, indicating that rows i and j are exchanged.
ii) The type-2 elementary row operation multiplies a row by a nonzero scalar. It is denoted by $E_{\alpha(i)}$, indicating that row i is multiplied by α.
iii) The type-3 elementary row operation adds a scalar times one row to another row. It is denoted by $E_{\alpha(i)+j}$, indicating that α times row i is added to row j.

- All elementary matrices are non-singular.
- If $A\vec{x} = \vec{b}$ and $B\vec{x} = \vec{c}$ are linear systems and D is the product of elementary matrices, then $B = DA$, $\vec{c} = D\vec{b}$ implies that the two systems have the same solutions.

Error Analysis for systems of linear equations

We say that $\vec{v} = \sum_{i=1}^{n} \xi_i \vec{e_i}$ is a standard basis vector in $\mathbb{R}^n$ and its norm is $\|\vec{v}\| = \left(\sum_i \xi_i^2\right)^{1/2}$.

If we impose the condition that $\|\vec{v}\| \leq 1$, then this implies that $|\xi_i| \leq 1 \ \forall \ i$.

Now for this vector $\vec{v}$ and any linear map A, we use the triangle inequality to get the norm of A :

$$\begin{aligned} \|A\vec{v}\| &= \|A \sum_{i=1}^{n} \xi_i \vec{e}_i\| = \|\sum_{i=1}^{n} \xi_i (A\vec{e}_i)\| \\ &\leq \sum_{i=1}^{n} |\xi_i| \ \|A\vec{e}_i\| \\ &\leq \sum_{i=1}^{n} \|A\vec{e}_i\| = M \ . \end{aligned}$$

This number M is called the norm of A (i.e. $M = \|A\|$).
Hence we have shown above that $\|A\vec{v}\| \leq \|A\|$, provided that $\|\vec{v}\| \leq 1$.

More generally, for any vector $\vec{v}$, it is always true that $\frac{1}{\|\vec{v}\|} \vec{v}$ has norm 1. Thus this implies that

$$\|A\vec{v}\| = \|\vec{v}\| \ \|A \frac{1}{\|\vec{v}\|}\| \leq \|A\| \ \|\vec{v}\|.$$

Thus we have proven that $\|A\vec{v}\| \leq \|A\| \ \|\vec{v}\|$ for any $\vec{v}$. ■

RESIDUALS

We have $f(x) = a$. Then given that $\hat{x}$ is an approximate solution, we have that
$f(x) - f(\hat{x})$ is called a residual.

In terms of linear mappings, suppose $A\vec{x} = \vec{b}$ is a linear system with nonsingular coefficient matrix A and suppose that $\hat{x}$ is a solution to such system.
Then set $\hat{b} = A\hat{x}$. Now we have that $\hat{r} = \vec{b} - \hat{b}$ is called the residual.
Notice that the residual is a vector. This is also the case for the standard error $\vec{e} = \vec{x} - \hat{x}$.
We now see that the norm error is bounded by the norm of $A^{-1}$and the norm of the residual, i.e. $\|\vec{e}\| \leq \|A^{-1}\| \ \|\vec{r}\|$:

$$\begin{aligned} \|\vec{e}\| = \|\vec{x} - \hat{x}\| &= \|A^{-1}\vec{b} - A^{-1}\hat{b}\| \\ &= \|A^{-1}(\vec{b} - \hat{b})\| \\ &\leq \|A^{-1}\| \ \|\vec{b} - \hat{b}\| = \|A^{-1}\| \ \|\vec{r}\| \quad \checkmark \end{aligned}$$

Also, as proven before, we have that $\|\vec{b}\| = \|A\vec{x}\| \leq \|A\| \ \|\vec{x}\|$.

Hence combining these inequalities we get a bound for the relative norm error $\frac{\|\vec{e}\|}{\|\vec{x}\|}$:

$$\frac{\|\vec{e}\|}{\|\vec{x}\|} = \frac{\|\vec{x}-\hat{x}\|}{\|\vec{x}\|} = \frac{\|A^{-1}\vec{b}-A^{-1}\hat{b}\|}{\|\vec{x}\|} = \frac{\|A^{-1}(\vec{b}-\hat{b})\|}{\|\vec{x}\|}$$
$$\leq \frac{\|A^{-1}\|\,\|\vec{b}-\hat{b}\|}{\|\vec{x}\|} \cdot \frac{\|\vec{b}\|}{\|\vec{b}\|}$$
$$= \frac{\|A^{-1}\|\,\|\vec{b}-\hat{b}\|}{\|\vec{x}\|} \cdot \frac{\|A\vec{x}\|}{\|\vec{b}\|}$$
$$\leq \frac{\|A^{-1}\|\,\|\vec{b}-\hat{b}\|}{\|\vec{x}\|} \cdot \frac{\|A\|\,\|\vec{x}\|}{\|\vec{b}\|}$$
$$= \|A\|\,\|A^{-1}\|\,\frac{\|\vec{b}-\hat{b}\|}{\|\vec{b}\|} = \|A\|\,\|A^{-1}\|\,\frac{\|\hat{r}\|}{\|\vec{b}\|}$$

Where $\frac{\|\hat{r}\|}{\|\vec{b}\|}$ is the relative norm residual and $\|A\|\,\|A^{-1}\|$ is the condition number (which is usually 1).

Definition: A linear transformation is said to be ill-conditioned if the condition number is much greater than 1.

Note: If α is an eigenvalue associated with A, we must have $|\alpha| \leq \|A\|$. Recall that if $\vec{v}$ is an eigenvector associated with α, then $A\vec{v} = \alpha\vec{v}$.

** For more proofs involving error analysis for systems of linear equations see HW#5 **

Functions of several variables: Finding roots and extrema

THE GRADIENT METHOD

Suppose we seek a minimum of f mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. Then

1) Start with a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ and an initial estimate $\vec{x}_0$ in $\mathbb{R}^n$.

2) a) If $m > 1$, replace f by $f \cdot f$ so that f takes values in $\mathbb{R}$.
   b) If $m = 1$, replace f by f^2 to ensure that f takes values in $\mathbb{R}$.

3) Calculate the gradient of f at the initial estimate, i.e. compute

$\vec{a} = \nabla f(\vec{x}_0) = \langle \alpha, \beta \rangle = \left\langle \frac{\partial f}{\partial x}\Big|_{x_0,y_0}, \frac{\partial f}{\partial y}\Big|_{x_0,y_0} \right\rangle$.

4) Set $l(t) = \vec{x}_0 + t\,\vec{a} = (x_0, y_0) + t\langle \alpha, \beta \rangle$.

5) Set $h = f(l(t))$.

6) Compute $h'(t)$ and solve $h'(t) = 0$ via FindRoot[] or Solve[] to yield t_0.

7) Set $\vec{x}_{\text{new}} = l(t_0)$.

8) a) If $f(\vec{x}_0) < f(\vec{x}_{\text{new}})$, then exit. (the process has failed)
b) If the iteration count exceeds the maximum, exit. (the process has failed)
c) If $|f(\vec{x}_{\text{new}}) - f(\vec{x}_0)|$ is sufficiently small, exit. (possible success)

9) Go back to Step 3 using $\vec{x}_{\text{new}}$.

** To see an example of this procedure, check HW#4 **

THE HESSIAN METHOD

Before proceeding we are going to state the following theorem, which will be used in the procedure below:

• Rolle's Theorem:
If $f(x_o) = f(x_1)$ then there exists a critical point $(f' = 0)$ on the interval $[x_o, x_1]$.

Also, recall that the Hessian of a function is given by the matrix of partial derivatives:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial f}{\partial x\,\partial y} \\ \frac{\partial f}{\partial y\,\partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}.$$

Taylor series to approximate the line using the Hessian:
$f(\vec{x}) \simeq f(\vec{x}_o) + \nabla f(\vec{x}_o)(\vec{x} - \vec{x}_o) + \frac{1}{2} H|_{\vec{x}_o} (\vec{x} - \vec{x}_o)^2 + \text{Remainder}$

For our purpose, we ignore the remainder (since it's a very small term).
Now if $f(\vec{x}) = f(\vec{x}_o)$, we expect a local extremum somewhere between x_o and some x_i in the interval we're analyzing (this follows from Rolle's theorem).
Then we have

$$-\nabla f(\vec{x}_o)\,(\vec{x} - \vec{x}_o) = \frac{1}{2} H|_{\vec{x}_o}\,(\vec{x} - \vec{x}_o)^2$$

Now ignoring the $\frac{1}{2}$ (because it's a very small term) we have

$$-\nabla f(\vec{x}_o) = H|_{\vec{x}_o}\,(\vec{x} - \vec{x}_o)$$

Now we solve this linear system and then replace $\vec{a}$ in step 3) above with the α and β defined as follows :

```
vect = LinearSolve[Hessian, −grad];
α = vect[[1]];
β = vect[[2]];
```

After plugging in these values for α and β into $\vec{a}$ in step 3), just proceed through all the remaining steps until you find the extremum.

** To see an example of this procedure, check HW#4 **

Interpolation and Fitting

Suppose we are given a set of points P_1, ..., P_n in the plane $\mathbb{R}^2$. Then we want to find a curve (function) that passes through the points (interpolating) or a curve that passes near the points (fitting). If we want the curve to pass through the points, then we may have to accept anomalies on the curve. If we are willing to accept a curve that only passes near the points, then we may place stronger restrictions on the curve. We will soon see how this 'give and take' materializes.
We will use several different interpolation and fitting techniques, such as polynomial interpolation, least squares fitting, Bezier interpolation, and the more widely used and more mathematically complex method: the splines.

POLYNOMIAL INTERPOLATION

Case 1 (Taylor Expansion): We are given data points and a function as well.

Say that we are given the function $f(x) = x^2\, e^{-x} - 0.1$. We can see that if we graph f, the curve it's pretty much cubic. Now say that we are given data points (4, 5, 6, 7). Then, in order to use a polynomial interpolation, we want a degree-3 (since f is pretty much cubic) Taylor series about $x_o = 5$ (since 5 is close to being halfway in this interval [4, 7]) .
The general formula of a degree-$\mathcal{N}$ Taylor series is given by

$$f(x) = \sum_{n=0}^{\mathcal{N}} \frac{f^{(n)}(x_0)\,(x - x_0)^n}{n!}$$

Hence, in this case we have

$$f(x) \simeq g(x) = f(5) + f'(5)\,(x-5) + \frac{1}{2} f''(5)\,(x-5)^2 + \frac{1}{6} f'''(5)\,(x-5)^3$$

Errors:

- A numerical value for the "goodness of fit" is given by ℓ_2 norm $\left(\sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}\right)$
- $\|f - g\| = \sqrt{\int_a^b (f-g)^2}$. This is the norm error.
- $\frac{1}{b-a}\sqrt{\int_a^b (f-g)^2}$. This is the mean norm error.

Case 2 (Vandermonde's Process): We are given data points but no function.

Let's say we have $n+1$ points (x_o, y_o), (x_1, y_1), ..., (x_n, y_n). Then we look at the standard basis for an n^{th} degree polynomial $\{1, x, x^2, \ldots, x^n\}$ and we set up a linear system

$$\begin{pmatrix} 1 & x_0 & x_0^2 & . & . & x_0^n \\ 1 & x_1 & x_1^2 & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & x_n & x_n^2 & . & . & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ . \\ . \\ . \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ . \\ . \\ . \\ y_n \end{pmatrix}$$

where the $n \times n$ matrix is called the Vandermonde matrix and the entries of the $n \times 1$ matrix it mutiplies are all scalar coefficients a_i.

Then we need to solve for these coefficients a_i
(In *Mathematica*, we use something like coeffvec = LinearSolve[Vand,yVec])
After solving coeffvec, we build the polynomial

$$p(x) = a_o + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

• Theorem:
The Vandermonde matrix is non-singular provided that no row is a multiple of the other, i.e. the entries x_i are distinct.

Proof:
Indeed the Vandermonde matrix is singular only if the columns are dependent, i.e. only if there exists at least a nonzero β_i scalar such that $\beta_0 \begin{pmatrix} 1 \\ . \\ . \\ 1 \end{pmatrix} + \beta_1 \begin{pmatrix} x_1 \\ . \\ . \\ x_n \end{pmatrix} + \ldots + \beta_n \begin{pmatrix} x_1^n \\ . \\ . \\ x_n^n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

Hence for each $i = 1, 2, \ldots, n+1$, we have $\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_n x_i^n = 0$

But this polynomial is degree n with at most n distinct roots. Thus we have $(x_i) + 1$ root $= n + 1$ roots. This means that we have $n + 1$ distinct roots $x_1, \ldots, x_{n+1}$. $(\Rightarrow\Leftarrow)$
Thus the Vandermonde matrix is nonsingular given distinct x_i. ■

Case 3 (Lagrange Polynomials): We are given data points but no function (just like in Vandermonde's case)

This is an alternative to Vandermonde's method. Let's say that, just like in the Vandermonde case, we have $n + 1$ points $(x_o, y_o), (x_1, y_1), \ldots, (x_n, y_n)$.
Then we can also do polynomial interpolation with these points using Lagrange bases instead of the standard basis $\{1, x, x^2, \ldots x^n\}$.

Our goal is to get a linear Lagrange polynomial

$$q(x) = \sum_{i=0}^{n} y_i \ell_i(x)$$

where the polynomials $\ell_i(x)$ are called Lagrange basis polynomials, which can be computed as follows :

$$\ell_i(x) = \prod_{\substack{0 \le j,i \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j}$$

Notice that $\ell_i(x_i) = 1$ and $\ell_i(x_j) = 0$.

• Theorem:
The polynomial we get by using Vandermonde's method and the one we build using Lagrange's method are the same, i.e. $p(x) = q(x)$.

Proof:
Set $r = p(x) - q(x)$. Then r is of degree n, since both $p(x)$ and $q(x)$ are of degree n. Since $p(x_i) = y_i = q(x_i)$ for each $i = 0, 2, \ldots, n$, then r has $n + 1$ roots $x_0, \ldots, x_n$. But since r is degree n, the fundamental theorem of algebra tells us that it can have at most n roots. Therefore, the only way that r can have $n + 1$ roots is by setting $r = 0$.
But then we have

$$r = 0 \Longrightarrow p(x) - q(x) = 0 \Longrightarrow p(x) = q(x) .$$ ■

If the point P_i lie on the graph of a function f, then it is natural to ask how well does p approximate f. If f has at least $n + 1$ continuous derivatives then we can estimate the error, $e(x) = f(x) - p(x)$.

• Theorem:

The error $e(x) = f(x) - p(x)$ is given by $e(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_i (x - x_i)$,

given a function f with $n+1$ derivatives on an interval $[a, b]$, where ξ is distinct from x_i.

Proof:

Define $g(x) = \frac{e(x)}{\prod_i (x - x_i)}$ so that $e(x) = g(x) \prod_i (x - x_i)$.

Next take $\zeta \in [a, b]$ distinct from x_i, and then set

$$h(x) = f(x) - p(x) - g(\zeta) \prod_i (x - x_i).$$

It is clear to see that each x_i is a root of h.

Also,

$$h(\xi) = e(\xi) - g(\zeta) \prod_i (x - x_i) = 0.$$

Thus h has $n+2$ roots on $[a, b]$.

Now by Rolle's Theorem:

↪ h has $n+2$ roots

↪ h' has $n+1$ roots

↪ $h^{(2)}$ has n roots...

↪ $h^{(n+1)}$ has 1 root ⟶ call this root $\xi = \xi_h$

Now we have

$$0 = h^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - g(\zeta) \frac{\partial^{(n+1)}}{\partial x^{(n+1)}} \prod_i (x - x_i) \Big|_{x=\xi}$$

But p is degree n, so $p^{(n+1)} = 0$, and also

$$\frac{\partial^{(n+1)}}{\partial x^{(n+1)}} \prod_i (x - x_i) = (n+1)! \quad \text{(regardless of what } \xi \text{ is)}.$$

Therefore

$$g(\zeta)\,(n+1)! = f^{(n+1)}(\xi) \qquad \text{or} \qquad e(\zeta) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_i (\zeta - x_i),$$

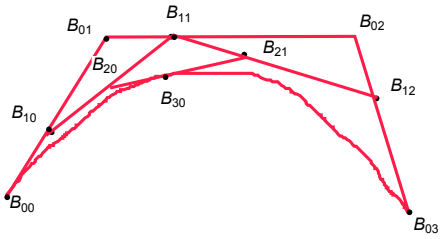
where ξ is derived via repeated applications of Rolle's Theorem and depends on the choice of ζ.

Finally, since h is defined for any x in the interval, then this last expression for the error is satisfied for all x.

Now for the final statement on the bound for the error magnitude we note that since f is $n+1$ times continuously differentiable, then $f^{(n+1)}$ is continuous and hence has maximum value on the interval.

■

BEZIER INTERPOLATION



Just as in the case of polynomial interpolation, we require a function and four points to do a cubic interpolation. However in this case the necessary information includes only two points on the curve (the starting point and ending point) and the slope of the curve at these two points. More complicated curves can be constructed by piecing successive Bezier curves together.

In the interval $t \in [0, 1]$, we have (following the above diagram)
$B(t) = B_{30}(t) = (1-t)^3 B_{00} + 3\,t(1-t)^2 B_{01} + 3\,t^2(1-t) B_{02} + t^3 B_{03}$
Our goal is then to determine the coefficients $B_{00}, B_{01}, B_{02}, B_{03}$:
$B(0) = B_{00}$ $\quad$ $B'(0) = 3\,(B_{01} - B_{00})$
$B(1) = B_{03}$ $\quad$ $B'(1) = 3\,(B_{03} - B_{02})$

$f(x) = \{x, f(x)\}$
$f'(x) = \{1, f'(x)\}$

Example:
For $x^2 e^{-x} - 0.1$ on $[1, 4]$,
$B_{00} = (1, f(1))$
$B_{03} = (4, f(4))$

$B(0) = f'(1) = 3\,(B_{01} - (1, f(1)))$
$B(1) = f'(4) = 3\,((4, f(4)) - B_{02}))$

ParametricPlot[$B(t)$, $\{t, 0, 1\}$]

** To see a complete example using Bezier interpolation check HW#7**

Definition: Point where Bezier curves meet is called a knot point.

LEAST SQUARES FITTING

We're trying to fit a line through $P_1, \ ..., \ P_n$, $P_i = (x_i, \ y_i)$.
The line is in the form $y = a\,x + b$, where a and b are the best choice to minimize the distance between the line and the P_i.
This distance is given by $\| (x_i, \ y_i) - (x_i, \ a\,x_i + b) \|$.
Now we want to get rid of the absolute value. We do so by squaring each of these terms and adding them to get a function σ with independent variables a and b. So we have
$\| (x_i, \ y_i) - (x_i, \ a\,x_i + b) \|^2 = \sigma(a, \ b) = \ \sum_i [y_i - (a\,x_i + b)]^2$

Now we apply standard max/min techniques to σ by differentiating with respect to each of the two independent variables :

$$\frac{d\sigma}{da} = \sum_i 2\,[y_i - (a\,x_i + b)]\,(-x_i) = 2\,\sum_i \left[a\,x_i^2 + (b - y_i)\,x_i\right]$$

$$\frac{d\sigma}{db} = \sum_i 2\,[y_i - (a\,x_i + b)]\,(-1) = 2\,\sum_i \left[a\,x_i + b - y_i\right]$$

Setting these two terms to zero and reorganizing them a little we get

$$0 = a\,\sum_i x_i^2 - \sum_i x_i\,y_i + b\,\sum_i x_i$$
$$0 = a\,\sum_i x_i - \sum_i y_i + n\,b$$

From the above we have:

$$\sum_i x_i\,y_i = a\,\sum_i x_i^2 + b\,\sum_i x_i$$
$$\sum_i y_i = a\,\sum_i x_i + n\,b$$

In matrix form

$$\begin{pmatrix} \sum_i x_i\,y_i \\ \sum_i y_i \end{pmatrix} = \begin{pmatrix} \sum_i x_i^2 & \sum_i x_i \\ \sum_i x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix}$$

Then in order to solve this it's easy to do the following :

We set $A = \begin{pmatrix} x_1 & . & . & x_n \\ 1 & . & . & 1 \end{pmatrix}$. Then it is immediately obvious that

$$A\,A^T = \begin{pmatrix} x_1 & . & . & x_n \\ 1 & . & . & 1 \end{pmatrix} \begin{pmatrix} x_1 & 1 \\ . & . \\ . & . \\ x_n & 1 \end{pmatrix} = \begin{pmatrix} \sum_i x^2{}_i & \sum_i x_i \\ \sum_i x_i & n \end{pmatrix}$$

Then $A.\begin{pmatrix} y_1 \\ . \\ . \\ y_n \end{pmatrix} = A\,A^T \begin{pmatrix} a \\ b \end{pmatrix}$

$$\begin{pmatrix} a \\ b \end{pmatrix} = \text{LinearSolve}\left[A^T\,A,\ A^T \begin{pmatrix} y_1 \\ . \\ . \\ y_n \end{pmatrix}\right]$$

** To see a complete example using Least Squares fitting check HW#7**

CUBIC / B SPLINES

Definition: We work on a closed interval $[a,\ b] \in \mathbb{R}$ such that $a = t_0 < t_1 < t_2 < \ldots < t_n = b$.
We call σ a cubic spline on $[a,\ b]$ if :

1) for each i, σ restricted to $[t_{i-1},\ t_i]$ is a parametric cubic, denoted by σ_i.
2) $\sigma_i(t_i) = \sigma_{i+1}(t_i)$
3) σ is C^2 on $[a,\ b]$.

The points $\sigma(t_i)$ are called the knot points. The splines developed in this section will be twice continuously differentiable (C^2) at the knot points. Before continuing, we remark that Bezier curves are splines with a single segment.

We begin with the mathematical analysis of the draftsman's spline. Specifically this spline is a thin elastic strip (traditionally made of wood), which is attached to a drafting table by pins and bent by weights (called ducks). Considering the spline as a beam and the ducks as concentrated loads on the beam, then the deformed beam satisfies the Bernoulli-Euler equation:

Bernoulli-EulerEquation:

$$(EI)\,k(x) = M(x),$$

where $k(x)$ is the curvature of the beam, $M(x)$ is the bending moment, and EI is a constant called the coefficient of rigidity. Furthermore, $M(x)$ is a linear function of x. Letting $\sigma(x)$ denote the arc represented by the bent beam, then the curvature in terms of σ is given by

$$k(x) = \frac{\sigma''}{\left(1+(\sigma')^2\right)^{3/2}}$$

▸ Assumption 1:

We assume that the beam deflections are small, i.e. $||\sigma'|| << 1$. Equivalently stated, σ' is negligible. Hence, according to the formula for the curvature defined above, we may suppose that $k(x) = \sigma''$. Since $M'' = 0$, then the fourth derivative $\sigma^{(4)} = 0$. This leads to the following...

▸ Assumption 2:

σ is C^2 at each duck (knot point) and cubic between the ducks.

We are now in a position to solve for σ:

Suppose that the pins have x-coordinates at a and b and at $a = x_0 < x_1 < ... < x_n = b$.
Then
(i) σ_i is parametric cubic on $[x_{i-1},\ x_i]$
(ii) σ is C^2 at each x_i
(iii) $\sigma_i(x_i) = y = \sigma_{i+1}(x_i)$
In turn we set
(iv) $\sigma'(x_i) = m_i$ and $\sigma''(x_i) = M_i$.

The second derivative of σ_i, denoted σ_i'', is linear. The first-order Lagrange polynomial for σ_i'' is given by

$$\sigma_i''(x) = M_{i-1}\ \frac{(x_i - x)}{\Delta i} + M_i\ \frac{(x - x_{i-1})}{\Delta i}\ ,$$

where $\Delta i = x_i - x_{i-1}$.

Now we integrate twice to get $\sigma_i(x)$:

$$\sigma_i'(x) = M_{i-1}\ \frac{(x_i - x)^2}{2\,\Delta i} + M_i\ \frac{(x - x_{i-1})^2}{2\,\Delta i} + C$$

$$\sigma_i(x) = M_{i-1}\ \frac{(x_i - x)^3}{6\,\Delta i} + M_i\ \frac{(x - x_{i-1})^3}{6\,\Delta i} + C\,x + D\ .$$

After tons of algebraic manipulations we get something like

$$\sigma_i(t) = \tfrac{1}{6}\,(-t^3 + 3\,t^2 - 3\,t + 1)\,P_i + \tfrac{1}{6}\,(3\,t^3 - 6\,t^2 + 4)\,P_{i+1} +$$
$$+ \tfrac{1}{6}\,(-3\,t^3 + 3\,t^2 + 3\,t + 1)\,P_{i+2} + \tfrac{1}{6}\,t^3\,P_{i+3}\ ,$$

where P_i are guide points.
$P = \{P_1,\ ...,\ P_n\}$, such that $P_i = (x_i,\ y_i)$
$q = \{\text{coefficient polys}\}$, $q_1 \to P_i$, $q_2 \to P_{i+1}$,etc....

In general if q_i are cubic bases, we have

$$\sigma_k(t) = \sum_{i=1}^{4}\ q_i\,P_{i+k} \quad \text{for } k = \{1,\ 2,\ 3,\ \ ...\ n-3\}$$

Example: for $n = 24$ we have
$q_1 P_{21} + q_2 P_{22} + q_3 P_{23} + q_4 P_{24}$

Properties of Splines:

- σ maintains smoothness at knot points (C^2 everywhere (including knot points)).
- Exhibit local control.