SFU

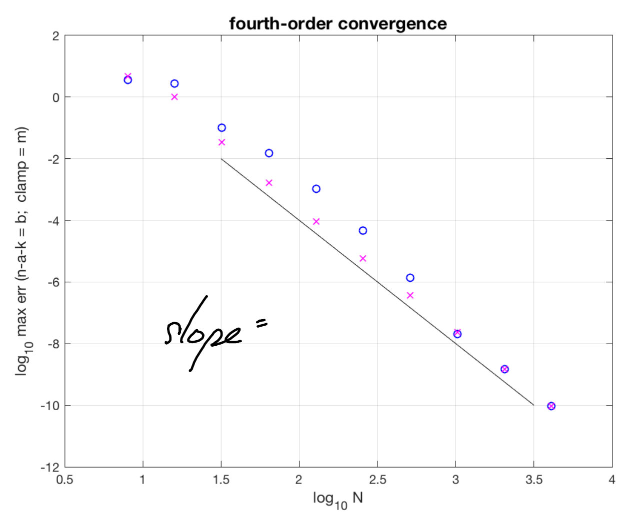
muraki

MONDAY

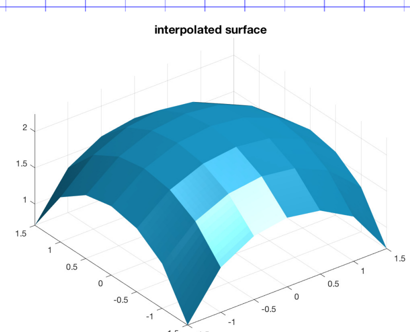
- CA4 due
- computing workshop @ 2^{30} (wait list)

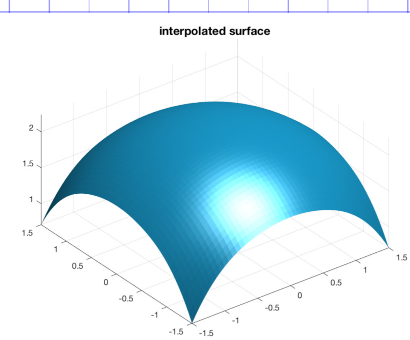
LAST DAY

- Matlab spline: interpolate & pre-compute
  clamped & not-a-knot
- ACC, EFF, ROB
  - uniform-spacing
    err = O(   )



- 2D "killer app"





muraki

POLYNOMIAL INTERPOLATION

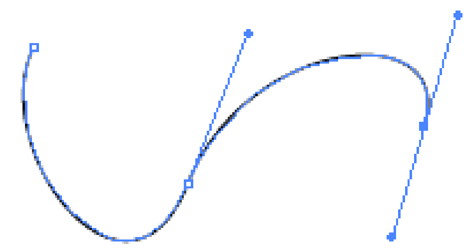
	MLI	cubic spline
# polys	of deg	, each
eval	$L_{MLI}(x)$	must find $\le x \le$
continuous derivs		, , only
large N	for most $\{x_k\}$	ROBUST
convergence (acc)	OK, but guaranteed	, not-a-knot
efficiency	O( ) pre-comp, O( ) eval	O( )

WHY DO WE STILL NEED CHOICE ?

- algorithms are seldom used in
  - algorithms in
  - for instance, derivative discontinuities

in a

code be problematic ...



OTHER CHOICES

- Newton
- Hermite (what if you know $f(x_k)$ & $f'(x_k)$?)

  → BEZIER CURVES

WHAT NEXT?

- too many points, OR data is imperfect?

THE REAL LEAST-SQUARES... (ch 8 + beyond...)

- the familiar LINEAR Least Squares (L-Sq)

  → the - through points.

- given N points. $\{(x_k, y_k)\}_{k=1\to N}$

  find best-fit line $y(x) =$

$$(a,b) = \sum_{k=1}^{N} (y_k - y(x_k))$$

error

-

SFU
muraki

$$\mathcal{E}(a,b) = \sum_{k=1}^{N} \left(y_k - (\quad + \quad) \right)^2$$

- L-Sq means minimizing $\mathcal{E}(a,b)$ over a & b.

1) $\frac{\partial \mathcal{E}}{\partial a} = \sum_{1}^{N} (y_k - a x_k - b)(\quad) =$

2) $\frac{\partial \mathcal{E}}{\partial b} = \sum_{1}^{N} (y_k - a x_k - b)(\quad) =$

1') $\left(\sum_{1}^{N} \quad\right) a + \left(\sum_{1}^{N} \quad\right) b = \left(\sum_{1}^{N} \quad\right)$

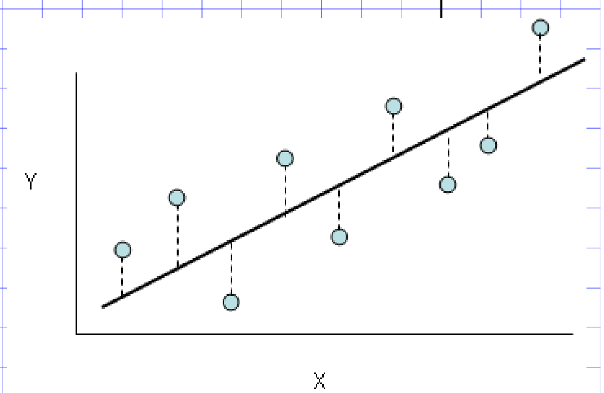
2') $\left(\sum_{1}^{N} \quad\right) a + \left(\quad \right) b = \left(\sum_{1}^{N} \quad\right)$

$\rightarrow$ x linear solve for & !

- THIS IS THE STANDARD PERSPECTIVE on

-

perspective



SFU
muraki

- A LINEAR SOLVE PERSPECTIVE
- what would a " - " line be ?

$\rightarrow \quad y_k = \quad + \quad$ for all ( )

- matrix equation for " fit"

$$[\;]\vec{\;} = \begin{bmatrix} x_1 & 1 \\ \\ x_N & 1 \end{bmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ \\ y_N \end{pmatrix} =$$

↳ but we expect solution $\vec{\;}$ when $N >$

- residual error: $\vec{e}(\vec{v}) = [\;]\vec{\;} - \vec{\;}$
- length of $\vec{e}$: $|\vec{e}|^2 = [\quad][\;]$

$$(\vec{\;}) = |\vec{e}(\vec{v})|^2 = \quad {}^T\left([A]\vec{v} - \vec{y}\right)$$

$$= \vec{\;}^T[A]^T[A]\vec{\;} - \vec{\;}^T[A]^T\vec{\;} - \vec{\;}^T[A]\vec{\;} + |\vec{\;}|^2$$

SFU
murakí

- find minimum of $\mathcal{E}(\vec{v})$, $\vec{v} = \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \quad \end{pmatrix}$

$\frac{\partial \vec{v}}{\partial v_1} = \hat{\ }$ & $\frac{\partial \vec{v}}{\partial v_2} = \hat{\ }$ basis vectors.

- derivatives

$$\frac{\partial \mathcal{E}}{\partial v_k} = {}^{T}[A]^{T}[A]\vec{v} - {}^{T}[A]^{T}\vec{y}$$
$$+ \vec{\ }^{T}[\quad]^{T}[\quad] - \vec{\ }^{T}[\quad] =$$

last 2 terms are

$$= \Big\{ \big([A^TA]\vec{v} - [A^T]\vec{y} \big) \Big\} = 0$$

!

th-component = , for every !

- the minimum of $(\vec{\ })$ occurs when

$$[\quad]\vec{\ } = [\quad]\vec{\ } \rightarrow \text{the}$$

- for N pts, $[A]$ is x matrix
  → $[A^T][A]$ is x !!

- for any $[A]$, $[A^TA]$ is a matrix

SFU
muraki

- WHY IS THIS THE REAL L-Sq ?
- For the linear system $[A]\vec{v} = \vec{y}$

$\vec{v} \in \mathbb{R}^N$ (unknowns)

but $[A]_{M\times N}$ is -square

with $\geq$ $(A) \geq N$

→ there is a $\vec{v}$

that gives residual $|\vec{e}|$ ≥

its value.

→ this $\vec{v}_{LS}$ satisfies the (square) linear system

$$[A^T A]\,\vec{v}_{LS} = [A^T]\,\vec{y}.$$

EQUATIONS

- Q: what does Matlab do when

$[A]$ $\vec{y}$

but $[A]_{\times}$ and > ?

a) error message

b) warning message

c) output value, no message

MATLAB EXPERIMENT

SFU

muraki