DATA SCIENCE BEFORE THE MACHINE LEARNING BANDWAGON CAME TO TOWN

Max Gunzburger

Department of Scientific Computing

Florida State University

mgunzburger@fsu.edu

April 16, 2022

All models are wrong, but some models are useful

George Box

Assumption has many advantages

Chiefly these are the same as those of theft over honest toil

Bertrand Russel

Computational results are believed by no one,
except for the person who wrote the code

Experimental results are believed by everyone,
except for the person who ran the experiment

Origin unknown

You can fool all the people some of the time

and some of the people all the time,

but you cannot fool all the people all the time

Abraham Lincoln (not original)

- Are the lectures comprehensive? absolutely not
  - the universe of data science is humongous, the lectures barely scratch the surface
- Are the lectures superficial? often yes
  - talking about even a single topic in true depth would take several lectures
- Will the lectures give you the latest and greatest versions of anything talked about? mostly no
  - even the plain vanilla versions of any topic can take a long time to present
- Will the subject of the lectures be of interest or of use to you? probability is small
  - because the universe of data science is humongous it is unlikely that a particular topic within the lectures will interest you

ergo, you may want to leave right now

- So what my goals for the short course?
  - they are rather modest

- All I can hope for is to give you a small inkling about the algorithmic breadth, depth, and usefulness of data science (whatever that means) before and even during the "machine learning" revolution

- By the way, everything I will talk about can be viewed as "machine learning," even if the topic was developed 200 years ago
  - back then and up to the 1950s, the "machines" were humans

- In fact, in my opinion, two of the greatest, perhaps the greatest, developments in data science are interpolation and least-squares approximation
  - I believe that today's data scientist are going to be very hard pressed to develop something new that will have the huge impact these two humble (200 to 250 year old) methods have had

- Finally, data science is not something new nor has it been around for just few hundred years
  - one has to only read about Newton's, Kepler's, and even Archimedes' contributions to what we might today refer to being "data science"

- But, back to my goal

- Lots of breadth and depth on the wall

- But to do justice to the breadth and depth of the universe of data science you would have to paper this wall

- My goal:

  what I want to accomplish is

  to give you a very small inkling of the breadth

  and depth of what has been done in data science

  before (and even concurrently with) what is being

  issued from the "machine learning" bandwagon

  - it is very much like

    facing a huge buffet having hundreds of items

    from which to choose from (even different classes

    of items such as meat, cheese, fruit, vegetables,

    cakes, ice creams, etc.)

  but

    you only get to taste a very tiny amount

    of a very few items in the buffet

  and hope that that

    is enough for you to have some idea of what is

    the breadth and depth of the items in the buffet

DATA SCIENCE ??????

Data ????

- Gathering step $\Longrightarrow$ many ways for creating data
  - laboratory experiment measurements, field observations, output of computational models, ...
  - the creation of data is often preceded by constructing an input
    - design of experiments: e.g., building mazes for experiments with rats
      $\Longrightarrow$ data is created by measuring rat behavior
    - writing a code: e.g., differential equations
      $\Longrightarrow$ data is created by outputs of the code
  - data is often corrupted or uncertain or both
  - data can come in different forms
    - tabular, functional, known probability distribution, ...

- What can one do with gathered data?
  - mine
  - compress
  - optimize
  - generate new data

- Sometime during the lectures all of these will enter into the picture
  as will the how to gather data "intelligently"

- Let's take a look at the possibilities for just tabular data
  similar discussions can be made about data in other forms

Tabular data

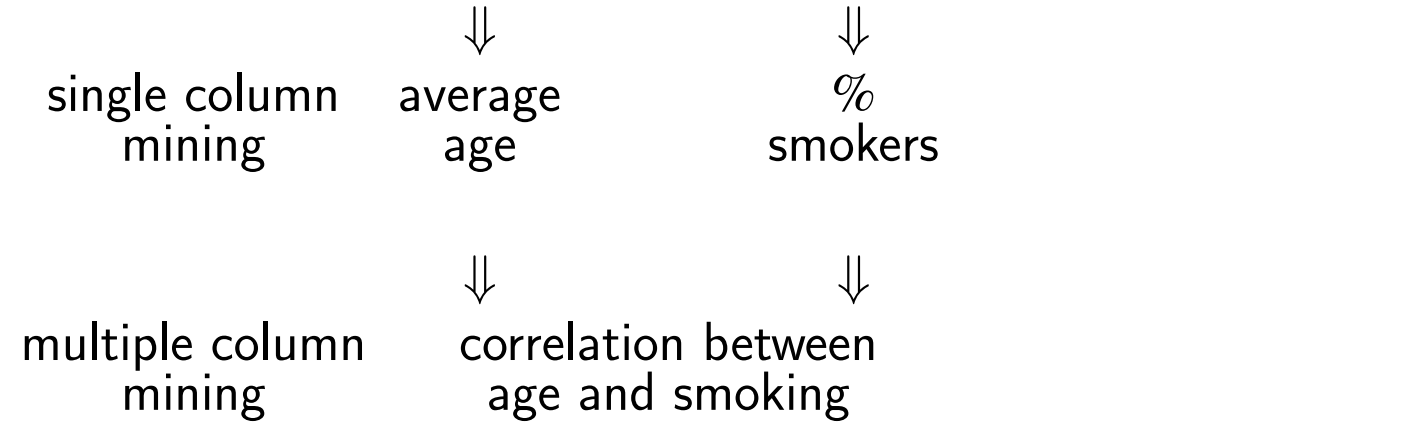
- Indexed tabular data

person	age	gender	smoker	address	bmi*	·····
1						
2						
3						
4						
.						
.						
.						
1,000,000						
	⇑	⇑	⇑	⇑	⇑	⇑
data type ⇒	integer	integer	yes/no	text	decimal	·····

* body mass index

– mining columns

person	age	gender	smoker	address	bmi	·····
1 2 3 4 . . . 1,000,000						



- Discrete spatial/temporal data

location	time	temperature	rainfall	humidity
$\boldsymbol{x}_1$	t_1			
$\boldsymbol{x}_1$	t_2			
$\vdots$	$\vdots$			
$\boldsymbol{x}_1$	$t_{M_{gather}}$			
$\boldsymbol{x}_2$	t_1			
$\boldsymbol{x}_2$	t_2			
$\vdots$	$\vdots$			
$\boldsymbol{x}_2$	$t_{M_{gather}}$			
$\vdots$	$\vdots$			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	t_1			
$\boldsymbol{x}_{M_{space}}$	t_2			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	$t_{M_{gather}}$			

– determining data at a time different from any time used to gathered data

GATHERED

location	time	temperature	rainfall	humidity
$\boldsymbol{x}_1$	t_1			
$\boldsymbol{x}_1$	t_2			
$\vdots$	$\vdots$			
$\boldsymbol{x}_1$	$t_{M_{gather}}$			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	t_1			
$\boldsymbol{x}_{M_{space}}$	t_2			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	$t_{M_{gather}}$			

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

TEMPORAL PREDICTION $(t_{prediction} > t_{M_{gather}})$

location	time	temperature	rainfall	humidity
$\boldsymbol{x}_1$	$t_{prediction}$			
$\boldsymbol{x}_1$	$t_{prediction}$			
$\vdots$	$\vdots$			
$\boldsymbol{x}_1$	$t_{prediction}$			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	$t_{prediction}$			
$\boldsymbol{x}_{M_{space}}$	$t_{prediction}$			
$\vdots$	$\vdots$			
$\boldsymbol{x}_{M_{space}}$	$t_{prediction}$			

– determining data at a point different from any point used to gather data

GATHERED

location	temperature	rainfall	humidity
$\boldsymbol{x}_1$			
$\boldsymbol{x}_2$			
$\vdots$			
$\boldsymbol{x}_{M_{space}}$			

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

SPATIAL PREDICTION

location	temperature	rainfall	humidity
$\boldsymbol{x}_{in}$			
$\boldsymbol{x}_{out}$			

$\{\boldsymbol{x}_{in}, \boldsymbol{x}_{out}\} \subsetneq \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$

$\boldsymbol{x}_{in}$ surrounded by $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$

e.g., $\boldsymbol{x}_{in} \in$ convex hull of $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$

$\boldsymbol{x}_{out}$ isolated from $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$

e.g., $\boldsymbol{x}_{out} \notin$ convex hull of $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$

– not surprising, usually $\boldsymbol{x}_{in}$ is "better" than $\boldsymbol{x}_{out}$

– finding optimal data

GATHERED

location	temperature
$\boldsymbol{x}_1$	T_1
$\boldsymbol{x}_2$	T_2
$\vdots$	$\vdots$
$\boldsymbol{x}_{M_{space}}$	T_M

$\Downarrow \quad \Downarrow \quad \Downarrow \quad \Downarrow$

OPTIMAL DATA

location	temperature
$\boldsymbol{x}_{max}$	$\Leftarrow T_{max}$
$\boldsymbol{x}_{min}$	$\Leftarrow T_{min}$

e.g., find $\boldsymbol{x}_{max} \in \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{M_{space}}\}$ at which the temperature is heighest

– constructing functions from data

GATHERED

location	temperature
$\boldsymbol{x}_1$	T_1
$\boldsymbol{x}_2$	T_2
$\vdots$	$\vdots$
$\boldsymbol{x}_M$	T_M

$$\text{given} \Longrightarrow \left\{ \begin{array}{c} (\boldsymbol{x}_1\,,\,T_1) \\ (\boldsymbol{x}_2\,,\,T_2) \\ \vdots \\ (\boldsymbol{x}_M\,,\,T_M) \end{array} \right\} \Longrightarrow \text{construct} \Longrightarrow \; f_T(\boldsymbol{x})$$

– of course, $f_T(\boldsymbol{x}) = f_T(\boldsymbol{x}; T_1, \ldots, T_M)$, i.e., depends on the data for T

– two classical ways to construct $f_T(\boldsymbol{x})$

- interpolation $\Longrightarrow f_T(\boldsymbol{x}_m) = T_m$ for $j = 1, \ldots, M$
- least squares approximation $\Longrightarrow f_T(\boldsymbol{x}_m) \neq T_m$ for $j = 1, \ldots, M$

– what can you do with $f_T(\boldsymbol{x})$?

- you can evaluate it at any $\boldsymbol{x}$ to obtain the temperature at points not in the input data set
- if obtaining the T_m's is expensive and one needs to obtain the temperature T_{new} at a new point $\boldsymbol{x}_{new}$, one can simply use (the appoximation) $f_T(\boldsymbol{x}_{new})$ at practically no cost

– add data to improve "results"

GATHERED

location	temperature
$\boldsymbol{x}_1$	T_1
$\boldsymbol{x}_2$	T_2
$\vdots$	$\vdots$
$\boldsymbol{x}_M$	T_M

– suppose we are not happy with these data so that we want to supplement them with a datum gathered at a new point $\boldsymbol{x}_{M+1}$ (or several new points)

NEW GATHERED

location	temperature
$\boldsymbol{x}_1$	T_1
$\boldsymbol{x}_2$	T_2
$\vdots$	$\vdots$
$\boldsymbol{x}_M$	T_M
$\boldsymbol{x}_{M+1}$	T_{M+1}

– ideally, we would like to choose the new point $\boldsymbol{x}_{M+1}$ so that we are now as happy as possible

– methods for doing so have been devised, but we can only be approximately happy

The curse of dimensionality (coined by Richard Bellman)

- Let's consider the humble example of the approximation of a function $f(\alpha,\beta)$ that depends on $p=2$ parameters $\{\alpha,\beta\}$ by a polynomial having the form

$$f_{approx1}(\alpha,\beta) = \sum_{i,j=0,1,2,i+j\le 2} a_{ij}\alpha^i\beta^j$$
$$= a_{00}+a_{10}\alpha+a_{10}\beta+a_{20}\alpha^2+a_{11}\alpha\beta+a_{02}\beta^2 \approx f(\alpha,\beta)$$

$$\Longrightarrow \quad \text{sum of exponents in } \alpha \text{ and } \beta \text{ is less than or equal to } n=2$$

  - to determine the six coefficient $\{a_{ij}\}_{i,j=0,1,2,i+j\le 2}$ we evaluate $f(\alpha,\beta)$ at 6 points $\{\alpha_k,\beta_k\}_{k=1}^6$ and require that

$$f_{approx1}(\alpha_k,\beta_k) = \sum_{i,j=0,1,2,i+j\le 2} a_{ij}\alpha_k^i\beta_k^j$$
$$= a_{00}+a_{10}\alpha_k+a_{10}\beta_k+a_{20}\alpha_k^2+a_{11}\alpha_k\beta_k+a_{02}\beta_k^2 = f(\alpha_k,\beta_k)$$

  - these are six equations in six unknowns so that (if the points $\{\alpha_k,\beta_k\}_{k=1}^6$ are distinct) we can find a unique set of six coefficients $\{a_{ij}\}_{i,j=0,1,2,i+j\le 2}$

– why would we want to do this

after all we know the function $f(\alpha,\beta)$ so
that we can evaluate it at any point (α,β)

– but suppose that

evaluating $f(\alpha,\beta)$ at a point is a very expensive task

and

we want to evaluate $f(\alpha,\beta)$ at lots of points

– so, instead of evaluating $f(\alpha,\beta)$ at all those points we evaluate $f_{approx1}(\alpha,\beta)$

– evaluating $f_{approx1}(\alpha,\beta)$ at a point is a very inexpensive task

– so, at the cost of only 6 expensive evaluations of $f(\alpha,\beta)$
we can find lots of values of the approximation
$f_{approx1}(\alpha,\beta)$ at very little cost

- Let's also consider another humble example in which we approximate the function $f(\alpha,\beta)$ of $p=2$ parameters by a polynomial now having the form

$$f_{approx2}(\alpha,\beta) = \sum_{i,j=0,1,2} a_{ij}\alpha^i\beta^j$$
$$= a_{00} + a_{10}\alpha + a_{10}\beta + a_{20}\alpha^2 + a_{11}\alpha\beta + a_{02}\beta^2$$
$$+a_{21}\alpha^2\beta + a_{22}\alpha^2\beta^2 + a_{12}\alpha\beta^2 \approx f(\alpha,\beta)$$

$$\Longrightarrow \quad \text{the exponent of each in } \alpha \text{ and } \beta \text{ is less than or equal to } n=2$$

  - we now have 9 coefficients to determine so we need to evaluate $f(\alpha,\beta)$ at 9 points
  - by the way, there are good practical reason why one would use $f_{approx2}(\alpha,\beta)$ and not $f_{approx1}(\alpha,\beta)$, even though the latter requires fewer expensive function evaluations

- What happens if we increase p and n?
  - specifically, how many expensive evaluations of $f(\vec{\alpha})$ (where $\vec{\alpha}=(\alpha_1,\alpha_2,\ldots,\alpha_p)$) do we have to do for larger p and n?

$p =$ number of parameters	$d =$ maximal degree of polynomials	number of evaluations of $f(\vec{\alpha})$ needed to constuct	
		$f_{approx1}(\vec{\alpha})$	$f_{approx2}(\vec{\alpha})$
3	3	20	64
	5	56	216
5	3	56	1,024
	5	252	7,776
10	3	286	1,048,576
	5	3,003	60,046,176
20	3	1,771	$> 1\times10^{12}$
	5	53,130	$> 3\times10^{15}$
100	3	176,851	$> 1\times10^{60}$
	5	96,560,646	$> 6\times10^{77}$
		$\Uparrow$	$\Uparrow$
		$\frac{(d+p)!}{d!p!}$	$(d+1)^p$

$\Longrightarrow$ number of atoms in the universe $\approx 10^{78}$

- For these examples, the curse of dimensionality =

  the explosive growth as p and/or n increase

  in the cost of constructing a polynomial approximation
  of a function that depends on

  p parameters and on the degree n of the polynomial

  – the curse appears in similar guises in many other settings

ELEMENTARY TOOLS

Very elementary linear algebra

- I assume that everyone knows what a vector and a matrix are, what are their transposes, what the identity matrix is, and other extremely elementary things about vectors and matrices

- Linear independence and rank of a matrix
  - a linear combination of a set of $\{z_1, z_2, \cdots, z_M\}$ is defined as

$$c_1 z_1 + c_2 z_2 + \cdots + c_m z_M$$

    where $\{c_1, c_2, \cdots c_M\}$ is a set of constants

  - a set of vectors $\{z_1, z_2, \cdots, z_M\}$ is linearly independent if
    none of the vectors in the set
    can be written as a linear combination of the other vectors

– viewing the columns (or rows) of a matrix $\mathbb{A}$ as vectors, the rank of a matrix
  is the maximum number of linearly independent
  columns (or equivalently rows) of the matrix
  - e.g., if $\mathbb{A} = (\boldsymbol{z}_1\ \boldsymbol{z}_2\ \cdots\ \boldsymbol{z}_M)$ where $\boldsymbol{z}_m$ denotes a column of $\mathbb{A}$
    the rank of $\mathbb{A}$ is the maximum number of columns that can
    be written as a linear combination of the other columns
  - row rank of a matrix $=$ column rank of a matrix

– a matrix $\mathbb{A}$ is said to have full column rank if the columns of
  the matrix are linearly independent
  - an $N \times M$ matrix $\mathbb{A}$ can have full column rank only if $M \leq N$
    i.e., if $M > N$, it is impossible for the matrix
    to have independent columns

– analogously, one an define full row rank

– clearly, only square matrices can have both full and column row ranks

- Symmetric and skew-symmetric matrices
  - symmetric matrix $\mathbb{A}^T = \mathbb{A}$
    - if a_{ij} denote the entries of $\mathbb{A}$, then $a_{ji} = a_{ij}$
  - skew-symmetric matrix $\mathbb{A}^T = -\mathbb{A}$
    - if a_{ij} denote the entries of $\mathbb{A}$, then $a_{ji} = -a_{ij}$
      the diagonal entries $a_{ii} = 0$

- Orthogonal (or orthonormal) matrices
  - square matrices $\mathbb{U}$ such that $\mathbb{U}\mathbb{U}^T = \mathbb{U}^T\mathbb{U} = \mathbb{I}$
    where $\mathbb{U}^T$ denotes the transpose of $\mathbb{U}$
    and $\mathbb{I}$ denotes the identity matrix
  - $\mathbb{U}^{-1} = \mathbb{U}$
  - if $\boldsymbol{u}_j$ and $\boldsymbol{u}_{j'}$ denote two columns of $\mathbb{U}$, then $\boldsymbol{u}_{j'}^T\boldsymbol{u}_j = \delta_{jj'}$
    and likewise for the rows

- Eigenvalues and eigenvectors of matrices
  - given a square matrix $\mathbb{A}$, then any pair λ and $\boldsymbol{u}$ that satisfy

$$\mathbb{A}\boldsymbol{z} = \lambda\boldsymbol{z}$$

    are referred to as an eigenvalue and eigenvector of $\mathbb{A}$, respectively

  - if $\mathbb{A}$ is a symmetric matrix, i.e, $\mathbb{A}^T = \mathbb{A}$, then
    the eigenvalues and eigenvectors of $\mathbb{A}$ are real

  - if $\mathbb{A}$ a is positive definite matrix
    i.e. $\boldsymbol{z}^T\mathbb{A}\boldsymbol{z} > 0$ for all vectors $\boldsymbol{z} \neq 0$
    then all the eigenvalues are positive

    - if $\boldsymbol{z}^T\mathbb{A}\boldsymbol{z} \geq 0$, then $\mathbb{A}$ is said to be positive semi-definite
      and can possibly have a null vector

      i.e., it can possibly have a vector $\boldsymbol{z}$ such that
      $\mathbb{A}\boldsymbol{z} = \mathbf{0}$, where $\mathbf{0}$ denotes the vector of all zeros

- Singular value decomposition

  – for any $N \times M$ matrix $\mathbb{A}$, there exist an
  $N \times N$ orthonormal matrix $\mathbb{U}$
  and an
  $M \times M$ orthonormal matrix $\mathbb{V}$
  such that

$$\mathbb{A} = \mathbb{U}\Sigma\mathbb{V}^T$$

  where
  the $N \times M$ matrix Σ is a rectangular diagonal matrix
  in which the diagonal entries are non-negative
  and appear in a non-increasing order

  - the columns of $\mathbb{U}$ are referred as the left-singular vectors of $\mathbb{A}$

  - the columns of $\mathbb{V}$ are referred as the right-singular vectors of $\mathbb{A}$

  - the diagonal entries σ_i of Σ are referred as the
  singular values of the matrix $\mathbb{A}$
  the number of positive singular values $=$ rank of $\mathbb{A}$

– we have that for any matrix $\mathbb{A}$

$$\mathbb{A}^T\mathbb{A} = \mathbb{V}\Sigma^T\mathbb{U}^T\mathbb{U}\Sigma\mathbb{V}^T = \mathbb{V}\Sigma^T\Sigma\mathbb{V}^T \quad \Longrightarrow \quad \mathbb{A}^T\mathbb{A}\mathbb{V} = \mathbb{V}\Sigma^T\Sigma$$

so that the columns $\mathbb{V}$ are the eigenvectors of $\mathbb{A}^T\mathbb{A}$ and the diagonal entries σ_i^2 of $\Sigma^T\Sigma$ are the eigenvalues of $\mathbb{A}^T\mathbb{A}$

- likewise

$$\mathbb{A}\mathbb{A}^T = \mathbb{U}\Sigma^T\mathbb{V}^T\mathbb{V}\Sigma\mathbb{U}^T = \mathbb{U}\Sigma\Sigma^T\mathbb{V} \quad \Longrightarrow \quad \mathbb{A}\mathbb{A}^T\mathbb{U} = \mathbb{U}\Sigma\Sigma^T$$

so that the columns $\mathbb{U}$ are the eigenvectors of $\mathbb{A}\mathbb{A}^T$ and the diagonal entries σ_i^2 of $\Sigma\Sigma^T$ are the eigenvalues of $\mathbb{A}\mathbb{A}^T$

- note that $\mathbb{A}\mathbb{A}^T$ and $\mathbb{A}\mathbb{A}$ have the same nonzero eigenvalues which are the squares of the nonzero singular values of $\mathbb{A}$

- note also that, in general, $\mathbb{A}\mathbb{A}$ and $\mathbb{A}\mathbb{A}^T$ are positive semi-definite matrices

Very elementary probability and statistics

- Given a random variable y, a

  probability density function (PDF) $\rho(y)$

  defined on an interval $[a,b]$ satisfies the following properties

  $$\rho(y) \geq 0 \text{ for all } y \in [a,b] \qquad \text{and} \qquad \int_a^b \rho(y)\,dy = 1$$

  – one may have $a = -\infty$ and/or $b = +\infty$

  – PDFs may also be defined on a finite set of points, i.e., given the point set $S = \{a_k\}_{k=1}^K$, we have that $\rho(y_k)$ defined over S is a function such that

  $$\rho(y_k) \geq 0 \text{ for all } y_k \in S \qquad \text{and} \qquad \sum_{k=1}^K \rho(y_k) = 1$$

- Given a random variable y and a PDF $\rho(y)$

  a cumulative density function (CDF) $F(y)$ is defined for $y \in [a,b]$ as

$$F(y) = \int_a^y \rho(y')\,dy'$$

  - clearly

$$F(a) = 0 \qquad F(b) = 1$$

$$0 \le F(y) \le 1 \text{ for } y \in (a,b) \qquad F(y') \ge F(y) \text{ whenever } y' > y$$

  - for any $a \le \widehat{a} \le \widehat{b} \le b$,

$$\text{Prob}(y \le \widehat{a}) = \{ \text{ probability that } y \le \widehat{a} \} = F(\widehat{a})$$

$$\text{Prob}(y \in (\widehat{a},\widehat{b})) = \{ \text{ probability that } y \in (\widehat{a},\widehat{b}) \} = F(\widehat{b}) - F(\widehat{a})$$

  - if $\rho(y)$ is continuous, then for any $\widehat{a} \in (a,b)$

$$\text{Prob}(y = \widehat{a}) = \{ \text{ probability that } y = \widehat{a} \} = 0$$

  - $F(y)$ is continuos and $\rho(y) = \dfrac{dF}{dy}$ if the derivative exists

- Given two random variables y and z equipped with PDFs $\rho_y(y)$ and $\rho_(z)$, respectively

  the conditional probability of an event[1] z

  is the probability that that event will occur

  knowing that the event y has already occurred

$$\text{conditional probability} = \rho(z|y) = \frac{\rho(y \cap z)}{\rho(y)}$$

$$= \frac{\text{probability that both events } y \text{ and } z \text{ will happen}}{\text{probability that the event } y \text{ already happened}}$$

  - if y and z are independent, then $\rho(z|y) = \rho(z)$

  - for three events y, z, and w, the probability of all three happening is

$$\rho(w \cap y \cap z) = \rho(y)\rho(y|z)\rho(w|(z \cap y))$$

[1] An event z is a draw from the PDF $\rho(y)$

- Then there is the Bayes formula

$$\rho(z|y) = \text{probability of } z \text{ given that } y \text{ is known} = \frac{\rho(y|z)\rho(z)}{\rho(y)}$$
$$= \frac{\big(\text{probability of } z \text{ given that } y \text{ is known}\big)\big(\text{probability of } z\big)}{\text{probability of } y}$$

  - Bayes formula is put to good use

    e.g., given data y, it can be used to determine the probability of z

- The expected value or mean of y is defined as

$$\mathsf{E}(y) = \int_a^b y\rho(y)\,dy$$

  - if $\mathsf{E}(y) = 0$, we say that y has zero mean
    or we say that the PDF $\rho(y)$ is centered

- The m-th moment of y is defined as

$$\mathsf{M}_m(y) = \mathsf{E}(y^m) = \int_a^b y^m\rho(y)\,dy$$

- The variance of y is defined as

$$\mathsf{V}(y) = \mathsf{E}\big((y - \mathsf{E}(y)(y - \mathsf{E}(y)\big) = \int_a^b \big(y - \mathsf{E}(y)\big)^2 \rho(y)\, dy$$

  - it is easily verified that

$$\mathsf{V}(y) = \mathsf{E}(y^2) - \big(\mathsf{E}(y)\big)^2 = \int_a^b y^2 \rho(y)\, dy - \Big(\int_a^b (y\rho(y)\, dy\Big)^2$$

  - clearly, if y has zero mean, we have $\mathsf{V}(y) = \mathsf{E}(y^2) = \mathsf{M}_2(y) = \int_a^b y^2 \rho(y)\, dy$

- The standard deviation of y is defined as

$$\sigma(y) = \sqrt{\mathsf{V}(y)}$$

  - a small standard deviation indicates that the PDF is clustered closely around the mean
  - a large standard deviation indicates that the PDF has appreciable values far away from the mean

- The covariance is defined as*

$$\mathrm{Cov}(y,y') = \mathrm{E}\big((y-\mathrm{E}(y)(y'-\mathrm{E}(y'))\big) = \mathrm{E}(yy') - \mathrm{E}(y)\mathrm{E}(y')$$

  - obviously $\mathrm{V}(y) = \mathrm{Cov}(y,y)$
  - the covariance is
    - symmetric $\mathrm{Cov}(y,y') = \mathrm{Cov}(y',y)$
    - nonnegative $\mathrm{Cov}(y,y') \geq 0$

- The correlation is defined as

$$K(y,y') = \frac{\mathrm{Cov}(y,y')}{\sigma(y)\sigma(y')}$$

  - obviously $K(y,y) = 1$
  - in general, $K(y,y') \leq 1$ for all y, y'

- If $K(y,y') = 0$, we say that y and y' are uncorrelated

---

* Here, y and y' are drawn from the same PDF

- Now, suppose we have N random variables $\{y_n\}_{n=1}^N$ defined over an N-dimensional set Γ

  – a joint probability density function (PDF) $\rho(y_1,\ldots,y_N)$ is a mapping from Γ into the real numbers such that

$$\rho(y_1,\ldots,y_N) \geq 0 \qquad \text{for all} \qquad \{y_1,\ldots,y_N\} \in \Gamma$$

  and

$$\int_\Gamma \rho(y_1,\ldots,y_N)\, dy_1 \cdots dy_N = 1$$

- For $n = 1, \ldots, N$, the mean or expected value of y_n is given by

$$\mu_n = \mathsf{E}(y_n) = \int_\Gamma y_n \rho(y_1, \ldots, y_N)\, dy_1 \cdots dy_N$$

  - if $\mu_n = 0$, then y_n is called centered

- The covariance of $\{y_n\}_{n=1}^N$ is the $N \times N$ matrix $\mathbb{C}$ given by*

$$\begin{aligned}
\mathbb{C}_{nn'} &= \mathsf{E}\Big((y_n - \mu_n)(y_{n'} - \mu_{n'}) \Big) \\
&= \int_\Gamma (y_n - \mu_n)(y_{n'} - \mu_{n'}) \rho(y_1, \ldots, y_N)\, dy_1 \cdots dy_N \\
&= \mathsf{E}(y_n y_{n'}) - \mathsf{E}(y_n)\mathsf{E}(y_{n'}) = \mathsf{E}(y_n y_{n'}) - \mu_n \mu_{n'}
\end{aligned}$$

  - if either y_n or $y_{n'}$ is centered $\qquad \mathbb{C}_{nn'} = \mathsf{E}(y_n y_{n'})$

---

* Here, y and y' are drawn from different PDFs

- The random variables $\{y_n\}_{n=1}^N$ are independent if the choice of values of any subset of variables does not depend on the choices made for the values of the remaining variables

  - the random variables $\{y_n\}_{n=1}^N$ are independent if and only if the joint PDF is a product of the PDFs for the individual variables

$$\rho(y_1,\ldots,y_N) = \prod_{n=1}^{N} \rho_n(y_n)$$

    where, for each $n = 1,\ldots,N$, $\rho_n(\cdot)$ is a mapping from an interval Γ_n to the real numbers that satisfies

$$\rho_n(y_n) \geq 0 \qquad \int_{\Gamma_n} \rho_n(y_n)\, dy_n = 1$$

  - note that then Γ is the hyper-rectangle $\Gamma = \Gamma_1 \otimes \Gamma_2 \otimes \cdots \otimes \Gamma_N$

  - note that the intervals Γ_n maybe of finite (e.g., for a uniform distribution) or infinite (e.g., for a Gaussian distribution) extent

- The random variables $\{y_n\}_{n=1}^N$ are uncorrelated only if

$$\mathbb{C}_{nn'} = \sigma_n^2 \delta_{nn'} = \begin{cases} \sigma_n^2 & \text{if } n = n' \\ 0 & \text{if } n \neq n' \end{cases} \qquad \forall n, n' = 1, \ldots, N$$

where σ_n^2 denotes the variance of y_n

$$\begin{aligned} \sigma_n^2 = \mathbb{C}_{nn} &= \mathsf{E}\Big((y_n - \mu_n)^2 \Big) \\ &= \int_\Gamma (y_n - \mu_n)^2 \rho(y_1, \ldots, y_N)\, dy_1 \cdots dy_N \\ &= \mathsf{E}(y_n^2) - \mu_n^2 \end{aligned}$$

– if y_n is centered $\qquad \sigma_n^2 = \mathbb{C}_{nn} = \mathsf{E}(y_n^2)$

- Independence implies uncorrelated
  - if $n \neq n'$, then

$$\mathbb{C}_{nn'} = \int_{\Gamma} (y_n - \mu_n)(y_{n'} - \mu_{n'}) \rho(y_1, \ldots, y_N)\, dy_1 \cdots dy_N$$

$$= \int_{\Gamma_1 \otimes \Gamma_2 \otimes \cdots \otimes \Gamma_N} (y_n - \mu_n)(y_{n'} - \mu_{n'}) \prod_{n=1}^{N} \rho_n(y_n)\, dy_1 \cdots dy_N$$

$$= \int_{\Gamma_n} (y_n - \mu_n)\rho_n(y_n)\, dy_n \int_{\Gamma_{n'}} (y_{n'} - \mu_{n'})\rho_{n'}(y_{n'})\, dy_{n'}$$
$$\times \prod_{n''=1, n'' \neq n, n'' \neq n'}^{N} \int_{\Gamma_{n''}} \rho_{n''}(y_{n''})\, dy_{n''}$$

$$= \int_{\Gamma_n} (y_n - \mu_n)\rho_n(y_n)\, dy_n \int_{\Gamma_{n'}} (y_{n'} - \mu_{n'})\rho_{n'}(y_{n'})\, dy_{n'}$$

$$= \mathsf{E}(y_n - \mu_n)\mathsf{E}(y_{n'} - \mu_{n'}) = 0$$

- However, uncorrelated does not necessarily imply independence
  - let y_1 be uniformly distributed on $[-1, 1]$

$$\implies \mathsf{E}(y_1) = 0 \qquad \mathsf{E}(y_1^3) = 0$$

  - let $y_2 = \frac{3}{2} y_1^2$
  - clearly, $\{y_1, y_2\}$ is not independent
  - however, $\{y_1, y_2\}$ is uncorrelated

$$\mathbb{C}_{12} = \mathsf{E}(y_1 y_2) - \mathsf{E}(y_1)\mathsf{E}(y_2) = \tfrac{3}{2}\mathsf{E}(y_1^3) = 0$$

- When does uncorrelated imply independence?
  - if and only if the variables follow a multivariate Gaussian distribution

SAMPLING IN HYPERCUBES

- A fundamental (perhaps the fundamental) data science task is to obtain samples of some entity at several "locations"

  here the nomenclature "locations"

  is used in a very generalized sense

  – the locations could be points in space and/or instants of time
    - e.g., sampling the temperature at 9:00 AM every day
      at fixed physical locations in Antarctica

  – the "locations" could be pixels in a digital image
    - e.g., sampling the color and intensity at pixels in an image

  – the "locations" could be people
    - e.g., sampling the age of a population segment

    •

    •

    •

- Where are "locations" located?
  - in some settings one does not have a choice of "locations"
    - e.g., if someone wants to sample the water quality at ponds and lakes in their state
  - in other settings, one is free to choose, among many choices, a predetermined set of "locations" that suits an objective
    - e.g., choosing quadrature points to meet an accuracy objective for approximating the integral of a function
  - in still other settings, one wants to sample at an optimal set of "locations" which are not predetermined in any way
    - e.g., in an petroleum field, find a few locations which are optimal for extraction purposes
  - then there is the case of an infinite number of "locations"
    - e.g., we are not dealing with discrete set of points, but with regions which contain a infinite number of points

- All these settings are worth exploring and all of them have a huge body of work devoted to them
  - we touch on all of these sampling settings during the lectures
  - however, due to time limitations,
    we only explore two of them in some detail
  - the first is point sampling in hyper-rectangles

Point sampling basics

- Point sampling in regions in $\mathbb{R}^d$ is useful in lots of settings
  - mesh generation
  - meshless computing methods
  - particle methods
  - parametric studies
  - response surface analyses
  - statistical analyses
  - process optimization
  - multi-dimensional integration
  - . . . . . .
- Point sampling is the central task in the design of experiments of either the laboratory or computational types; it answers the question
  - how does one choose the parameters that are used in an experiment?

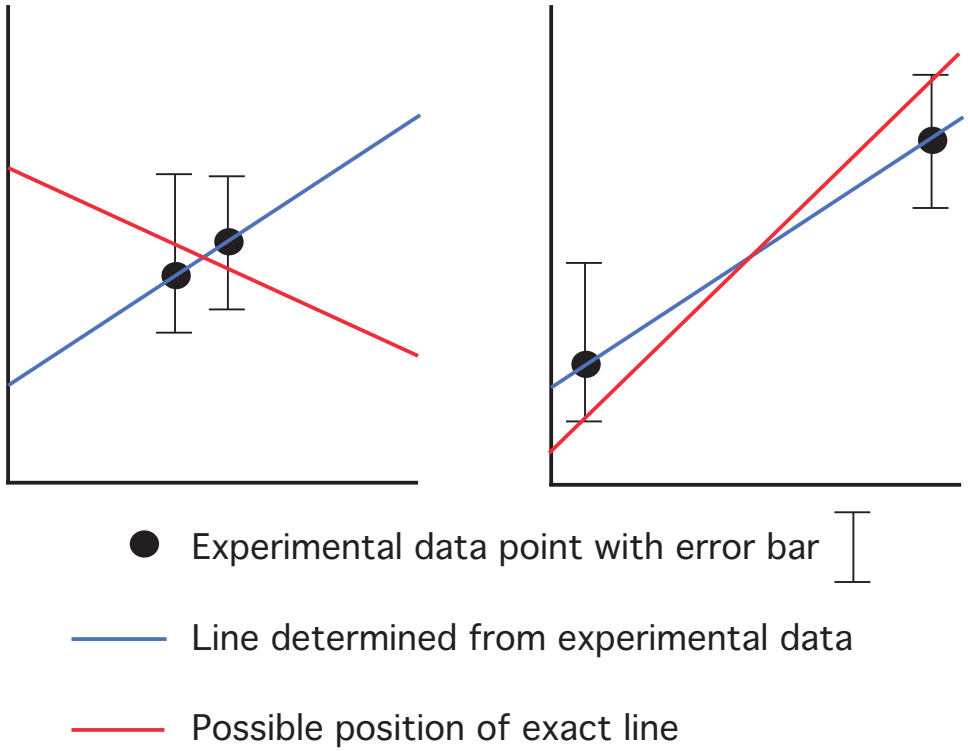
- All combinations of the dimensionality and cardinality of point sets are of interest, depending on the application
  - large number of points in high dimensions
  - large number of points in low dimensions
  - small number of points in high dimensions
  - small number of points in low dimensions
  - ..... and everything in between

- All types of point distributions are of interest as well, again depending on the application
  - uniformly distributed points in simple regions (e.g., hypercubes)
  - general regions
  - nonuniformly distributed points
  - anisotropically distributed points
  - ..... and combinations thereof

- Cautionary note 1
  - sampling methods that are known to be good for a large number of sampling points may not be so good for sparse sampling
  - likewise, theorems that hold "as the number of sampling points goes to ∞" are often useless in practice
  - unfortunately, most sampling methods and the accompanying theorems have been developed for the case of a large number of sampling points

- Cautionary note 2
  - there is a fundamental difference between the design of laboratory and computational experiments
    - computer experiments are repeatable
    - due to noise, laboratory experiments are not exactly repeatable

- Effect of sampling strategy in the design of a laboratory experiment having one free parameter



Left: points sampled near center of sampling interval

Right: points sampled near the end of the sampling interval

- One should take advantage of any information known about the parameters
  - if the parameters are constrained, one should sample in the feasible region
  - if there is a known bias towards a certain sub-range of parameter values, one should sample nonuniformly
  - if there are known correlations between the parameters, one should sample anisotropically

- If one knows nothing, then one should just sample uniformly in a hypercube
  - even this requires some rough knowledge about upper and lower bounds on the possible parameter values
  - because, more often than not, that is what one will end up doing for the design of computational experiments

POINT SAMPLING IN HYPER-RECTANGLES

- One is tempted to believe that sampling within a simple domain such as a hyper-rectangle is rather uninteresting
  - however, this is far from the truth
  - in fact, there are lots of applications for which sampling in a hyper-rectangle is needed
    - uncertainty quantification, optimization, control, image processing, ...
  - in fact, designing, analyzing, and applying strategies for sampling in hyper-rectangles is a huge industry
    - so much so that there have been many algorithms devised specifically for sampling in hyper-rectangles that cannot be used for sampling in more general domains

- A hyper-rectangle Γ is simply a high-dimensional box
  - canonical bounded hyper-rectangles include
    $1 \times 1 \times \cdots \times 1$ hypercubes having a vertex at the origin
    $2 \times 2 \times \cdots \times 2$ hypercubes having their center at the origin
  - hyper-rectangles of interest include those having one, or some, edges that have semi-infinite or infinite lengths
    - a rectangle with one side $[0, 1]$ and the other a semi-infinite side $[0, \infty]$
    - a rectangle with one side $[0, 1]$ and the other an infinite side $[-\infty, \infty]$
  - so an N-dimensional hyper-rectangle is defined in terms of its one-dimensional edges as

$$[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N]$$

    where $-\infty \leq a_n < b_n \leq \infty$ for $n = 1, \ldots, N$

Monte Carlo (MC) sampling

- By far, the most used sampling scheme is Monte Carlo sampling
  - sample points are chosen randomly
  - sampling can be done sequentially
    - given a Monte Carlo set of samples, one can add another sample point without having to move any point in the given set
  - sampling can be effected on general domains, not just hyper-rectangles
    - but for now we stick to hyper-rectangles

- If each component y_n of the vector $\boldsymbol{y}$ can be chosen independently from the other coordinates

  i.e., if we have the joint PDF $\rho(\boldsymbol{y}) = \rho_n(y_1)\rho_2(y_2)\cdots\rho_N(y_N)$

  then

  each component y_n of $\boldsymbol{y}$ can be independently sampled using its own PDF $\rho_n(y_n)$

  – specifically

  - randomly select, for each n, a point $y_n \in [a_n, b_n]$ according to the PDF $\rho_n(y_n)$

  - then, the sample point in the hyper-rectangle is simply $\boldsymbol{y} = (y_1,\, y_2,\, \cdots\, y_n)$

- If the hyper-rectangle is bounded

  – then the most commonly used strategy is to sample uniformly

  – of course, one cannot sample uniformly if an edge is infinite ($a_n = -\infty,\, b_n = \infty$) or semi-infinite (e.g., $a_n = 0,\, b_n = \infty$)

- The more general case has a joint PDF $\rho(\boldsymbol{y})$ that is not separable
  - e.g., the general multivariate Gaussian PDF is not separable

$$\frac{1}{\sqrt{2\pi^N|\Sigma|}} e^{\frac{1}{2}(\boldsymbol{y}-\mathsf{E}(\boldsymbol{y}))^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\mathsf{E}(\boldsymbol{y}))}$$

$$\begin{aligned} \boldsymbol{\Sigma} &\Leftarrow N \times N \text{ symmetric, positive definite matrix} \\ &\quad \text{of correlations between } y_n \text{ and } y_{n'} \\ |\boldsymbol{\Sigma}| &\Leftarrow \text{determinant of } \boldsymbol{\Sigma} \\ \mathsf{E}(\cdot) &\Leftarrow \text{expected (or mean) value} \end{aligned}$$

  - if the components of $\boldsymbol{y}$ are uncorrelated
    then $\boldsymbol{\Sigma}$ is a diagonal matrix with diagonal entries σ_n^2, $n = 1, \ldots, N$

$$\frac{1}{\sqrt{(2\pi)^N \, \Pi_{n=1}^N \sigma_n}} e^{\frac{|\boldsymbol{y}-\mathsf{E}(\boldsymbol{y})|^2}{2\sigma^2}} = \left(\frac{1}{\sqrt{2\pi}\,\sigma_1} e^{\frac{(y_1-\mathsf{E}(y_1))^2}{2\sigma_1^2}}\right)\left(\frac{1}{\sqrt{2\pi}\,\sigma_2} e^{\frac{(y_1-\mathsf{E}(y_2))^2}{2\sigma_2^2}}\right) \cdots \left(\frac{1}{\sqrt{2\pi}\,\sigma_n} e^{\frac{(y_N-\mathsf{E}(y_n))^2}{2\sigma_3^2}}\right)$$

$$\sigma_n^2 \Leftarrow \text{variance of } y_n$$

- indeed, this PDF is the product of N one-dimensional Gaussian PDFs

- this is a very special PDF

- How does one sample non-uniformly?
  - sets of points distributed according to a joint PDF $\rho(\boldsymbol{y})$ can be sampled from uniformly distributed point sets by an appropriate mapping
    - in fact, this holds for any sampling method
    - constructing the needed mapping may not always be a simple matter

– alternately, for non-uniform Monte Carlo sampling

- the sampling itself can incorporate the PDF $\rho(\boldsymbol{y})$
- one means for doing so is to use a rejection method

– one such rejection method proceeds as follows

given a joint PDF $\rho(\boldsymbol{y})$ defined for $\boldsymbol{y} \in \Gamma$

- set $\rho_{max} = \max_{\boldsymbol{y} \in \Gamma} \rho(\boldsymbol{y})$
- then sample a point $\boldsymbol{y} \in \Gamma$ according to the uniform PDF
- also sample a point $\widehat{y} \in [0, 1]$ according to the one-dimensional uniform PDF
- if $\widehat{y} < \dfrac{\rho(\boldsymbol{y})}{\rho_{max}}$, then $\boldsymbol{y}$ is accepted as one of the desired sample points
- otherwise, it is rejected
- one continues the process until one obtains the desired number of sample points

Quasi-Monte Carlo (QMC) sequences

- The descriptor "sequences" refers to the fact that the QMC points are sampled one at a time so that an M-point set retains all the points of the $M-1$ point set

  - in this regard, QMC and MC sampling are alike

  – but, unlike MC, the QMC samples are (usually) deterministically defined

  – also unlike MC, QMC sampling can not be effected for general domains

- Many QMC sequences have been defined, including the

  Faure, Halton, Niederreiter, and Sobol sequences

  just to name a few

  – we look at a few such sampling approaches for uniform sampling in a unit hypercube

  – there is a huge literature about QMC sequences

- As an example, consider Halton sequences which are determined according to the following procedure

  - given a prime number p

    any positive integer q can be uniquely represented as $q = \sum_i q_i p^i$ for some integers q_i

  - define the mapping $H_p(q) = \sum_i q_i / p^{i+1}$

    we then have that $H_p(q) \in [0, 1]$

  - then, the Halton sequence of M points in N dimensions is given by

$$\left\{ H_{p_1}(q), H_{p_2}(q), \ldots, H_{p_N}(q) \right\}_{m=1}^{M}$$

    where $\{p_n\}_{n=1}^{N}$ is a set of N prime numbers

- Hammersley sampling is technically not a QMC method because
  the sample points are not determined sequentially

  – however, Hammersley sampling is also
    deterministic
    and
    relies on Halton sequences
    so that it often lumped together with true QMC sequences

  – Hammersley sampling in the unit hypercube in $\mathbb{R}^N$ proceeds as follows

    - the first coordinate of the sample points is determined
      by a uniform partition of the unit interval

    - the remaining coordinates are determined from an
      $(N-1)$-dimensional Halton sequence

Latin hypercube sampling (LHS)

- Many variations of LHS sampling have been developed

  - here we describe the basic technique

- A set of LHS sampling of M points in the unit hypercube in R^N are determined randomly and non-sequentially by the following process

  – first, the unit cube is divided into M^N congruent cubical bins

  i.e., into M bins in each of the N coordinate directions

  – then, M of the cubical bins are chosen according to N random permutations of $\{1, 2, \ldots, M\}$

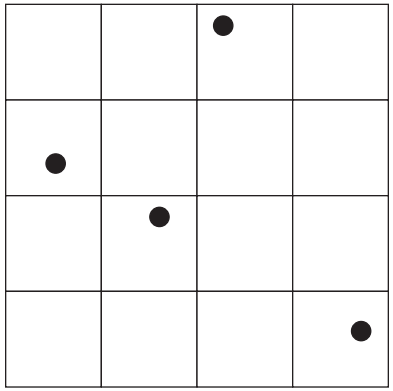
  – finally

  a random point is sampled within each of the M cubical bins so chosen

  or alternately

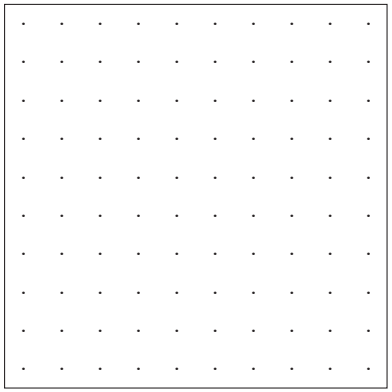
  the points are simply chosen to be the center points of those bins

- The example LHS sample set given in the figure corresponds to $N = 2$, $M = 4$, and the permutations $\{3, 2, 4, 1\}$ and $\{4, 2, 1, 3\}$

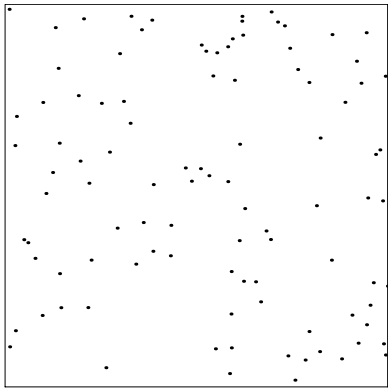


  - note that, by construction
    - in each row of bins, there is only one sample point
    - in each column of bins, there is only one sample point
  - this is the general feature of Latin hypercube samples

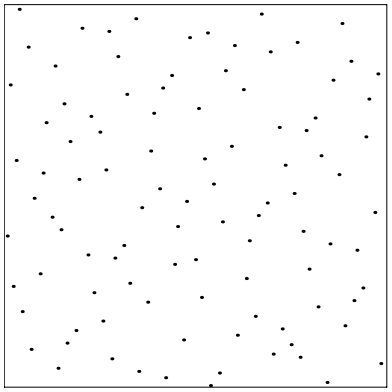
Illustrations of five "uniform" 100 point sets in a square



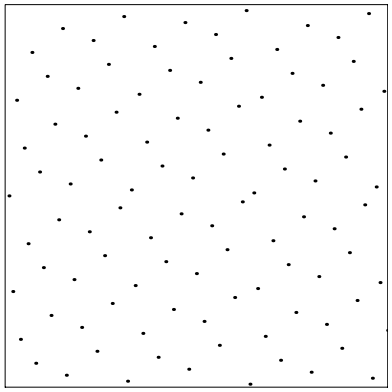
Cartesian



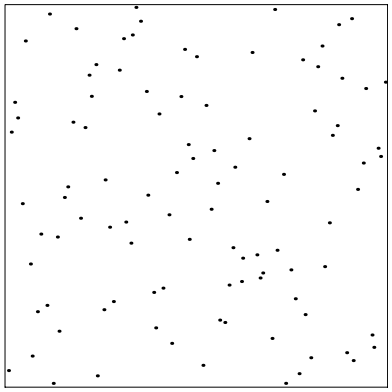
Monte Carlo



Halton



Hammersley



Latin hypercube

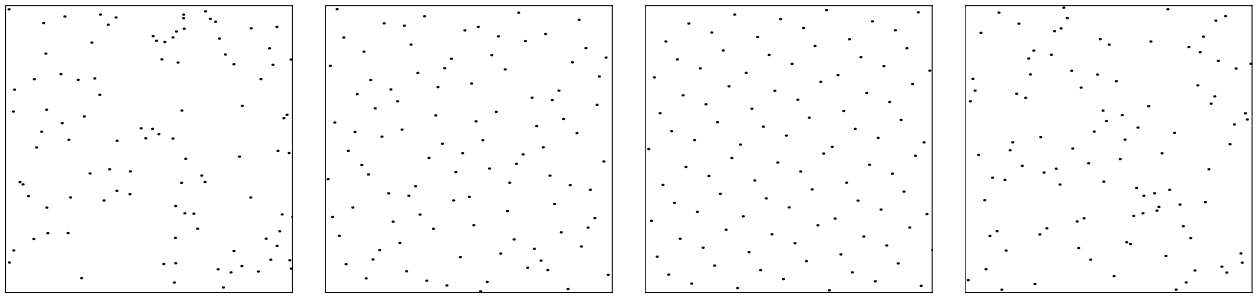
- Certainly, the tensor-product Cartesian set is the most "uniform", but of course
  - the number of points M is restricted to be an integer power of the parameter dimension N
  - furthermore, for more than one reason, Cartesian sampling turns out to be a very bad choice of sample points for some important settings

- For the other four cases
  - the sample points are certainly not uniformly distributed
  - note that Hammersley "looks" more uniform than does Halton, which is one of the reasons why this variant of Halton was developed

– can the uniformity of "uniform" point sets be quantified?

- absolutely, and in many ways

- Uniformity comparisons using quantitative measures of uniformity
  - Utopian uniformly distributed point samples have three attributes
    - the points are equally spaced
    - the points cover the region, i.e., there are no relatively large subregions that contain no points
    - the points are isotropically distributed, i.e., there is no directional bias in the placement of points
  - in 2D, the eye does a great job in using all three attributes when comparing the uniformity of different point sets



MC Halton Hammersley LHS

- Hammersley wins

– some popular quantitative measures of uniformity are flawed in that they only consider the spacing between points

– we consider eight measures of uniformity for sampling via

MC, 4 QMCs + Hammersley, LHS

and two other schemes we have not mentioned

	cov	γ	h	μ	χ	ν	τ	d
Ideal	0	1	0.0707	1	1.414	1	0	0
Monte Carlo	0.5075	88.75	0.1767	3.792	17.17	25.11	0.2833	0.01246
Halton	0.2911	3.37	0.1266	2.325	5.213	5.386	0.1732	0.01057
Hammersley	0.1559	2.94	0.1424	2.003	4.084	2.744	0.1257	0.00457
Faure	0.2552	2.70	0.1472	2.245	4.640	3.081	0.1432	0.00926
Sobol	0.5246	12.55	0.1378	2.068	20.16	3.369	0.1453	0.01865
Niederreiter	0.3072	3.05	0.1279	1.879	5.172	3.026	0.1294	0.01630
Latin hypercube	0.4771	8.60	0.1690	3.382	13.11	10.04	0.2907	0.02130
IHS	0.1588	2.46	0.1225	2.115	5.551	3.103	0.1033	0.00549
???	0.0509	1.30	0.0792	1.311	1.720	1.456	0.0355	0.00107

Eight measures of uniformity for different types of 100 "uniformly" distributed points in the unit square[1]

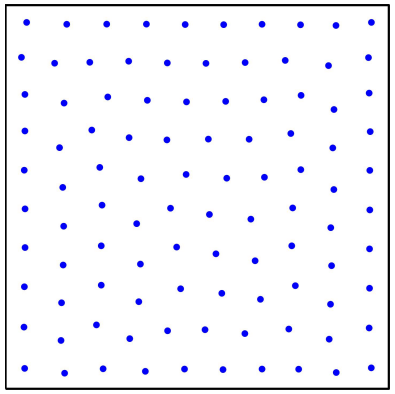
brown – flawed measures of sample uniformity green – good measures of sample uniformity

red - best blue - second best

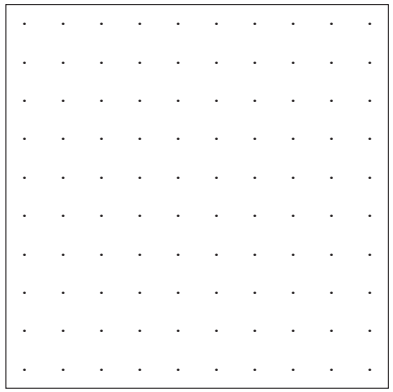
[1] IHS = improved hypercube sampling; this nomenclature is usurped for a specific method, when in fact, anything other than MC can is viewed in one way or another as an "improved hypercube sampling" method

– clearly, we need to say more about ??? sampling

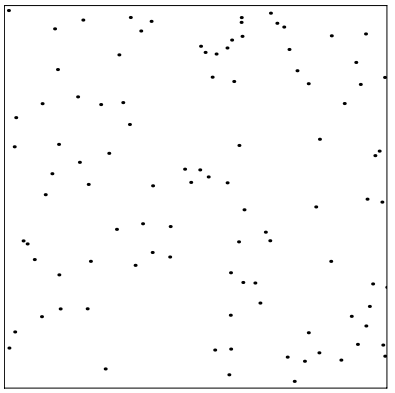
- we will do so later



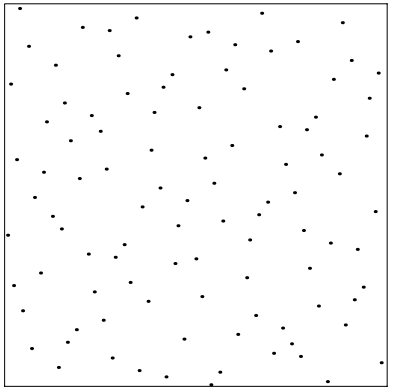
???



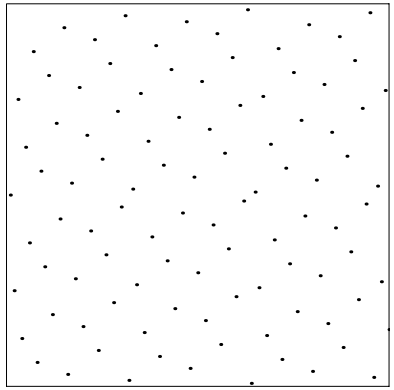
Cartesian



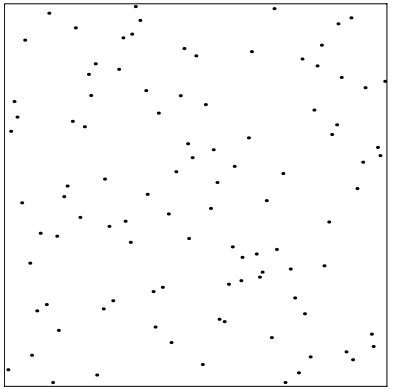
Monte Carlo



Halton



Hammersley



Latin hypercube

Nonuniform sampling in hypercubes

- Approaches for nonuniform point sampling similar to the two we discussed for Monte Carlo sampling can be applied for nonuniform QMC sequences and Hammersley sampling

  – here, we consider an approach that is specific to LHS

- For non-uniform Latin hypercube sampling

  - again, the sampling itself can incorporate the PDF $\rho(\boldsymbol{y})$

  – one means for doing so proceeds as follows

    - given a PDF $\rho(\boldsymbol{y}) = \prod_{n=1}^{N} \rho_n(y_n)$ defined for $\boldsymbol{y} \in \Gamma$ that has independent components

    - for each $n = 1, \ldots, N$,

      choose subintervals $\{[y_{n,n'-1}, y_{n,n'}]\}_{n'=1}^{N}$ so that

      $$0 = y_{n,0} < y_{n,1} < \ldots < y_{n,N-1} < y_{n,N} = 1$$

      and so that

      $$\int_{y_{n,n'-1}}^{y_{n,n'}} \rho(y_n) dy_n \qquad \text{is independent of } n'$$

    e.g., so that if $\rho(y_1) = 5$ for $y_1 \in (y_{1,2}, y_{1,3})$ and $\rho(y_1) = 3$ for $y_1 \in (y_{1,7}, y_{1,8})$, we have

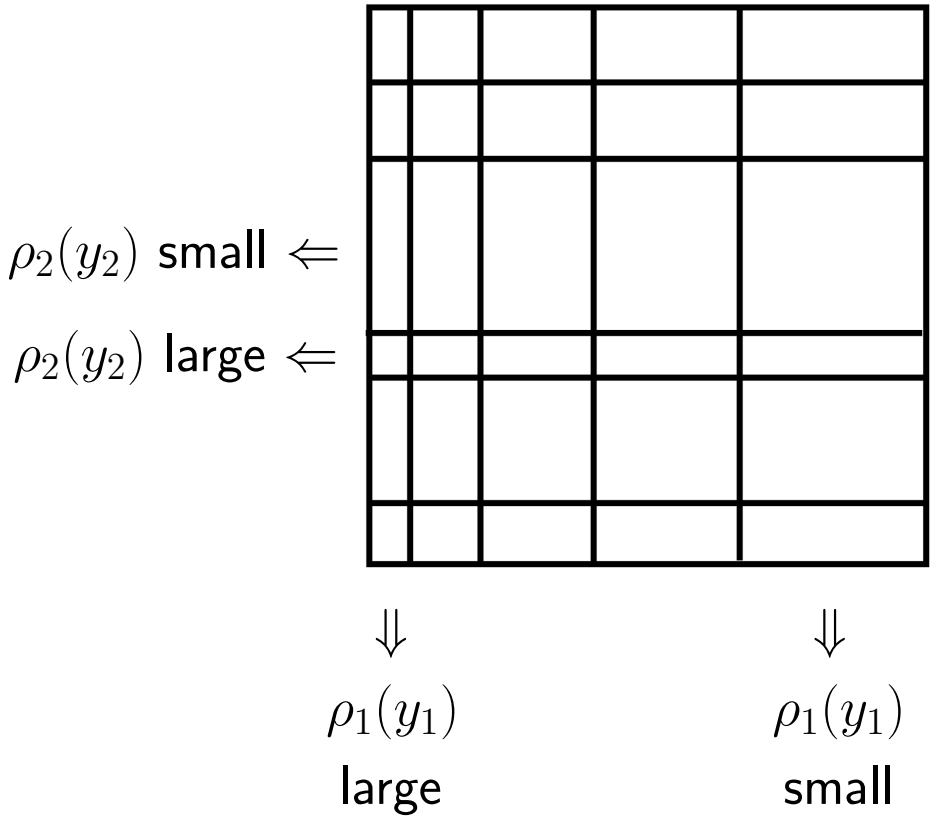
    $$\int_{y_{1,2}}^{y_{1,3}} 5 dy_1 = \int_{y_{1,7}}^{y_{1,8}} 3 dy_1$$

– as a result, the probability that a sample point is in a subinterval $[y_{n,n'-1}, y_{n,n'}]$ is the same for all subintervals

thus large PDF $\Rightarrow$ narrow subbintervals

small PDF $\Rightarrow$ wide subintervals

$\Longrightarrow$ points cluster in narrow subintervals in which the PDF large

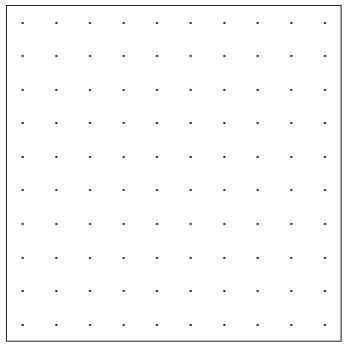


What makes a set of sample points good?
– regardless of dimension, i.e., even for $N = 2$
regardless of the number of sample points M

- Of course "good" can depend on the use one makes of the point set
  - here we presuppose we are going to use them for quadrature

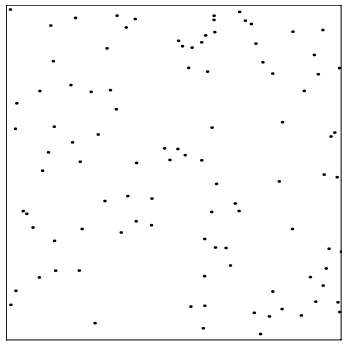
  although
  - what we discuss applies to other settings as well

- We have already considered good coverage, good spacing, and good isotropy
  - using these criteria, we were led to come observations about the relative quality of different point sets



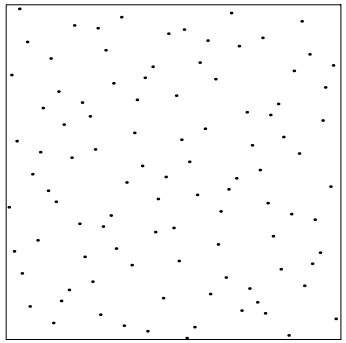
Cartesian

great coverage

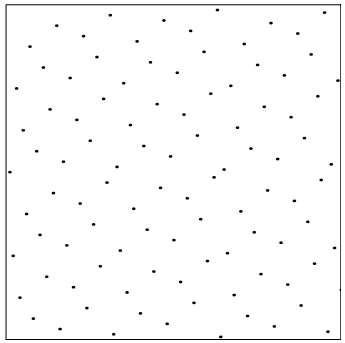


Monte Carlo

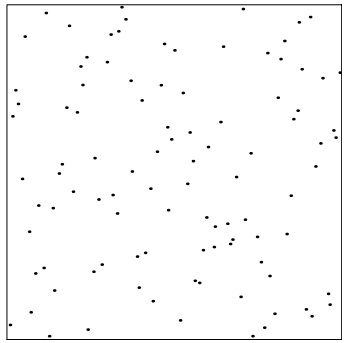
big areas devoid of sample points



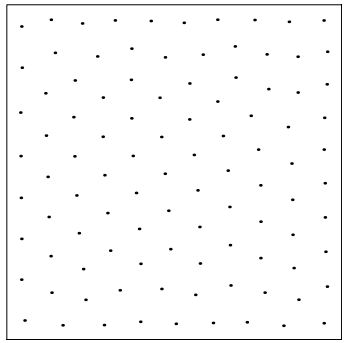
Halton



Hammersley



Latin hypercube



???

not as bad as Monte Carlo – – – but Halton and LHS still quite bad

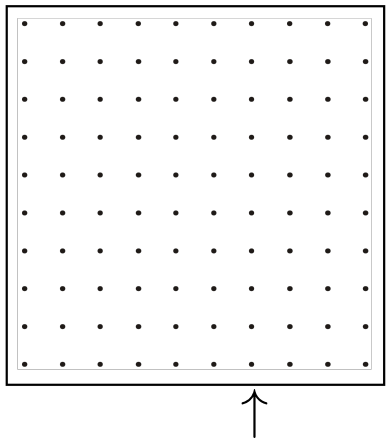
Hammersley not too bad, at least in 2D – – – ??? best

- However, there are additional criteria that should be met before one can label a particular sampling scheme as being a better than other schemes
  at least if they are to be used for quadrature

  – one such criteria is that

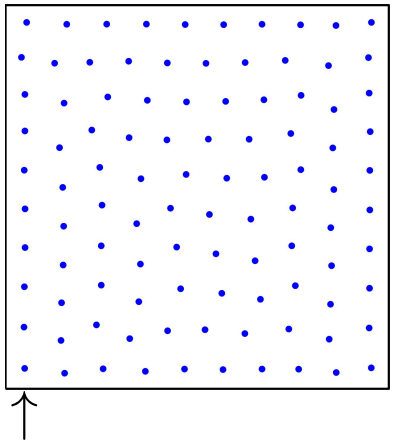
    projections of sample points onto lower-dimensional surfaces
    should not cluster, and certainly should not coincide

  – Cartesian samples are terrible even for $N = 2$



all 10 sample points in this column project to a single point on the bottom side

– ??? samples are bad in this respect



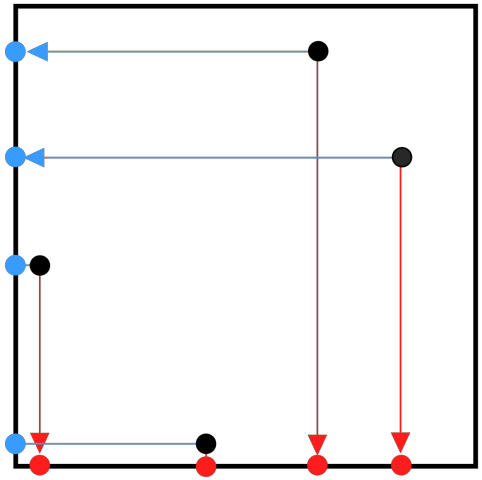
the 8 sample points in this
column project to a very close
clustering of points on the bottom side

- we note that although this feature of ??? point sampling
is bad for the settings such as the
quadrature rules we will soon considerx

it is actually very good for other settings

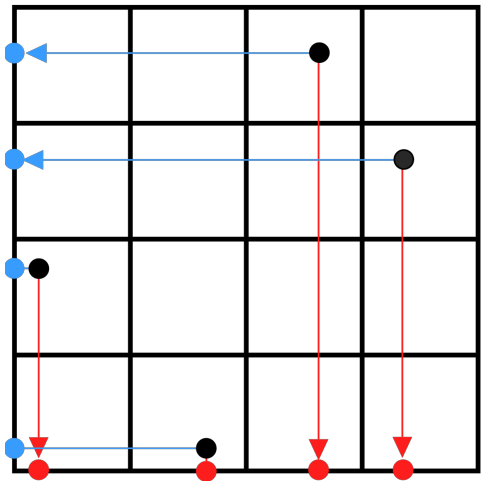
– coincident or closely-clustered projected points are a bad thing because, e.g., they can compromise the accuracy of sampling+averaging quadrature rules

- we want projections onto lower-dimensional faces to look like



– we will get back to this issue

– for this purpose, Latin hypercube samples are great



- e.g.,there is only one sample point in each column
  so when projected to the bottom side
  there is only one point in each sub-interval

- of course, the same goes for projections onto the left side

- in fact, the projection of Latin hypercubes points in N dimensions onto faces of lower dimension are also Latin samples

- for example,

if one has Latin hypercubes points in a cube, i.e., $N = 3$

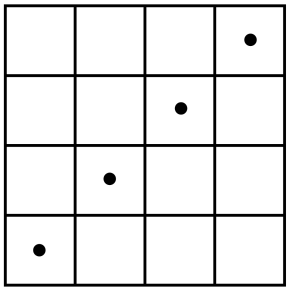
then

the projection of those points onto the six faces are all Latin hypercubes points in squares $(N = 2)$

and

the further projection onto the twelve edges are all Latin hypercubes points in intervals

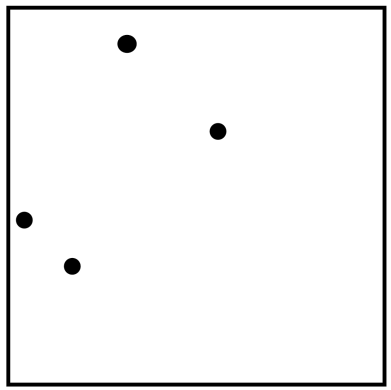
- Since LHS is so good at this criteria, why don't we go ahead and use it?
  - in fact, LHS is very popular
  - however, LHS sample points can have very bad coverage
    - an extreme case is the legitimate LHS point set



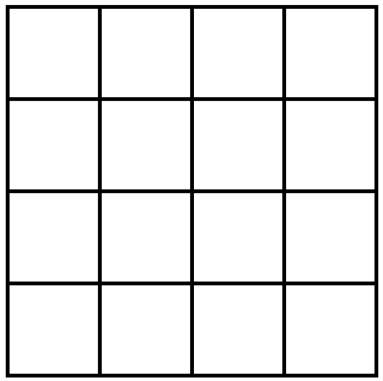
- Fortunately, it is a simple matter to take a set of sample points, including those with good coverage, and turn them into an LHS set

  (Cartesian points excluded - we are not interested in them for other reasons)
  - we refer to this transformation as the Latinization of a point set

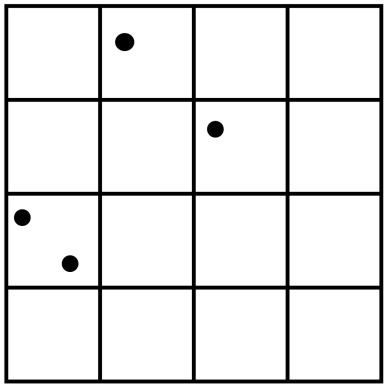
- Here, we illustrate of the Latinization process



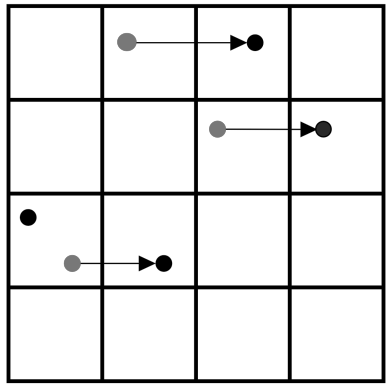
a point set



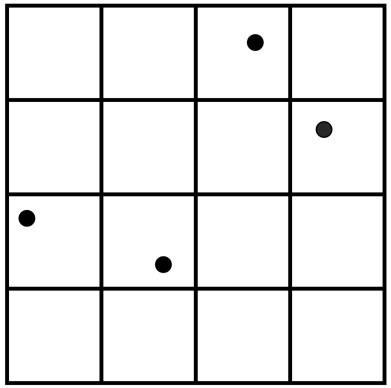
4×4 bins



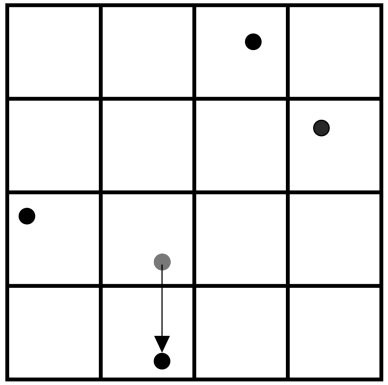
points in bins –
not Latin



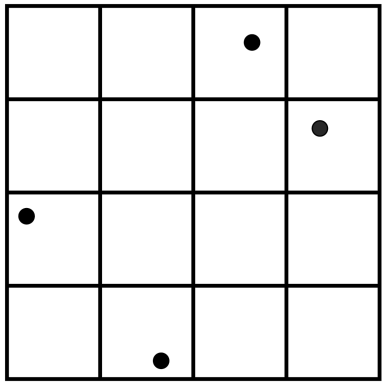
shifting y_1
coordinates



result of shifting –
y_1 coordinates now
in different y_1 intervals

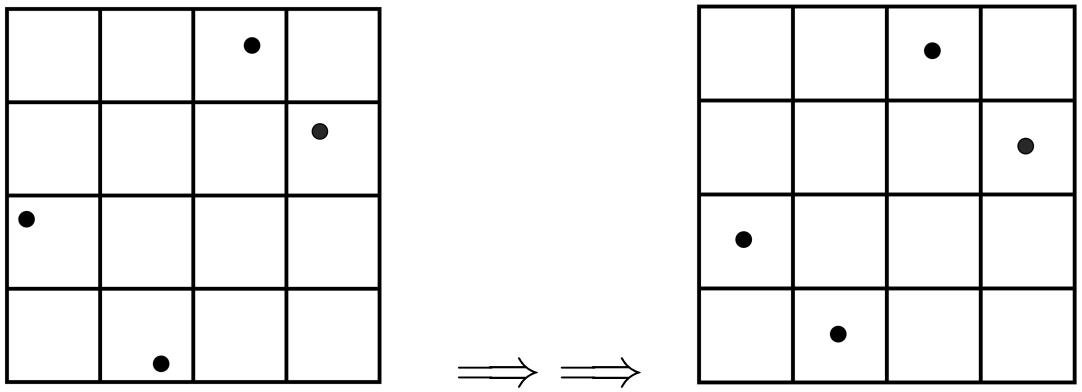


shifting y_2
coordinates



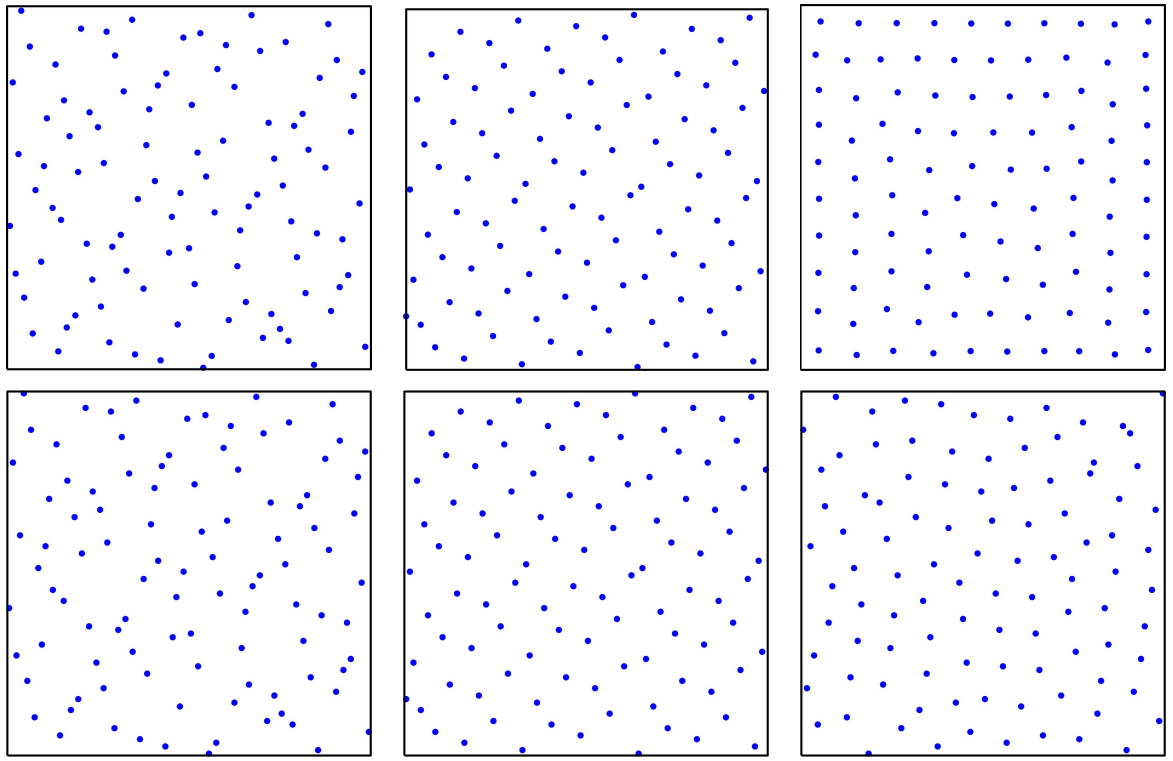
result of shifting –
now a
Latin sampling

- At this stage we could, if we so desire, move all points to the centers of their bin



  - we could also effect the centering as we step through the different coordinates

    - e.g., when we are dealing with y_7, we shift the points to the centers of the y_7 intevals



Top row: *100 point Halton, Hammersley, and ??? point sets*

Bottom row: *Latinized versions of the sample points in the top row*

- to the naked eye, the movement of points is almost not visible
  the LHS property is difficult to discern
- we also observe that Latinization does not do much damage to coverage

- What makes coincident or closely clustered projected points bad?

  – suppose we want to approximate an integral of a function $f(\boldsymbol{y})$ by averaging over M sample points, i.e.,

$$\int_{\Gamma} f(\boldsymbol{y})\, d\boldsymbol{y} \approx \frac{1}{M} \sum_{m=1}^{M} f(\boldsymbol{y}_m)$$

    - this is a very, very common use of the hyper-rectangle sampling schemes we have been talking about

  – it is known that the higher the star discrepancy of a set of points $\{\boldsymbol{y}_m\}_{m=1}^{M}$ the worse is the approximation to the integral

    (or at least for the estimates for the error in the approximation)

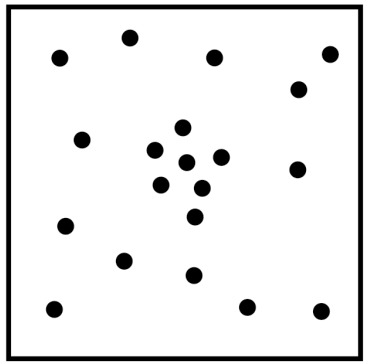
  – what is the star discrepancy of a set of points?

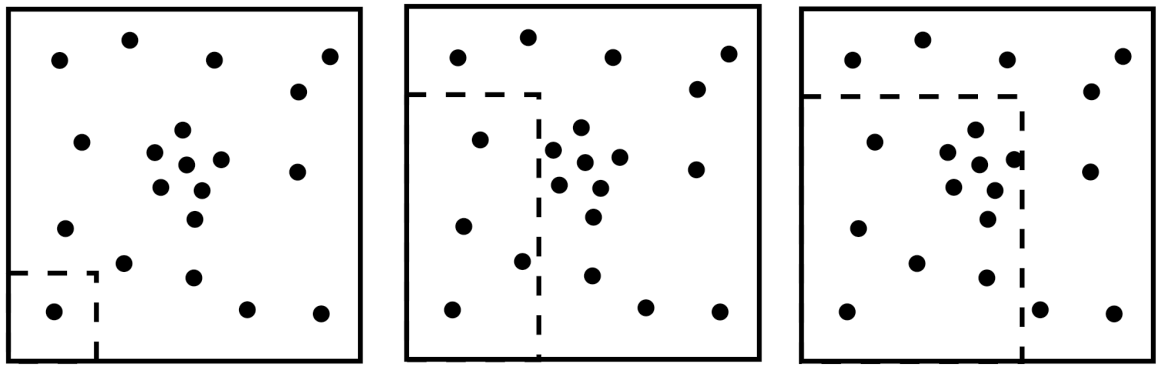
– the star discrepancy of a set of $\{\boldsymbol{y}_m\}_{m=1}^{M}$ of M points in the unit hypercube Γ is defined as

$$D_M^* = \max_{R \in \mathcal{R}} \underbrace{\left| \frac{\text{number of points in rectangle } R}{M} - \text{volume of rectangle } R \right|}_{D_R}$$

where $\mathcal{R}$ denotes the set of all rectangles $R \in \Gamma$ having one corner at the origin

– example: 20 points in the unit square





number of points in R	1	4	12
volume of R	$\frac{2}{32}$	$\frac{9}{32}$	$\frac{15}{32}$
D_R	0.125	0.08125	0.13125

- so that the star discrepancy of these 20 points is at least 0.13125

– it is proved that

for any point set $\{\boldsymbol{y}_m\}_{m=1}^{M}$ in Γ

and for any $\epsilon > 0$

there exists an infinitely differentiable function $f(\boldsymbol{y})$ such that

$$\left| \int_{\Gamma} f(\boldsymbol{y})\, d\boldsymbol{y} - \frac{1}{M} \sum_{m=1}^{M} f(\boldsymbol{y}_m) \right| \;\geq\; D_M^* - \epsilon$$

and

$$\left| \int_{\Gamma} f(\boldsymbol{y})\, d\boldsymbol{y} - \frac{1}{M} \sum_{m=1}^{Q} f(\boldsymbol{y}_m) \right| \;\leq\; (\text{variance of } f) \times D_M^*$$

– it is conjectured that for any finite set of M points where C_N depends on N but not on M

$$\Longrightarrow \quad D_M^* \geq C_N \frac{(\ln M)^{N-1}}{M}$$

– it is also conjectured that for any infinite set of M points where C_N depends on N but not on M

$$\Longrightarrow \quad D_M^* \geq C_N \frac{(\ln M)^{N}}{M}$$

- these conjectures are proven only for $N \leq 2$

- note that $\frac{(\ln M)^{N-1}}{M} \to \infty$ and $\frac{(\ln M)^{N}}{M} \to \infty$ as $N \to \infty$

– for some sampling schemes, upper bounds for the star discrepancy have been established

- for Halton sequences $D_M^* \leq C'_N \frac{(\ln M)^N}{M}$

so that $C_N \frac{(\ln M)^N}{M} \leq D_M^* \leq C'_N \frac{(\ln M)^N}{M}$

- for Hammersley sequences $C_N \frac{(\ln M)^{N-1}}{M} \leq D_M^* \leq M'_N \frac{(\ln M)^N}{M}$

– such bounds are often pessemistic

- for example, it has been proven that

for any $M \geq 1, \;\; N \geq 1, \;\;$ and $p \in (0, 1)$

a M-point Monte Carlo point set satisifies

$$D_M^* \leq 5.7 \sqrt{4.9 + \frac{\ln((1-p)^{-1})}{N}} \frac{N^{1/2}}{M^{1/2}}$$

with probability p

- note that

the bound increases with increasing p

in fact, the bound $\rightarrow \infty$ as $p \rightarrow 1$

the bound decreases with increasing M

if one chooses $p = 0.9$ then the

discrepancy bound is 15.30 for $M = 1$

and 12.65 for $M = 100$

SAMPLING + AVERAGING QUADRATURE RULES

- A very common use of the sampling schemes in hyper-rectangles such as the ones we have discussed is for the

  approximation of integrals by sampling+averaging quadrature rules

- We consider integrals of the form

$$\int_{\Gamma} G(\boldsymbol{y}) w(\boldsymbol{y})\, dy$$

  where

  Γ is an N-dimensional hypercube

  $w(\boldsymbol{y})$ is a PDF in a UQ setting or simply

  a weight function in more general settings

  - here, we assume it is a PDF $\rho(\boldsymbol{y})$

- We have in hand M points $\{\boldsymbol{y}_m\}_{m=1}^{M}$ in Γ obtained using one of the sampling schemes

- We can then take two approaches for approximating an integral by a sampling + averaging quadrature rule

$$\int_\Gamma G(\boldsymbol{y})\rho(\boldsymbol{y})\,dy \approx \frac{1}{M}\sum_{m=1}^{M} G(\boldsymbol{y}_m) \qquad \text{if one samples the PDF } \rho(\boldsymbol{y})$$

$$\int_\Gamma G(\boldsymbol{y})\rho(\boldsymbol{y})\,dy \approx \frac{1}{M}\sum_{m=1}^{M} \rho(\boldsymbol{y}_m)G(\boldsymbol{y}_m) \qquad \text{if one samples the points uniformly}$$

  - for the first choice, the quadrature weights are all $1/M$
    - they do not depend on the position of the points $\{\boldsymbol{y}_m\}_{m=1}^{M}$ or on any other geometric quantities
    - this is, of course, different from classical quadrature rules such as Simpson's rule which have different weights for different points
  - for the second choice, the quadrature weight for a sample point $\boldsymbol{y}_m$ is $\rho(\boldsymbol{y}_m)/M$

- of course, if Γ has edges of infinite or semi-infinite length, one has no choice, one has to sample non-uniformly according to the PDF $\rho(\boldsymbol{y})$

- The second approach seems simpler, but is wasteful
  - the density of points is the same
    in regions where $\rho(\cdot)$ is small (low probability)
    as it is
    in regions where $\rho(\cdot)$ is large (high probability)
  - unfortunately, many sampling methods
    can only be used to sample uniformly
    or are
    less efficient when sampling non-uniformly

Monte Carlo quadrature

- The canonical (and ubiquitous) sampling + averaging quadrature rule is, of course, the Monte Carlo rule

- Monte Carlo quadrature has two very great virtues (in addition to its simplicity)

  – virtue 1
  its convergence behavior is independent of the dimension N
  i.e., independent of the number of parameters

    - in this sense, Monte Carlo quadrature
      does not suffer from the curse of dimensionality

    - all other stochastic quadrature rules used in practice
      suffer from the curse

  – virtue 2
  its convergence behavior is not affected by the smoothness of the integrand

- Unfortunately, Monte Carlo quadrature has also has two great faults

  – Fault 1
  its convergence behavior is slow Error $= \mathcal{O}\left(\frac{\sigma}{\sqrt{M}}\right)$

  - so that if one wants to obtain one more digit of accuracy than that obtained using M samples, one has to obtain $100M$ samples

  – Fault 2
  its convergence behavior is not affected by the smoothness of the integrand

  - Monte Carlo cannot take advantage of smooth integrands to reduce the number of samples needed

  - other quadratures rules can take advantage of greater smoothness

- Because the complexity of other sampling methods depends on the parameter dimension N

  Monte Carlo will always beat other methods if N is large enough

  – so all other methods considered are merely efforts (which have been huge) directed at increasing the value of N at which Monte Carlo starts winning

Other sampling+averaging quadrature rules that are (not always) better than MC

- There have been many sampling+averaging quadrature rules proposed as replacements for Monte Carlo quadrature, including

  variance reduction Monte Carlo methods

  quasi-Monte Carlo methods

  stratified sampling

  Latin hypercube sampling and its many "improved" versions

  orthogonal arrays

  lattice rules

  importance sampling

  etc.

- In general, these "improved" rules have, in theory, improved rates of convergence, at least for not too large a value of N

  – a typical theoretical result is

$$\text{Error} = \mathcal{O}\left(\frac{(\ln M)^N}{M}\right) \qquad \Longleftarrow \text{ note the dependence on } N$$

  – for small N such as $N = 1, 2$ or 3, the $1/M$ term dominates

    - example: for $M = 100$ and $N = 3 \Rightarrow (\ln 100)^3/100 < 0.05$

      compare to MC: error $= \mathcal{O}\left(\frac{1}{\sqrt{M}}\right) \Rightarrow 1/\sqrt{100} = 0.1$

    for $M = 10,000$, QMC error $\approx$ 0.001, MC error $\approx$ 0.01

  – but, for even moderate N, the $(\ln M)^N$ term dominates

    - example: for $M = 100$ and $N = 10 \Rightarrow (\ln 100)^{10}/100 > 42,000$
    - this is a manifestation of the curse of dimensionality

  – this estimate is often (very) pessimistic but, still, the error is large even for moderate values of N and M

- How does one beat the curse?
  - especially in the setting of expensive functional evaluations
  - all than one can hope to do is to try to push the boundary,
    - e.g., develop methods that can treat a larger number of parameters
  - not surprisingly, doing something to beat the curse is a huge industry in the mathematical, natural, social, and engineering sciences

- Some of the approaches use towards doing something about the curse include
  - designing better sampling methods so that the number of expensive functional evaluations can be reduced
  - making better used of sampled input data so as to obtain accurate outputs while incurring lower costs
  - reducing the number of parameters by getting rid of parameters that are not influential
  - building surrogates models having outputs that are cheaper to obtain but are still useful

- We briefly consider one approach towards mitigating costs

- Later on we will also consider other point sets for quadrature rules that are also useful for interpolation

Analysis of variance (ANOVA) expansions

- An ANOVA-type expansion of a function $f(\boldsymbol{y})$ of N variables defined over the hypercube $\Gamma_N \in R^N$ is given by

$$f(\boldsymbol{y}) = f_0 + \sum_{i=1}^{N} f_i(y_i) + \sum_{i<j} f_{ij}(y_i, y_j) + \sum_{i<j<k} f_{ijk}(y_i, y_j, y_k) + \cdots + f_{12\cdots N}(\boldsymbol{y})$$

- If the the L^2 measure is used, we obtain the standard ANOVA expansion for which

$$f_0 = \int_{\Gamma_N} f(\boldsymbol{y})\, d\boldsymbol{y}$$

$$f_i(y_i) = \int_{K_{N-1}} f(\boldsymbol{y}) \prod_{k \neq i} dy_k - f_0 \qquad \text{for } i = 1, \ldots, N$$

$$f_{ij}(y_i, y_j) = \int_{K_{N-2}} f(\boldsymbol{y}) \prod_{k \notin \{i,j\}} dy_k - \sum_{n=1}^{N} f_k(y_k) - f_0 \quad \text{for } i, j = 1, \ldots, N, \quad i < j$$

etc.

- ANOVA-type expansions have many interesting properties
  - the terms in the expansions are all mutually orthogonal and define projections
  - the approximation properties of the expansion are essentially independent of the specific type of expansion

- Since the last term of an ANOVA-type expansion is itself a function of N-variables
  - it is obvious that any function has an ANOVA-type expansion
  - so it seems like we have not accomplished anything by writing a function in the form of an ANOVA expansion

- However

  suppose that the higher terms in an ANOVA-type expansion are "small"

  so that the given function $f(\boldsymbol{y})$ can be well approximated by the first few terms

– for example, suppose we have that

$$f(\boldsymbol{y}) \approx f_0 + \sum_{n=1}^{N} f_n(y_n) + \sum_{n'<n} f_{n'n}(y'_n, y_n)$$

with the sum of the neglected terms being small

– this would then mean that the given function $f(\boldsymbol{y})$ of N-variables is essentially the sum of a constant function, univariate functions, and bivariate functions

- Let's take a look at the simplest situation for which $N = 2$ and we have the ANOVA approximation

$$f(y_1, y_2) \approx f_0 + f_1(y_1) + f_2(y_2)$$

  - we use the ANOVA expansion to approximate the integral of $f(y_1, y_2)$ over the unit square $\Gamma_2 = [0, 1] \times [0, 1]$

$$\begin{aligned}
&\int_{\Gamma_2} f(y_1, y_2) dy_1 dy_2 \\
&\approx \int_0^1 \int_0^1 f_0 dy_1 dy_2 + \int_0^1 \int_0^1 f_1(y_1) dy_1 dy_2 + \int_0^1 \int_0^1 f_2(y_2) dy_1 dy_2 \\
&= f_0 + \int_0^1 f_1(y_1) dy_1 + \int_0^1 f_2(y_2) dy_2
\end{aligned}$$

    - this is great because now we only have to deal with one-dimensional integrals

– also, let's suppose we want to approximate the two-dimensional integral via sampling

$$\int_{\Gamma_2} f(y_1, y_2) dy_1 dy_2 \approx \frac{1}{M} \sum_{m=1}^{M} f(y_{1,m}, y_{2,m})$$

where $\{y_{1,m}, y_{2,m}\}_{m=1}^{M}$ are M sample points in Γ_2

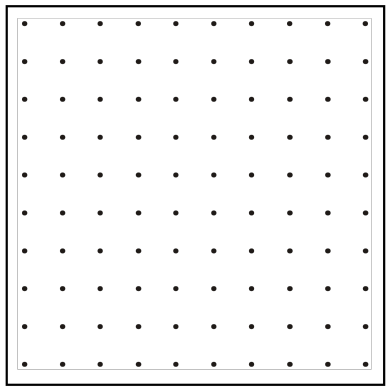
– if we further approximate by using the truncated ANOVA expansion of $f(y_1, y_2)$, we now have

$$\int_{\Gamma_2} f(y_1, y_2) dy_1 dy_2 \approx \frac{1}{M} \sum_{m=1}^{M} f_0 + \frac{1}{M} \sum_{m=1}^{M} f_1(y_{1,m}) + \frac{1}{M} \sum_{m=1}^{M} f_2(y_{2,m})$$

– doing one-dimensional sums is less expensive than doing two-dimensional sums

– but we have to be careful about what sampling scheme we use

– next, let's look at the worst case sampling scenario,
namely, Cartesian sampling over $\sqrt{M} \times \sqrt{M}$ points



as in the $M = 100$ case

and let's take a look at the $\frac{1}{M}\sum_{m=1}^{M} f_1(y_{1,m})$
term in the ANOVA expansion

- in the y_1 direction we only have a one-dimensional
integral with only $\sqrt{M}$ unique points so that

$$\frac{1}{M}\sum_{m=1}^{M} f_1(y_{1,m}) = \frac{1}{\sqrt{M}}\sum_{m=1}^{\sqrt{M}} f_1(y_{1,m})$$

so that we have a quadrature rule with only
$\sqrt{M}$ samples instead of M samples

- In the only slightly better scenario for which

  the projection of points onto lower dimensional

  faces form tight clusters

  we may still have an M point quadrature rule, but

  the quadrature points are poorly distributed

  so that the accuracy of the ANOVA terms is compromised

  – although we have used the context of approximating integrals

    it is clear that the above discussion extends

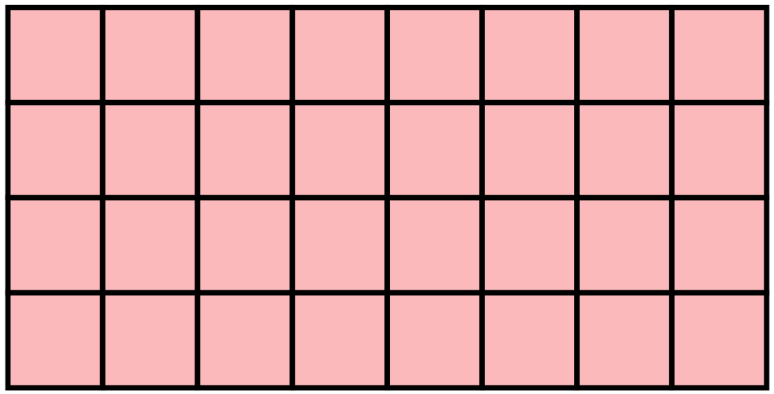
    to other approximation settings

- This discussion about the relation between point sampling and effective ANOVA expansions can be based on star discrepancies

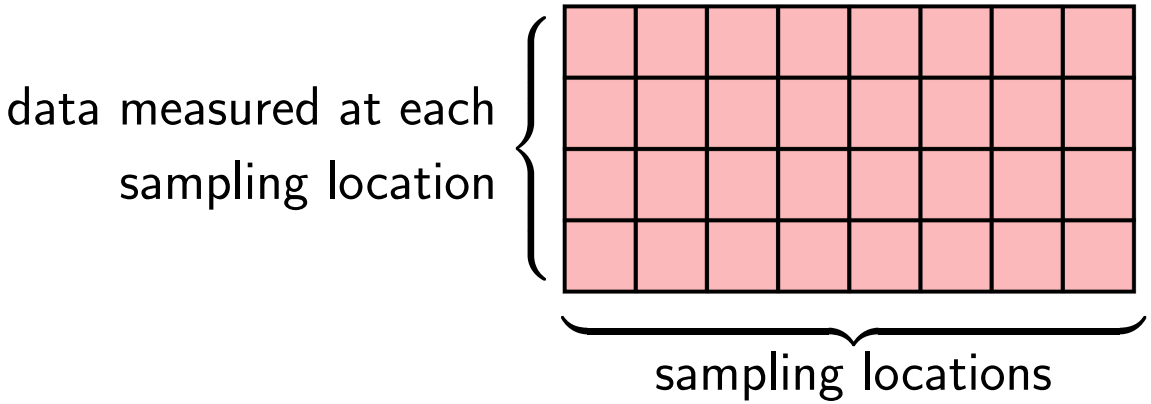
  – the clustering of projected points leads to large star discrepancies

UNDERDETERMINED LINEAR ALGEBRAIC SYSTEMS

- Underdetermined linear algebraic systems arise in myriad applications
  - such linear systems have matrices that have more columns than rows



  - for example, we could have the following situation

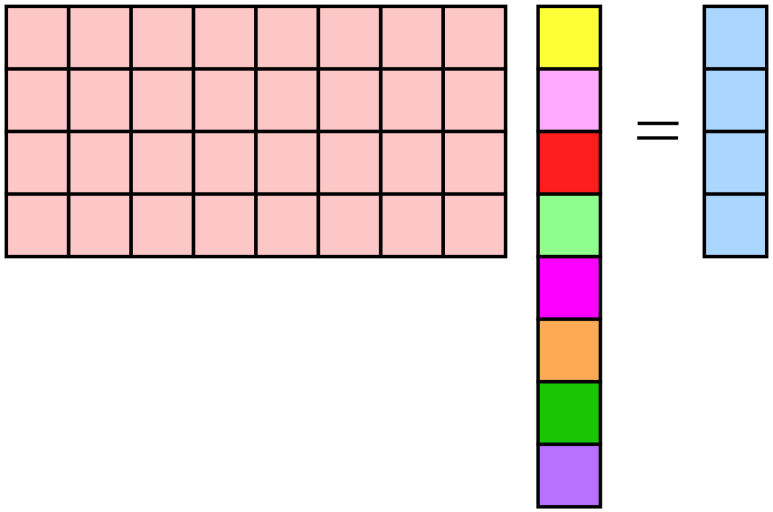


- Thus we consider linear algebraic systems

$$\mathbb{A}\boldsymbol{z} = \boldsymbol{b} \quad \text{where} \quad \begin{cases} \mathbb{A} \text{ is an } N \times M \text{ matrix with } M > N \text{ (more columns than rows)} \\ \boldsymbol{z} \text{ is an } M \times 1 \text{ column vector} \\ \boldsymbol{b} \text{ is an } N \times 1 \text{ column vector} \end{cases}$$

where $\mathbb{A}$ and $\boldsymbol{b}$ are a given matrix and a given right-hand side vector

  - the case $N = 4$ and $M = 8$ looks like

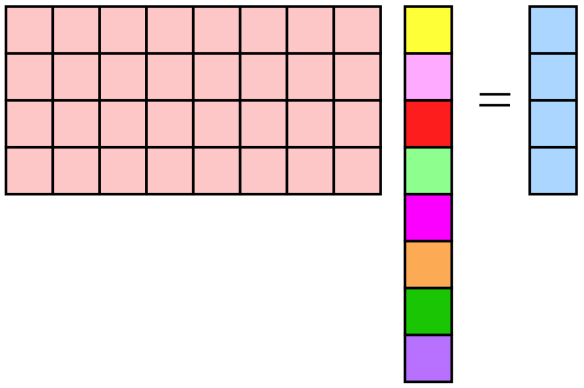


  - the task at hand is to determine the vector $\boldsymbol{z}$

– to keep things simple, we assume that the matrix $\mathbb{A}$ has full row rank, i.e.,
no row in the matrix can be expressed
as a linear combination of the other rows
- in this case
the linear system $\mathbb{A}\boldsymbol{z} = \boldsymbol{b}$ has an infinite number of solutions

– of course the columns cannot be linearly independent

- All sort of constraints can be placed on the solution $\boldsymbol{z}$ that enable one to find a unique solution of $\mathbb{A}\boldsymbol{z} = \boldsymbol{b}$
  – we consider two popular means for extracting a unique solution of the linear system



linear algebra class $\Longrightarrow$ find minimum Euclidean norm solution

compressed sensing $\Longrightarrow$ somehow find a solution vector $\boldsymbol{z}$ for which many of the components are zero $\Longrightarrow$ minimize sparsity-inducing norms

- Thus, we consider the two problems

$$\text{minimize } \|\boldsymbol{z}\|_2 \quad \text{subject to} \quad \mathbb{A}\boldsymbol{z} = \boldsymbol{b}$$

and

$$\text{minimize } \|\boldsymbol{z}\|_1 \quad \text{subject to} \quad \mathbb{A}\boldsymbol{z} = \boldsymbol{b}$$

where if $\boldsymbol{z}$ has dimension M, the 2-norm is defined as

$$\|\boldsymbol{z}\|_2 = \sqrt{z_1^2 + z_2^2 + \cdots + z_M^2}$$

and 1-norm is defined as

$$\|\boldsymbol{z}\|_1 = |z_1| + |z_2| + \cdots + |z_M|$$

- The difference between the outcomes obtained using the two norms can be explained in the simplest case, namely

$$N = 1 \text{ equation} \quad \text{in} \quad M = 2 \text{ unknowns}$$

- In the two-dimensions we have one equation with two unknowns z_1 and z_2

$$az_1 + bz_2 = c$$

where a, b, and c are given

  - clearly, this equation is underdetermined, i.e., there are an infinite number of pairs (z_1, z_2) that satisfy this equation

  - we examine two means for extracting a specific pair (z_1, z_2) that satisfies $az_1 + bz_2 = c$

- First, we minimize the 2-norm $\sqrt{z_1^2 + z_2^2}$ (the standard linear algebra approach)

$$\text{substitution} \qquad az_1 + bz_2 = c \implies z_1^2 + z_2^2 = z_1^2 + \frac{1}{b^2}(c - az_1)^2$$

$$\text{set first derivative} = 0 \qquad 2z_1 - \frac{a}{b^2}(c - az_1) = 0 \implies 2z_1(b^2 - a^2) = \frac{ac}{b^2}$$

$$\text{minimizing point} \qquad z_1 = \frac{ac}{2b^2(b^2 - a^2)} \neq 0 \qquad z_2 = \frac{bc}{2a^2(a^2 - b^2)} \neq 0$$

- in case it is of interest, the 2-norm of the minimizing solution is given by

$$\|\boldsymbol{z}\|_2 = \sqrt{z_1^2 + z_2^2} = \left(\frac{c^2}{4(a^2 - b^2)^2} \left(\frac{a}{b} + \frac{b}{a} \right) \right)^{1/2}$$

- Instead, we minimize the 1-norm $|z_1| + |z_2|$ (the compressed sensing approach)

$|z_1| + |z_2| \geq \max\{|z_1|, |z_2|\}$

if $|z_1| \geq |z_2| \implies |z_1| + |z_2|$ is minimized if $z_2 = 0$

$$\implies z_1 = \frac{c}{a} \implies |z_1| + |z_2| = \left|\frac{c}{a}\right|$$

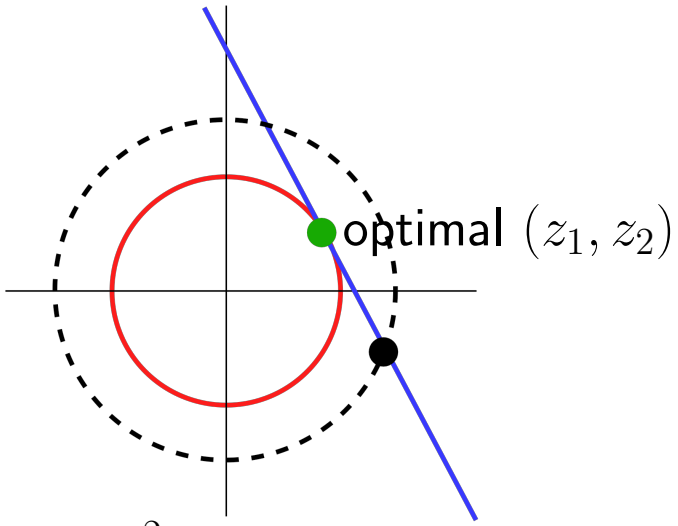
if $|z_2| \geq |z_1| \implies |z_1| + |z_2|$ is minimized if $z_1 = 0$

$$\implies z_2 = \frac{c}{b} \implies |z_1| + |z_2| = \left|\frac{c}{b}\right|$$

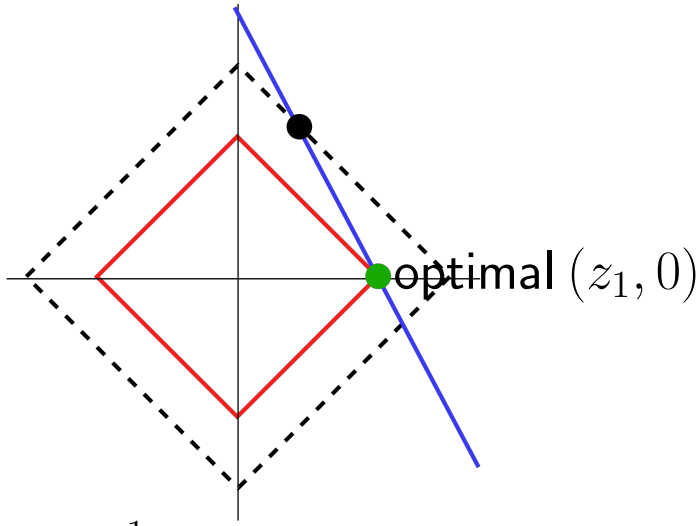
so, if $|a| > |b| \implies |z_1| + |z_2|$ is minimized if $z_2 = 0$ and $z_1 = \frac{c}{a}$

so, if $|b| > |a| \implies |z_1| + |z_2|$ is minimized if $z_1 = 0$ and $z_2 = \frac{c}{b}$

– a geometric interpretation of what is happening



ℓ^2-norm ball



ℓ^1-norm ball

blue line: $az_1 + bz_2 = c$

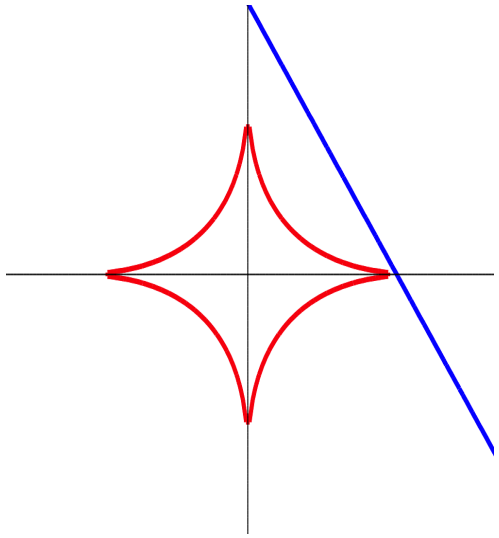
points on the circles all have the same $z_1^2 + z_2^2$
points on the diamonds all have the same $|z_1| + |z_2|$

in each case, the green dots depict the points
on the blue line that are closest to the origin

the black dots depict points on the blue line
that are further away from the origin

– any norm $\|\cdot\|_\bullet$ such that $0 \leq \bullet \leq 1$ is a sparsity-induced norms

- for $\|\cdot\|_{1/2}$, the picture looks like



- For the Euclidean norm, generalization to general M and $N > M$ is well known
  - one means for determining the $\boldsymbol{z}$ that minimizes the Euclidean norm of solutions of $A\boldsymbol{z} = \boldsymbol{b}$ is to solve

$$\mathbb{A}(\mathbb{A}^T\mathbb{A})^{-1}\boldsymbol{z} = \boldsymbol{f}$$

    - this applies for the full-rank case (i.e., the rank $r = M$)
    - for the rank defficient case ($r < M$)
      $(\mathbb{A}^T\mathbb{A})^{-1}$ is replaced by a pseudo-inverse of $\mathbb{A}^T\mathbb{A}$

  - another means for determining the optimal $\boldsymbol{z}$ is through the singular-value decomposition $\mathbb{A} = \mathbb{U}\boldsymbol{\Sigma}\mathbb{V}^T$
    - for both the full-rank and rank-defficient cases, the optimal $\boldsymbol{z}$ is

$$\boldsymbol{z} = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i^T\boldsymbol{b}}{\sigma_i}\boldsymbol{v}_i$$

    where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ denote the columns of $\mathbb{U}$ and $\mathbb{V}$, respectively, and σ_i denote the singular values

- Determining the $\boldsymbol{z}$ that minimizes the 1-norm of solutions of $A\boldsymbol{z} = \boldsymbol{b}$ is much less straighforward

GLOBAL POLYNOMIAL APPROXIMATIONS

ORTHOGONAL POLYNOMIALS[†]

AND

LAGRANGE INTERPOLATORY POLYNOMIALS

[†]In the uncertainty quantification community, orthogonal polynomial approximation is very unfortunately very often referred to as "polynomial chaos"

- Polynomials are among the most studied subjects in mathematics

- Great mathematicians (mostly from the late 18th century and the first half of the 19th century) made fundamental contribution such as

  defining polynomials that have a specific and useful structure

  and also

  rigorously studying their approximation properties

- For example, one of the most (justifiable) celebrated theorems in all of mathematics is the Weierstrass approximation theorem

  Suppose that $f(y)$ *is a continuous function defined on the interval* $[a, b]$

  Then, for every $\epsilon > 0$*, there exists a polynomial* $p(y)$ *such that for all* $y \in [a, b]$, $|f(y) - p(y)| < \epsilon$

Global polynomial approximations

- Let P_M denote the set of all polynomials of degree less than or equal to M
  - let $\{\Theta_m(y)\}_{m=0}^{M}$ denote a basis for P_M
  - of course, there are numerous possible bases in use
    - the simplest is the monomial basis for which $\Theta_m(y) = y^m$ for $m = 0, 1, \ldots, M$
  - then all polynomials $p_M(y) \in P_M$
    i.e., all polynomials of degree less that or equal to M
    can be expressed in the form

$$p_M(y) = \sum_{m=0}^{M} C_m y^m \quad \text{for any set of constants } \{C_m\}_{m=0}^{M}$$

  - later, we consider other, more practical, bases

- Now consider the multivariate case for which we have an

  N-vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$ of variables

  – in general, for each y_n, one could use polynomials of degree M_n and a basis $\{\Theta_{n,m}(y_n)\}_{m=1}^{M_n}$

    - for the sake of simplicity, we assume that $M_n = M$ for all n

    - there are often good reasons for sometimes choosing different degree polynomials for each y_n

- Denote by $P_{n,M}$ the set of all polynomials in y_n of degree less than or equal to M

  – let $\{\Theta_{n,m}(y_n)\}_{m=0}^{M}$ denote a basis for $P_{n,M}$

- Let $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_N)$ denote a multi-index

  $\Longrightarrow$ an N-vector whose components are non-negative integers

  – then, a set of multi-dimensional polynomials $P_{N,M}$ contains polynomials $p_{N,M}$ which are a sum of terms having the form

  $$y_1^{\mu_1} y_2^{\mu_2} \cdots y_N^{\mu_N}$$

  – let $|\boldsymbol{\mu}| = \sum_{n=1}^{N} \mu_n$

  – choosing different constraints on $\boldsymbol{\mu}$ lead to different types of polynomials

- Assumption: We have in hand a function $f(\boldsymbol{y})$ that is smooth with respect to an N-vector $\boldsymbol{y}$ of variables

  smooth $\Longleftrightarrow$ $f(\boldsymbol{y})$ possesses a "sufficient" number of derivatives

  Consequence: $f(\boldsymbol{y})$ can be well-approximated by by multivariate global polynomials

  – what does "sufficient" mean?

  - to achieve full accuracy

    $f(\boldsymbol{y})$ has to possess $M+1$ derivatives

    $\longrightarrow$ the larger M is, the better the accuracy

  - however

    if $f(\boldsymbol{y})$ possess $\widetilde{M}+1$ derivatives, where $0 \leq \widetilde{M} \leq M$

    $\longrightarrow$ one only achieves the accuracy of a polynomial of degree $\widetilde{M} < M$

  $\Longrightarrow$ the smoothness of $f(\boldsymbol{y})$ matters!

Total degree polynomials

- For a given integer $M \geq 0$, let $\{\psi_k^{td}(\boldsymbol{y})\}_{k=1}^{K_{td}}$ denote the set of distinct multivariate polynomials such that

$$\left\{\psi_k^{td}(\boldsymbol{y})\right\}_{k=1}^{K_{td}} = \left\{\prod_{n=1}^{N} \Theta_{n,m_n}(y_n)\right\}$$

where $\Theta_{n,m_n}(y_n) \in P_{n,M}$ and $|\boldsymbol{\mu}| = \sum_{n=1}^{N} \mu_n \leq M$

  - the highest degree term in any of the multivariate polynomials is M
    - thus, if $N = 2$ and $M = 2$, we have $|\boldsymbol{\mu}| = \mu_1 + \mu_2 \leq 2$ so that we have terms such as y_1^2 and $y_1 y_2$ but not terms like $y_1^2 y_2$
  - then, polynomials such that $|\boldsymbol{\mu}| \leq M$ have the form

$$p_{td}(\boldsymbol{y}) = p_{td}(y_1, y_2, \ldots, y_N) = \sum_{k=1}^{K_{td}} C_k \psi_k^{td}(\boldsymbol{y})$$

for any set of constants $\{C_k\}_{k=1}^{K_{td}}$

– the degrees of freedom, i.e., the number of terms,
in such polynomials is given by

$$K_{td} = \frac{(N+M)!}{N!\,M!}$$

where $N =$ dimension of the vector $\boldsymbol{y}$

$M =$ maximal degree of any of the
N-dimensional global poloynomials used

– we can choose any set of multivariate polynomials $\left\{\psi_k^{td}(\boldsymbol{y})\right\}_{k=1}^{K_{td}}$
having total degree $\leq M$ and that are linearly independent†

- they do not have to be monomials

- in fact, one should not use monomial bases

– for example, if $N = 2$ and $M = 3$, we have

$$|\boldsymbol{\mu}| = \mu_1 + \mu_2 \leq M = 3$$

and

$$K_{td} = \frac{(N+M)!}{N!\,M!} = \frac{(2+3)!}{2!\,3!} = 10$$

and we have the set of 10 distinct basis functions $\left\{\psi_1(y_1, y_2)\,,\,\ldots\,,\,\psi_{10}(y_1, y_2)\right\}$ such that

$$\text{span}\left\{\begin{array}{c}\psi_1^{td}(y_1,y_2)\\ \psi_2^{td}(y_1,y_2)\\ \psi_3^{td}(y_1,y_2)\\ \psi_4^{td}(y_1,y_2)\\ \psi_5^{td}(y_1,y_2)\\ \psi_6^{td}(y_1,y_2)\\ \psi_7^{td}y_1,y_2)\\ \psi_8^{td}(y_1,y_2)\\ \psi_9^{td}(y_1,y_2)\\ \psi_{10}^{td}(y_1,y_2)\end{array}\right\} = \text{span}\left\{\begin{array}{c}\Theta_{1,0}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,2}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,2}(y_2)\\ \Theta_{1,3}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,3}(y_2)\end{array}\right\} = \text{span}\left\{\begin{array}{c}1\\ y_1\\ y_2\\ y_1y_2\\ y_1^2\\ y_2^2\\ y_1^2y_2\\ y_1y_2^2\\ y_1^3\\ y_2^3\end{array}\right\}$$

$$\Downarrow \qquad\qquad \Downarrow$$

this does not imply that, e.g., $\Theta_{1,2}(y_1)\,\Theta_{2,1}(y_2) = y_1^2y_2$

Tensor product polynomials

- Alternately, one could use a tensor product basis

$$\left\{\psi_k^{tp}(\boldsymbol{y})\right\}_{k=1}^{K_{tp}} = \left\{\prod_{n=1}^{N}\Theta_{n,m}(y_n)\right\}$$

where

$$\Theta_{n,m}(y_n) \in P_{n,M} \qquad \text{and} \qquad \mu_n \leq M \text{ for all } n$$

  - now the highest degree term in any of the polynomials is M in each y_n

    - thus, if $M = 2$, we not only have terms like y_1^2 and $y_1 y_2$, but we now also have terms like $y_1^2 y_2^1$ and $y_2^2 y_2^2$

– we still have polynomials having the form

$$p_{tp}(\boldsymbol{y}) = p_{tp}(y_1, y_2, \ldots, y_N) = \sum_{k=1}^{K_{tp}} C_k \psi_k^{tp}(\boldsymbol{y})$$

for any set of constants $\{C_k\}_{k=1}^{K_{tp}}$, but now the degrees of freedom, i.e, the number of terms, is

$$K_{tp} = (M+1)^N > K_{td}$$

where $N =$ dimension of the vector $\boldsymbol{y}$

$M =$ maximal degree in any variable y_n of any of the N-dimensional global poloynomials used

– for example, if $N = 2$ and $M = 3$, we now have

$$K_{tp} = (M+1)^N = (3+1)^2 = 16$$

and we now have the set of 16 distinct basis functions $\{\psi_1(y_1, y_2), \ldots, \psi_{16}(y_1, y_2)\}$ such that

$$\text{span}\left\{\begin{array}{c}\psi_1^{tp}(y_1,y_2)\\ \psi_2^{tp}(y_1,y_2)\\ \psi_3^{tp}y_1,y_2)\\ \psi_4^{tp}(y_1,y_2)\\ \psi_5^{tp}(y_1,y_2)\\ \psi_6^{tp}(y_1,y_2)\\ \psi_7^{tp}(y_1,y_2)\\ \psi_8^{tp}(y_1,y_2)\\ \psi_9^{tp}(y_1,y_2)\\ \psi_{10}^{tp}(y_1,y_2)\\ \psi_{11}^{tp}(y_1,y_2)\\ \psi_{12}^{tp}(y_1,y_2)\\ \psi_{13}^{tp}(y_1,y_2)\\ \psi_{14}^{tp}(y_1,y_2)\\ \psi_{15}^{tp}(y_1,y_2)\\ \psi_{16}^{tp}(y_1,y_2)\end{array}\right\} = \text{span}\left\{\begin{array}{c}\Theta_{1,0}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,2}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,2}(y_2)\\ \Theta_{1,3}(y_1)\,\Theta_{2,0}(y_2)\\ \Theta_{1,0}(y_1)\,\Theta_{2,3}(y_2)\\ \Theta_{1,1}(y_1)\,\Theta_{2,3}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,3}(y_2)\\ \Theta_{1,3}(y_1)\,\Theta_{2,3}(y_2)\\ \Theta_{1,2}(y_1)\,\Theta_{2,2}(y_2)\\ \Theta_{1,3}(y_1)\,\Theta_{2,1}(y_2)\\ \Theta_{1,3}(y_1)\,\Theta_{2,2}(y_2)\end{array}\right\} = \text{span}\left\{\begin{array}{c}1\\ y_1\\ y_2\\ y_1y_2\\ y_1^2\\ y_2^2\\ y_1^2y_2\\ y_1y_2^2\\ y_1^3\\ y_2^3\\ y_1y_2^3\\ y_1^2y_2^3\\ y_1^3y_2^3\\ y_1^2y_2^2\\ y_1^3y_2^1\\ y_1^3y_2^2\end{array}\right\}$$

– but now we fall prey to the the dreaded curse of dimensionality

Global polynomial degrees of freedom			
$N =$ dimension of $\boldsymbol{y}$	$M =$ maximal degree of polynomials	$K =$ number of terms in the polynomial	
		using total degree bases	using tensor product bases
3	3	20	64
	5	56	216
5	3	56	1,024
	5	252	7,776
10	3	286	1,048,576
	5	3,003	60,046,176
20	3	1,771	$> 1\times10^{12}$
	5	53,130	$> 3\times10^{15}$
100	3	176,851	$> 1\times10^{60}$
	5	96,560,646	$> 6\times10^{77}$

- As N and M increases, it seems that
  - using total degree polynomials is a bad idea
  - using tensor product polynomials is a really bad idea

– tensor product polynomial approximation involves
more terms compared to total degree approximation

- unfortunately, the additional terms do not improve the
convergence rate (as M increases) of the approximation

- although the convergence rates are the same

tensor product polynomial approximation could
result in smaller absolute errors

but, of course, incurring (possibly much) larger costs

$\Longrightarrow$ you cannot do better than total degree approximation!

• Before proceeding further, we mention some global
approximations that we do not consider in detail

- Of course, when thinking about global approximations of a function, the first thing that comes to mind for anyone who has taken a calculus course is Taylor polynomial approximation

  – the Taylor polynomial approximation of a given function $f(\boldsymbol{y})$ with respect to $\widehat{\boldsymbol{y}}$ is

$$f(\boldsymbol{y}) \approx \sum_{0<|\boldsymbol{\mu}|\leq M} t_{\boldsymbol{\mu}}(\boldsymbol{y}-\widehat{\boldsymbol{y}})^{\boldsymbol{\mu}}$$

  where

$$(y^{\boldsymbol{\mu}}-\widehat{y})^{\boldsymbol{\mu}} = \prod_{n=1}^{N}(y_n-\widehat{y}_n)^{\mu_n} \quad \text{and} \quad t_{\boldsymbol{\mu}} = \frac{1}{\mu_1!\mu_2!\cdots\mu_N!}\partial_{\boldsymbol{y}}^{\boldsymbol{\mu}} f(\widehat{\boldsymbol{y}})$$

  – note that Taylor polynomials are expressed in terms of monomials, i.e., each term is a power of $(y_n-\widehat{y}_n)$

- Further afield are global approximations of a function via non-polynomial bases
  - Fourier series are the prime example
  - we do not consider such approximations, but note that they are useful in many settings
  - here, we consider

    global orthogonal polynomial approximation

    and

    global Lagrange polynomial interpolation

GLOBAL ORTHONORMAL POLYNOMIALS

- For $n = 1, \ldots, N$, let $\{\Theta^{go}_{n,m}(y_n)\}_{m=0}^{M}$ denote the set of polynomials in $\mathbb{R}$ of degree less than or equal to M that are orthonormal with respect to an

  interval $[a_n, b_n]$ and a weight function $\phi_n(y_n)$

  - we have that, for $n = 1, \ldots, N$,

$$\int_{a_n}^{b_n} \Theta^{go}_{n,m}(y_n)\Theta^{go}_{n,m'}(y_n)\phi_n(y_n)\, dy_n = \delta_{mm'} \qquad \text{for } m, m' \in \{0, \ldots, M\}$$

  - note that the cases $b_n = -\infty$ or $a_n = +\infty$ or both are allowable

  - note that the set $\{\Theta^{go}_{n,m}(y_n)\}_{m=0}^{M}$ is hierarchical in the sense that

$$\text{degree}(\Theta^{go}_{n,m}) = m$$

$$\Longrightarrow \text{ the polynomials in } \{\Theta^{go}_{n,m}(y_n)\}_{m=0}^{M} \text{ are linearly independent}$$

- Let

$$\psi_k^{go}(\boldsymbol{y}) = \prod_{n=1}^{N} \Theta_{n,m_n}^{go}(y_n) \quad \text{for all } m_n \in \{0,\ldots,M\} \text{ such that } \sum_{n=1}^{N} m_n \leq M$$

  - we then have that $k \in \left\{1,\ldots,K_{go_{td}} = \dfrac{(N+M)!}{N!M!}\right\}$
  - for example, if $M=1$ and $N=3$ we have the $K_{go_{td}}=4$ basis functions†

$$\Theta_{1,0}^{go}(y_1)\Theta_{2,0}^{go}(y_2)\Theta_{3,0}^{go}(y_3)$$
$$\Theta_{1,1}^{go}(y_1)\Theta_{2,0}^{go}(y_2)\Theta_{3,0}^{go}(y_3) \quad \Theta_{1,0}^{go}(y_1)\Theta_{2,1}^{go}(y_2)\Theta_{3,0}^{go}(y_3) \quad \Theta_{1,0}^{go}(y_1)\Theta_{2,0}^{go}(y_2)\Theta_{3,1}^{go}(y_3)$$

  whereas for if $M=2$ and $N=3$ we have the $K_{go_{td}}=10$ basis functions (suppressing noting the explicit dependences on the $\boldsymbol{y}_n$'s)

$$\Theta_{1,0}^{go}\Theta_{2,0}^{go}\Theta_{3,0}^{go}$$
$$\Theta_{1,1}^{go}\Theta_{2,0}^{go}\Theta_{3,0}^{go} \quad \Theta_{1,0}^{go}\Theta_{2,1}^{go}\Theta_{3,0}^{go} \quad \Theta_{1,0}^{go}\Theta_{2,0}^{go}\Theta_{3,1}^{go}$$
$$\Theta_{1,2}^{go}\Theta_{2,0}^{go}\Theta_{3,0}^{go} \quad \Theta_{1,1}^{go}\Theta_{2,1}^{go}\Theta_{3,0}^{go} \quad \Theta_{1,1}^{go}\Theta_{2,0}^{go}\Theta_{3,1}^{go} \quad \Theta_{1,0}^{go}\Theta_{2,2}^{go}\Theta_{3,0}^{go} \quad \Theta_{1,0}^{go}\Theta_{2,1}^{go}\Theta_{3,1}^{go} \quad \Theta_{1,0}^{go}\Theta_{2,0}^{go}\Theta_{3,2}^{go}$$

† It is convenient to write the N-dimensional polynomials so that each row contains the polynomials of the same total degree $\sum_{n=1}^{N} m_n$; thus the first row contains all possible products of the N one-dimensional polynomials of total degree 0, the second row has total degree 1, etc.

- We see that the functions $\psi_k^{go}(\boldsymbol{y})$ are products of one-dimensional orthonormal polynomials and have total degree less than or equal to M
  - we then have that, with $\Gamma = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_N, b_N]$,

$$\int_\Gamma \psi_k^{go}(\boldsymbol{y}) \psi_{k'}^{go}(\boldsymbol{y}) \phi(\boldsymbol{y})\, d\boldsymbol{y} = \int_\Gamma \psi_k^{go}(\boldsymbol{y}) \psi_{k'}^{go}(\boldsymbol{y}) \Pi_{n=1}^N \phi_n(y_n)\, d\boldsymbol{y}$$

$$= \prod_{n=1}^N \int_{a_n}^{b_n} \Theta_{n,m_n}^{go}(y_n) \Theta_{n,m_n'}^{go}(y_n) \phi_n(y_n)\, dy_n = \delta_{kk'}$$

  - note that we need to write $\phi(\boldsymbol{y}) = \prod_{n=1}^N \phi_n(y_n)$, i.e., as a product as well, so that we know what $\Theta_{n,m}^{go}(\cdot)$ is orthonormal with respect to
  - thus, we are restricted to independent random variables and to domains Γ that are (possibly infinite) hyper-rectangles

- It is easy to see that the set $\{\psi_k^{go}\}_{k=1}^{K_{go_{td}}}$ of N-dimensional polynomials is a basis for the complete polynomial space of degree M, i.e.,

$$\text{span}\{\psi_k^{go}\}_{k=1}^{K_{go_{td}}} = \text{all polynomials of total degree} \leq M$$

- We now have global polynomials of the form

$$p_{go}(\boldsymbol{y}) = \sum_{k=1}^{K_{go}} C_k \psi_k^{go}(\boldsymbol{y})$$

  – for a given function $f(\boldsymbol{y})$, the coefficients $\{C_k\}_{k=1}^{K_{go}}$ are determined by setting

$$\begin{aligned}
\int_\Gamma f(\boldsymbol{y})\psi_{k'}^{go}(\boldsymbol{y})d\boldsymbol{y} &= \int_\Gamma p_{go}(\boldsymbol{y})\psi_k^{go}(\boldsymbol{y})d\boldsymbol{y} \\
&= \int_\Gamma \Big(\sum_{k=1}^{K_{go}} C_k \psi_k^{go}(\boldsymbol{y})\Big)\psi_{k'}^{go}(\boldsymbol{y})d\boldsymbol{y} \\
&= \sum_{k=1}^{K_{go}} C_k \int_\Gamma \psi_k^{go}(\boldsymbol{y})\psi_{k'}^{go}(\boldsymbol{y})d\boldsymbol{y} = \sum_{k=1}^{K_{go}} C_k \delta_{kk'} = C_{k'}
\end{aligned}$$

$\Longrightarrow$ the orthonormal polynomial approximation of $f(\boldsymbol{y})$ is given by

$$f(\boldsymbol{y}) \approx \sum_{k=1}^{K_{go}} C_k \psi_k^{go}(\boldsymbol{y}) \quad \text{with} \quad C_k = \int_\Gamma f(\boldsymbol{y})\psi_k^{go}(\boldsymbol{y})d\boldsymbol{y}$$

- Note that one has to compute the integrals

$$C_k = \int_\Gamma f(\boldsymbol{y}) \psi_k^{go}(\boldsymbol{y}) d\boldsymbol{y} \quad \text{for } k = 0, \ldots, K_{go}$$

  - in general, one has to approximate the integrals using a quadrature rule

$$C_k = \int_\Gamma f(\boldsymbol{y}) \psi_k^{go}(\boldsymbol{y}) d\boldsymbol{y} \approx \sum_{q=1}^{Q} W_q f(\boldsymbol{y}_q) \psi_k^{go}(\boldsymbol{y}_q) \quad \text{for } k = 0, \ldots, K_{go}$$

  - doing so, one has to evaluate $f(\boldsymbol{y})$ at the possibly many quadrature points

- Why does it make sense to do all of this?
  - suppose that
    1. evaluating $f(\boldsymbol{y})$ is an expensive proposition
    2. you want to evaluate $f(\boldsymbol{y})$ at zillions of $\boldsymbol{y}$ values
  - to approximate all C_k, you have to evaluate $f(\boldsymbol{y})$ Q times
  - after you have done that, you have in hand the approximation

$$p_{go}(\boldsymbol{y}) = \sum_{k=1}^{K_{go}} C_k \psi_k^{go}(\boldsymbol{y}) \approx f(\boldsymbol{y})$$

  - then,

    instead of evaluating $f(\boldsymbol{y})$ at zillions of new $\boldsymbol{y}$ values
    you evaluate $p_{go}(\boldsymbol{y})$ at those values

  - if evaluating $p_{go}(\boldsymbol{y})$ is much cheaper than evaluating $f(\boldsymbol{y})$, you win big

- If one instead uses tensor product polynomials, as we have seen, such a choice leads to hugely more costly approximations†

---

†The tensor product basis is given by

$$\psi_k^{go}(\boldsymbol{y}) = \prod_{n=1}^{N} \Theta_{n,m_n}^{go}(y_n) \quad \text{for all } m_n \in \{0,\dots,M\} \text{ such that } m_n \leq M$$

in this case, span$\{\psi_k^{go}\}_{k=1}^{K}$ is the tensor product space of polynomials such that the degree in any coordinate y_n is less than or equal to M; if we do this, we end up with $K = (M+1)^N$ basis functions; for example, if $M = 1$ and $N = 3$, we have the 8 polynomials (the 4 we had before plus 4 additional ones)

$$\begin{array}{ccc} & \Theta_{1,0}^{go}\Theta_{2,0}^{go}\Theta_{3,0}^{go} & \\ \Theta_{1,1}^{go}\Theta_{2,0}^{go}\Theta_{3,0}^{go} & \Theta_{1,0}^{go}\Theta_{2,1}^{go}\Theta_{3,0}^{go} & \Theta_{1,0}^{go}\Theta_{2,0}^{go}\Theta_{3,1}^{go} \\ \Theta_{1,1}^{go}\Theta_{2,1}^{go}\Theta_{3,0}^{go} & \Theta_{1,1}^{go}\Theta_{2,0}^{go}\Theta_{3,1}^{go} & \Theta_{1,0}^{go}\Theta_{2,1}^{go}\Theta_{3,1}^{go} \\ & \Theta_{1,1}^{go}\Theta_{2,1}^{go}\Theta_{3,1}^{go} & \end{array}$$

for $N > 1$ and $M > 0$ we have that $(M+1)^N > \frac{(N+M)!}{N!M!}$; for a moderate number of parameters or for a moderately high degree polynomial, we in fact have that $(M+1)^N \gg \frac{(N+M)!}{N!M!}$; for example,

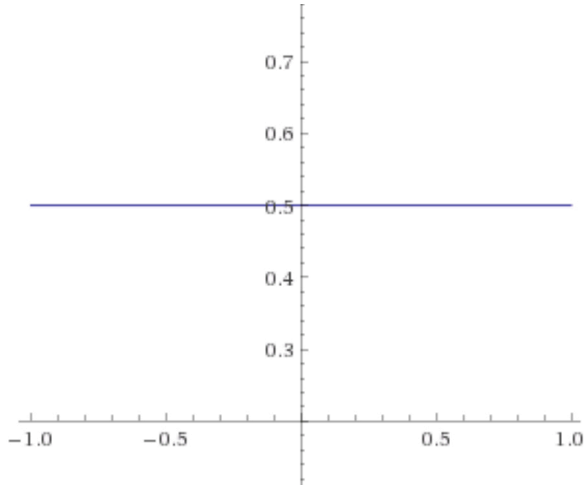
$$\text{if } M = 6 \text{ and } N = 3 \Longrightarrow (N+M)!/(N!M!) = 84 \text{ and } (M+1)^N = 343$$
$$\text{if } M = 4 \text{ and } N = 5 \Longrightarrow (N+M)!/(N!M!) = 126 \text{ and } (M+1)^N = 3125$$
$$\text{if } M = 2 \text{ and } N = 7 \Longrightarrow (N+M)!/(N!M!) = 36 \text{ and } (M+1)^N = 2187$$

the disparity gets worse as, say, N increases; for example,

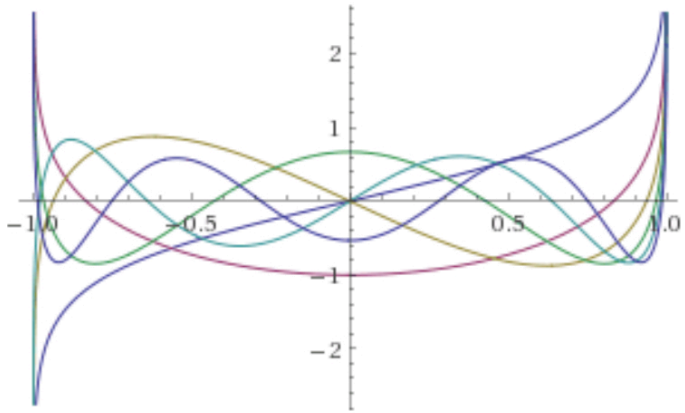
$$\text{if } M = 2 \text{ and } N = 10 \Longrightarrow (N+M)!/(N!M!) = 66 \text{ and } (M+1)^N = 59059$$

on the other hand, since the accuracy, i.e., the rate of convergence of global polynomial approximation, is determined by the degree of the largest complete polynomial space contained in the approximate space, for the same M, the accuracy obtained using a tensor product space is the same as that obtained using a complete polynomial space; as a result, by using the latter one can obtain the same accuracy with substantially fewer degrees of freedom

- Some orthonormal bases
  - the univariate Legendre polynomials are orthogonal polynomials over the interval $[-1, 1]$ with respect to the weight function $w(y) = \frac{1}{2}$



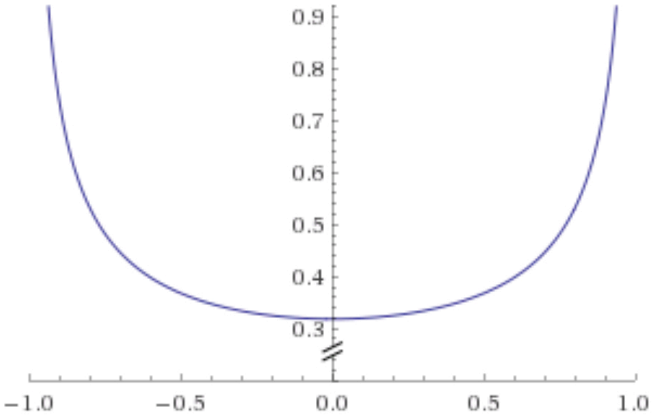
weight



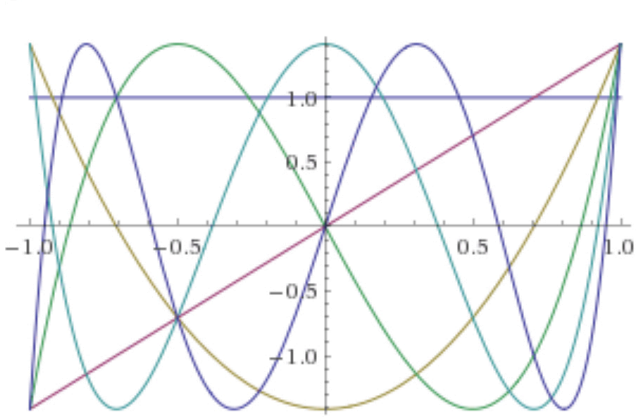
polynomials

- the multivariate Legendre polynomials are orthogonal polynomials over the domain $[-1, 1]^N$ with respect to the weight function $w(\boldsymbol{y}) = (1/2)^N$

– the univariate Chebyshev polynomials are orthogonal polynomials over the interval $[-1, 1]$ with respect to the weight function $w(y) = \pi^{-1}(1 - y^2)^{-1/2}$



weight



polynomials

- the multivariate Chebyshev polynomials are orthogonal polynomials over the domain $[-1, 1]^N$ with respect to the weight function $w(\boldsymbol{y}) = \pi^{-N} \prod_{n=1}^{N} \frac{1}{(1-y_n^2)^{-1/2}}$

– some other univariate orthogonal polynomials from which one can define multivariate orthogonal polynomials

weight function	support	polynomial
$\frac{1}{\sqrt{2\pi}}e^{-y^2/2}$	$(-\infty, \infty)$	Hermite $H_k(y)$
$\frac{1}{2}$	$[-1, 1]$	Legendre $P_n(y)$
e^{-y}	$[0, \infty)$	Laguerre $L_n(y)$
⋮	⋮	⋮
⋮	⋮	⋮
hundreds - even infinitely - many more		

– of course, one does not have to use the same type of orthogonal polynomials for all y_n

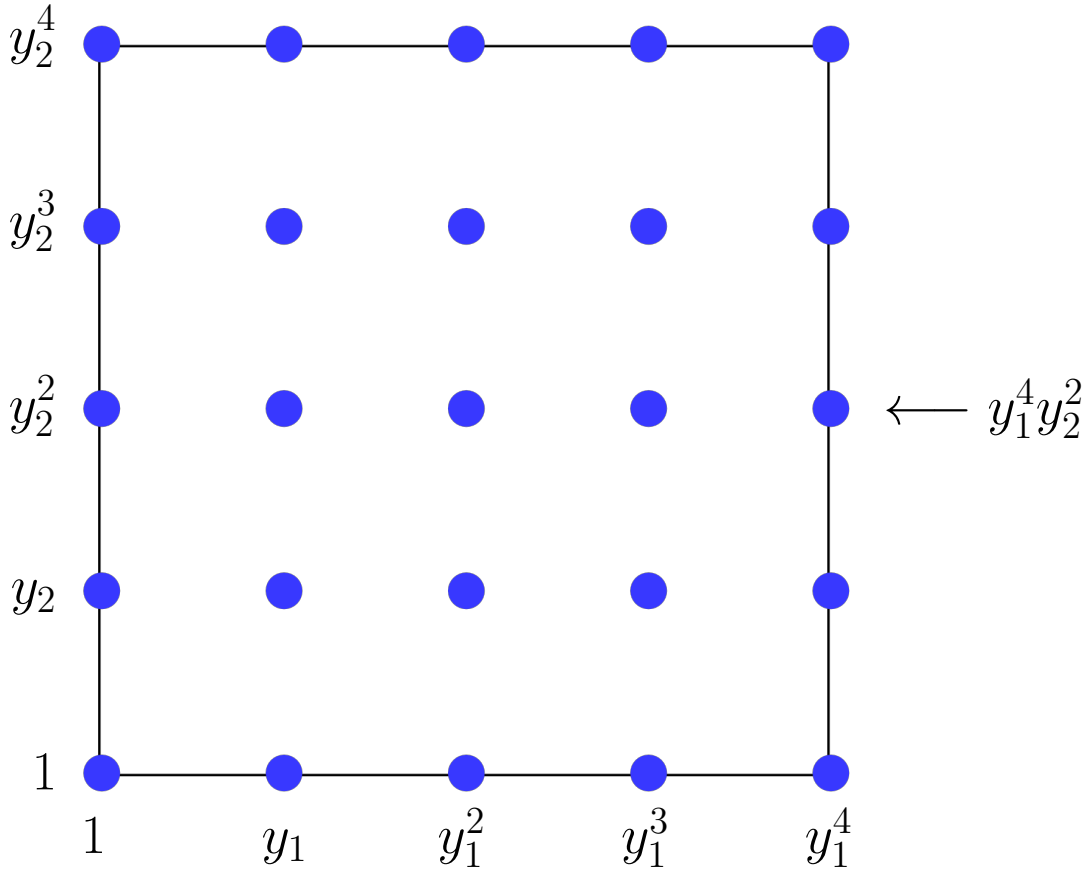
- for example, one can use univariate Chebyshev orthogonal polynomials for y_1 and univariate Hermite orthogonal polynomials for y_2 which result in the

domain $[-1, 1] \times (-\infty, \infty)$

and the weight function $\phi(y_1, y_2) = \left(\frac{1}{\pi(1-y_1^2)^{-1/2}}\right)\left(\frac{1}{\sqrt{2\pi}}e^{-y_2^2/2}\right)$

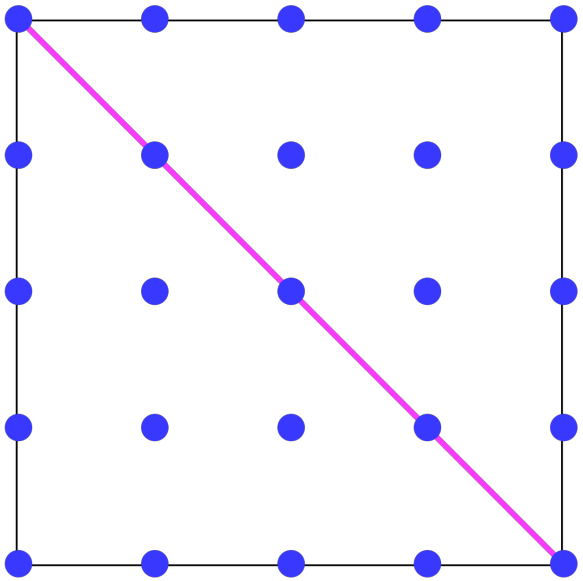
- What is a good polynomial approximation subspace?

- Several choices for the multi-index $\boldsymbol{\mu}$ have been proposed
  - we will use the following diagram as a base for illustrating the outcomes of different choices for $\boldsymbol{\mu}$
  - this diagram corresponds to the tensor product case with $N = 2$ and $M = 4$ for which there are 25 terms in the polynomial



– this diagram corresponds to the total degree case with $N = 2$ and $M = 4$
for which only the points on or below the magenta line are included

- there are only 15 terms in the polynomial



- however, the rates of convergence of the total degree polynomial
and the tensor product polynomial are the same

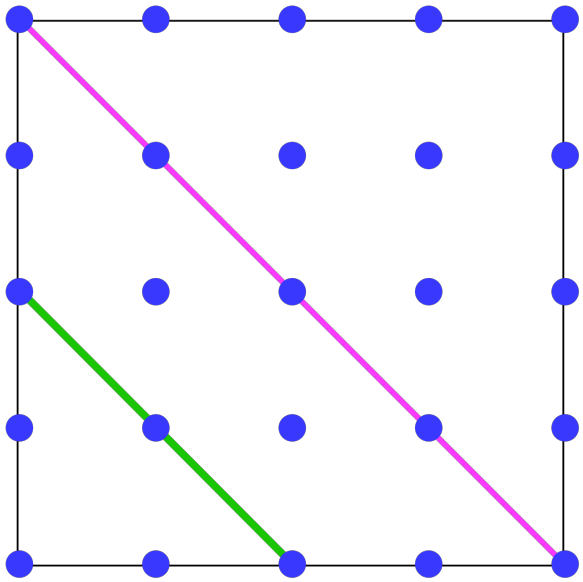
$\Longrightarrow$ the terms above the line do not
improve the rate of convergence

– in every following illustration, the lines delineates a total degree polynomial

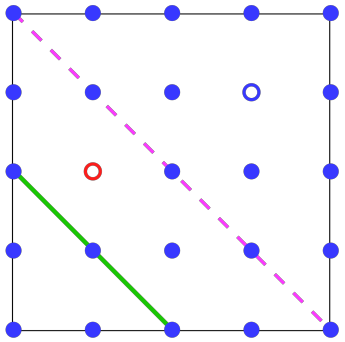
- the 25 the dots correspond to a tensor product polynomials with $M = 4$

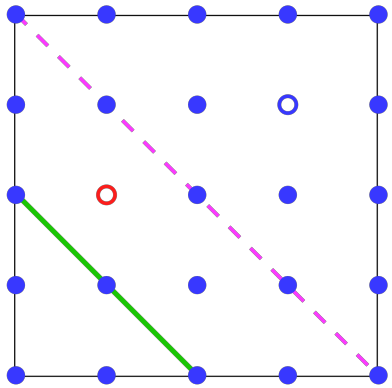
- the 15 dots on or below the magenta line correspond to a total degree polynomials with $M = 4$

- the 6 dots on or below the green line correspond to a total degree polynomials with $M = 2$



- Let P denote a set of polynomials
  - regardless of what set P of polynomials one chooses the accuracy (i.e., the rate of convergence) of the polynomial approximation is determined by the largest subset of total degree polynomials within the given set P that has largest degree
  - lets start out with the set P_{tp} being the set of all tensor product polynomials with $N = 2$ and $M = 4$
    - now let P_{td} denote the subset of P_{tp} that contains all total degree polynomial of the highest degree possible (all the dots on or below the magenta line result in degree 4 total degree polynomials)





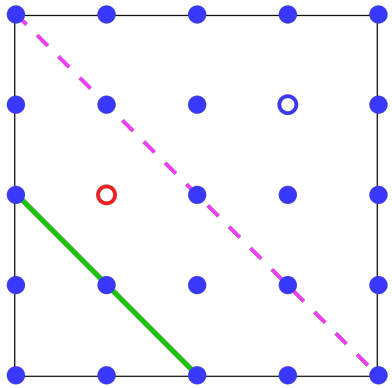
- suppose that for some reason

we decide to exclude from P_{tp} all polynomials
that have the $y_1^3 y_2^3$ term (the blue circle)

because that term is not included in the total degree polynomial

losing that one term does not affect the accuracy

$\Longrightarrow$ the accuracy remains the same as that of
a degree 4 total degree polynomial P_{td}



- now suppose we do something really silly

  we decide to exclude from P_{tp} all polynomials
  that have the $y_1 y_2^2$ term (the red circle)

  because that term is excluded from the degree 4 (and for that matter, also from the degree 3) total degree polynomials

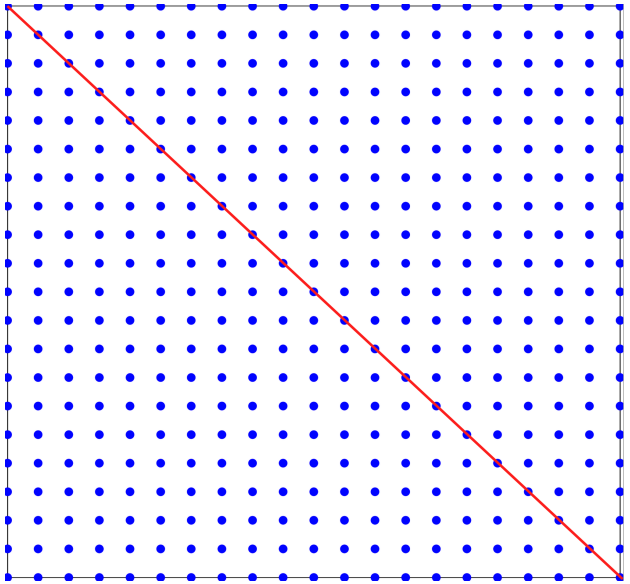
  losing that single term leaves us with the
  accuracy of a degree 2 total degree polynomial P_{td}

  corresponding to the dots on or below the green line

- Are we really being silly?

– the tensor product case: $\max_{1 \le n \le N} \mu_n \le M$

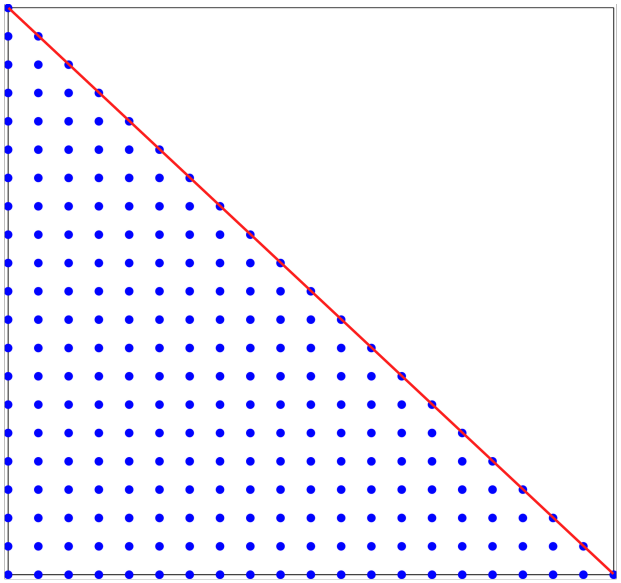
- for $N = 2$ and $M = 20$, the picture looks like



- on and below the red line, we have all the terms we need to have degree $M = 20$ total degree polynomials

– the total degree case: $\sum_{n=1}^{N} \mu_n \leq M$

- for $N = 2$ and $M = 20$, the picture looks like

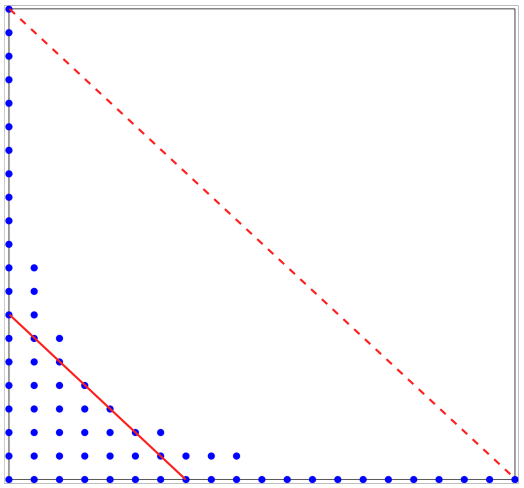


- throwing out all terms corresponding to the dots above the red line does not compromise the accuracy

the rates of convergence for the degree 20 tensor product and total degree polynomials are the same

– the hyperbolic cross case: $\prod_{n=1}^{N}(\mu_n + 1) \leq M + 1$

- for $N = 2$ and $M = 20$, the picture looks like

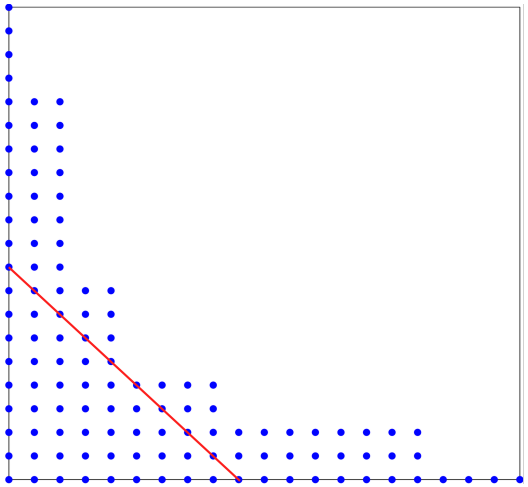


- now we have thrown out terms above and below the red dotted line that corresponded to degree 20 total degree polynomials
  so that the total degree polynomials correspond
  to the dots on or below the new solid red line
  resulting in degree 7 total degree polynomials

- so we have a great reduction in the number of terms
  but have also suffered a serious loss of accuracy

– the Smolyak case: $\sum_{n=1}^{N} \lceil \log_2(\mu_n) \rceil \leq \log_2(M)$ with $\mu_n \geq 2$

- for $N = 2$ and $M = 20$, the picture looks like



- now we have again thrown out terms above and below the red dotted line that corresponded to degree 20 total degree polynomials

  so that the total degree polynomials correspond

  to the dots on or below the new solid red line

  resulting in degree 10 total degree polynomials

- so we have a great reduction in the number of terms but have also suffered a serious loss of accuracy

- Anisotropic multivariate polynomials
  - by introducing weights into the definitions of the different types of polynomials, we can induce different polynomial degrees in different coordinates

    the tensor product case: $\max_{1 \le n \le N} \alpha_n \mu_n \le M$

    the total degree case: $\sum_{n=1}^{N} \alpha_n \mu_n \le M$

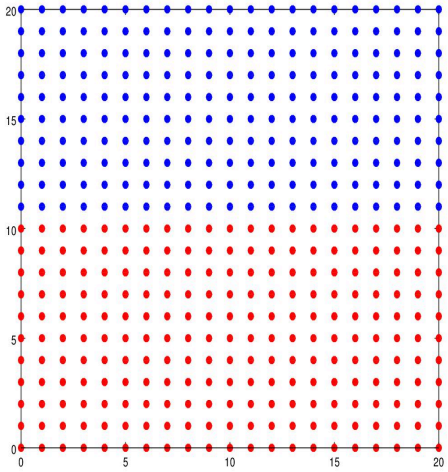
    the hyperbolic cross case: $\prod_{n=1}^{N} (\mu_n + 1)^{\alpha_n} \le M + 1$

    the Smolyak case: $\sum_{n=1}^{N} \alpha_n \lceil \log_2(\mu_n) \rceil \le \log_2(M)$ with $\mu_n \ge 2$

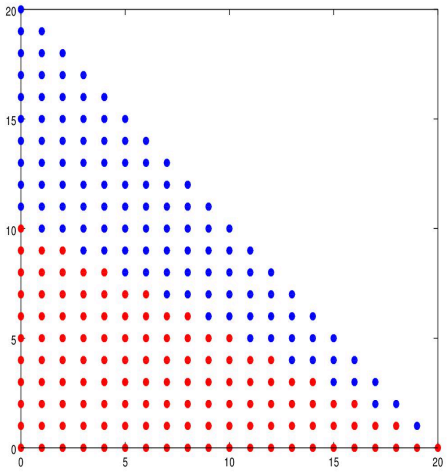
  - in some cases, the weights can be predetermined

    whereas

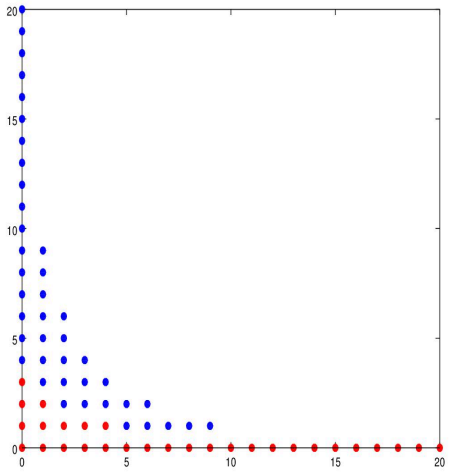
    in other cases the weights can be determined adaptively



TP



TD



HC

blue + red: standard

red: anisotropic with $\alpha_1 = 1$ and $\alpha_2 = 1/2$

GLOBAL LAGRANGE INTERPOLATION

- Instead of using global orthogonal polynomials, one can use interpolatory polynomials

- Given a set $\{\widetilde{\boldsymbol{y}}_k\}_{k=0}^{K_{li}}$ of K_{li} N-vectors

  – for $k \in \{1, \ldots, K_{li}\}$, let $L_k^{li}(\boldsymbol{y}; \widetilde{\boldsymbol{y}}_0, \widetilde{\boldsymbol{y}}_1, \ldots, \widetilde{\boldsymbol{y}}_{K_{li}})$ denote the set of Lagrange interpolating polynomials for these points

  – these polynomials are required to satisfy the interpolation conditions

$$L_k^{li}(\boldsymbol{y}_{k'}; \widetilde{\boldsymbol{y}}_0, \widetilde{\boldsymbol{y}}_1, \ldots, \widetilde{\boldsymbol{y}}_{K_{li}}) = \delta_{kk'} \qquad \text{for all } k, k' \in \{0, \ldots, K_{li}\}$$

  – note that each of these polynomials depends on all the chosen N-vectors $\widetilde{\boldsymbol{y}}_0, \widetilde{\boldsymbol{y}}_1, \ldots, \widetilde{\boldsymbol{y}}_{K_{li}}$

– we now have global polynomials of the form

$$p_{li}(\boldsymbol{y}) = \sum_{k=0}^{K_{li}} C_k L_k^{li}(\boldsymbol{y}; \widetilde{\boldsymbol{y}}_0, \widetilde{\boldsymbol{y}}_1, \ldots, \widetilde{\boldsymbol{y}}_{K_{li}})$$

– for a given function $f(\boldsymbol{y})$, the coefficients $\{C_k\}_{k=1}^{K_{go}}$ are determined by

$$f(\boldsymbol{y}_{k'}) = p_{li}(\boldsymbol{y}_{k'}) = \sum_{k=0}^{K_{li}} C_k L_k^{li}(\boldsymbol{y}_{k'}; \cdots\cdots) = \sum_{k=0}^{K_{li}} C_k \delta_{kk'} = C_{k'}$$

$\Longrightarrow$ the Lagrange interpolatory polynomial approximation of $f(\boldsymbol{y})$ is given by

$$f(\boldsymbol{y}) \approx \sum_{k=0}^{K_{li}} f(\boldsymbol{y}_k) L_k^{li}(\boldsymbol{y})$$

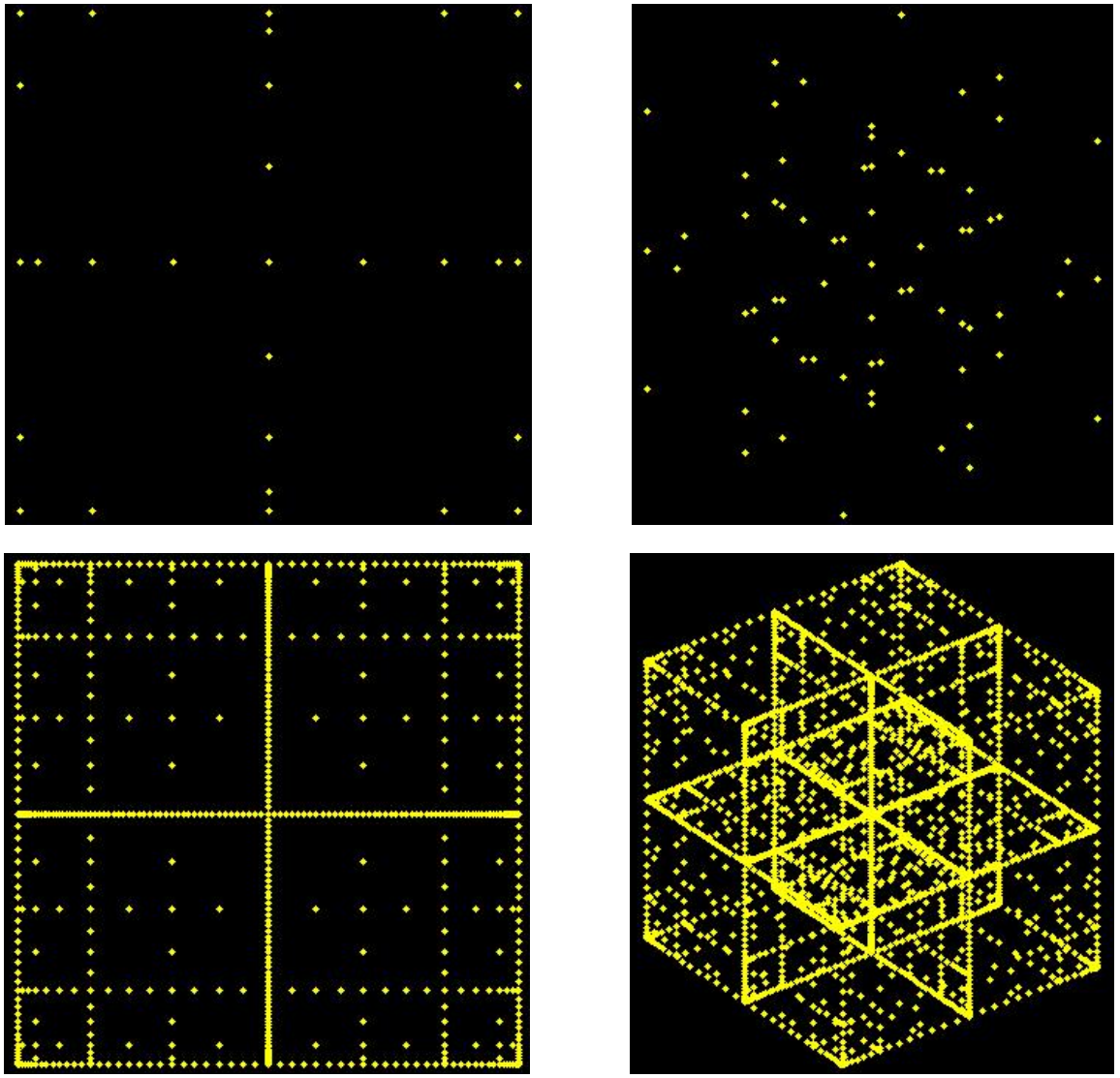
- Unfortunately, even for a moderate number of variables, it is not an easy to define a "good" set of interpolation points that can be used to determine a complete Lagrange interpolant
  - it is an easy matter to define a set of interpolation points that can be used to define a tensor product Lagrange interpolatory polynomial[†]
  - however, as we have seen, this leads to a very inefficient approximation compared to complete polynomial approximation
- There exists intermediate choices, e.g., Smolyak point sets,

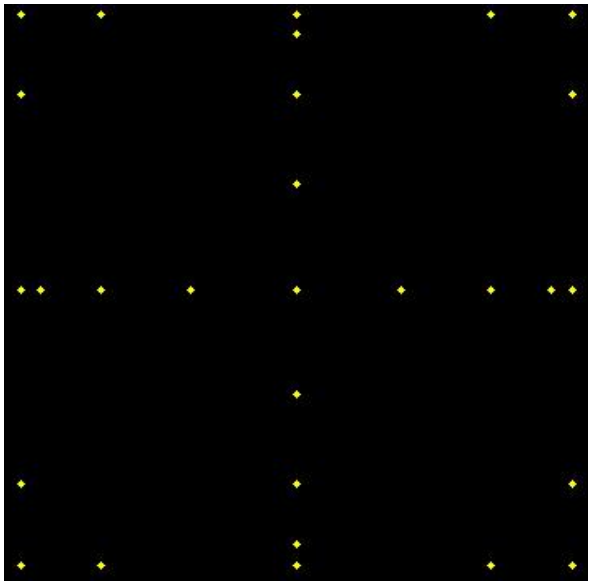
  that can be systematically defined in any dimension
  - for the Smolyak point sets, $K_{li}^{smolyak} > \dfrac{(M+N)!}{N!M!}$ so that they require more points compared to total degree polynomial interpolation
  - however, we have that $K_{li}^{smolyak} \ll (M+1)^N$ so that they requires much fewer points compared to tensor product interpolation

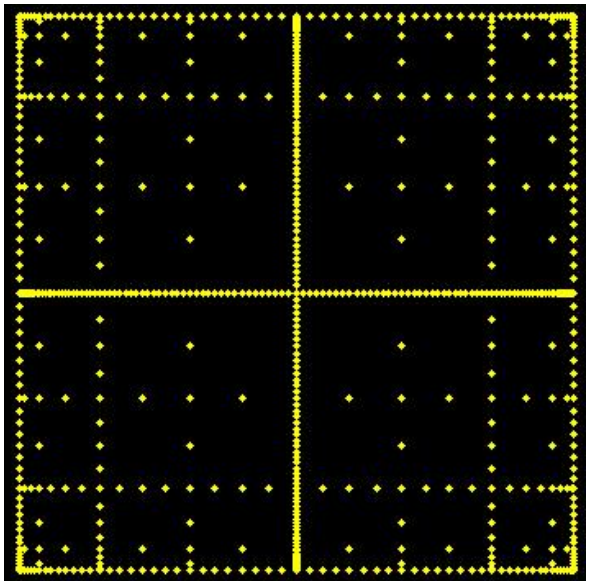
---

[†]Unlike the case for orthogonal polynomials, for Lagrange interpolating polynomials it is not easy to define a total degree polynomial basis from the tensor product basis; for the Lagrange interpolation case, the tensor product basis is not hierarchical since all Lagrange polynomials are of the same degree

- Some Smolyak grids in two and three dimensions





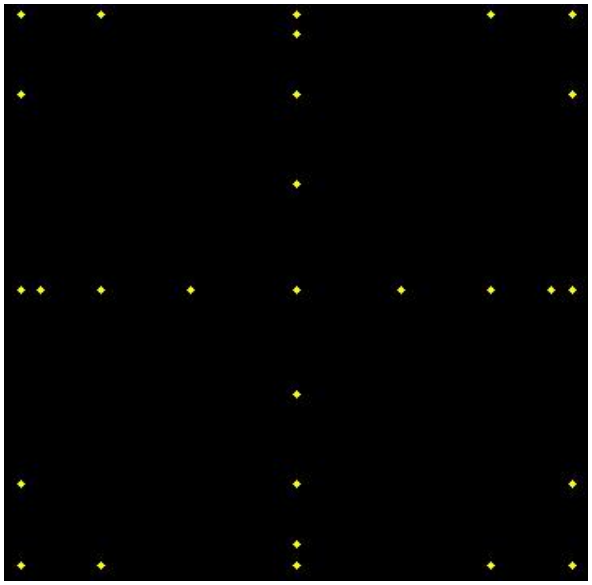


– the construction of these particular Smolyak grids is based on a univariate Chebyshev grid

  - in fact, you can see the univariate grids along the coordinate axes

– note that the Chebyshev points are not uniformly distributed

  - using Chebyshev points results in better accuracy than using uniformly distributed points



– this Smolyak grid has 29 grid points so that 29 function evaluations have to be made to build the interpolatory polynomial

– the corresponding 29-point Smolyak interpolant has the same "nominal accuracy" as does the 15-point total degree interpolant

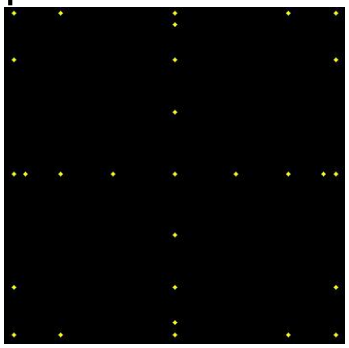
- however, for even very moderate N, we have no idea where to place the 15 points
- a "bad" placement of the points can result in big errors

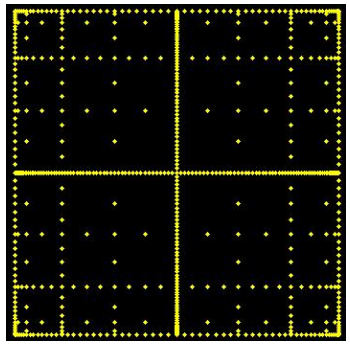
- We ask the question:

  is the restriction of only being able to handle points that are within a hyper-rectangle make global orthogonal polynomials or Lagrange interpolatory polynomials useless?

  - fortunately, there are settings for which hyper-rectangles is exactly what one encounters
  - most notably, this is the common situation in the uncertainty quantification (UQ) setting

- What about coverage, equal spacing, and star displacement?
  - clearly, sample points such as





    break all the rules

    - there are big holes, bad projections, and star discrepancies are bad

  - we are saved because

    - interpolants or quadrature rules using these points
      require that the functions involved be smooth
    - one does not get the accuracy of such approximations unless
      the functions involve have a sufficient number of derivatives

    in such cases the smoothness of the functions involved save the day because the approximations are accurate everywhere in the domain, including the holes

– furthermore, the quadrature rules associated with
such points involve specific choices of quadrature points

they are not simple sampling & averaging rules

- the weights are chosen by requiring that the quadrature
rule achieves the highest possible accuracy level

UQ examples

- There are many UQ tasks for which

  $\boldsymbol{y} \in \Gamma$ is a vector or random variables which are distributed according to a given PDF $\rho(\boldsymbol{y})$

  and for which

  Γ is an N-dimensional hyper-rectangle

  – a partial list includes

    determining statistical quantities such as expectations, moments, variances, covariances, etc.

    estimating the probability of failures

    finding extremal values

  – here, we consider the first two tasks

- First consider the

  evaluation of integrals having integrands that depend on a function $f(\boldsymbol{y})$

  – specifically, consider integrals of the form

$$\int_{\Gamma} G\big(f(\boldsymbol{y})\big)\rho(\boldsymbol{y})\,d\boldsymbol{y}$$

  – the function $G(\cdot)$ determines what statistical information about $f(\boldsymbol{y})$ is desired

$$G(f) = f$$

expected value of f $\quad\Leftarrow\quad \mathsf{E}(f) = \displaystyle\int_{\Gamma} f(\boldsymbol{y})\rho(\boldsymbol{y})\,d\boldsymbol{y}$

$$G(f) = f^2$$

second moment of f $\quad\Leftarrow\quad \mathsf{M}_2(f) = \mathsf{E}(f^2) = \displaystyle\int_{\Gamma} \big(f(\boldsymbol{y})\big)^2\rho(\boldsymbol{y})\,d\boldsymbol{y}$

$$G(f) = (f - \mathsf{E}(f))^2$$

variance of f $\quad\Leftarrow\quad \mathsf{V}(f) = \displaystyle\int_{\Gamma} \big(f(\boldsymbol{y}) - \mathsf{E}(f)\big)^2\rho(\boldsymbol{y})\,d\boldsymbol{y}$

. . . . . covariances, higher moments . . . . . .

– in general, we have to approximate such integrals using a quadrature rule

$$\int_{\Gamma} G\big((f(\boldsymbol{y})\big)\rho(\boldsymbol{y})\,d\boldsymbol{y} \approx \sum_{q=1}^{Q} W_q \rho(\boldsymbol{y}_q) G\big(f(\boldsymbol{y}_q)\big)$$

  - requires Q evaluations of $f(\boldsymbol{y})$

  - question: what are good quadrature points to use?

– a good approach is to choose the quadrature points to be the interpolation points of a good interpolant of $f(\boldsymbol{y})$

  - doing so, one can
    - systematically determine the weights W_q
    - derive estimates for quadrature errors from estimates for interpolation error

– for example, the Smolyak points that were used for Smolyak interpolation can be also used as quadrature points

- Another UQ task is to determine failure probabilities

$$\text{Prob}[f(\boldsymbol{y}) > f_0] \quad \text{for a given } f_0$$

  - e.g., what is the probability that $f(\boldsymbol{y})$ exceeds the value f_0 at which the bridge falls down?

$$\text{Prob}[f(\boldsymbol{y}) > f_0] = \int_\Gamma \rho(\boldsymbol{y}) \mathcal{X}_{f(\boldsymbol{y})>f_0}(\boldsymbol{y}) d\boldsymbol{y}$$

  where $\mathcal{X}_\cdot(\boldsymbol{y})$ denotes the indicator function

  – this integral is often approximated by a massive sampling/rejection method so that lots evaluations of $f(\boldsymbol{y})$ are utilized

    - if instead of sampling $f(\boldsymbol{y})$, we sample an approximation of $f(\boldsymbol{y})$, be it an orthogonal polynomial or an interpolant or some other surrogate

  substantial saving could be made, especially if sampling $f(\boldsymbol{y})$ is expensive

The need for smoothness

- The product rules we have discussed, including the sparse-grid rules, putatively achieve greater accuracy as the polynomial precision increases

  – actually achieving the increased accuracy requires
  the integrand be "sufficiently" smooth,
  i.e., to have a "sufficient" number of continuous derivatives

- For an integrand that is not sufficiently smooth

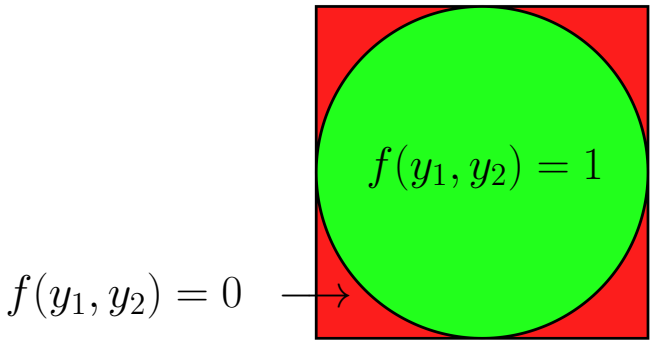
  e.g., for the absolute value function or any discontinuous function

  the sparse-grid approach using global polynomial interpolatory quadrature rules result in generally very bad approximations

  – the same comment holds for global
  orthogonal polynomial approximations

- For example, in dimension $N = 6$, let $f(\boldsymbol{y})$ denote the indicator function of the unit ball embedded in a unit cube

  here we illustrate with the $N = 2$ setting



– the table shows the non-convergence of
CC sparse grid quadrature for this example

– for comparison purposes, the table also provides the convergence history of Monte Carlo approximate integration for the same function

Q	SG estimate	SG error	MC estimate	MC error
1	4.000	1.167	0.00000	5.16771
13	64.000	58.832	0.00000	5.16771
85	-42.667	-47.834	3.01176	2.15595
389	-118.519	-123.686	4.77121	0.39650
1457	148.250	143.082	5.16771	0.01555
4865	-24.682	-29.850	5.41994	0.25226
exact	5.16771	–	5.16771	–

Comparison of sparse grid and Monte Carlo approximations
of the integral of a discontinuous function

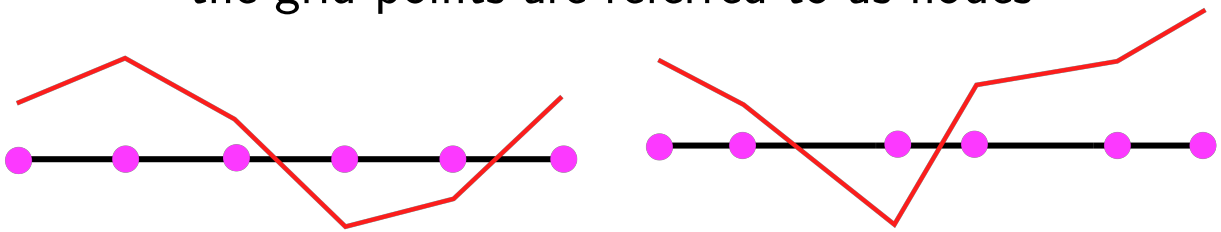
PIECEWISE POLYNOMIAL APPROXIMATIONS

- We briefly consider piecewise polynomial approximations of a function $f(\boldsymbol{y})$
- There are several reasons why such approximations are of interest
  - they are simple to construct
  - they can deal with general domains Γ, not just hyper-rectangles
  - they can deal with discontinuous functions $f(\boldsymbol{y})$
- Let's start with the one-dimensional case
  - piecewise linear and piecewise constant functions

– piecewise linear and piecewise constant functions in one dimensions

uniform grid nonuniform grid

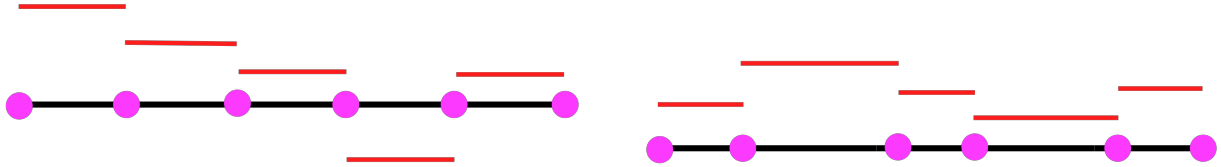
the grid points are referred to as nodes



continuous piecewise linear polynomials

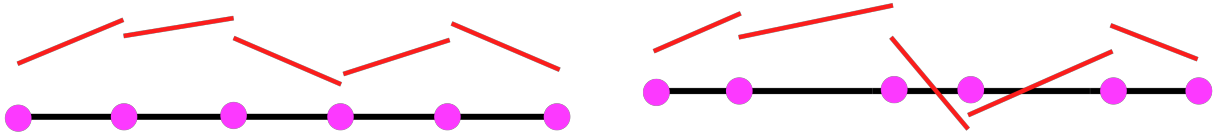
a different linear polynomial $ay + b$ in each subinterval

continuous over the whole interval



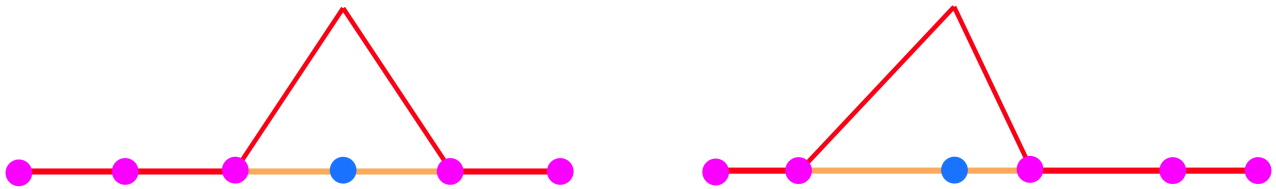
discontinuous piecewise constant polynomials

constant in each subinterval, discontinuous at the nodes



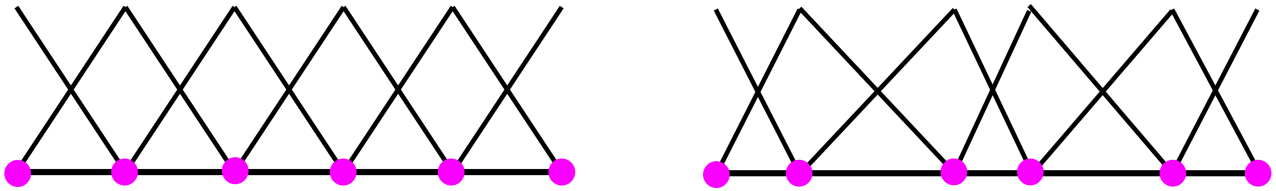
discontinuous piecewise linear polynomials

linear in each subinterval, discontinuous at the nodes

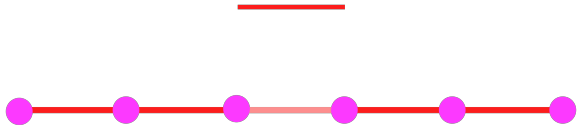


a compactly supported continuous piecewise linear polynomial corresponding to the blue node

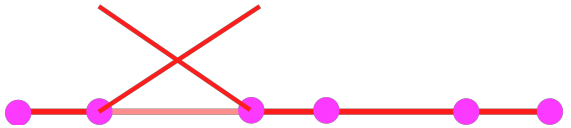
the function is $\neq 0$ only on the subintervals that contain the blue node



the set of basis functions which span all continuous piecewise linear polynomials



a basis function for the set of discontinuous piecewise constant polynomials



two basis functions for the set of discontinuous piecewise linear polynomials

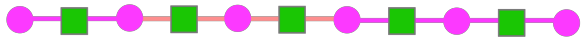
basis functions are $\neq 0$ on only a single subinterval

– piecewise linear and piecewise quadratic functions in one dimension

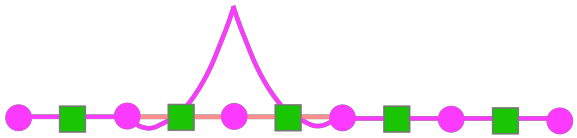
we have an interval divided into subintervals (just the same as for the piecewise linear or piecewise constant polynomial cases)

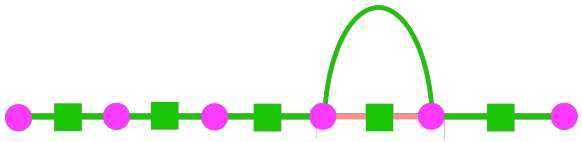
in each subinterval, we have different quadratic polynomial $a + by + cy^2$

to uniquely define a quadratic polynomial within each subinterval we need to know its value at 3 points



for this reason, we add the green nodes which are located in the middle of each subinterval (the green nodes are not grid points) so that in each subinterval we have 3 points (2 magenta and one green)





we can then define a set of compact continuous basis functions

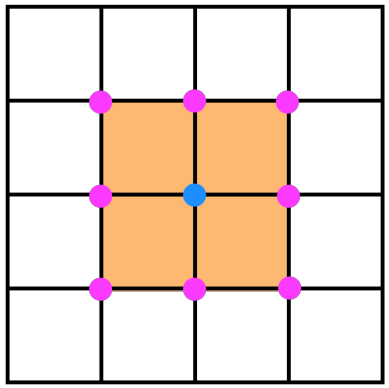
a basis function which is 1 at a magenta point and 0 at all the other magenta points and at all the green points

a basis function which is 1 at a green point and 0 at all the other green points and at all the magenta points

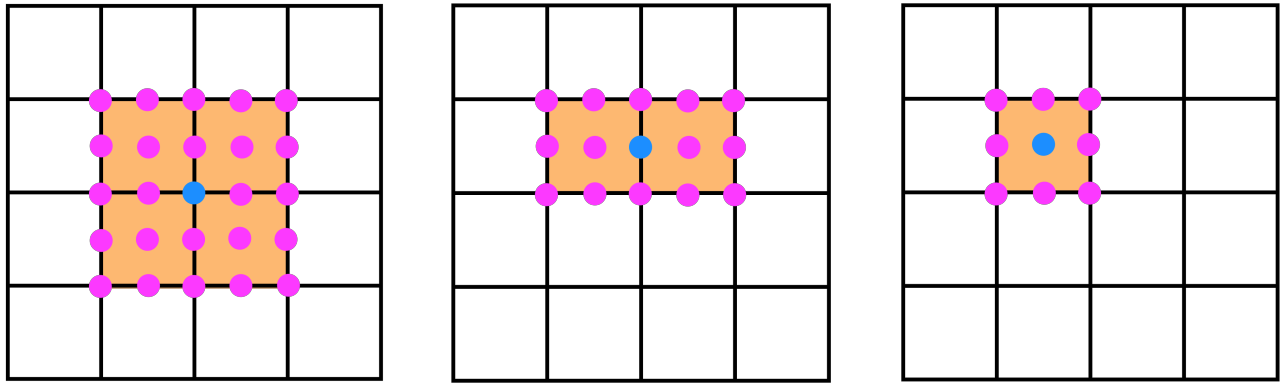
– can also define discontinuous piecewise quadratic basis functions

– can also define piecewise higher-degree basis function, both contintinuous and discontinuous

- Let's take a brief look at 2D



the support of a bilinear[1] basis function $a + by_1 + cy_2 + dy_1y_2$

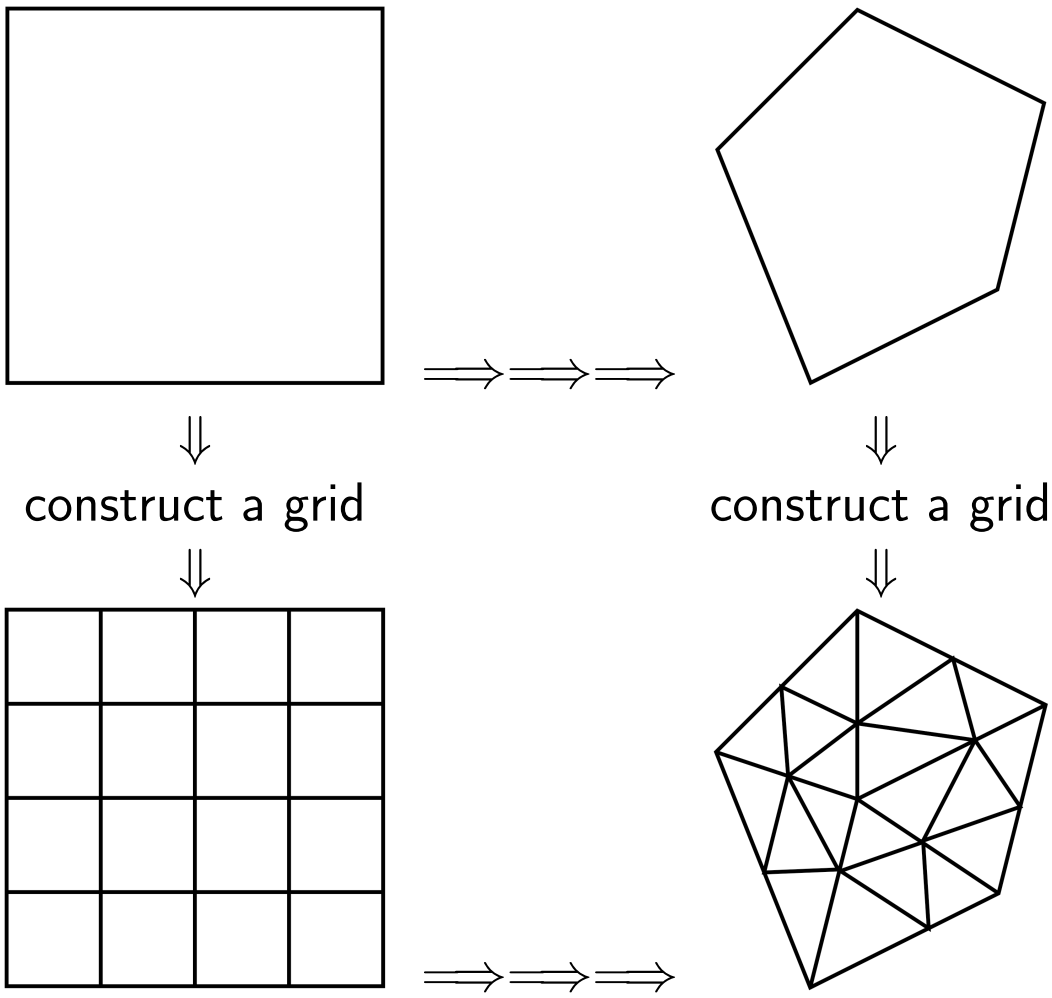


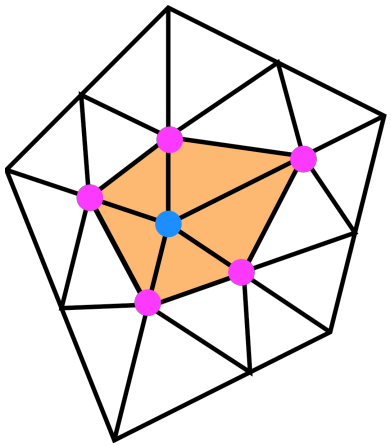
the support of a biquadratic basis function

$$a + by_1 + cy_2 + dy_1^2 + ey_1y_2 + fy_2^2 + gy_1^2y_2 + hy_1y_2^2 + jy_1^2y_2^2$$

[1] Bilinear polynomials are linear in each variable; biquadratic polynomials are quadratic in each variable

– But, one can use piecewise polynomials for arbitrary domains Γ, not just hyper-rectangles

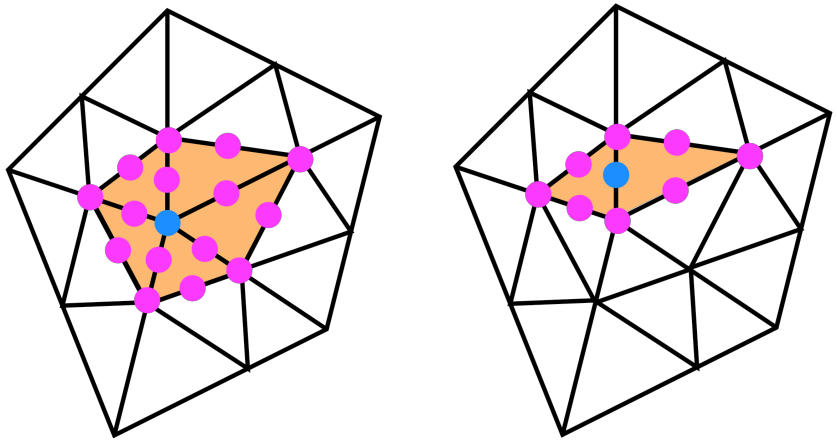




support of the basis function for the blue node

basis function is a piecewise linear function $a + by_1 + cy_2$ in each triangle

the basis function is 1 at the blue point and zero at the magenta points and is zero outside the 4 orange triangles



support of two types of basis function for the blue node in case of piecewise quadratic polynomials

basis functions are a piecewise quadratic polynomial $a + by_1 + cy_2 + dy_1^2 + ey_1y_2 + ey_2^2$ in each triangle

the basis functions are 1 at the blue point and zero at the magenta points and is zero outside the orange triangles

- It is then an easy matter to construct piecewise polynomials interpolants
  - linear interpolants

$$f_{pl}(\boldsymbol{y}) = \sum_{\text{all vertices } \boldsymbol{y}_k} f(\boldsymbol{y}_k)\psi_k^{pl}(\boldsymbol{y})$$

    where $\psi_k^{pl}(\boldsymbol{y})$ denotes the piecewise linear basis functions corresponding to a vertex $\boldsymbol{y}_k$

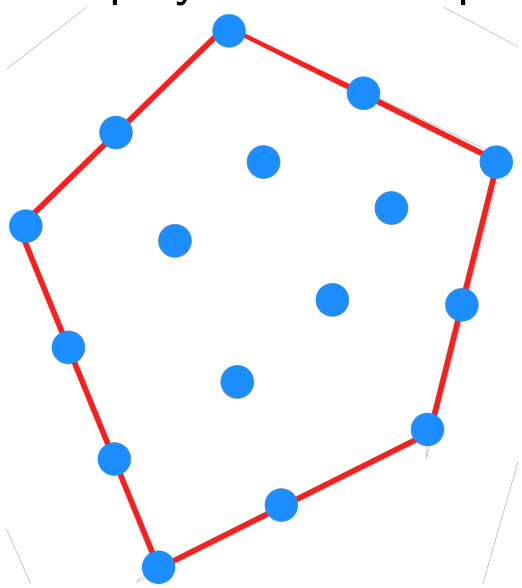
  - quadratic interpolants

$$f_{pq}(\boldsymbol{y}) = \sum_{\text{all vertices and midsides } \boldsymbol{y}_k} f(\boldsymbol{y}_k)\psi_k^{pq}(\boldsymbol{y})$$

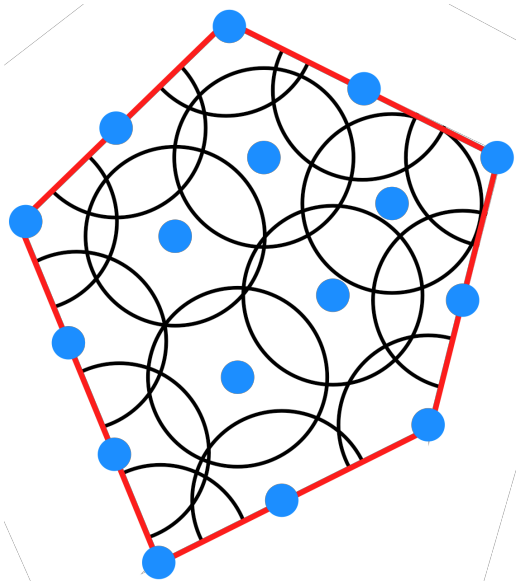
    where $\psi_k^{pl}(\boldsymbol{y})$ denotes the piecewise quadratic basis functions corresponding to a vertex or a mid-side $\boldsymbol{y}_k$

- There are other approaches to interpolation that
  - do not rely on grids
  - do not rely on polynomials, piecewise or otherwise

  – for example, let's start with the same points that we just used for piecewise polynomial interpolation



– with each point we associate a circle



– with each point we also associate, for example,
a Gaussian function centered at that point

– we then use the set of all Gaussian functions associated
with the points as a basis for interpolation

– a class of methods of this type are referred to as
radial basis functions methods

LEAST-SQUARES APPROXIMATIONS

- After having pontificated about the very high pedestal on which I placed interpolation and least-squares approximation (regression) in the universe of data science, I would be remiss if I did't say a few words about the latter

- First, we have a set of pairs $\{\boldsymbol{y}_m, f_m\}_{m=1}^M$ where

  $\{\boldsymbol{y}_m\}_{m=1}^M$ is a set of points in a region Ω

  $\{f_m\}_{m=1}^M$ is a set values associated with the points $\boldsymbol{y}_m$

  – f_m could be a quantity that is measured at a given $\boldsymbol{y}_m$

  alternately, we could have that

  $f_m = f(\boldsymbol{y}_m) \Longleftarrow f_m$ is the value of a given function $f(\boldsymbol{y})$ at the point $\boldsymbol{y}_m$

- Second, we have in hand basis set of cardinality $K \leq M$, even $K \ll M$, of our choosing

$$\left\{\psi_k(\boldsymbol{y})\right\}_{k=1}^{K}$$

  - it could be a global or local polynomial basis, or a Fourier basis (sines and cosines), or any of the many, many other bases that are available

  – given the basis, we can choose as set $\{C_k\}_{k=1}^{K}$ of constants to define the function

$$p_K(\boldsymbol{y}) = \sum_{k=1}^{K} C_k \psi_k(\boldsymbol{y})$$

- Given the data $\{\boldsymbol{y}_m, f_m\}_{m=1}^{M}$ and having chosen a basis $\left\{\psi_k(\boldsymbol{y})\right\}_{k=1}^{K}$ we want to find a data-informed set of coefficients $\{C_k\}_{k=1}^{K}$

- We do so by

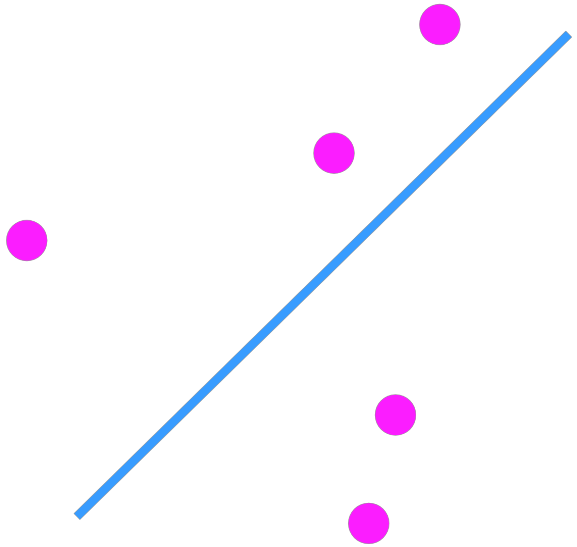
$$\text{minimizing over all possible sets } \{C_k\}_{k=1}^{K} \left(\sum_{m=1}^{M} \left| p_{M,K}^{ls}(\boldsymbol{y}_m) - f_m \right|^2 \right)$$

  - for example, if $K = M$ we can set $p_{M,M}^{ls}(\boldsymbol{y}_m) = f_m$ so that $p_{M,M}^{ls}(\boldsymbol{y})$ is an interpolant which obviously is a minimizer $\Longleftarrow \sum_{m-1}^{M} \left| p_{M,K}^{ls}(\boldsymbol{y}_m) - f_m \right|^2 = 0$

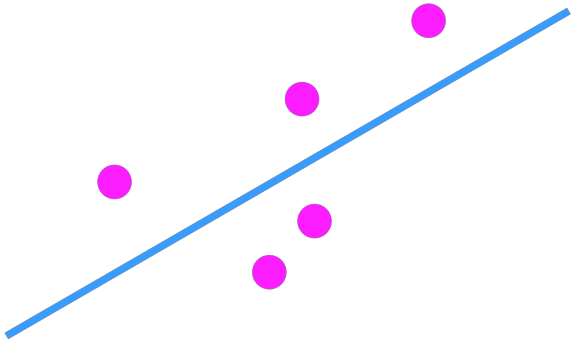
- Least-squares can do things that interpolation cannot do

  - least-squares can deal with general domains in multi-dimensions
  - least-squares can deal with scattered data
  - other than having $M > K$, the cardinalities M of the data set and K of the basis do not have to be related

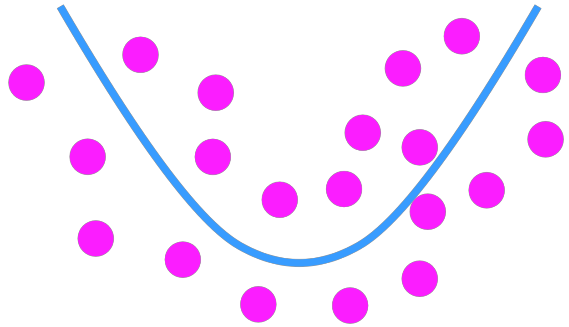
- We can have 5 data points and a linear polynomial



- A linear polynomial is perhaps more useful
  if the data points are not widely scattered



- For this data set a quadratic polynomial does a better job than a linear polynomial



- One can also consider minimizing

$$\sum_{m=1}^{M} \left| p_{M,K}^{\ell 1}(\boldsymbol{y}_m) - f_m \right|^1$$

  - solving this minimization problem is considerably more difficult compared to solving the least-squares minimization problem

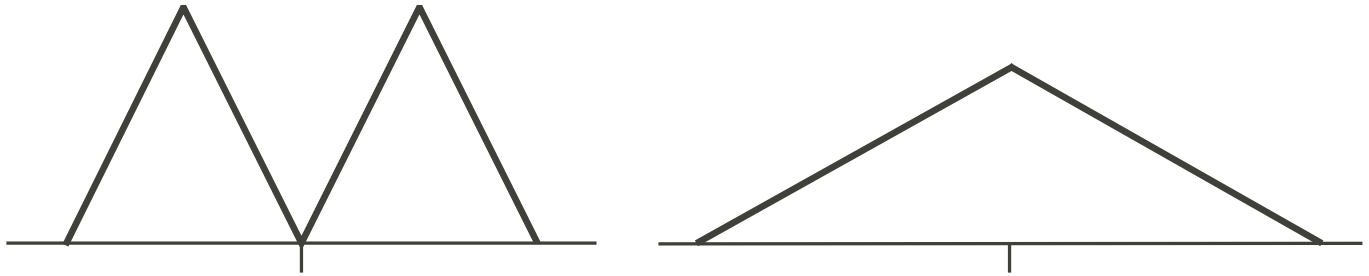
DETERMINING PDFs

Input PDFs – does one know what they are?
What about output PDFs?

- Very often, it is tacitly assumed
  - that one knows the PDFs $\rho_n(y_n)$ of the input parameters
  - or even the joint PDF $\rho(\boldsymbol{y})$ of an input parameter vector

- Actually, in practice,
  one often does not know much about the statistics of the input parameters
  - one is lucky if
    - one only knows a range of values for an input parameter
      i.e., maximum and minimum values
    - in which case there is not much one can do but assume that
      the parameter is uniformly distributed over that range
  - if one is luckier, one knows
    - the mean and variance of the input parameter
    - in which case there is not much one can do other than
      assume that the parameter is normally distributed

– of course, one may be completely wrong in assuming simple probability distributions for a parameter



bi-modal and uni-modal PDFs having the same mean and variance

- The difficulty arises because only the Gaussian PDF is uniquely determined by its mean and variance
  - i.e., by its first and second moments

– all other PDFs also possess higher moments

– the situation is even worse in the multi-parameter case for which only very special Gaussian PDFs (but not general ones) can be determined from the parameter means and variances
  - for general multivariate Gaussian PDFs, one also has to know the covariances between parameters

- Any additional information one has about an input PDF can result in a better "guess"
  - one knows the support of the PDF can only take positive values
    (in such a case, log-normal PDFs are often used)
  - one knows that the PDF is bi-modal

- The lack of knowledge about input PDFs leads to the need to solve stochastic model calibration problems
  i.e., can we do better than guessing
  - naturally, this is very active field of research

- But, can we really do better than guessing?
  - yes we can if we have data available from, e.g., experimental measurements, field observations, simulations, etc.

- What about output PDFs?
  - the ideal QoI task is to
    - given a model output $f(\boldsymbol{y})$ (which may be continuous or discrete) and its input PDF $\rho_y(\boldsymbol{y})$
    - determine (at least approximately) the output PDF $\rho_f(f)$ for f itself
  - having obtained $\rho_f(f)$, one can obtain values of the model output f by simply sampling $\rho_f(f)$ instead of resorting to solving the model

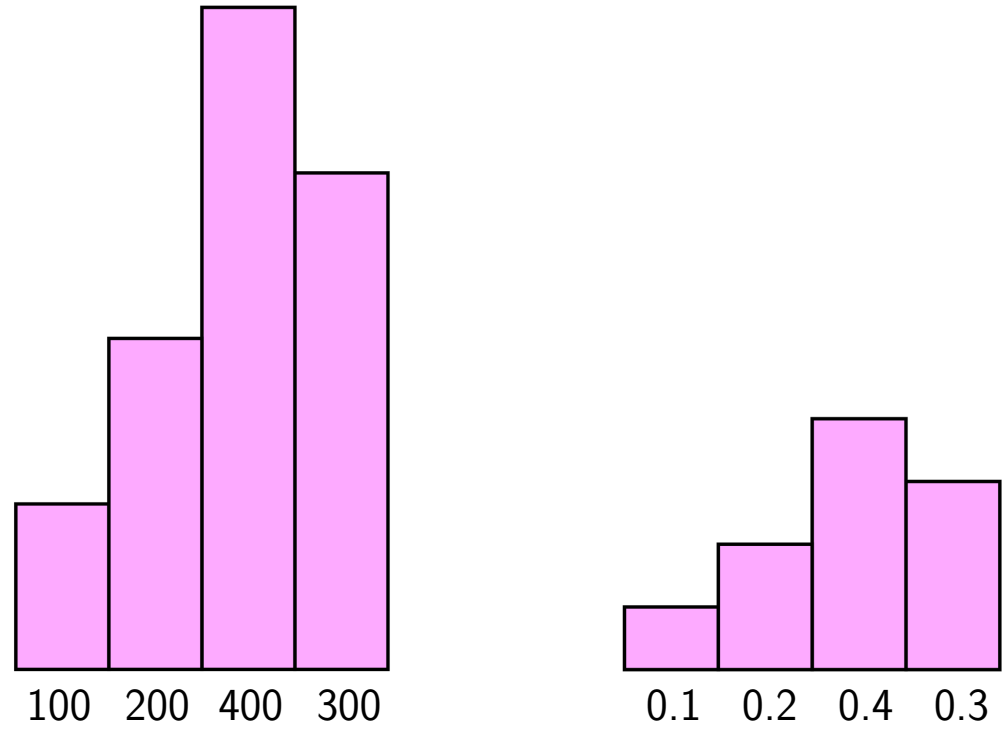
- The main difference between constructing input and output PDFs is the source of the data needed
    - for the input PDF, the data can come from many sources
    - for the output PDF, the data usually comes from model simulations

  - we consider three approaches for obtaining (approximate) PDFs from data

Histograms

- A histogram approximation of a PDF $\rho(z)$ is constructed via sampling and binning

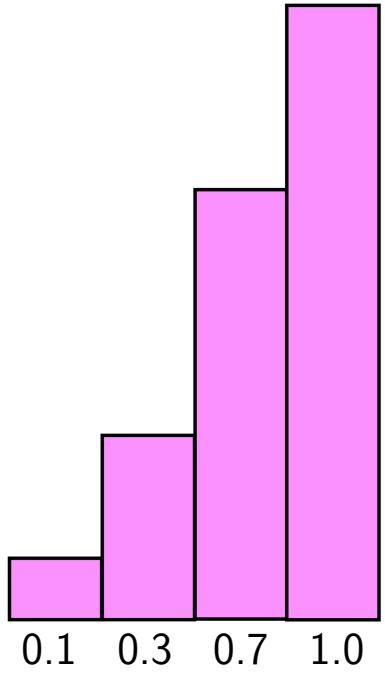
- Assume that we know that the values of z are restricted to a finite interval $[a, b]$
  - then, subdivide the interval $[a, b]$ into K bins
    - choose all the bins to have the same length $\dfrac{(b-a)}{K}$
  - we have available a data set $\{z^{(i)}\}_{i=1}^{big\ number}$
  - we count the number of $z^{(i)}$ that are in each bin
    - if a $z^{(i)} \notin [a, b]$, we can add bins
  - we normalize the bin heights so that the total area of the histogram is $= 1$



1000 samples distributed into bins

then normalized so that the total area = 1

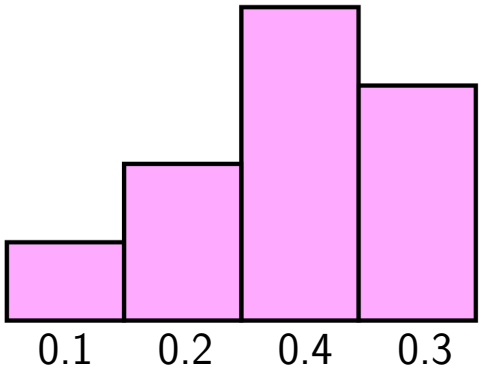
- of course, can get the corresponding CDF by stacking bins, left to right



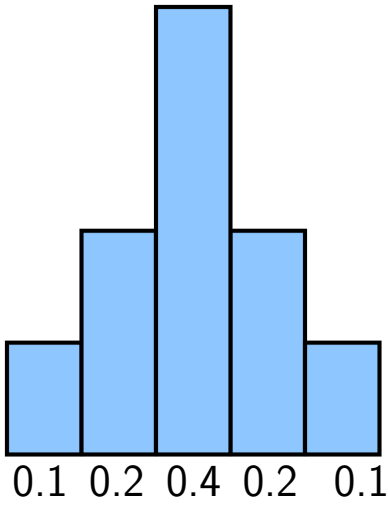
- Now suppose we have
  - a vector $\boldsymbol{z} = (z_1, z_2)$ of independent random variables

    with $z_1 \in [a_1, b_1]$ and $z_2 \in [a_2, b_2]$

    and with the joint PDF $\rho(\boldsymbol{z}) = \rho_1(z_1)\rho_2(z_2)$
  - a sample point is determined in each direction by sampling and binning according to each PDF

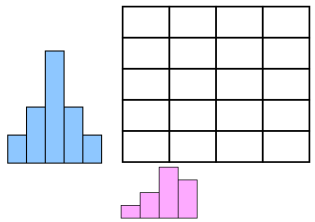


binning samples z_1
according to the PDF $\rho_1(z_1)$



binning samples z_2
according to the PDF $\rho_2(z_2)$

- The bi-variate histogram then has 20 rectangular bins
  - the areas above each bin are entered into the boxes below
  - the sum of the areas $= 1$



0.1	0.01	0.02	0.04	0.03	$\Sigma = 0.1$
0.2	0.02	0.04	0.08	0.06	$\Sigma = 0.2$
0.4	0.04	0.08	0.16	0.12	$\Sigma = 0.4$
0.2	0.02	0.04	0.08	0.06	$\Sigma = 0.2$
0.1	0.01	0.02	0.04	0.03	$\Sigma = 0.1$
	0.1	0.2	0.4	0.3	

- The binning approaches just considered can be straightforwardly generalized to handle, e.g., in one dimension, bins of different lengths

- For the multivariate case, it can also be generalized to handle non-separable PDFs

- The multivariate binning approach just considered is well and good if one is dealing with two or three parameters, but as we have seen in other settings, it is not practical even in moderately higher dimensions

  - it falls prey to the curse of dimensionality

  – for one thing, one would have to have a humongous number of samples to sufficiently fill the bins so that one obtains a useful histogram

    - if one samples uniformly

      10 bins in each coordinate in 6 dimensions $\Longrightarrow$ 1,000,000 bins

      1,000,000 samples $\Longrightarrow$ on the average, one sample per bin $\Longleftarrow$ 1,000,000 bins

Kernel density estimators

- Kernel density estimators (KDE) have the form

$$\rho(z) \approx \frac{1}{Kb} \sum_{k=1}^{K} \Phi\Big(\frac{z - z_k}{b}\Big)$$

where

z_k is a data point

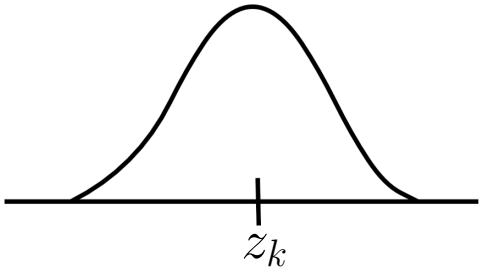
$\Phi(\cdot)$ is the kernel function (in other contexts it is referred to as something else)

usually chosen to be a smooth function

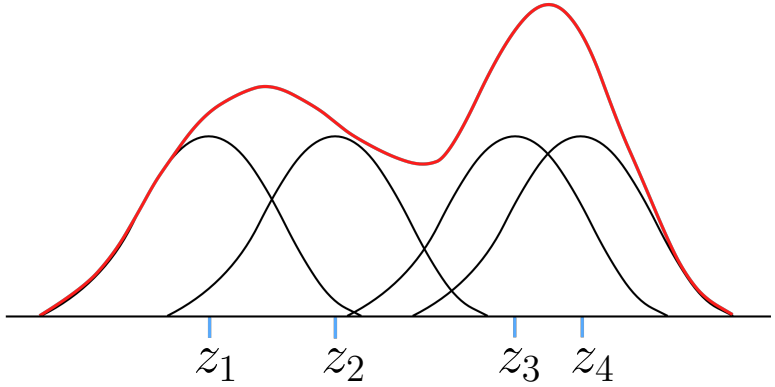
such as a (perhaps modified) Gaussian

b is a parameter that determines how fast the kernel function decays and what is its support

– a typical kernel function looks like



– a typical KDE estimator (the red curve) of the PDF looks like the



- for any z

  the height of the red curve is the sum of

  the heights of all the kernel at that point

- if the data points z_k are close to each other,

  the estimated PDF is larger

  than when the data points z_k are less close

- Generalization to the multivariate case is straightforward

$$\rho(\boldsymbol{z}) \approx \frac{1}{Kb} \sum_{k=1}^{K} \Phi\left(\frac{|\boldsymbol{z} - \boldsymbol{z}_k|}{b}\right)$$

- Kernel density estimators

  – result in smooth approximate PDFs
  so that, e.g., they may be differentiated

  – can be used for scattered data sets $\{\boldsymbol{z}_k\}_{k=1}^{K}$

    - points in such sets need not be in a hyper-rectangle

Moment expansions

- We consider the classical problem of determining an approximate PDF from its moments

  – warning: in general, PDFs are not uniquely determined by their moments

    - e.g., none of the moments corresponding to the PDF

$$\rho(z) = \frac{1}{z\sqrt{2\pi}} e^{-\frac{1}{2}(\ln z)^2} \big(1 + a \sin(2\pi \ln z)\big) \qquad -1 \le a \le 1$$

      depend on a

  – there have been many methods proposed for PDF recovery from moments

  – here, we consider the Gram-Charlier expansion of a PDF in terms of its moments

    - we will consider such expansions in the case of a single parameter

    - extensions to multiple parameters exist

    - several generalizations and improvements of the Gram-Charlier approach have also been developed

- If z is random parameter and $\rho(z)$ its associated PDF, we have that the m^{th} moment M_m is defined by

$$M_m = \int_{-\infty}^{\infty} z^m \rho(z)\, dz$$

  - the zeroth moment is equal to 1 and the first moment is equal to the mean (expected value) of z
  - we assume, without loss of generality, that z has zero mean and unit variance
    - the general case can be reduced to this case by removing the mean by subtraction and normalizing the variance by scaling

- Let $g(z) = \frac{1}{\sqrt{2\pi}} e^{\frac{-z^2}{2}}$ denote the PDF for the standard normal distribution

- Let $H_m(z)$ denote the normalized Hermite polynomial of degree m, where by normalized we mean that $H_m(0) = 1$

– we have that

$$\int_{-\infty}^{\infty} H_m(z) H_{m'}(z) g(z)\, dz = m! \delta_{mm'}$$

- Then, the PDF $\rho(z)$ has the expansion

$$\rho(z) = \sum_{m=0}^{\infty} c_m H_m(z) g(z) = \frac{1}{\sqrt{2\pi}} \sum_{m=0}^{\infty} c_m H_m(z) e^{\frac{-z^2}{2}}$$

where the coefficients c_m, $m = 1, \ldots,$, are given by

$$c_m = \frac{1}{m!}\left(M_m - \frac{b_{m2}}{2 \cdot 1!} M_{m-2} + \frac{b_{m4}}{2^2 \cdot 2!} M_{m-4} - \frac{b_{m6}}{2^3 \cdot 3!} M_{m-6} + \cdots \right)$$

where $\quad b_{mm'} = m!/(m - m')!$

- Calculating up to first five terms, we have

$$\rho(z) = \left(1 + \frac{1}{6} M_3 H_3(z) + \frac{1}{24} (M_4 - 3) H_4(z) + \cdots \right) e^{\frac{-z^2}{2}}$$

- In practice, one uses truncations of the Gram-Charlier expansion to approximate $\rho(z)$
  - the zeroth approximation is the Gaussian PDF $g(z) = e^{\frac{-z^2}{2}}$
  - thus, truncated Gram-Charlier expansions yield approximations of $\rho(z)$ that are "corrections" of the Gaussian PDF
  - often, Gram-Charlier expansions converge slowly which is why there is continuous search for improvements

- How does one obtain the moments of an output of interest F to use in the Gram-Charlier formula?
  - we assume that $F(\boldsymbol{y})$ is a scalar quantity that depends on an input parameter vector $\boldsymbol{y}$ which has an associated joint PDF $\widehat{\rho}(\boldsymbol{y})$

- The m-th moment we want is

$$M_m = \int_{-\infty}^{\infty} F^m \rho(F)\, dF$$

  where $\rho(F)$ is the PDF for the output F which we don't know (and in fact, which we are trying to find)
  - this is not the same as

$$\int_{\Gamma} F^m(\boldsymbol{y})\rho(\boldsymbol{y})\, dy$$

- Thus, we are left with trying to estimate M_m from data

- We do this by generating the numbers $F_k = F(\boldsymbol{y}_k)$, $k = 1, \ldots, K$, where $\boldsymbol{y}_k$ are sample points in Γ

- Then the m-th moment is estimated by

$$M_m \approx \frac{1}{K} \sum_{k=1}^{K} F_k^m$$

  – this estimate looses accuracy as m increases so that in practice, one seldom goes beyond $m = 4$

- We have given a very bare-bones description of the use of Gram-Charlier expansions as a means for determining approximate output PDFs

  – in practice, there are many nuances about such expansions that have to be taken care of to render the useful for the intended purpose

    - one such "nuance" is that without making some additional assumptions, Gram-Charlier may be asymptotic expansions and not necessarily convergent ones

    - in such a case, there is an optimal number of terms i.e., adding more terms makes things worse

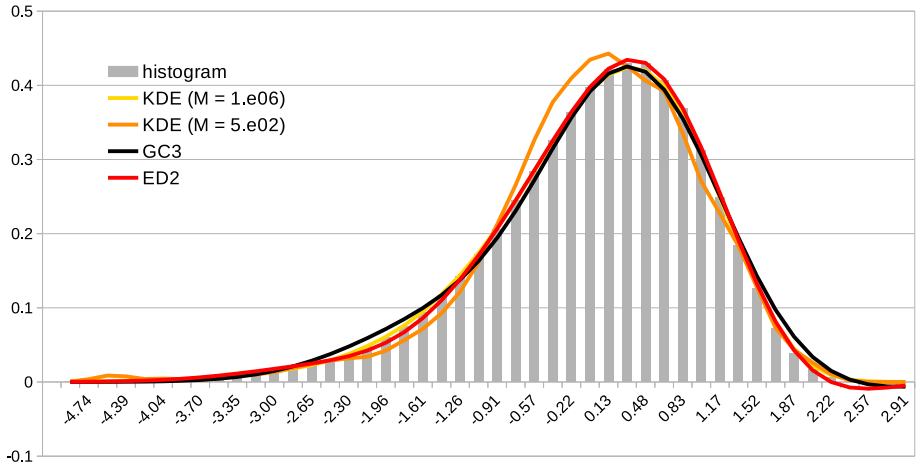
- There exist other moment expansions that can be used to estimate PDFs

  perhaps most notable are Edgeworth expansions

- We compare the use of Gram-Charlier and Edgeworth expansions with kernel density estimation (KDE)

CPU times for a single function evaluation [sec]						
	KDE ($K = 10^6$)		KDE ($K = 500$)		G-C	
N	Offline	Online	Offline	Online	Offline	Online
1	negligible	0.048431	negligible	0.000034	0.007968	negligible
2	negligible	0.047788	negligible	0.000036	0.012642	negligible
3	negligible	0.052763	negligible	0.000050	0.044448	negligible
4	negligible	0.047178	negligible	0.000037	0.334770	negligible

Computational times for a single evaluation of the KDE and the G-C methods (10^4 samples are used for the crude histogram).



KDE, Gram-Charlier, Edgeworth, and histogram approximations

Bayes

Bayes

- This is a homework assignment

REDUCED-ORDER MODELING

- "Reduced-order modeling" should be in quotation marks in the title slide because
  - on the one hand
    there are a huge number of data science tasks than one can reasonably characterized as being "reduced-order modeling"
      - in fact, anytime one replaces an expensive task with a cheaper one that still produces good enough outputs can be viewed as an exercise in reduced-order modeling
  - on the other hand
    there are communities in which "reduced-order modeling" has a narrower definition
      - as will be become apparent, here, we take the second view
      - but fear not, "reduced-order modeling" has been as is still alive and kicking in the other lectures

- The need for reduced-order modeling
  - the approximate solution of complex models using standard approaches is often expensive with respect to both storage and CPU costs
  - as a result, it is difficult if not impossible, to deal with a number of situations such as
    - continuation or homotopy methods for computing model solutions
    - parametric studies of model solutions (multiple model solutions)
    - optimization and optimal control problems (multiple model solutions)
    - feedback control settings (real-time model solutions)
    - uncertainty quantification (multiple model solutions)
    - . . . . . .
  - not surprisingly, lots of attention has been paid to
    reducing the costs expensive model solutions
    by using
    reduced-order models

- What do we mean by reduced-order modeling?
  - we have in hand a model $\Longrightarrow$ $\mathcal{M}(U;\alpha) = 0$
    α is a given input
    U is the desired output
    - so the task at hand is given α, determine U
  - the model may be a discrete (e.g. molecular dynamics) or continuous (e.g., partial differential equations)
    - in the latter case, one can seldom find an exact solution so that the continuous model is discretized
    - so, we go ahead and assume that the model $\mathcal{M}(U;\alpha) = 0$ is a discrete one
  - we are interested in settings (such as uncertainty quantification, optimization, ...) in which
    - determining the solution U of the model $\mathcal{M}(U;\alpha) = 0$ is computationally (very) expensive
    - many solutions U_i of the model $\mathcal{M}(U_i;\alpha_i) = 0$ corresponding to many inputs α_i are needed

– to ameliorate this situation

we construct a reduced-order model $\mathcal{M}_{rom}(U_{rom}; \alpha) = 0$

for which

- determining U_{rom} is computationally very inexpensive
- given α, U_{rom} is an acceptably accurate approximation of the expensive full-model solution U

– to construct the reduced-order models we consider

- one must first determine a "few" solutions of the the expensive full-order model $\mathcal{M}(U; \alpha) = 0$
- the high cost of determining those few expensive solutions

is hopefully amortized

by

using the reduced-order model $\mathcal{M}_{rom}(U_{rom}; \alpha) = 0$

(instead of the expensive full-order model)

to determine the large number of solutions needed

- The reduced-order models we construct are based on the collection of snapshots
  - snapshots are full-order model outputs U for several model inputs α that are costly to obtain
  - the snapshots are used to define a reduced-order model having outputs that are cheaper to obtain
- The reduced-order models of the type we consider are totally dependent on the information content in the snapshots
  - thus, a necessary (but not sufficient) condition for producing effective reduced-order models is the generation of "good" snapshots

    if it ain't in the snapshot set,
    it ain't in the reduced-order model
  - thus, producing "good snapshots" is key to the effectiveness of any reduced-order model

    - doing so is as much and art as it is a science

- All cases we consider are based on the construction and then use of a basis set

- We focus on four types of reduced-order models
  - for the first two, the basis is simply all the snapshots
    - the reduced-basis method
    - greedy reduced-basis method
  - the other two remove redundant or near redundant information from the snapshots so that the basis is smaller in dimension than the number of snapshots
    - proper orthogonal decomposition
    - ??????

SNAPSHOTS

- The solution of a complex model is determined by parameters that appear in the specification of the model
  - in physics or engineering models, parameters can appear in, e.g.,
    - geometrical specifications
    - initial and boundary conditions
    - source terms
    - media-dependent coefficients
    - . . . . . .

- The solution of the model also depends on independent variables appearing in the model
  - in physics or engineering models these could be spatial position and time

- Snapshot sets consist of full-order model solutions
  - corresponding to several parameter values
  - evaluated at several values of one or more of the independent variables
  - combinations of the two

- Thus, for examples, a snapshot set can consist of
  - full-order model solutions corresponding to several sets of design parameters
  - full-order model solutions for evaluated at several time instants during the evolution process
  - or both

- Snapshot sets are determined by obtaining solutions of a large-dimensional full-order discrete system
  - the discrete system could come in the form of
    - inherently discrete model
    - a discretization of a continuous model
    - experimental or observational data

- The generation of snapshot sets is an exercise in the design of experiments
- The basic question is
  - how does one choose the sets of parameters at which the model solutions are to be calculated (using expensive, high-fidelity, full-order computations) in order to generate the snapshot set?
  - clearly, some a priori knowledge about features of the model solutions are very useful in this regard; unfortunately, such knowledge is not always available
- There is certainly the need for

  developing systematic, rational, justifiable, and effective

  methodologies for generating good snapshot sets
- We will focus on the intelligent sampling of parameter space
  - unintelligent sampling of parameter space can result in
    - "bad" snapshot sets
    - the need to do "too many" high-dimensional full-order simulations

Sampling needs for snapshot generation

- Perhaps we only know bounds for the allowable values of the parameters
  e.g., $a_i \leq \alpha_i \leq b_i$
  - so, we need "intelligent" sampling in hyper-rectangles in parameter space

- Perhaps we know some other constraint relations between parameters
  e.g., $\sum_i \alpha_i^2 \leq 1$
  - so, we need "intelligent" sampling in more general regions in parameter space

- Without any additional information about the parameters
  - samples should be uniformly distributed

- If there is additional information available about parameters
  (correlation information, probability distributions, ...)
  - samples should be nonuniformly distributed

- Clearly, properly sampling parameter space is important
  1. in many cases, we do not want to adaptively update the reduced basis
     - thus, we only get one chance to generate snapshots that will be useful throughout many simulations needed in, e.g. an optimization or control process or an uncertainty quantification exercise
  2. we want to use as few runs of the high-dimensional simulation code as possible in order to generate the snapshots
     - thus, for the generation of snapshots, we want to run the simulation code for relatively few values of the parameters
  3. finally, there may be many parameters appearing in the description of the model that one may want to vary
     - thus, parameter sampling may be done in a high-dimensional parameter space

- Thus, we need methods for (1) effective and (2) sparse sampling in (3) regions in high dimensions

- Questions & answers

  – there are a huge number of available methods for sampling in hypercubes

    which one should be used for snapshot generation?

    - in an earlier lecture, we already discussed,
      at some length, point sampling in hypercubes

    - in a later lecture, we will add to that discussion

  – what about sampling in general regions and sampling nonuniformly?

    - in a later lecture, we also consider an effective
      sampling strategy for general regions and
      for nonuniform sampling

THE REDUCED-BASIS METHOD

- As was the case for "reduced-order models" which is referred to in a narrow sense when, in fact, that terminology is applicable to a much wider class of approximate models
  - the terminology "reduced-basis methods" is used to refer to a narrow class of reduced-order models which are based on using snapshots to define a low-dimensional basis which is then used to define a reduced-order model
  - in fact, "reduced-basis methods" commonly refers to such methods in which the snapshots themselves constitute the basis
  - this is in contract to POD reduced-order modeling (which we will look at shortly) in which the basis, while still using the snapshots, is defined by removing redundant or nearly redundant information from the snapshot set

- As a result

  approximate model solutions have the form of
  a linear combination of all the snapshots

- In this class of methods, there are many
  choices for the reduced basis
  - we consider
    - Lagrange bases
    - Hermite bases
    - Taylor bases

- Lagrange bases
  - Lagrange bases consist of snapshots of model solutions corresponding to several different values of the input parameter vector $\boldsymbol{\alpha}$

    - one first samples M parameter vectors $\{\boldsymbol{\alpha}_m\}_{m=1}^M$

    - for each parameter vector
      a full-order model solution $U(\boldsymbol{\alpha}_m)$ is obtained
      using a standard (and expensive) technique
      such as, in the PDE case, finite element or finite volume methods

    - the Lagrange reduced basis is then simply all M solutions $\{U(\boldsymbol{\alpha}_m)\}_{m=1}^M$

  - for a parameter $\boldsymbol{\alpha} \notin \{\boldsymbol{\alpha}_m\}_{m=1}^M$, the Lagrange reduced-basis approximation $U_{lagrange}(\boldsymbol{\alpha})$ then has the form

$$U_{lagrange}(\boldsymbol{\alpha}) = \sum_{m=1}^{M} C_m U(\boldsymbol{\alpha}_m)$$

– the coefficients $\{C_m\}_{m=1}^{M}$ are determined

by solving the model using the reduced basis

instead of the standard a standard full-order basis

such as a finite element basis

- Hermite bases
  - Hermite bases consist of snapshots of model solutions and derivatives of model solutions corresponding to several different values of the input parameter vector $\boldsymbol{\alpha}$ that has N components

    - one first samples M parameter vectors $\{\boldsymbol{\alpha}_m\}_{m=1}^{M}$

    - for each parameter vector $\boldsymbol{\alpha}_m$
      a model solution $U(\boldsymbol{\alpha}_m)$ is obtained
      as are
      the N derivatives $\{U^{(n)}(\boldsymbol{\alpha}_m)\}_{n=1}^{N}$ of $U(\boldsymbol{\alpha}_m)$ with respect each of the N components of the parameter vector

    - the model solution $U(\boldsymbol{\alpha}_m)$ is obtained by a standard (and expensive) full-order technique
      in fact, they are the same as those obtained for the Lagrange reduced-basis

- the derivatives $\{U^{(n)}(\boldsymbol{\alpha}_m)\}_{n=1}^{N}$ of the model solution are obtained by
  first differentiating the model equations with respect
  to each N components of the parameter vector
  and then
  solving the resulting differentiated models by a
  standard (and expensive) full-order technique

- note that if even if one has a nonlinear model, all the differentiated models are linear in the derivatives $U^{(j)}(\boldsymbol{\alpha}_n)$

- the Hermite reduced basis is then simply
  all M solutions $\{U(\boldsymbol{\alpha}_m)\}_{m=1}^{M}$
  plus
  all MN derivative solutions $\left\{\{U^{(n)}(\boldsymbol{\alpha}_m)\}_{n=1}^{N}\right\}_{m=1}^{M}$

– for a parameter $\boldsymbol{\alpha} \notin \{\boldsymbol{\alpha}_m\}_{m=1}^{M}$, the Hermite reduced-basis approximation $U_{hermite}(\boldsymbol{\alpha})$ then has the form

$$U_{hermite}(\boldsymbol{\alpha}) = \sum_{m=1}^{M} C_{m,0} U(\boldsymbol{\alpha}_m) + \sum_{m=1}^{M} \sum_{n=1}^{N} C_{m,n} U^{(n)}(\boldsymbol{\alpha}_m)$$

– the coefficients $\left\{\{C_{m,n}\}_{n=0}^{N}\right\}_{m=1}^{M}$ are determined by

solving the model using the reduced basis

instead of a standard full-order basis
such as a finite element basis

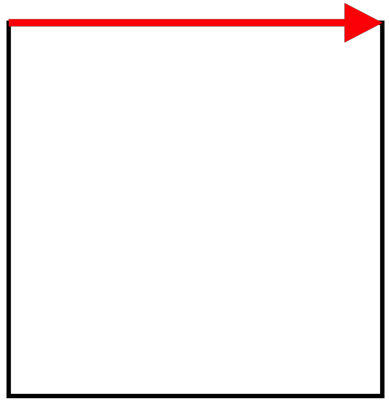
- Taylor bases
  - Taylor bases are like Hermite bases but which not not only
    include model solutions and first derivatives of model solutions
    but additionally
    include higher-order derivatives of model solutions

  - the number of higher-order derivatives grows very rapidly
    as the number of design parameters increase
    - e.g., if one has $N = 10$ design parameters
      there are 55 different second derivatives

  - thus, the dimension of the Taylor reduced basis grows
    quickly with the number of parameters and the number
    of derivatives used

Computational examples

- The driven cavity problem
  - flow in a box with the top lid moving from left to right

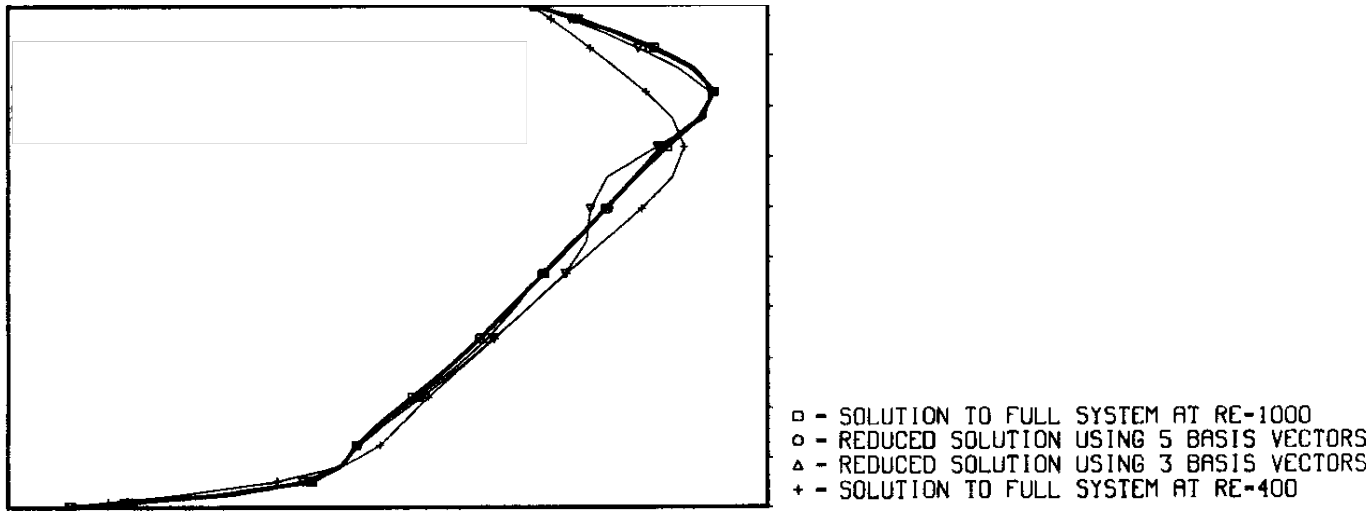


Horizontal velocity component along the vertical mid-line of the cavity

two Taylor bases generated at a Reynolds number = 400

shown are the full-model solutions for Reynolds number = 400 and 1000

also shown are the reduced-order solutions for Reynolds number = 1000
using Taylor bases with up to second derivatives and up to fourth derivatives



the basis is constructed taking snapshots with the Reynolds number 400

and is then used to produce solutions at Reynolds number 1000

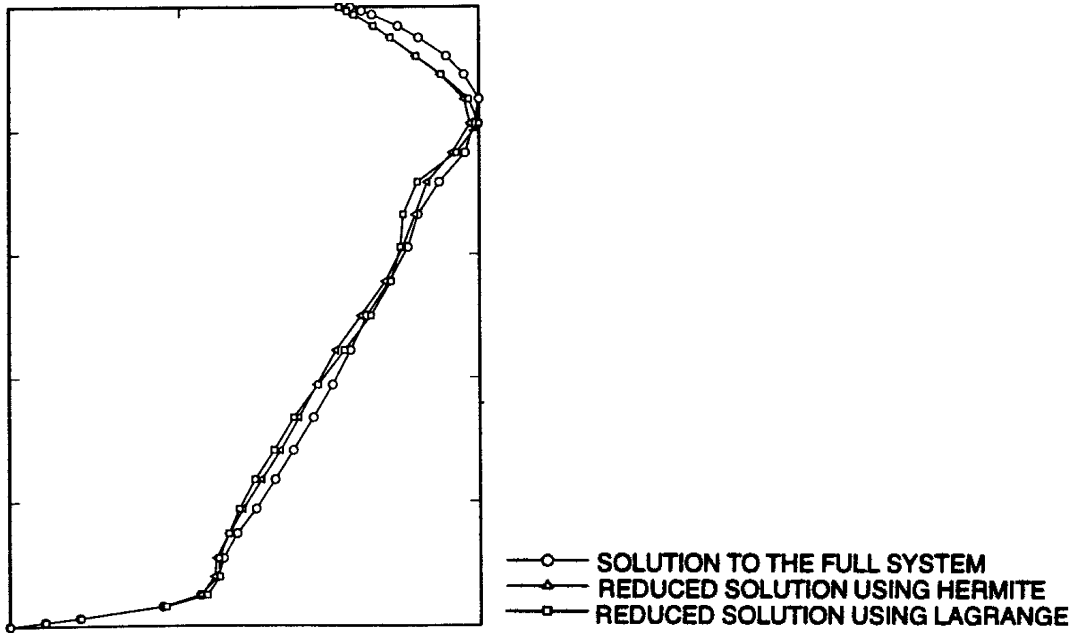
Horizontal velocity component along the vertical mid-line of the cavity

Lagrange basis generated using Reynolds numbers = 100, 300, 500, 700

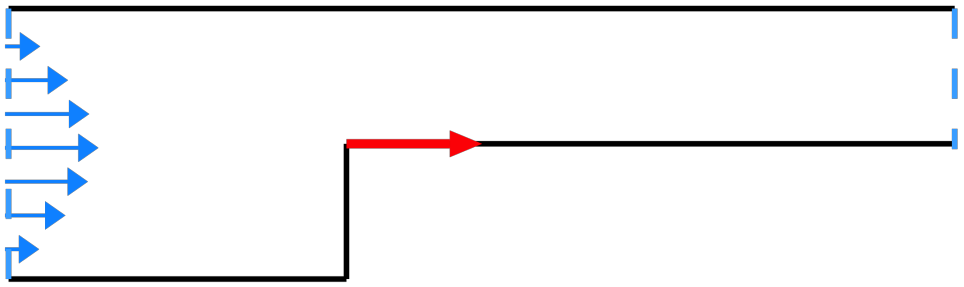
Hermite basis generated using Reynolds numbers = 500, 700

Shown are the full-model and the two reduced-basis solutions for Reynolds number = 1200

- note that this Reynolds number is larger than
  the Reynolds numbers used to generate the reduced basis



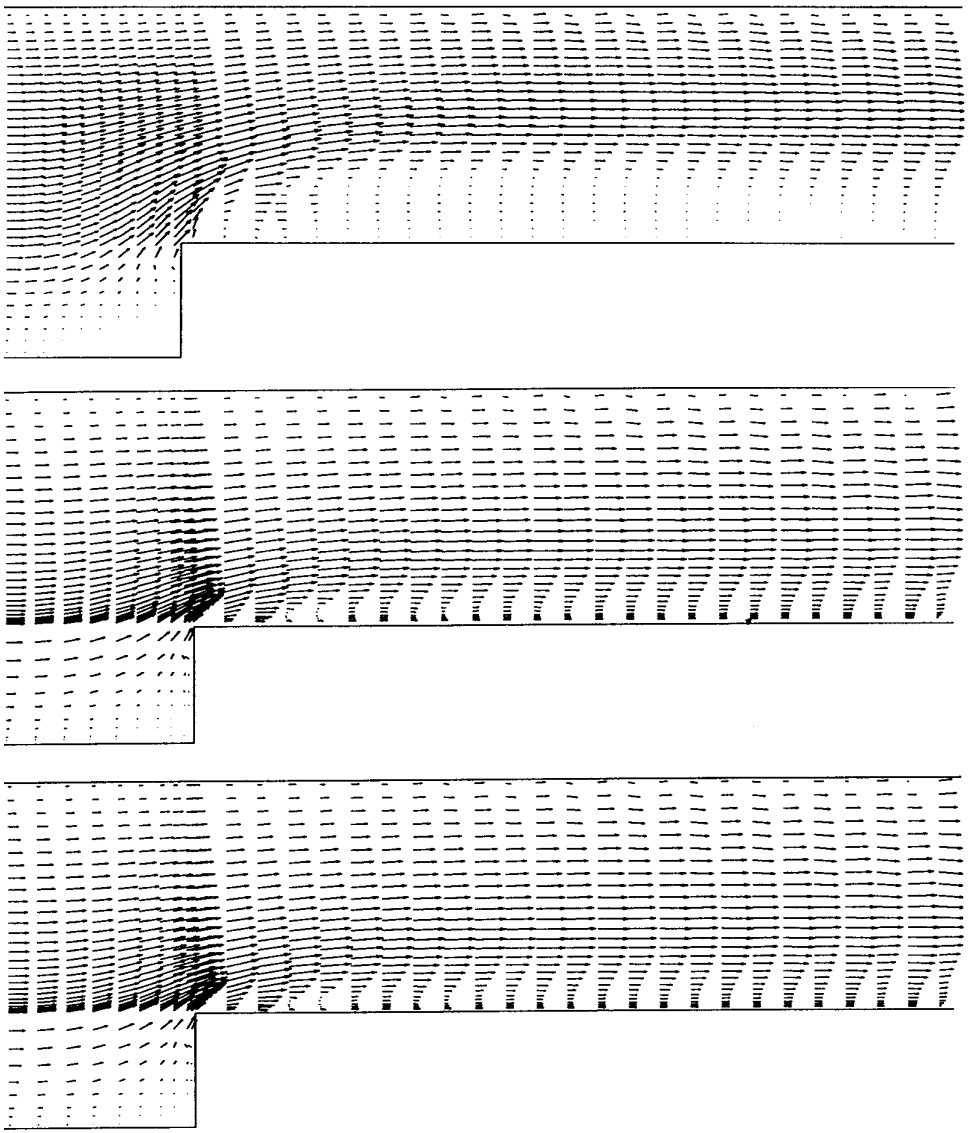
- Optimal control of flow over a forward facing step



- scalar control: constant horizontal velocity component
  along a section of the lower wall
  immediately behind step

- functional to be minimized: L^2 norm of the vorticity

- Lagrange basis functions were generated using
  6 values for the scalar control
  $0,\ 0.1,\ 0.15,\ 0.2,\ 0.25,\ 0.3$



Top to bottom: uncontrolled flow, full-model optimally controlled flow,
Lagrange reduced-order model optimally controlled flow

GREEDY REDUCED-BASIS METHODS

Greedy reduced-basis methods are attempts to sample snapshots more intelligently and thus reduce the number of full-order model solutions needed to construct a good reduced basis

Construction of the greedy reduced basis

- To begin
  - choose a training set Ξ_{train} of M_{train} points $\{\boldsymbol{z}_m\}_{m=1}^{M_{train}}$ in a sample domain Γ
    - these points could be chosen randomly or deterministically
    - to keep the exposition simple, we number the points in the order they are used
  - using the expensive full-order model compute the solutions $U_m^{fo}(\boldsymbol{z}_m)$, $m = 1, \ldots, M_{train}$, corresponding to $\boldsymbol{z}_m$

– choose an initial set of points $\{\boldsymbol{z}_m \in \Xi_{\text{train}}\}_{m=1}^{M_{initial}}$

- if $M_{initial} = 1$, i.e., only one point is chosen,
  it is usually chosen somewhere near the center of Γ

- if multiple points are chosen,
  then they can distributed uniformly in Γ
  or distributed non-uniformly if something is known a priori about
  where one should concentrate the points

- For $M = M_{initial}, M_{initial} + 1, \ldots$
  - assume we have in hand $\Longleftarrow$ clearly this is true for $M = M_{initial}$

    - a set of points $\Xi_M = \{\boldsymbol{z}_m\}_{m=1}^M \subset \Xi_{\text{train}}$

    - a set of corresponding expensive model solutions $\{U_m^{fo}(\boldsymbol{z}_m)\}_{m=1}^M$

    $\Longleftarrow$ this is the current candidate for the reduced basis
    - it contains M snapshots of the full-order model

  - the reduced-basis solution is a linear combination of the current set of snapshots

$$U_M^{rb}(\boldsymbol{z}) = \sum_{m=1}^{M} C_{m,M}(\boldsymbol{z}) U_m^{fo}(\boldsymbol{z}_m)$$

  where the coefficients $\{C_{m,M}(\boldsymbol{z})\}_{m=1}^M$ are determined by solving the model using the reduced basis

- The next step is to determine the point

$$\boldsymbol{z}_{\widetilde{m}} \in \Xi_{train} \setminus \{\boldsymbol{z}_m\}_{m=1}^{M}$$

i.e., a point in the training set that is not one of the points already visited

such that, in some suitable norm,

$$\|U_M^{rb}(\boldsymbol{z}_{\widetilde{m}}) - U_{\widetilde{m}}^{fo}(\boldsymbol{z}_{\widetilde{m}})\| \geq \|U_M^{rb}(\boldsymbol{z}_m) - U_{\widetilde{m}}^{fo}(\boldsymbol{z}_m)\|$$

for all $m = M+1, \ldots, M_{train}$

$\Longleftarrow$ the unvisited point $\boldsymbol{z}_{\widetilde{m}}$ in the training set at which the reduced-basis solution differs the most from the full-order solution

- if $\|U_M^{rb}(\boldsymbol{z}_{\widetilde{m}}) - U_{\widetilde{m}}^{fo}(\boldsymbol{z}_{\widetilde{m}})\|$ is less than a desired tolerance

  - terminate

  - and define the final reduced basis by

$$\{U_m^{fo}(\boldsymbol{z}_m)\}_{m=1}^{M}$$

- if $\|U_M^{rb}(\boldsymbol{z}_{\widetilde{m}}) - U_{\widetilde{m}}^{fo}(\boldsymbol{z}_{\widetilde{m}})\|$ is greater than a desired tolerance

  continue the loop by

  - choosing $\boldsymbol{z}_{M+1} = \widetilde{\boldsymbol{z}}$ and $U_{M+1}^{fo}(\boldsymbol{z}_{M+1}) = U_M^{fo}(\boldsymbol{z}_{\widetilde{m}})$

  - adding $U_{M+1}^{rb}(\boldsymbol{z}_{M+1})$ to the reduced basis

    $\Longrightarrow$ we have the new reduced basis $\{U_m^{fo}(\boldsymbol{z}_m)\}_{m=1}^{M+1}$

- After termination, one can determine a reduced-basis approximation model solution at any $\boldsymbol{z} \in \Gamma$
  - of course, as is true about any decent reduced-order model, this can be done cheaply because the number of reduced-basis degrees of freedom is much smaller than the number of full-order degrees of freedom

- Unfortunately, this is not a practical procedure because it requires M_{train} solutions of the expensive full-order model
  - the number M_{train} has to be sufficiently large to effect good coverage of the sampling domain Γ
  - obtaining a large number of full-order soluions
    is exactly what we are trying to avoid!

- This bottleneck caused by
  - using the norm of the difference between the reduced-basis and full-order solutions

  can be avoided by instead

  - using the residual obtained when one plugs in the reduced-basis approximation into the full-order model

$$\text{residual}_m = \mathcal{M}\left(U_M^{rb}(\boldsymbol{z}_m); \alpha)\right) \qquad \text{for } m = M+1, \ldots, M_{train}$$

    and then choosing, to add to the reduced basis, the snapshot $\boldsymbol{z}_m$ which yields the largest residual

- The steps taken to determine the final reduced bases are much the same for the impractical approach and the improved residual-based approach, so we do not provide details about the latter

- We also do not present anything about other variants for greedy-type reduced-basis modeling or about available convergence estimates

PROPER ORTHOGONAL DECOMPOSITION (POD)

Principal Component Analysis **Empirical Orthogonal Eigenfunctions**

Karhunen-Loève Expansions **Singular Value Decompostion**

- Given M snapshot N-vectors $\{\boldsymbol{z}_m\}_{m=1}^M$

- For $K \leq M \leq N$, let $\{\boldsymbol{\psi}_k\}_{k=1}^K$ denote an arbitrary orthonormal basis of N-vectors

$$\Longrightarrow \qquad \boldsymbol{\psi}_{k'}^T \boldsymbol{\psi}_k = \delta_{kk'} \quad \text{for } k, k' = 1, \ldots, K$$

- Let $P_{\psi,K}\boldsymbol{z}_m$ denote the projection of a snapshot $\boldsymbol{z}_m$ onto the K-dimensional space spanned by $\{\boldsymbol{\psi}_k\}_{k=1}^K$

$$\text{for} \quad m = 1, \ldots, M \quad \Longrightarrow \quad \begin{cases} P_{\psi,K}\boldsymbol{z}_m = \displaystyle\sum_{k=1}^{K} c_{mk}\boldsymbol{\psi}_k \\ \\ c_{mk} = \boldsymbol{\psi}_k^T \boldsymbol{z}_m \quad \text{for} \quad k = 1, \ldots, K \end{cases}$$

  - thus $P_{\psi,K}\boldsymbol{z}_m$ is an approximation of the snapshot $\boldsymbol{z}_m$

– then $\boldsymbol{z}_m - P_{\psi,k}\boldsymbol{z}_m$ denotes

the difference between a snapshot $\boldsymbol{z}_m$
and its projection $P_{\psi,K}\boldsymbol{z}_m$
onto the subspace spanned by $\{\boldsymbol{\psi}_k\}_{k=1}^K$

- Let the POD error be defined by

$$\mathcal{E}_{\text{pod}} = \sum_{m=1}^{M} |\boldsymbol{z}_m - P_{\psi,k}\boldsymbol{z}_m|^2 = \sum_{m=1}^{M} \left|\boldsymbol{z}_m - \sum_{k=1}^{K} (\psi_k^T \boldsymbol{z}_m)\psi_k\right|^2$$

$\Longrightarrow$ the sum of the squares of the differences between
the snapshots $\boldsymbol{z}_m$ and their projection $P_{\psi,k}\boldsymbol{z}_m$

- We then pose the minimization problem

minimize
over all K-dimensional orthonormal sets $\{\boldsymbol{\psi}_k\}_{k=1}^K$ of N-vectors
$\sum_{m=1}^{M} \left|\boldsymbol{z}_m - \sum_{k=1}^{K} (\psi_k^T \boldsymbol{z}_m)\psi_k\right|^2$

- The POD basis of dimension K is defined as the set $\{\phi_k\}_{k=1}^K$ that solves this minimization problem

- Let $\mathbb{S}$ denote the $N \times M$ snapshot matrix

  i.e., the matrix

$$\mathbb{S} = \left(\boldsymbol{z}_1 \, , \; \boldsymbol{z}_2 \, , \; \ldots \; , \; \boldsymbol{z}_M \right)$$

  whose columns are the snapshots $\boldsymbol{z}_m$, $m = 1, \ldots, M$

  – consider the singular value decomposition (SVD)

$$\underbrace{\mathbb{S}}_{N \times M} = \underbrace{\mathbb{U}}_{N \times N} \; \underbrace{\boldsymbol{\Sigma}}_{N \times M} \; \underbrace{\mathbb{V}^T}_{M \times M}$$

  where

  $\mathbb{U}$ is an $N \times N$ orthonormal matrix

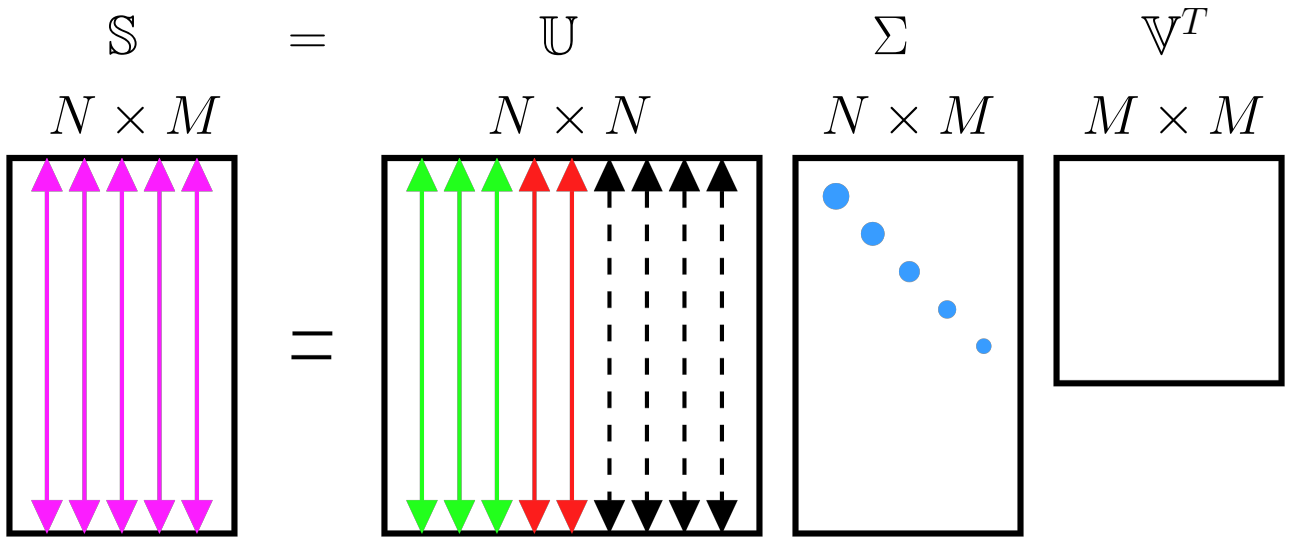
  $\mathbb{V}$ is an $M \times M$ orthonormal matrix

  $\boldsymbol{\Sigma}$ is an $N \times M$ diagonal matrix with non-increasing diagonal entries[1] (the singular values)

$$\sigma_1 \geq \sigma_2 \geq \; \cdots \; \geq \sigma_M \geq 0$$

  – then, the POD basis of dimension $K \leq M \leq N$ is given by the first K columns of $\mathbb{U}$

[1] Some singular values vanish if the snapshot matrix $\mathbb{S}$ does not have full column rank



magenta: $M = 5$ snapshot vectors

green + red: the first $M = 5$ columns of $\mathbb{U}$ span the same 5-dimensional space as do the 5 columns of $\mathbb{S}$

green: the first $K = 3$ columns of $\mathbb{U}$ form a POD basis of dimension 3

blue: the $M = 5$ singular values which appear in non-increasing size

- It is easy to show that the error of the K-dimensional POD subspace is given by

$$\mathcal{E}_{\text{pod}} = \sum_{m=K+1}^{M} \sigma_m^2$$

  - thus, if one wishes for the relative error to be less than a prescribed tolerance ϵ, i.e., if one wants

$$\mathcal{E}_{\text{pod}} \leq \delta \sum_{m=1}^{M} |\boldsymbol{z}_m|^2$$

  one should

choose K to be the smallest integer such that

$$\frac{\sum_{m=K+1}^{M} \sigma_m^2}{\sum_{m=1}^{M} \sigma_m^2} \geq 1 - \delta$$

- in practice, we truncate the sums at some $M_{trucate} > K$

Variations on POD

There have been several variations introduced in attempts to "improve" POD

- Weighted POD – gives more weight to some members of the snapshot set
  - can be accomplished, e.g., by including multiple copies of an "important" snapshot in the snapshot set

- POD with derivatives – get more information into the snapshot set in order to get a "better" POD basis
  - add derivatives or numerical approximations of derivatives of model solutions to the snapshot set

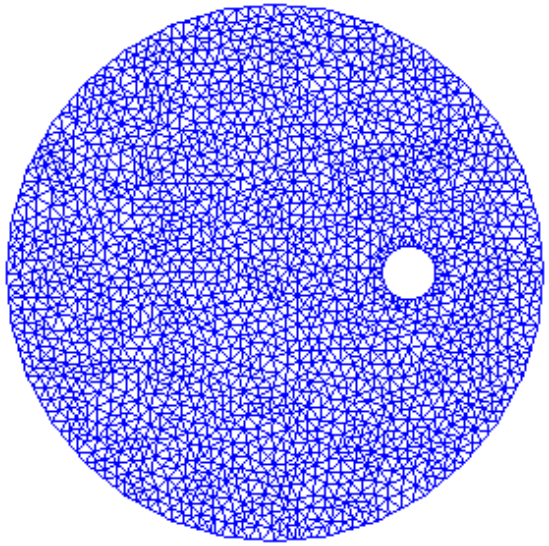
- H^1 POD – change the error measure for POD in order to get a "better" POD basis
  - use H^1 norms and inner products (instead of L^2) in the definition and construction of POD bases

- Constrained POD – impose a constraint
  - e.g. - symmetry of the POD basis
    - basis that leads to conservation of energy
    - basis that preserve the Hamiltonian structure of the full-order model

- Adaptive POD – change the POD basis when
  it no longer seems to be working
  - requires detection of failure of the POD basis
  - requires the determination of new snapshot vectors
  - naively requires doing a new SVD of the new snapshot matrix
    - there are means for partially avoiding this naive approach

POD for partial differential equations

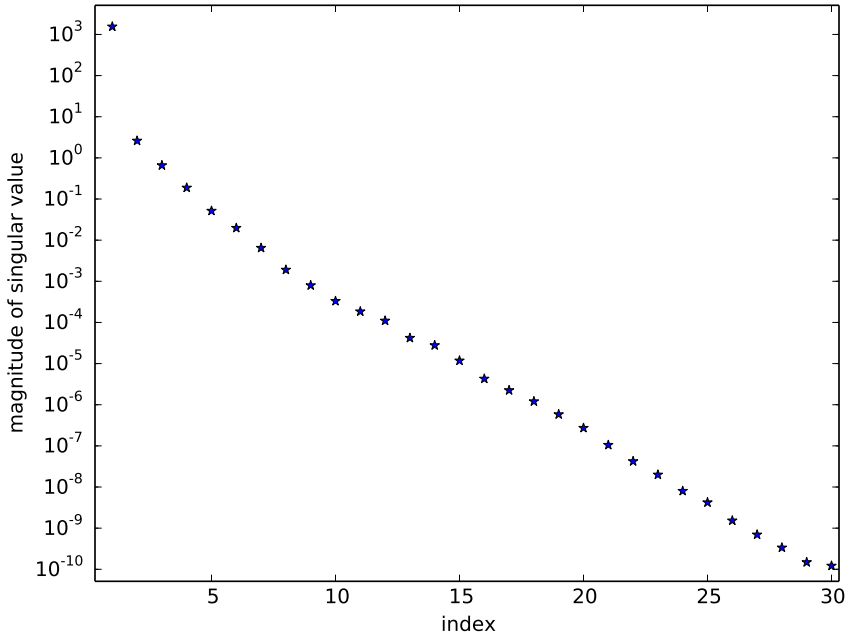
- We consider the two-dimensional Navier-Stokes flow between two offset circles driven by a counterclockwise rotational body force
  - the Navier-Stokes equations are discretized via a finite-element method having 16457 degrees of freedom
    - i.e., we are dealing with snapshot vectors having dimension $N = 16457$
    - this is what the geometry looks like and what a grid with 7405 degrees of freedom looks like



– time-stepping is effected over the time interval $[0, 5]$ using a time step $\Delta t = 0.01$sec

- Snapshots are taken every 0.04sec so that we end up with $M = 125$ snapshots

  - thus the snapshot matrix $\mathbb{S}$ is an 16457×125 matrix



The 30 largest singular values of the snapshot matrix

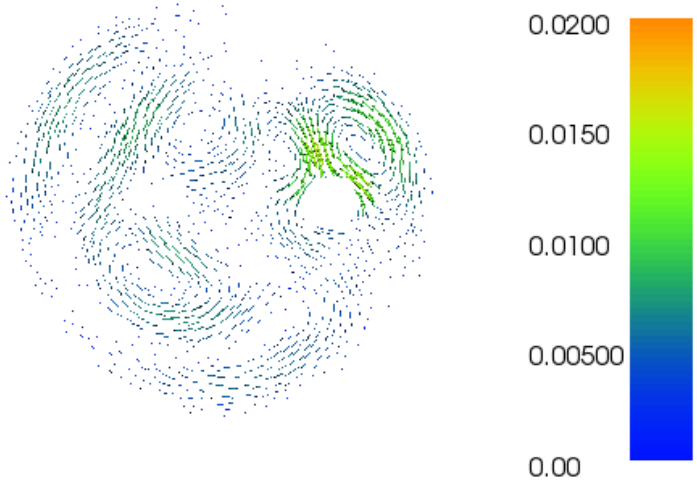
- We compare the performance of POD approximation in two settings

  1. a data mining setting

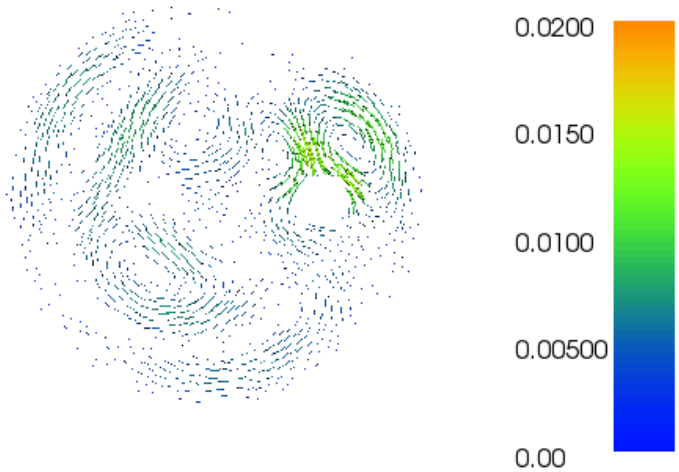
     - we compare the POD approximation with the full finite element solution for the case in which all model inputs are the same as those used to generate the snapshots

  2. a prediction setting

     - we again compare the POD approximation with the full finite element solution for but now model inputs are different from those used to generate the snapshots



data mining



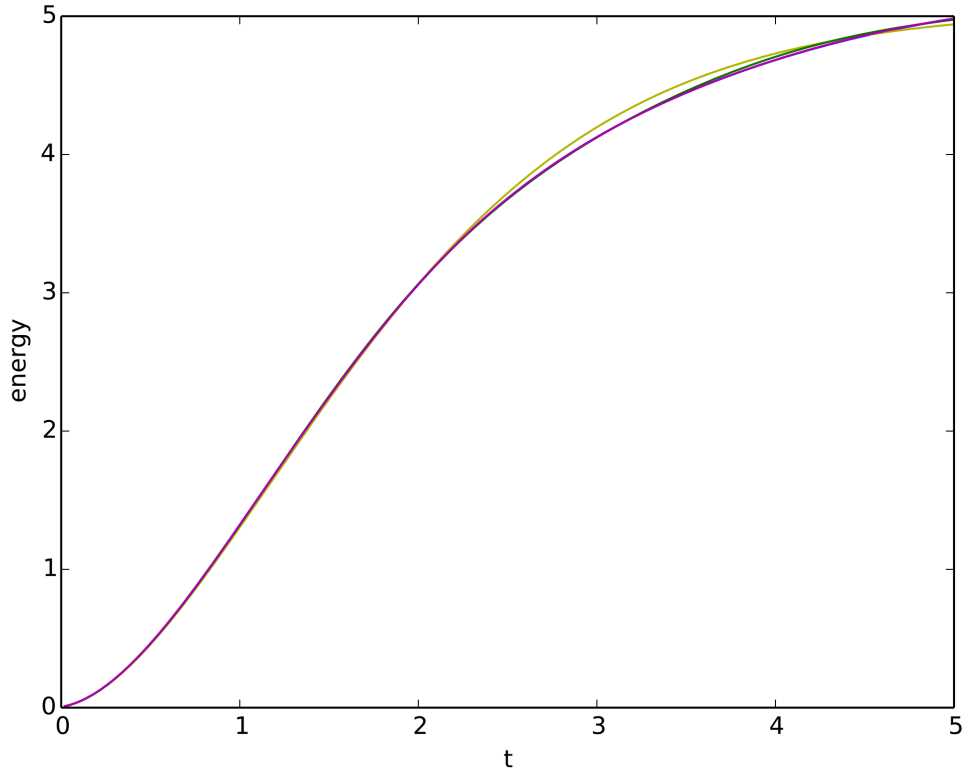
prediction

At the final time $T = 5$

the difference between the velocity fields

of the full finite element solution and

the POD approximation with $K = 6$ POD basis



For the prediction setting and for $0 \leq t \leq 5$, the energy
determined from the the full finite element solutions
and from the POD approximation of dimensions $K = 2,\ 4,\ 6$

data mining

K	error
2	0.035785
3	0.021379
4	0.013802
5	0.009067
6	0.004886

prediction

K	error
2	0.035869
3	0.021437
4	0.013910
5	0.009073
6	0.004969

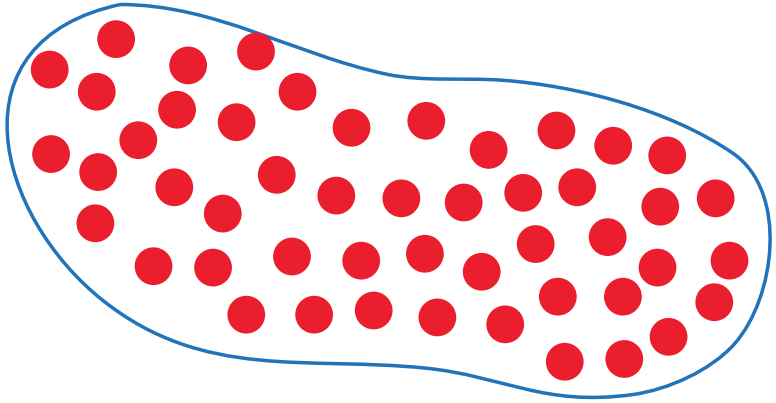
The L^2 relative error between

the full finite element solution

and the POD approximations of dimensions $K = 2$ to 6

POD for molecular dynamics

- We have a system of particles (point masses)



- MD calculations look deceptively simple
  - one has a system of ODEs to solve
- However, MD calculations are expensive
  - zillions of atoms are involved $\Longrightarrow$ huge ODE systems
  - long-time calculations may be needed $\Longrightarrow$ huge number of time steps
  - work per time step can be big

- Let

  $D =$ spatial dimension

  $Q =$ the number of particles

  $\boldsymbol{z}_q(t)$ denote the $3D$ vector of the position of particle q at time t

  $\boldsymbol{Z}(t)$ denote the $N = 3DQ$ vector of all the particle positions at time t

$$\Longrightarrow \qquad \boldsymbol{Z}(t) = \begin{pmatrix} \boldsymbol{z}_1(t) \\ \boldsymbol{z}_2(t) \\ \vdots \\ \vdots \\ \boldsymbol{z}_{Q-1}(t) \\ \boldsymbol{z}_Q(t) \end{pmatrix}$$

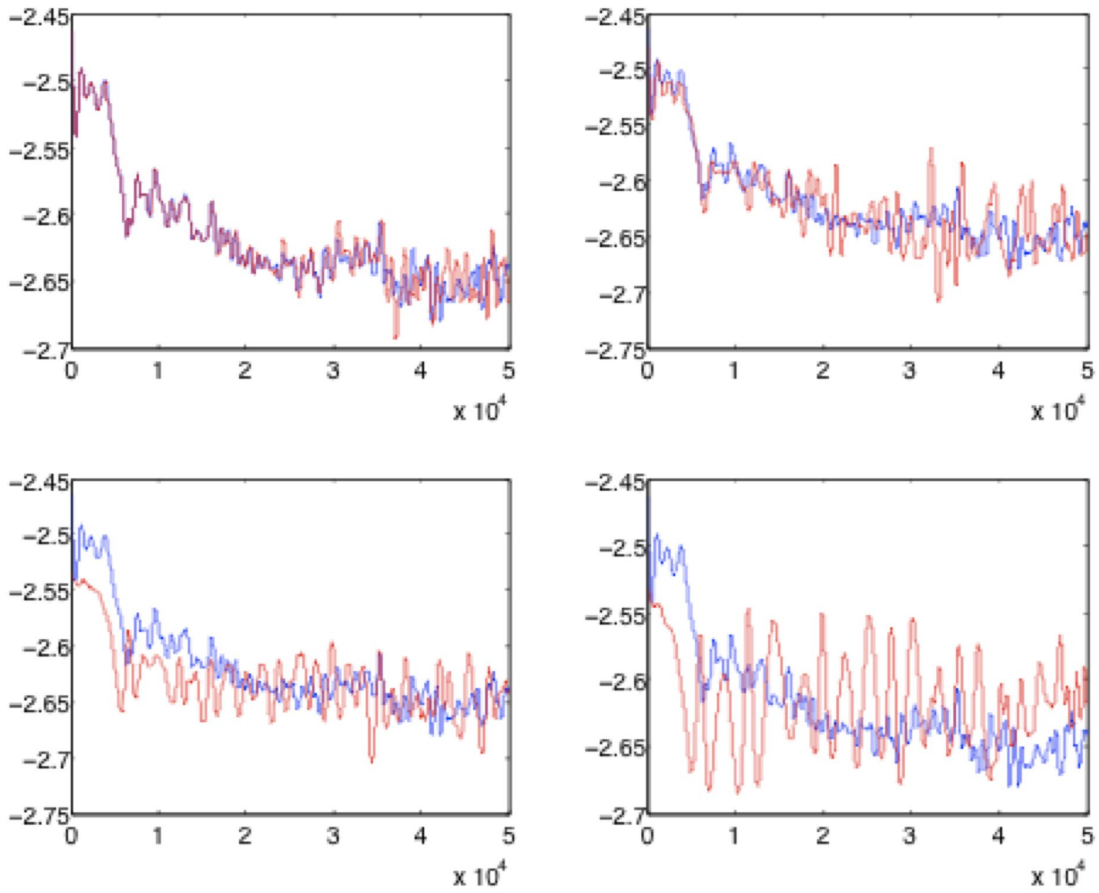
- Snapshots are determined by sampling at K time instants
  - we then have the $3DQ \times K$ snapshot matrix

$$\Longrightarrow \qquad \mathbb{S} = \begin{pmatrix} \boldsymbol{z}_1(t_1) & \boldsymbol{z}_1(t_2) & \boldsymbol{z}_1(t_3) & \dots & \boldsymbol{z}_1(t_K) \\ \boldsymbol{z}_2(t_1) & \boldsymbol{z}_2(t_2) & \boldsymbol{z}_2(t_3) & \dots & \boldsymbol{z}_1(t_K) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{z}_Q(t_1) & \boldsymbol{z}_Q(t_2) & \boldsymbol{z}_Q(t_3) & \dots & \boldsymbol{z}_Q(t_K) \end{pmatrix}$$

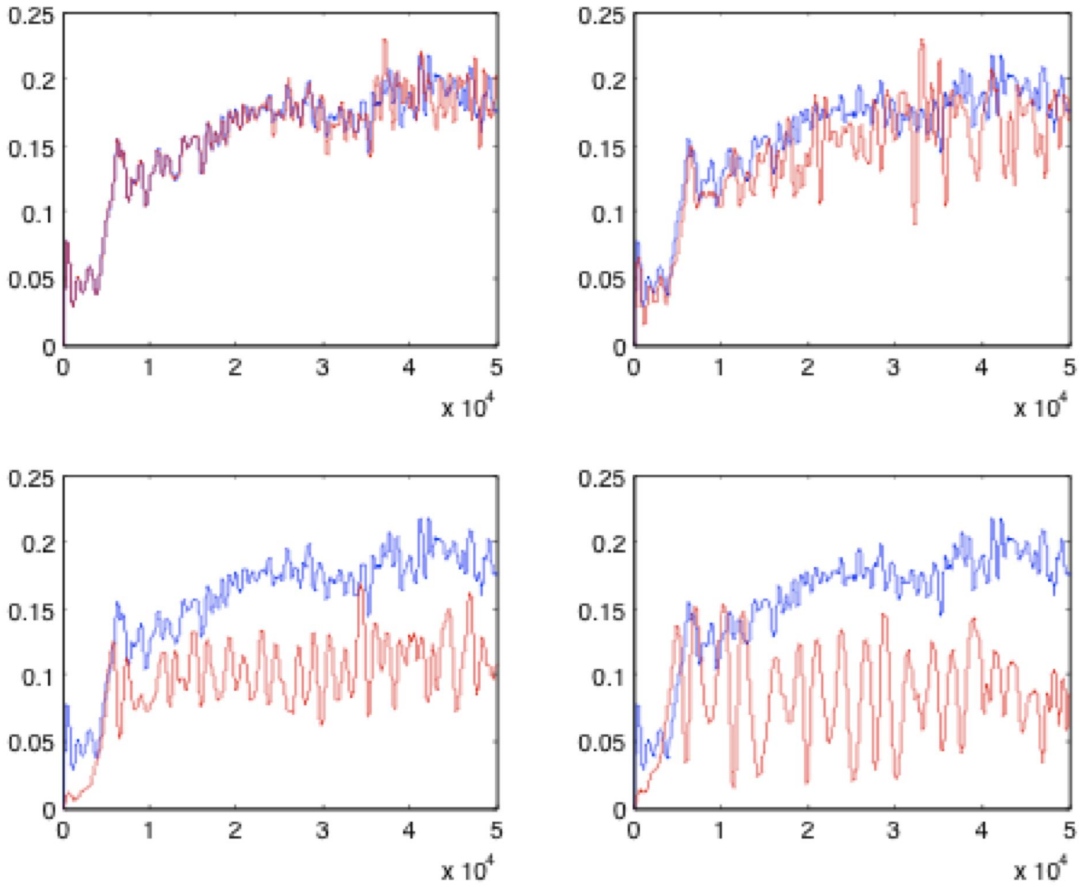
  - note that that the rows of $\mathbb{S}$ are samplings of the trajectories of the particles

- We then define the POD basis based on the singular value decomposition $\mathbb{S}$

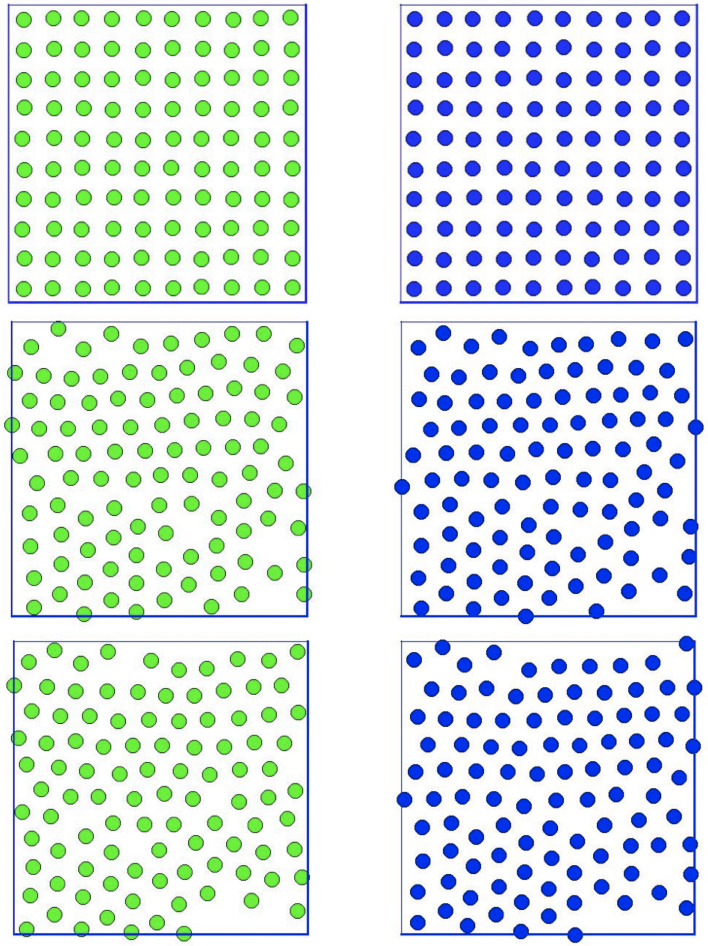
- Toy computational example
  - we consider a case with 100 particles in a 2-dimensional box
    - the dimension N of the snapshot vectors is 200
  - we choose PODs based on $K = 100$, 50, 25, and 10 POD basis functions
  - we plot the potential energy and kinetic energies of the system vs. time for the full MD solutions and for the POD approximations solutions
  - we also plot, at a few time instants, the positions of the 100 particles as determined by both the full MD and POD models



Potential energy of 100 atom system vs. time as determined from the full MD simulation (blue) and ROM simulation (red); dimension of ROM basis is 100 (top left), 50 (top right), 25 (bottom left), and 10 (bottom right)



Kinetic energy of 100 atom system vs. time as determined from the full MD simulation (blue) and ROM simulation (red); dimension of ROM basis is 100 (top left), 50 (top right), 25 (bottom left), and 10 (bottom right)



Position of the atoms at time steps 1, 250, and 500 (top to bottom)

Left: full MD simulation

Right: POD simulation with a POD basis of dimension $K = 50$

BUT, WHAT IS SWEPT UNDER THE RUG?
WHAT IS THE DIRTY LITTLE SECRET?

- First recall the reduced-order modeling mantra
  - we are willing to pay the off-line cost incurred by obtaining snapshots through the use of an expensive model $\mathcal{M}(U;\alpha)$

  because then
  - we can amortize that cost over zillions of on-line reduced-order model $\mathcal{M}_{rom}(U_{rom};\alpha)$ solutions

- Thus, reduced-order modeling is wonderful
  - once we have assembled the reduced order model $\mathcal{M}_{rom}(U_{rom};\alpha)$ we can indeed use it to obtain zillions of solutions at very little cost
  - so what's the problem?

- Here's the problem

- Whereas
  - the solution of a ROM $\mathcal{M}_{rom}(U_{rom};\alpha)$ can be obtained

    at a cost that depends only on the
    dimension K of the reduced basis

    so that

    the cost does not depend on the
    dimension N of the snapshots,
    i.e., on the dimension of full-order
    model $\mathcal{M}(U;\alpha)$

  - the simplest approach towards the construction of a ROM $\mathcal{M}_{rom}(U_{rom};\alpha)$

    incurs online costs that depend on the dimension
    of the N of the snapshot vectors

- Of course, having on-line costs that depend on the dimension of the full-order model $\mathcal{M}(U;\alpha)$ defeats the very purpose of doing reduced-order modeling

- It is then not a surprise that there has been considerable efforts directed at getting around this bottleneck
  - some of which have been successful for wide class of models

  but
  - all of which are limited in scope with regards to their application to general classes of models

MULTILEVEL AND MULTIFIDELITY METHODS

SETTING

- Given inputs
  - random N-vectors $\boldsymbol{y}$
    - each component y_n, $n = 1, \ldots, N$, is randomly chosen
  - a mapping that transforms an N-vector $\boldsymbol{y}$ to a J-vector $\boldsymbol{U}(\boldsymbol{y})$
    - of course, the J-vector is then also a random vector

- Outputs
  - step-one outputs: for each $m = 1, \ldots, M$,
    - we choose an input vector $\boldsymbol{y}_m$

    and
    - then determine the corresponding output J-vector $\boldsymbol{U}_m(\boldsymbol{y}_m)$
  - step-two output: determine
    - a scalar QoI (quantity of interest) that depends on $\{\boldsymbol{U}_m\}_{m=1}^M$

- Setting
  - obtaining $\boldsymbol{U}_m$ from $\boldsymbol{y}_m$ is very costly
  - to obtain an accurate QoI, M has to be large
    - we have to obtain many expensive $\boldsymbol{U}_m$

- To provide a concrete context, we consider

  multifidelity methods in the context of uncertainty quantification (UQ)

  and

  a QoI of the form

$$\text{QoI}_M(\Phi) = \sum_{m=1}^{M} w_m \rho(\boldsymbol{y}_m) \Phi(\boldsymbol{y}_m)$$

  where $\Phi(\boldsymbol{y}) = G\big(\boldsymbol{U}(\boldsymbol{y})\big)$ and $\rho(\boldsymbol{y})$ is a given PDF

  - e.g., if $G(\boldsymbol{U}) = \boldsymbol{U} \implies \text{QoI}_M(\Phi) \approx \text{E}(\boldsymbol{U}) =$ expected value of $\boldsymbol{U}$

    if $G(\boldsymbol{U}) = \big(\boldsymbol{U} - \text{E}(\boldsymbol{U})\big)^2 \implies \text{QoI}_M(\Phi) \approx \text{V}(\boldsymbol{U}) =$ variance of $\boldsymbol{U}$

  – for example, we can have that $\text{QoI}_M(\Phi)$ is a quadrature rule approximation of a stochastic integral

$$\text{QoI}_M(\Phi) = \sum_{m=1}^{M} w_m \rho(\boldsymbol{y}_m) \Phi(\boldsymbol{y}_m) \approx \int_{\Gamma} \Phi(\boldsymbol{y}) \rho(\boldsymbol{y}) d\boldsymbol{y}$$

  where w_m and and $\boldsymbol{y}_m$ denote quadrature weights and points

Goal of multifidelity methods for UQ

- We have
  - the high-fidelity model $\Leftarrow$ the expensive and accurate model we want to use
  - a parameter sampling scheme

    $\Longrightarrow$ high-fidelity model + sampling scheme

    $\Rightarrow$ accurate but costs too much

- We also have some surrogate models $\Leftarrow$ cheaper and less accurate models

  $\Longrightarrow$ any surrogate model by itself + sampling scheme

  $\Rightarrow$ costs less but is also less accurate

- Goal: find ways to

  combine {high-fidelity model + surrogate models} + sampling scheme

  $\Rightarrow$ so that total costs are lower but accuracy is the same as that of the high-fidelity model

- Several choices for models and sampling schemes

Models of varying fidelity, all depending on the parameters y	Parameter sampling schemes
interpolants	Monte Carlo
least-squares approximations	quasi-Monte Carlo
reduced-order models	Latin hypercube
simplified physics	CVT
coarser grid sizes	sparse grids
unconverged iterations	.....
machine learning models	
experimental data	
.....	

  - we will look at three choices

- Multilevel Monte Carlo
  - truth model = expensive (e.g., a fine-grid discretization of a PDE)
  - surrogates and sampling scheme used $\Longrightarrow$

Deterministic models of varying fidelity all depending on the parameters $\boldsymbol{y}$	Parameter sampling schemes
interpolants	Monte Carlo
least-squares approximations	quasi-Monte Carlo
reduced-order models	Latin hypercube
simplified physics	CVT
coarser grids	sparse grids
unconverged iterations	.....
machine learning models	
experimental data	
.....	

  - more generally, we can have sequence of parameters such that
    as they increase, the models become cheaper but are less accurate

- Multilevel stochastic collocation
  - truth model = expensive (e.g., a fine grid discretization of a PDE)
  - surrogates and sampling scheme used $\Longrightarrow$

Deterministic models of varying fidelity all depending on the parameters $\boldsymbol{y}$	Parameter sampling schemes
interpolants	Monte Carlo
least-squares approximations	quasi-Monte Carlo
reduced-order models	Latin hypercube
simplified physics	CVT
coarser grids	sparse grids
unconverged iterations	.....
machine learning models	
experimental data	
.....	

  - more generally, we can have sequence of parameters such that
    as they increase, the models become cheaper but are less accurate

– We also consider a more general mutifidelity method

- several different types of surrogates are used

- sampling via Monte Carlo

Monte Carlo methods

- Classical Monte Carlo (MC) methods determine the approximation to a QoI by the simple random sampling and averaging formula

$$\text{QoI}_{M_{mc},h_{hf}} = \frac{1}{M_{mc}} \sum_{m=1}^{M_{mc}} G_{h_{hf}}(\boldsymbol{y}_m)$$

$$\{\boldsymbol{y}_m\}_{m=1}^{M_{mc}} \Leftarrow M_{mc} \text{ i.i.d. random points in } \Gamma$$

where $G_{h_{hf}}(\boldsymbol{y}_m) = G\Big(\big(F\big(\boldsymbol{U}_{h_{hf}}(\boldsymbol{y}_m)\big)\big)\Big) \Leftarrow$ depends on $\boldsymbol{U}_{h_{hf}}(\boldsymbol{y}_m)$ which is expensive to obtain because h_{hf} is small

  - h_{hf} is chosen so that, e.g.,

    the "error" $\sum_{m=1}^{M} w_m \rho(\boldsymbol{y}_m)\Phi(\boldsymbol{y}_m) - \int_\Gamma \Phi(\boldsymbol{y})\rho(\boldsymbol{y})d\boldsymbol{y} = \mathcal{O}(h_{hf}^{\alpha}) = \mathcal{O}(\epsilon)$

    where ϵ is a prescribed tolerance

    - so that smaller the tolerance, the greater the cost

– the number of samples M_{mc} is chosen so that the $O(1/\sqrt{M_{mc}})$ sampling error is commensurate with the $O(h_{hf}^{\alpha})$ error

– in this way the total error due to both sources is of $O(\epsilon)$

– all

M_{mc} outputs $\{\boldsymbol{U}_{hf}(\boldsymbol{y}_m))\}_{m=1}^{M_{mc}}$

of the model corresponding to the

M_{mc} input sample points $\{\boldsymbol{y}_m\}_{m=1}^{M_{mc}}$

are determined

using the h_{hf}

- Multilevel Monte Carlo (MLMC) methods are designed to obtain an approximation of the QoI that has the same nominal accuracy ϵ but at less cost

Multilevel Monte Carlo methods

- MLMC methods make use of a hierarchy of grid sizes

$$h_l = \frac{h_{l-1}}{\eta} \qquad \text{or} \qquad h_l = \frac{h_0}{\eta^l} \qquad \text{for } l = 1, \ldots, L$$

  - h_0 is a given coarse grid size
  - $\eta > 1$ is usually chosen to be an integer (most often 2)
  - L is chosen as $h_L = h_{hf}$ so that there is hope that the approach meets the same accuracy threshold ϵ
    - the actual MLMC algorithm then ensures that this is a certainty

- MLMC methods also make use of a hierarchy of MC quadrature rules

$$\mathcal{Q}_{M_l}^{mc}(\Phi) = \frac{1}{M_l} \sum_{m_l=1}^{M_l} \Phi(\boldsymbol{y}_{m_l}) \approx \int_\Gamma \Phi(\boldsymbol{y}) \rho(\boldsymbol{y}) \, d\boldsymbol{y} \qquad \text{for } l = 0, 1, \ldots, L$$

  using the randomly sampled points $\{\boldsymbol{y}_{m_l}\}_{m_l=1}^{M_l}$ in Γ

- At each level l and for any $\boldsymbol{y} \in \Gamma$, we have the approximate model output $\boldsymbol{U}_{h_l}(\boldsymbol{y})$

- We can then evaluate, for each level l and for any $\boldsymbol{y} \in \Gamma$, the approximation $G_{h_l}(\boldsymbol{y}) = G\Big(\big(F\big(\boldsymbol{U}_{h_l}(\boldsymbol{y})\big)\Big)$ of the integrand $G(\boldsymbol{y})$ in the QoI

- Then, for each level l, we have the spatial approximation of the QoI given by

$$\mathsf{QoI}_{h_l} = \int_{\Gamma} G_{h_l}(\boldsymbol{y}) \rho(\boldsymbol{y})\, d(\boldsymbol{y}) \qquad \text{for } l = 0, \ldots, L$$

- Obviously, the approximation QoI_{h_L} of the QoI on the finest spatial grid $h_L = h_{hf}$ can be written in the form of the telescoping sum

$$\mathsf{QoI}_{h_L} = \mathsf{QoI}_{h_0} + \sum_{l=1}^{L} \big(\mathsf{QoI}_{h_l} - \mathsf{QoI}_{h_{l-1}}\big)$$

- We express this more economically as

$$\mathsf{QoI}_{h_L} = \sum_{l=0}^{L} \Delta_{h_l}$$

where

$$\Delta_{h_0} = \mathsf{QoI}_{h_0} \quad \text{and} \quad \Delta_{h_l} = \mathsf{QoI}_{h_l} - \mathsf{QoI}_{h_{l-1}} \quad \text{for } l = 1, \ldots, L$$

- For any level l, $l = 1, \ldots, L$, we determine an MC approximation of Δ_{h_l} using the level l MC quadrature rule $\mathcal{Q}_{M_l}^{mc}$ using the points $\{\boldsymbol{y}_{m_l}\}_{m_l=1}^{M_l}$

  i.e., we have

$$\Delta_{M_l,h_l}^{mc} = \mathcal{Q}_{M_l}^{mc}\big(G_{h_l} - G_{h_{l-1}}\big)$$

$$= \frac{1}{M_l} \sum_{m_l=1}^{M_l} \big(G_{h_l}(\boldsymbol{y}_{m_l}) - G_{h_{l-1}}(\boldsymbol{y}_{m_l})\big) \approx \Delta_{h_l}$$

- The MLMC approximation of the QoI is then given as

$$\text{QoI} \approx \text{QoI}_{h_L} \approx \text{QoI}_{h_L}^{mlmc} = \sum_{l=0}^{L} \Delta_{M_l,h_l}^{mc}$$

  - note that we do not apply the MC method to any $G_{h_l}(\boldsymbol{y})$ for $l > 0$ but rather to the differences $\Delta_{h_l}(\boldsymbol{y}) = G_{h_l}(\boldsymbol{y}) - G_{h_{l-1}}(\boldsymbol{y})$

- The total number of samples taken is $M = \sum_{l=0}^{L} M_l$

  - note that because $h_l < h_{l-1}$
    the cost of obtaining samples of $\Delta_{h_l}(\boldsymbol{y})$ increases as the level l increases

- How does one choose the number of samples M_l for each level $l = 0, \ldots, L$?

  - again, the aim is to have the error in the MLMC approximation to be less than a given tolerance ε

  - then, M_l, $l = 0, \ldots, L$, are determined by minimizing the total sampling cost subject to the constraint that the total sampling error is of $O(\varepsilon)$

- The result of the optimization process is that
  the number of needed samples M_l decreases as the level l increases

- Thus, we see that
  - M_l is large when h_l is large
    - one has to do relatively lots of sampling when the realizations of the solution of the discretized PDE are relatively cheap
  - M_l is small when h_l is small
    - one has to do relatively little sampling when the realizations of the solution of the discretized PDE are relatively expensive

- Thus, there is a tradeoff in using MLMC compared to MC
  - to obtain the same error for both methods
    - the total number of samples $\sum_{l=0}^{L} M_l$ taken by MLMC may be larger than the total number of samples used by MC

    however
    - all the MC samples are taken on the finest spatial grid

    whereas
    - some of the MLMC samples are taken on coarser spatial grids

- Who wins?
  - does the MLMC method save over the MC?
    - the answer is yes

- Why does MLMC win?
  - the sampling error at the level l is proportional to $\sigma_l/\sqrt{M_l}$

    σ_l is a measure of the standard deviation of $\Delta_{h_l}(\boldsymbol{y})$
  - the key is that the variances σ_l^2 of the differences $\Delta_{h_l}(\boldsymbol{y}) = u_{h_l}(\boldsymbol{y}) - u_{h_{l-1}}(\boldsymbol{y})$ decrease as l increases
  - thus

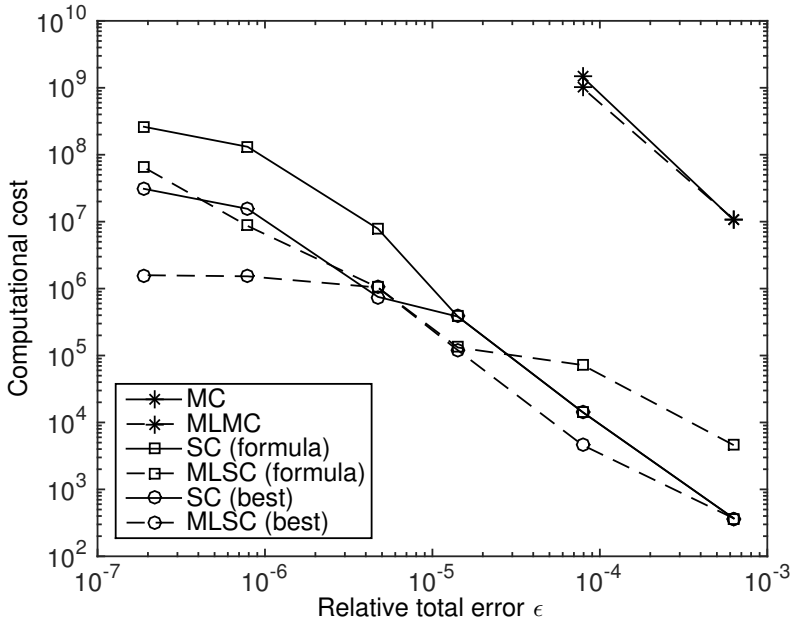
    to equilibrate errors across all levels

    one can use

    a smaller M_l for larger l

Multilevel stochastic collocation methods

- One can keep everything we just presented for multilevel Monte Carlo methods except for changing the sampling method
  - e.g., we can use sparse grid (such as Smolyak) sampling instead of Monte Carlo sampling

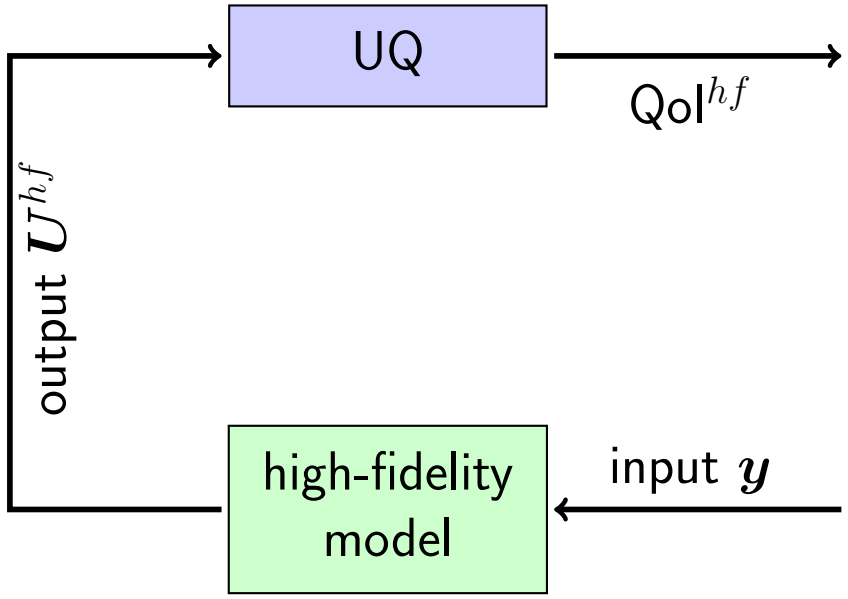


Computational cost versus error ε

Optimal multimodel, multifidelity Monte Carlo method

Multifidelity	$\Leftarrow$	a plethora of surrogates; implicit model hierarchy
Sampling	$\Leftarrow$	Monte Carlo

$$
\begin{cases}
\boldsymbol{y} \in \Gamma \subset \mathbb{R}^N & \Longleftarrow \quad \text{input vector of random inputs} \\
\boldsymbol{U}^{hf}(\boldsymbol{y}) : \Gamma \to \mathbb{R} & \Longleftarrow \quad \text{high-fidelity model output functional of interest} \\
\text{QoI}^{hf} & \Longleftarrow \quad \text{statistical information about } \boldsymbol{U}^{hf}(\boldsymbol{y}) \\
\mathcal{C}_{hf} & \Longleftarrow \quad \text{cost to obtain } \boldsymbol{U}^{hf}(\boldsymbol{y}) \text{ for any given } \boldsymbol{y} \in \Gamma
\end{cases}
$$



- $\boldsymbol{y} \in \Gamma \subset \mathbb{R}^N \quad \Longleftarrow \quad$ vector of random inputs

$\Downarrow$

high-fidelity model solves at sample points in Γ

$\Downarrow$

- $\boldsymbol{U}^{hf}(\boldsymbol{y}) : \Gamma \to \mathbb{R} \quad \Longleftarrow \quad$ high-fidelity output functional of interest

$\Downarrow$

use the high-fidelity model solves to approximate the QoI

$\Downarrow$

$\text{QoI}^{hf} \quad \Longleftarrow \quad$ statistical information about the output of interest

- $\mathcal{C}_{hf} \geq 0 \quad \Longleftarrow \quad$ cost to obtain $\boldsymbol{U}^{hf}$ for any given $\boldsymbol{y} \in \Gamma$
$\approx$ cost of the high-fidelity model solve

- $G^{hf}(\boldsymbol{y}) = G\big(\boldsymbol{U}^{hf}(\boldsymbol{y})\big) \quad \Longleftarrow \quad$ integrand in QoI^{hf}

- Then, the high-fidelity Monte Carlo estimator of the QoI is given by

$$\mathsf{QoI}_{M_{hf}}^{hf} = \frac{1}{M_{hf}} \sum_{m=1}^{M_{hf}} \boldsymbol{U}^{hf}(\boldsymbol{y}_m) \approx \mathsf{QoI}$$

  - $\{\boldsymbol{y}_m\}_{m=1}^{M_{hf}}$ are M_{hf} i.i.d. samples in Γ
  - requires M_{hf} high-fidelity solves of the expensive model
  - cost of doing the M_{hf} solves is $M_{hf}\mathcal{C}_{hf}$

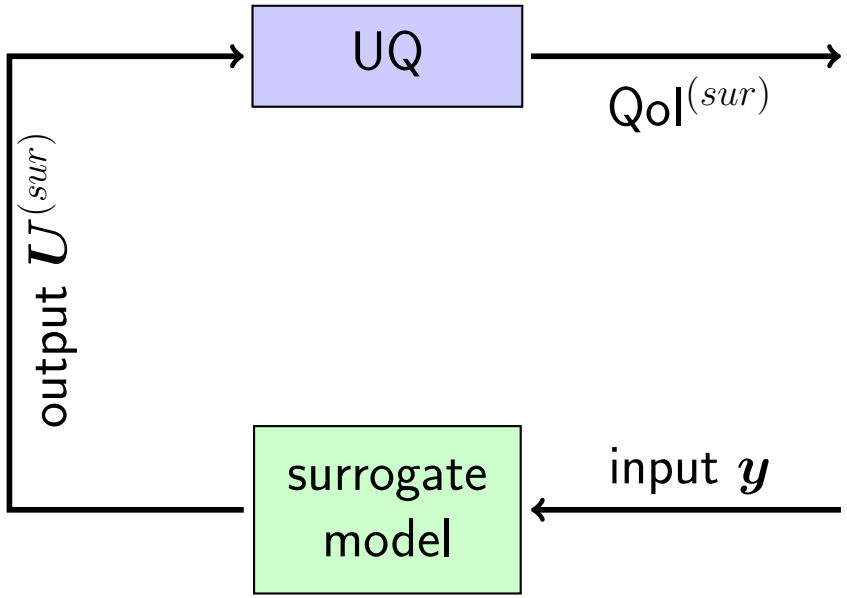
- Mean-square error of the high-fidelity Monte Carlo estimator

$$\textit{error}\big(\mathsf{QoI}_{M_{hf}}^{hf}\big) = \mathsf{E}\Big[\big(\mathsf{E}[\boldsymbol{U}^{hf}] - \boldsymbol{U}^{hf}\big)^2\Big] = \frac{1}{M_{hf}}\mathbb{V}[\boldsymbol{U}^{hf}]$$

- If $\mathcal{C}_{hf} \gg 1$, one looks for ways to reduce the cost of approximating the QoI

- In many situations, a surrogate model is available for which the corresponding approximation $\boldsymbol{U}^{(sur)}(\boldsymbol{y})$ is less expensive to obtain
  - if $\mathcal{C}^{(sur)}$ denotes the cost of obtaining $\boldsymbol{U}^{(sur)}(\boldsymbol{y})$, we have

$$\mathcal{C}^{(sur)} \ll \mathcal{C}^{hf}$$



- The construction of some surrogates,

  e.g., interpolants, reduced-order models

  require the solution of the expensive high-fidelity model

  – the hope is that the high offline construction cost is amortized over many subsequent online solves using the surrogate

- The Monte Carlo estimator of the QoI is now given by

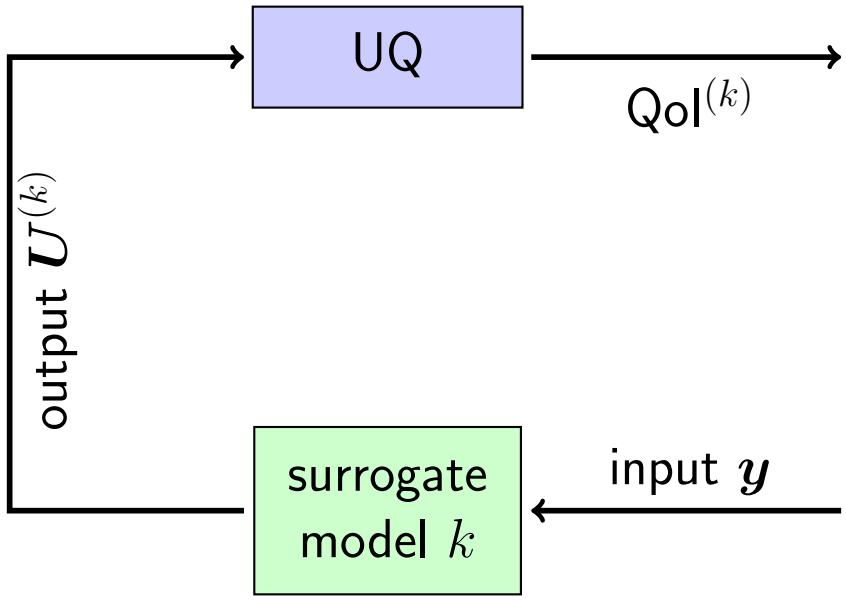
$$\text{QoI}_{M_{sur}}^{(sur)} = \frac{1}{M_{sur}} \sum_{m=1}^{M_{sur}} G^{(sur)}(\boldsymbol{y}_m) = \frac{1}{M_{sur}} \sum_{m=1}^{M_{sur}} G\big(\boldsymbol{U}^{(sur)}(\boldsymbol{y}_m)\big)$$

  – by using the surrogate instead of the high-fidelity model, the cost of determining the approximation of the QoI reduces from $M_{hf}\mathcal{C}_{hf}$ to the much smaller $M_{sur}\mathcal{C}_{sur}$

  – unfortunately, less costly surrogates are also usually less accurate

- In many situations, several surrogates $\boldsymbol{U}^{(2)}(\boldsymbol{y}), \ldots, \boldsymbol{U}^{(K)}(\boldsymbol{y})$ for the output of interest are available

  - for $k = 2, \ldots, K$



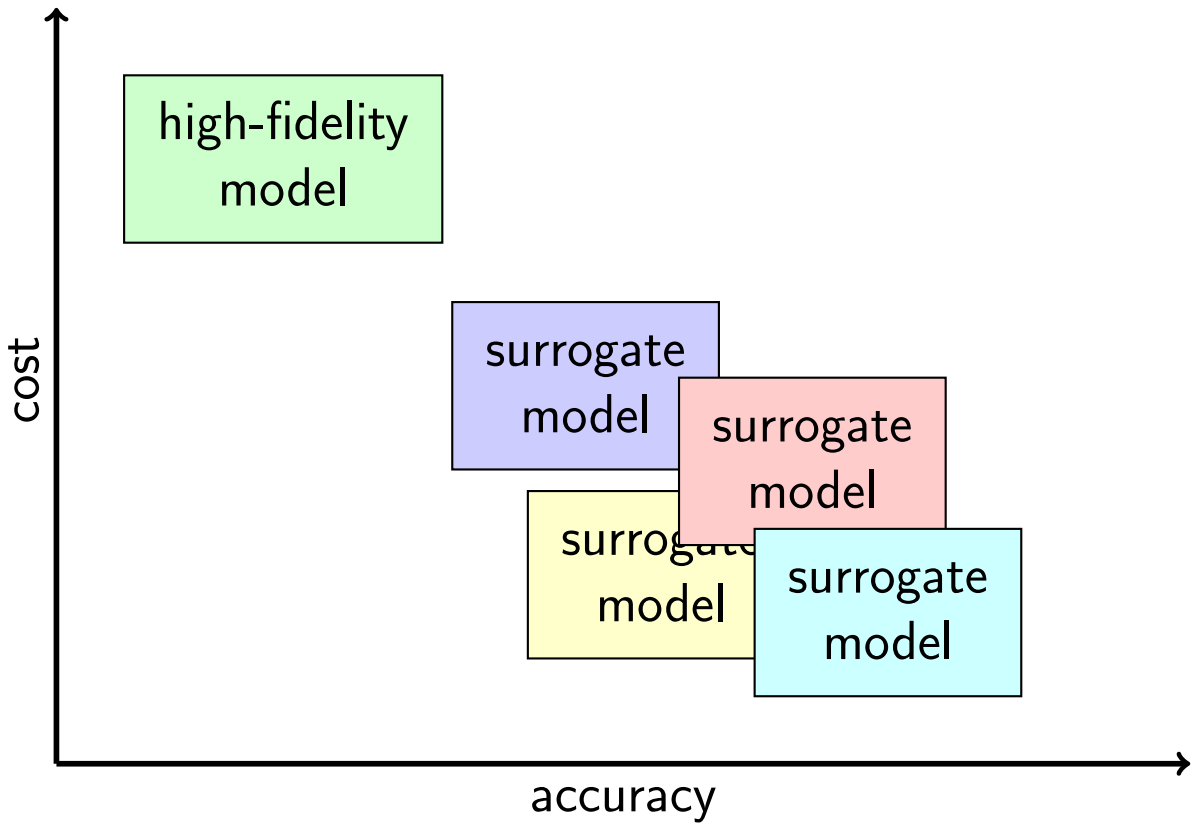
- For $k = 2, \ldots, K$

  - the cost $\mathcal{C}_k$ for obtaining $\boldsymbol{U}^{(k)}(\boldsymbol{y})$ is often significantly less than is the cost $\mathcal{C}_1 = \mathcal{C}_{hf}$ for obtaining $\boldsymbol{U}^{(1)}(\boldsymbol{y}) = \boldsymbol{U}^{hf}(\boldsymbol{y})$

  - QoI$^{(k)}$ can viewed as a "lower" fidelity "approximation" of QoI$^{(1)}$ = QoI^{hf}

- Many types of surrogate models are possible
  - interpolants, least-squares approximations
  - projection-based reduced-order models
  - simplified-physics models
    - linearized PDEs
    - averaged models (e.g., RANS)
  - coarse-grid approximations
  - stopping iterations early resulting in higher residuals
  - machine-learning-based models
  - experimental data

– these are very different from each other
with respect to cost, fidelity, and construction

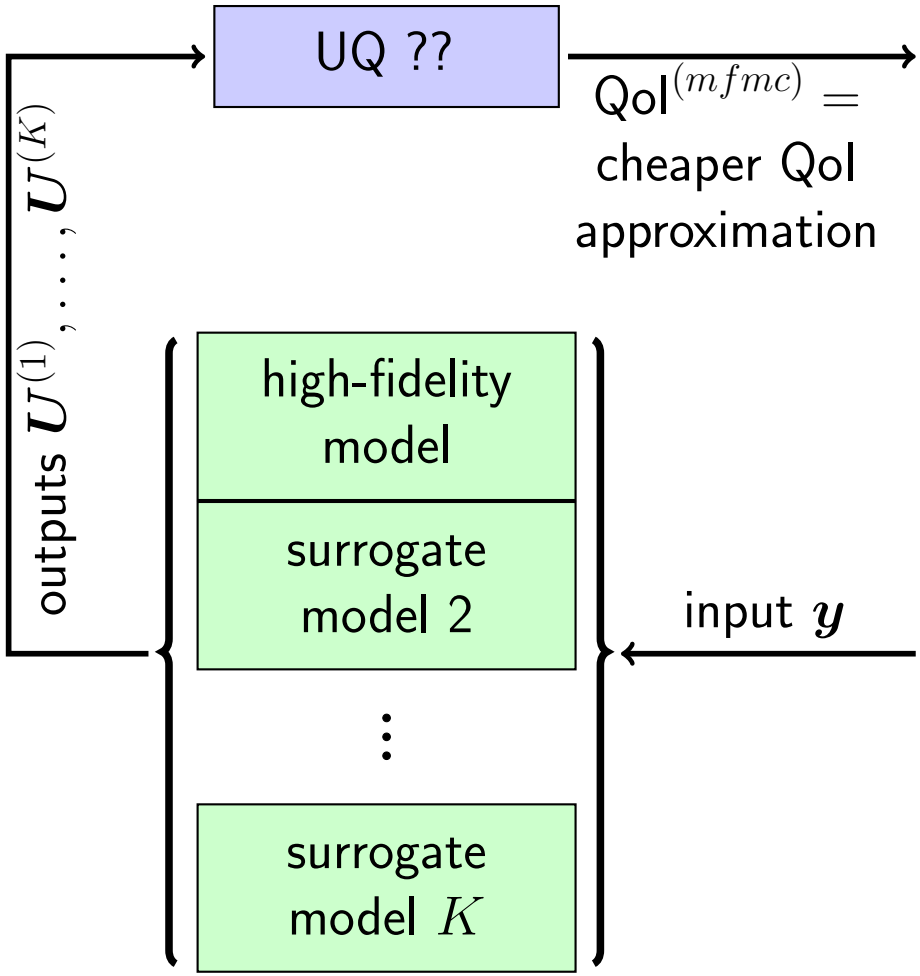


- The hope is that somehow the cheaper surrogate models can be used to speed up, without compromising accuracy,

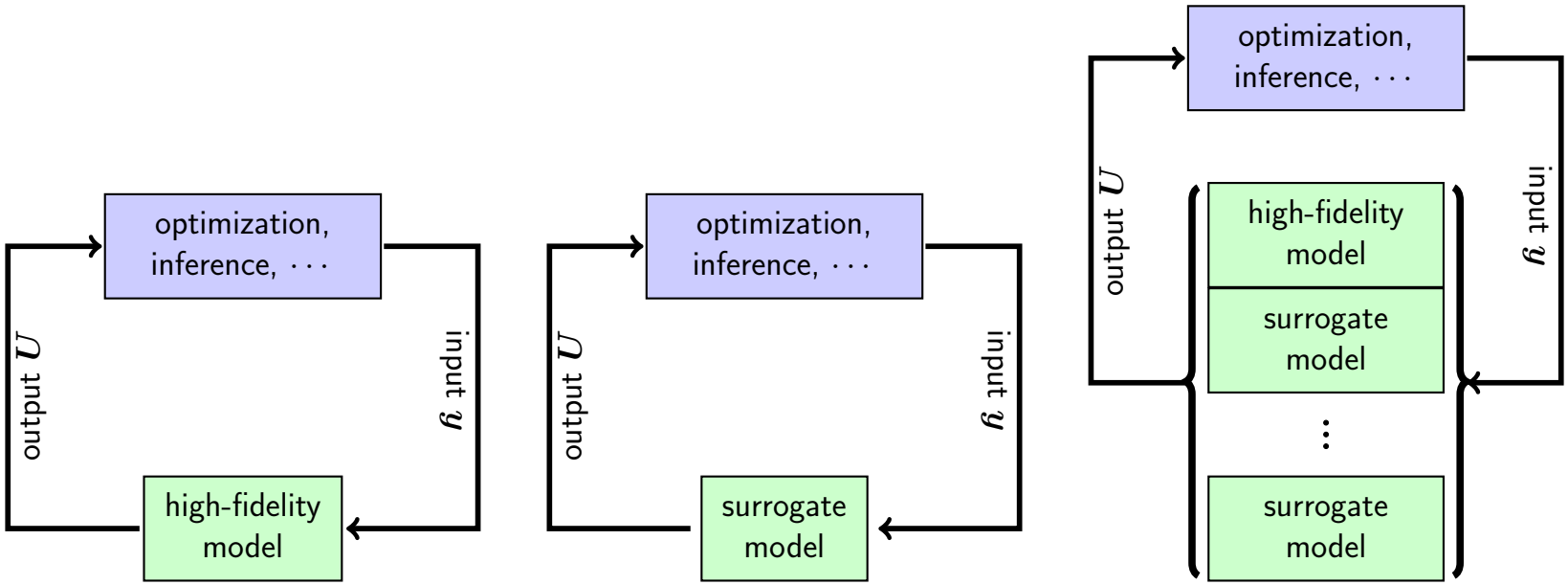
the approximation of the QoI corresponding to the high-fidelity model

- Model management
  - combine all model outputs of interest $\boldsymbol{U}^{(1)}, \ \boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(K)}$
  - use surrogates $\boldsymbol{U}^{(2)}, \ldots, \boldsymbol{U}^{(K)}$ for speedup
  - use $\boldsymbol{U}^{(1)}$ for accuracy
  - balance the number of model evaluations (number of samples) among models
  - establish accuracy guarantees

- Because we want to be able to incorporate models of vastly different nature, we generally do not have available estimates of the relative accuracy of those models
  - this is unlike the situation for MLMC and MLSC methods for which we have explicit information about those differences because the different models are generated solely by changing, e.g., the grid size
  - thus, we do not assume we have any a priori information about fidelities of any of any model relative to the other ones


UQ ??
$\text{QoI}^{(mfmc)} =$
cheaper QoI
approximation
outputs $\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(K)}$
high-fidelity
model
surrogate
model 2
surrogate
model K
input $\boldsymbol{y}$

- Before going on to remove the ?? in the UQ box, we note that there are other applications in which surrogates have been used to improve the efficiency of a high-fidelity computation, including closed-loop settings
  - optimization: corrections
  - inference: delayed acceptance MCMC



Multifidelity Monte Carlo method

- Setup
  - $\boldsymbol{y} \in \Gamma$ $\Leftarrow$ vector of random parameters with joint PDF $\rho(\boldsymbol{y})$
  - $\boldsymbol{U}^{(1)}(\boldsymbol{y})$ $\Leftarrow$ output of high-fidelity model ("truth")
  - $\boldsymbol{U}^{(2)}(\boldsymbol{y}), \ldots, \boldsymbol{U}^{(K)}(\boldsymbol{y})$ $\Leftarrow$ output of $K-1$ surrogate models
  - $\mathcal{C}_k$ $\Leftarrow$ cost for obtaining $\boldsymbol{U}^{(k)}(\boldsymbol{y})$ for given $\boldsymbol{y} \in \Gamma$
  - M_k $\Leftarrow$ number of model evaluations of $\boldsymbol{U}^{(k)}$ with $0 < M_1 \leq M_2 \leq \cdots \leq M_K$
  - $\mathsf{QoI}^{(1)} = \mathsf{QoI}^{hf}$ $\Leftarrow$ statistical information about the high-fidelity output $\boldsymbol{U}^{(1)}(\boldsymbol{y}) = \boldsymbol{U}^{hf}(\boldsymbol{y})$
- Goal: estimate $\mathsf{QoI}^{(1)}$ specifically $\Leftarrow$ $\mathsf{QoI}^{(1)} = \mathsf{E}[G^{(1)}(\boldsymbol{y})]$

- Problem definition: given a computational budget $B > 0$
  - derive estimator of $\text{QoI}^{(1)} = \text{E}[G^{(1)}(\boldsymbol{y})]$ that uses surrogate models
  - estimator has cost B
  - estimator is unbiased with respect to $\text{QoI}^{(1)}$
  - estimator has lower MSE than the MC estimator also having cost B

- No assumptions about surrogate models
  - use as many surrogate models of any type as one has available
  - no a priori error bounds
  - no a posteriori error estimators
  - models do not have to form an explicit hierarchy through known error bounds, e.g., as in MLMC or MLSC where such bounds are known in terms of, e.g., the grid size
    - the hierarchy is established implicitly through correlations between models

- Model management questions
  - how to combine models? $\Leftarrow$ control variates
  - how to balance model evaluations among them? $\Leftarrow$ optimization

- Model sampling
  - draw M_k (i.i.d.) realizations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{M_k} \in \Gamma$ of $\boldsymbol{y}$
  - for $k = 1, \ldots, K$
    - evaluate $\boldsymbol{U}^{(k)}(\boldsymbol{y})$ at the realizations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{M_k}$ to obtain

$$\boldsymbol{U}^{(k)}(\boldsymbol{y}_1), \ldots, \boldsymbol{U}^{(k)}(\boldsymbol{y}_{M_k})$$

    - construct the kth MC estimate $\text{QoI}_{M_k}^{(k)} = \frac{1}{M_k} \sum_{m=1}^{M_k} G^{(k)}(\boldsymbol{y}_m)$
    - reuse evaluations to estimate $\text{QoI}_{M_{k-1}}^{(k)} = \frac{1}{M_{k-1}} \sum_{m=1}^{M_{k-1}} G^{(k)}(\boldsymbol{y}_m)$

- The multifidelity Monte Carlo (MFMC) estimate for QoI^{mfmc} is then defined as

$$\mathsf{QoI}^{(mfmc)} = \mathsf{QoI}^{(1)}_{M_1} + \sum_{k=2}^{K} \alpha_k \left(\mathsf{QoI}^{(k)}_{M_k} - \mathsf{QoI}^{(k)}_{M_{k-1}} \right)$$

  where $\{\alpha_k\}$ are positive weights

- Properties of the MFMC estimator
  - cost: $\widehat{\mathcal{C}}_{M_K} = \sum_{k=1}^{K} M_k \mathcal{C}_k$
  - the MFMC estimator is unbiased

- Given a computational budget B, one can analytically determine the
  - optimal values $\{\alpha_k^*\}_{k=2,\ldots,K}$ for the weights
  - optimal values for $\{M_k^*\}_{k=1,\ldots,K}$ for the number of samples one should take of the high-fidelity model and for each of the $K-1$ surrogates

  such that

$$
\begin{aligned}
M_1^* &\geq 1 \\
M_k^* - M_{k-1}^* &\geq 0 \quad \text{for } k = 2, \ldots, K \\
\sum_{k=1}^{K} M_k^* \mathcal{C}_k &= B
\end{aligned}
$$

  - $M_1^* \geq 1$ ensures that the accuracy of MLMC estimator is not compromised

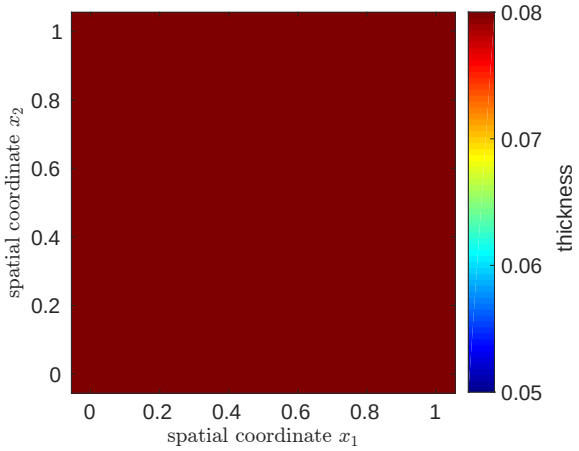
- Two competing features of a surrogate model come into play during the optimization process, namely
  - cost
  - how well they are correlated with the high-fidelity model

  – during the optimization process, the surrogates are re-ordered so that the the number of samples M_k increase with k and the costs decrease with k

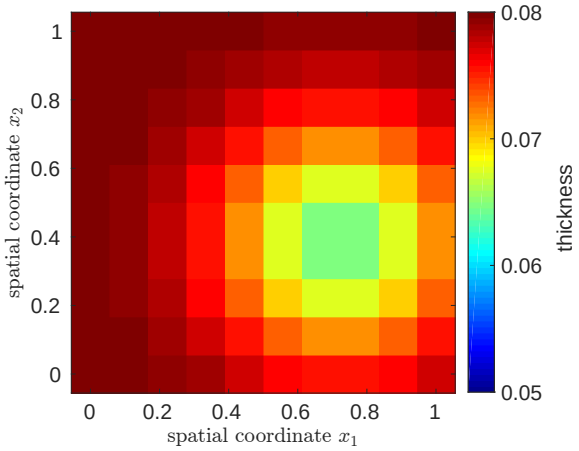
  – furthermore, the optimization process is capable of removing surrogate models whose "high" costs and "poor" correlations render them as being not useful

Numerical example: locally damaged plate in bending

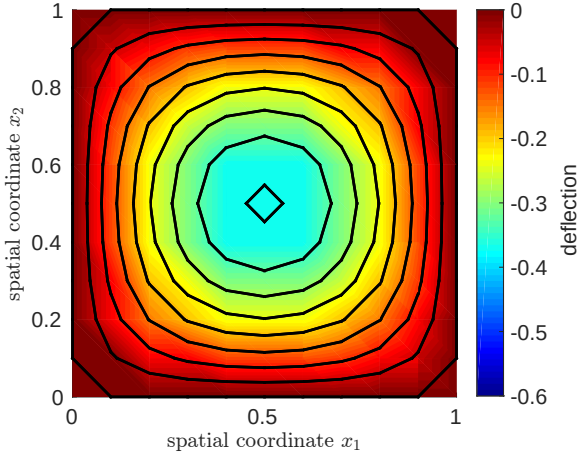
- Input $\boldsymbol{y} \in \mathbb{R}^4$: nominal thickness, load, damage
  - uniformly distributed in $[0.05, 0.1] \times [1, 100] \times [0, 0.2] \times [0, 0.05]$
- Output of interest $G(\boldsymbol{y})$: maximum deflection of plate
- QoI: the expected value of $G(\boldsymbol{y})$
- Six models
  - high-fidelity model: FEM, 300 DoFs
  - reduced-order model: POD with 10 DoFs
  - reduced-order model: POD with 5 DoFs
  - reduced-order model: POD with 2 DoFs
  - piecewise-linear interpolant with 256 points, using data from high-fidelity model
  - support vector machine with 256 points



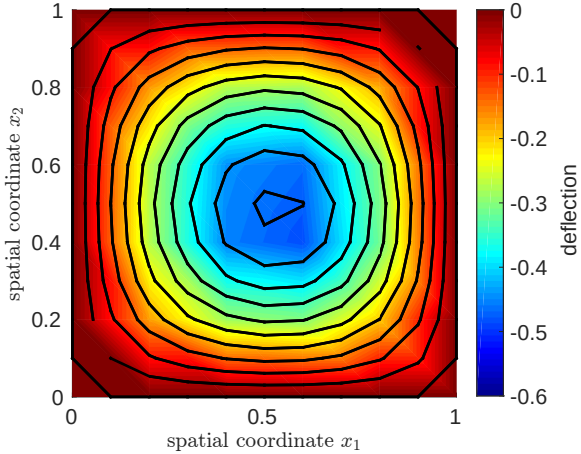
thickness: no damage



thickness: damage up to 20%

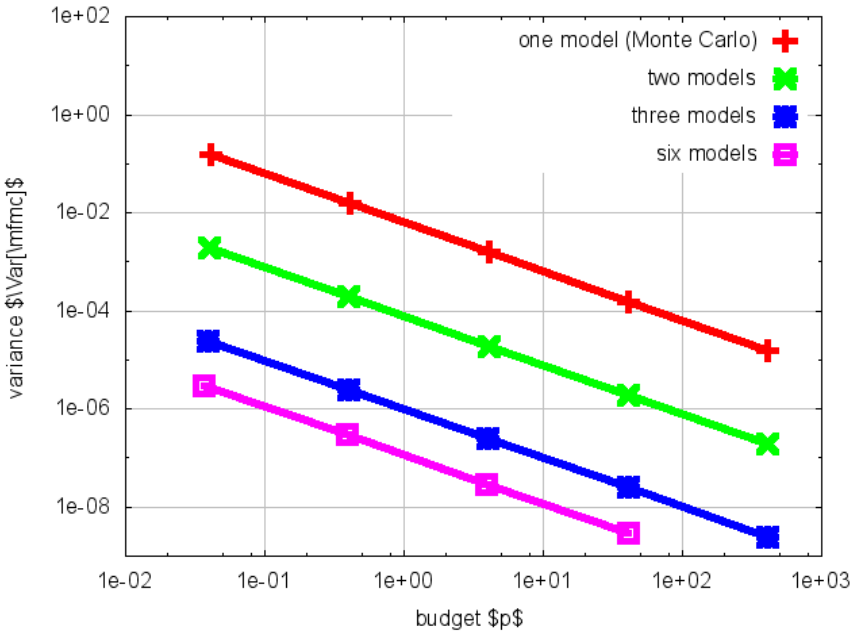


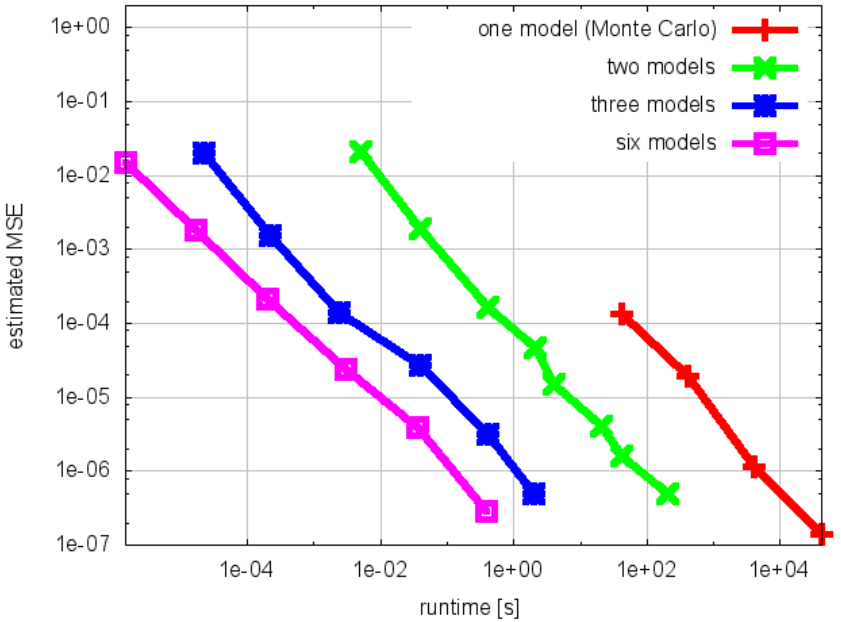
deflection: no damage



deflection: damage up to 20%

- Variance and mean square error (MSE) for several model combinations

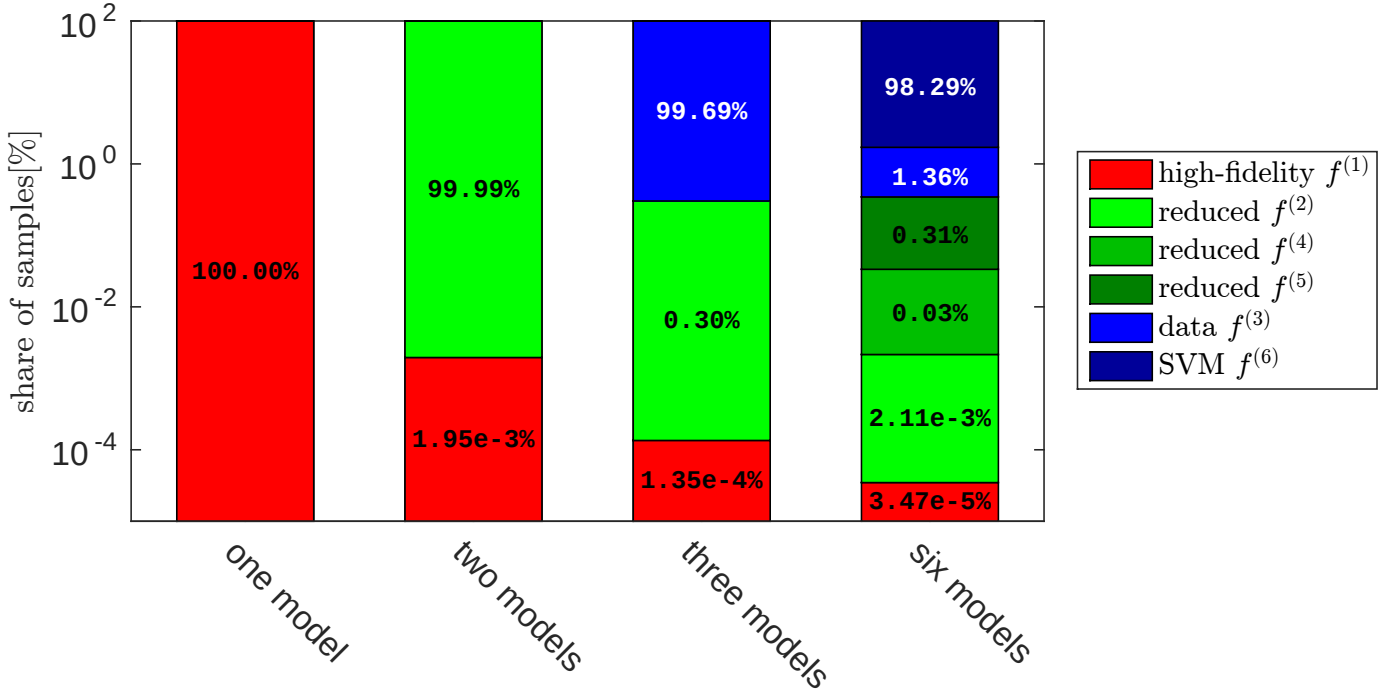




  - largest improvement from one $\rightarrow$ two and two $\rightarrow$ three models
  - adding additional models effects smaller improvement
  - theoretical and computational MSEs match well

- MFMC provides for optimal load balancing on HPCs because we know a priori how many times each model will be evaluated

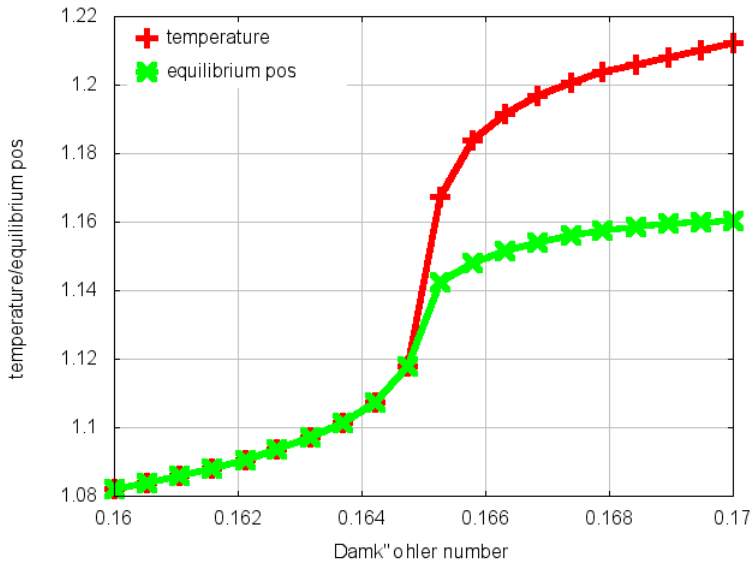
- Distribution of number of samples (model evaluations) among models

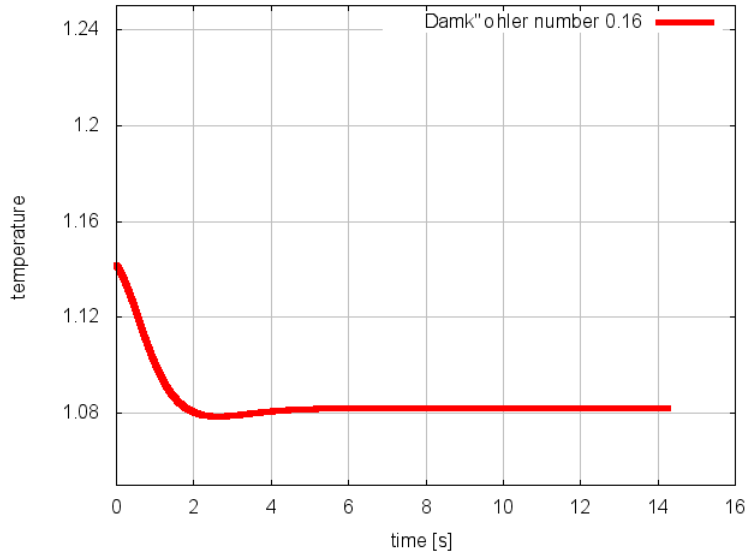


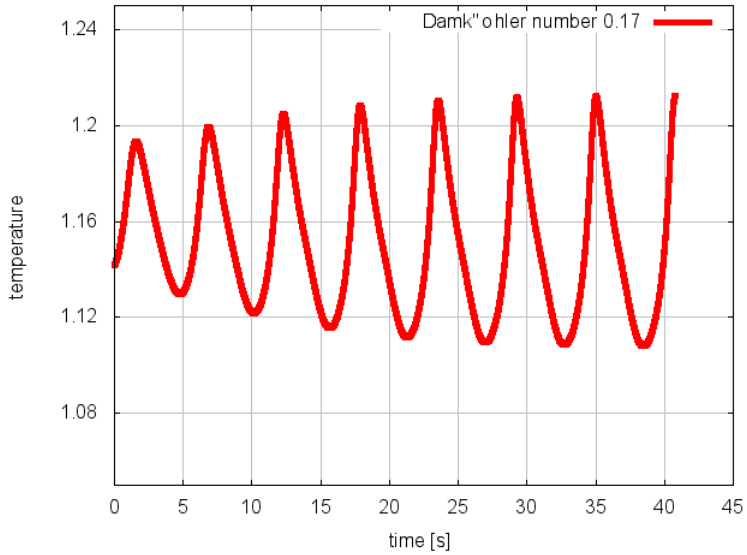
  - MFMC distributes samples among models depending on correlations and costs
  - number of samples changes exponentially between models
  - highest number of samples for data-fit and SVM models (cost ratio $\mathcal{C}_1/\mathcal{C}_6 \approx 10^6$)

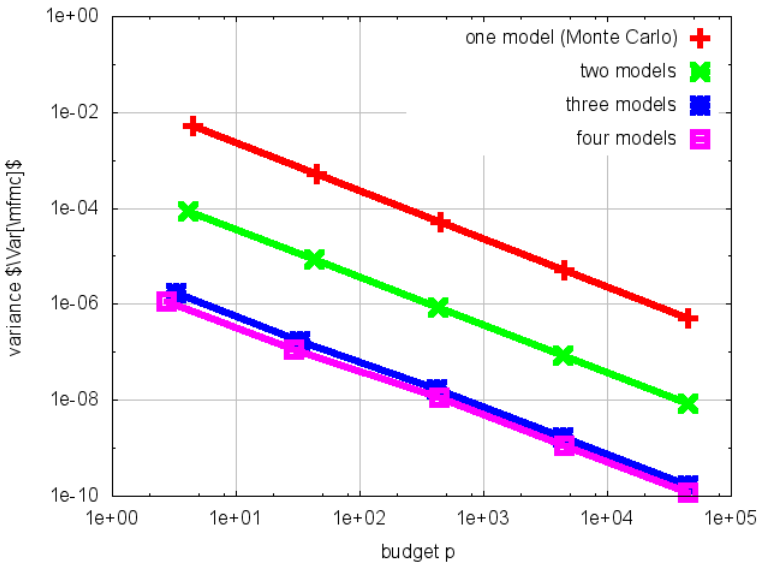
Numerical nonlinear example: limit cycle oscillation (LCO)

- Non-adiabatic tubular reactor
- Arrhenius-type (exponential) nonlinear term
- Input: Damköhler number $\sim \mathcal{N}(0.167, 0.03)$
- Output of interest: LCO amplitude
- QoI: expected value of the output of interest
- Four models
  - high-fidelity model: finite difference with 198 DoFs
  - reduced-order model: POD with 10 DoFs
  - reduced-order model: POD + DEIM with 10 DoFs
  - interpolation with 10 points

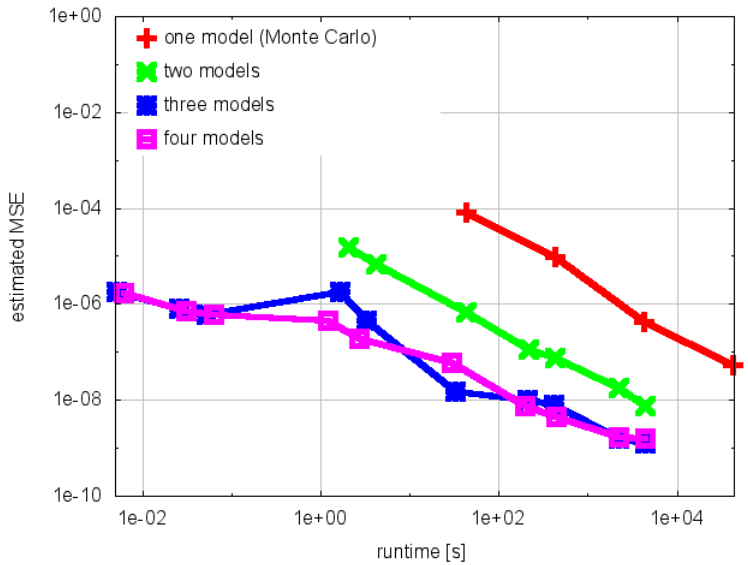

temperature
equilibrium pos
temperature/equilibrium pos
Damk"ohler number
1.22
1.2
1.18
1.16
1.14
1.12
1.1
1.08
0.16
0.162
0.164
0.166
0.168
0.17


Damk"ohler number 0.16
temperature
time [s]
1.24
1.2
1.16
1.12
1.08
0
2
4
6
8
10
12
14
16


Damk"ohler number 0.17
temperature
time [s]
1.24
1.2
1.16
1.12
1.08
0
5
10
15
20
25
30
35
40
45



variance



MSE

- Speedup of four orders of magnitude with high-fidelity + reduced order (POD) + interpolant
- Adding another reduced model (DEIM), only a slight improvement
- Agreement of variance (theoretical) and estimated MSE (numerical)

OPTIMAL CLUSTERING

K-MEANS & CENTROIDAL VORONOI TESSELATIONS

- Given a set of objects

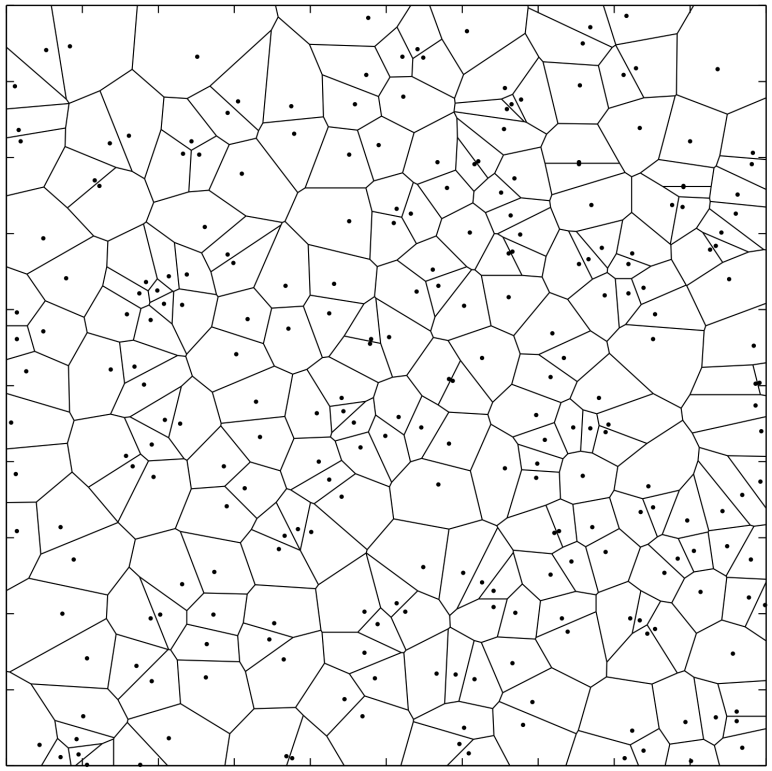
  clustering is the exercise of dividing the set into groups (= clusters) based on some attributes possessed by the objects

  – both the set of objects and their attributes can be very diverse

objects = people in Timbuktu	attribute = age
objects = people in Timbuktu	attribute = income
objects = people in Kalbarri	attribute = age
objects = pixels in an image	attribute = color
objects = pixels in an image	attribute = intensity
objects = points on a map	attribute = altitude
objects = points on a map	attribute = distance from airports

- Example
  - the dots are locations of fictitious existing Starbucks in a fictitious square city which has a uniform population density

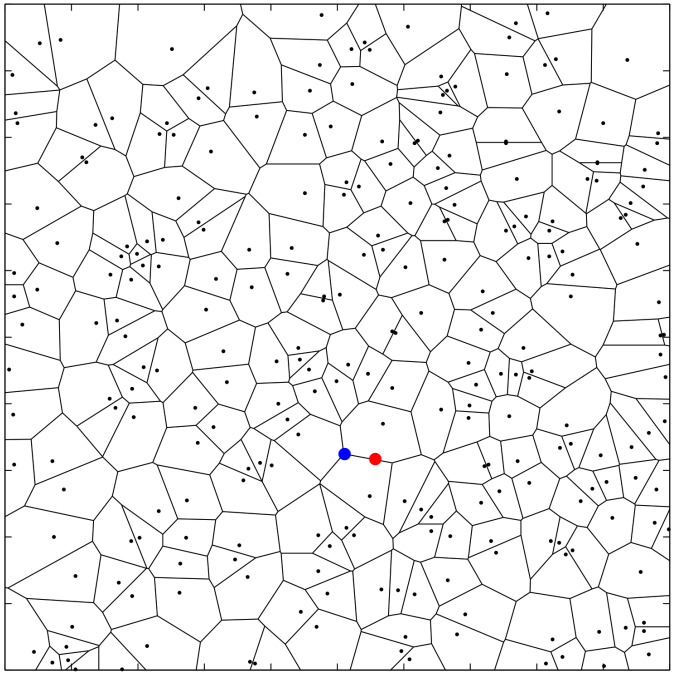
    - each polygon contains all points (all people)
      that are closer to the point in that polygon (a Starbucks location)
      than to any other points (closer than to any other Starbucks)



- objects to be clustered: people in the city
  clustering objective: group the population into clusters based on how close they are to an existing Starbucks

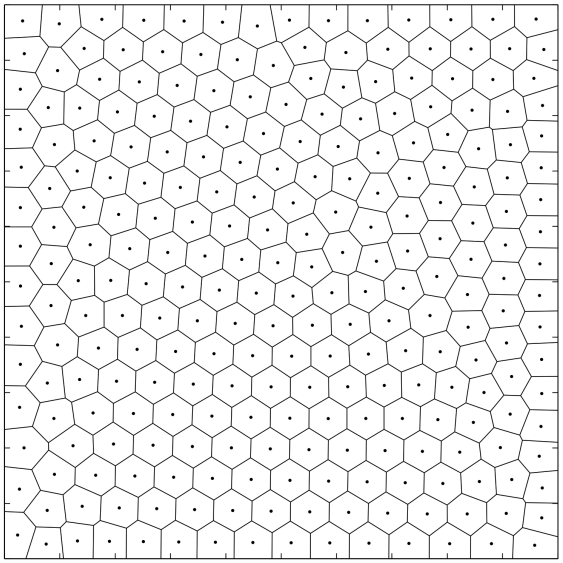
- Starbucks wants to know this information because
  they want to open another site
  in doing so they want to lessen the distance
  to a Starbucks for some of population

- e.g., they may choose to locate the new Starbucks at the red or blue location



– but suppose Amazon decides to enter into the coffee shop market

- they note that some people have to drive further to their closest Starbucks than do other people
- because they are starting from scratch, they are free to chose the locations of their coffee shops
- naturally, they decide to locate them so that the furthest drive to a closest shop is roughly the same for each shop
- Amazon decides to distribute their coffee shops as in the figure

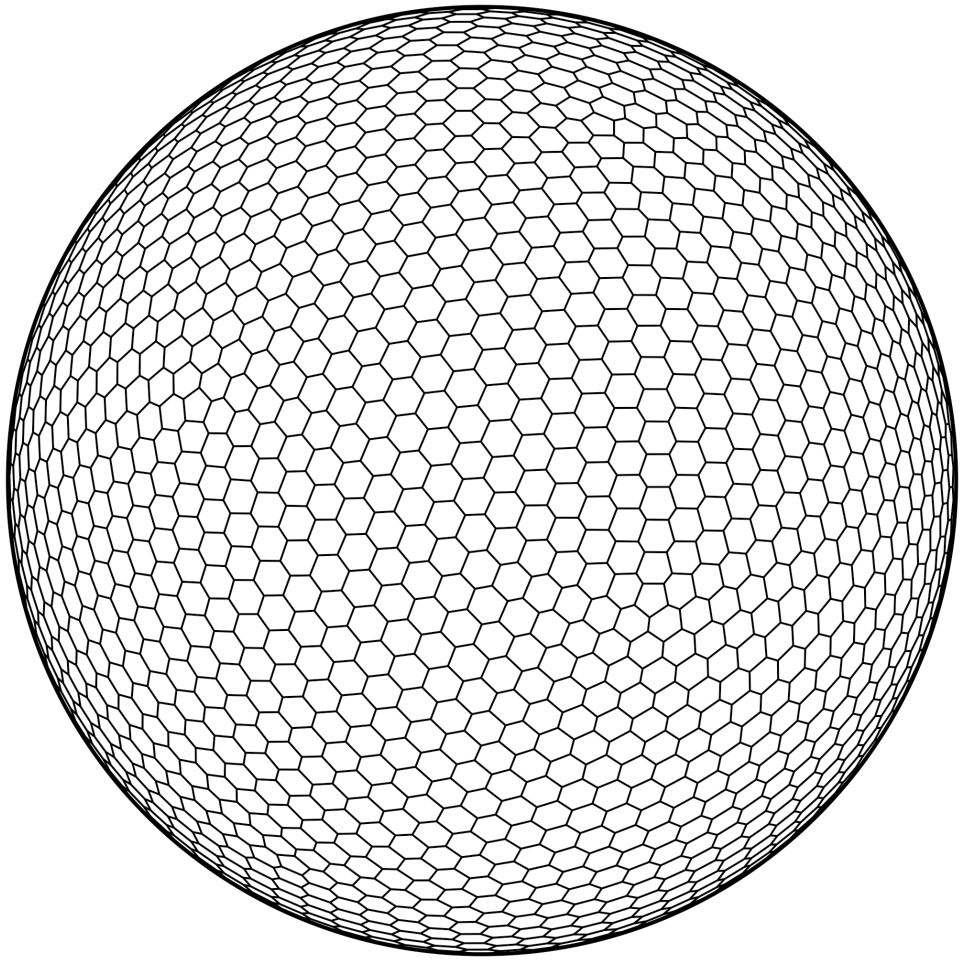


- how did Amazon determine this distribution of coffee shops?

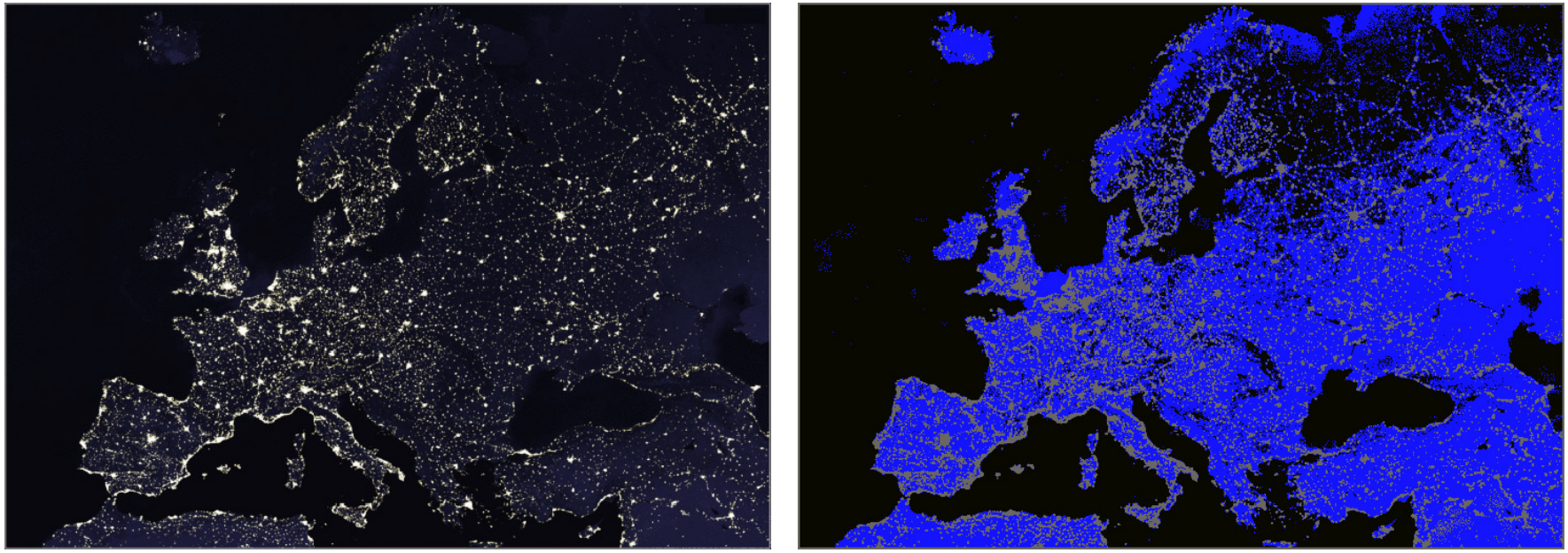
- Amazon's problem is an example of the generic optimal clustering task
  - given a set of objects — people in a city
    given attributes possessed by the objects — distance to the nearest coffee shop
    given an objective — minimize the furthest drive to a closest shop

    cluster the objects so that the objective is met — determine the locations of the coffee shops so that the objective is met

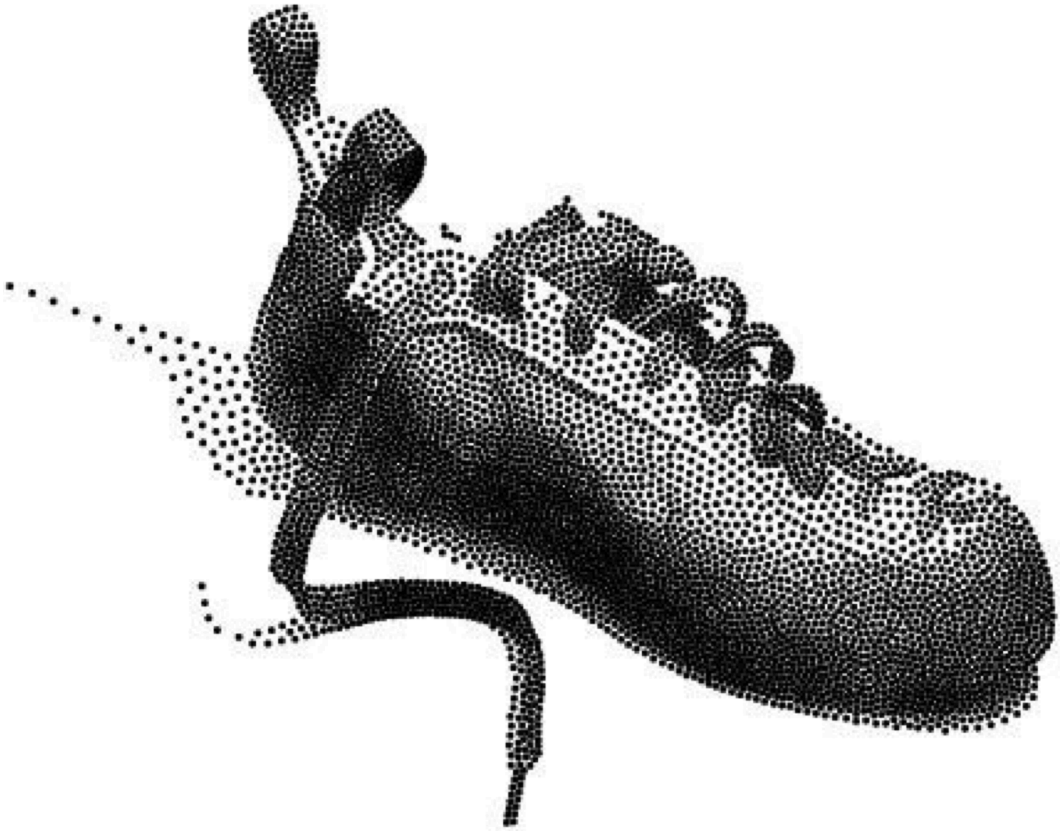
- This is an optimization problem
  - one wants to minimize or maximize (depending on the setting) the objective
  - problems of this nature arise in a very wide variety of settings
  - e.g., the three slides that follow are all outcomes of just such a type of optimization problem
    - they three differ in that they have different sets of objects, different attributes, and different objectives



"Uniform" hexagonal grid on the sphere



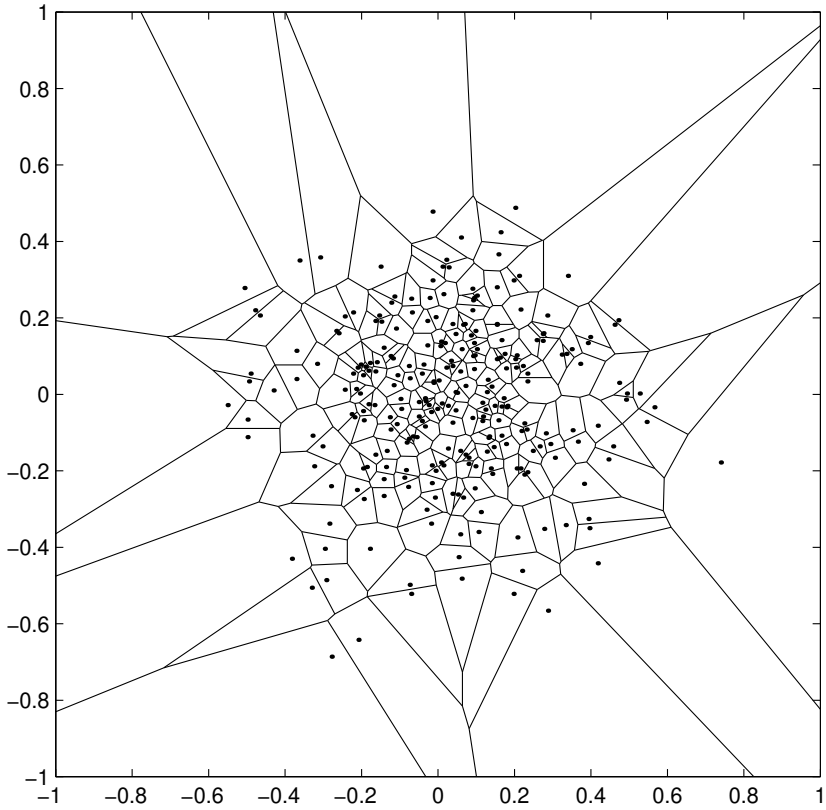
"Europe-by-night" satellite image (left) and its segmentation into three segments (right)



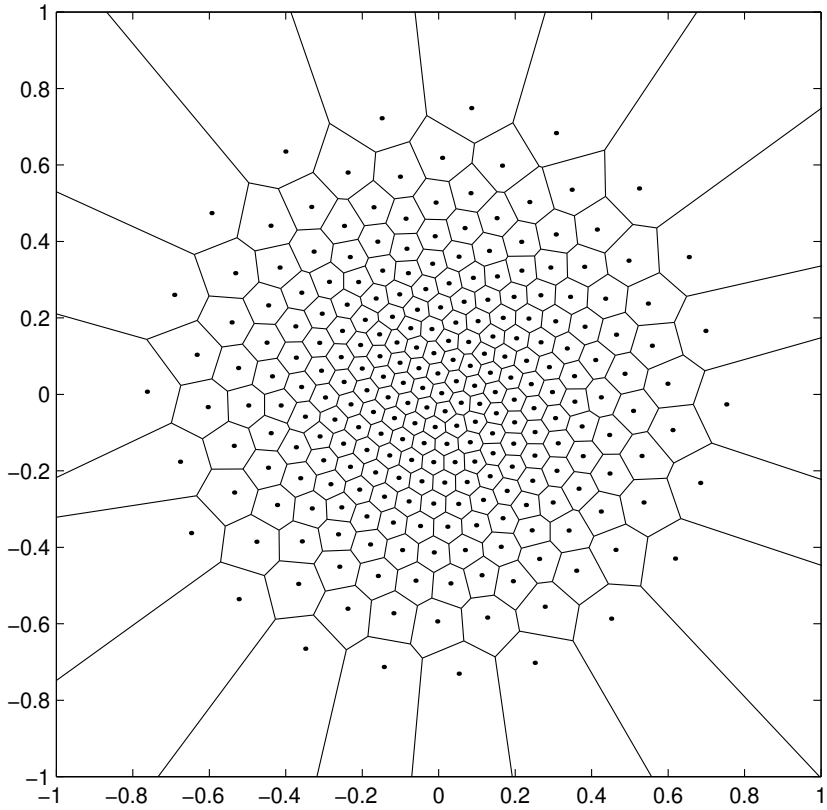
Stippled image

- What happens if we change the input information?
  - e.g., suppose that the population is not distributed uniformly
    but instead
    the population density is higher in the middle of the city?
  - naturally, there would be more Starbucks near the city center
    than out in the suburbs

- Amazon's objective may now be somewhat different
  - now it wants to equilibrate the sum of the distances
    that people have to travel to one of its coffee shops
  - Amazon can again cast the problem of determining its
    coffee shop locations as an optimization problem



Existing Starbuck locations are closer to each other in high population areas



Amazon's choice of locations

- There are surely many approaches known for solving the optimization problems we have defined
  - for discrete models

    i.e., the objects are discrete (e.g., can be expressed in terms of vectors)

    here, we consider only a k-means method
  - for continuous models

    e.g., the objects are functions (such as solutions of differential equations)

    here, we consider centroidal Voronoi tesselations (CVTs)
  - often, one discretizes the continuous model

    - in which case one ends up with a discrete model so one may be able to use k-means
  - however, CVTs, as we will define them, when discretized can result in variants of the k-means methods

- We define weighted[1] k-means
  - given a set of cardinality M of N-vectors $S = \{\boldsymbol{z}_1, \boldsymbol{z}_3, \cdots, \boldsymbol{z}_M\}$
  - one subdivides $S = \{S_1, S_2, \cdots, S_K\}$ into K non-overlapping, covering subsets of cardinality K_k
    - we have $S_k \cap S_{k'} = \emptyset \quad \text{if } k' \neq k$

      $$S_1 \cup S_2 \cup \cdots \cup S_K = S$$

      $$K = \sum_{k=1}^{K} K_k$$

    - of course, there are many ways to define such a subdivision
  - also given are a set of positive weights $\{w_k\}_{k=1}^{K}$

[1] We will omit the adjective "weighted" from here on in

– the object of k-means clustering is, given a set of objects $S = \{\boldsymbol{z}_m\}_{m=1}^{M}$ determine a subdivision $S = \{S_1, S_2, \cdots, S_K\}$ such that[2]

$$\sum_{k=1}^{K} \sum_{\boldsymbol{z}_m \in S_k} w_m |\boldsymbol{z}_m - \boldsymbol{\mu}_k|^2 \quad \text{is minimized over all possible subsets } \{S_k\}_{k=1}^{K}$$

where $\boldsymbol{\mu}_k = \frac{1}{K_k} \sum_{\boldsymbol{z}_m \in S_k} \boldsymbol{z}_m$ = the average of the K_k vectors $\boldsymbol{z}_k \in S_k$

- Later on, we discuss some approaches for solving the minimization problem
- k-means can has been generalized in many directions
  - hierarchical k-means
  - we can use a different distance metric instead of the Euclidean distance

    e.g., instead of using $|\cdot|^2$, use $|\cdots|$ so that we have a sum $\sum_{k=1}^{K} \sum_{\boldsymbol{z}_m \in S_k} w_m |\boldsymbol{z}_m - \boldsymbol{\mu}_k|$ of absolute values instead of a sum of squares
  - . . . . . .

[2] $|\cdot|$ denotes the Euclidean distance

Voronoi tessellations

- Given a set S, divide it into K subsets $S_1, S_2, \ldots, S_K$ such that
  - no member of a subset S_k is a member of another subset S_j
    the sets do not overlap
  - every member of S belongs to one of the sets S_k
    the union of the subsets is the original set

- We refer to the collection of such subsets $\{S_1, S_2, \ldots, S_K\}$ as a tessellation[3] of S

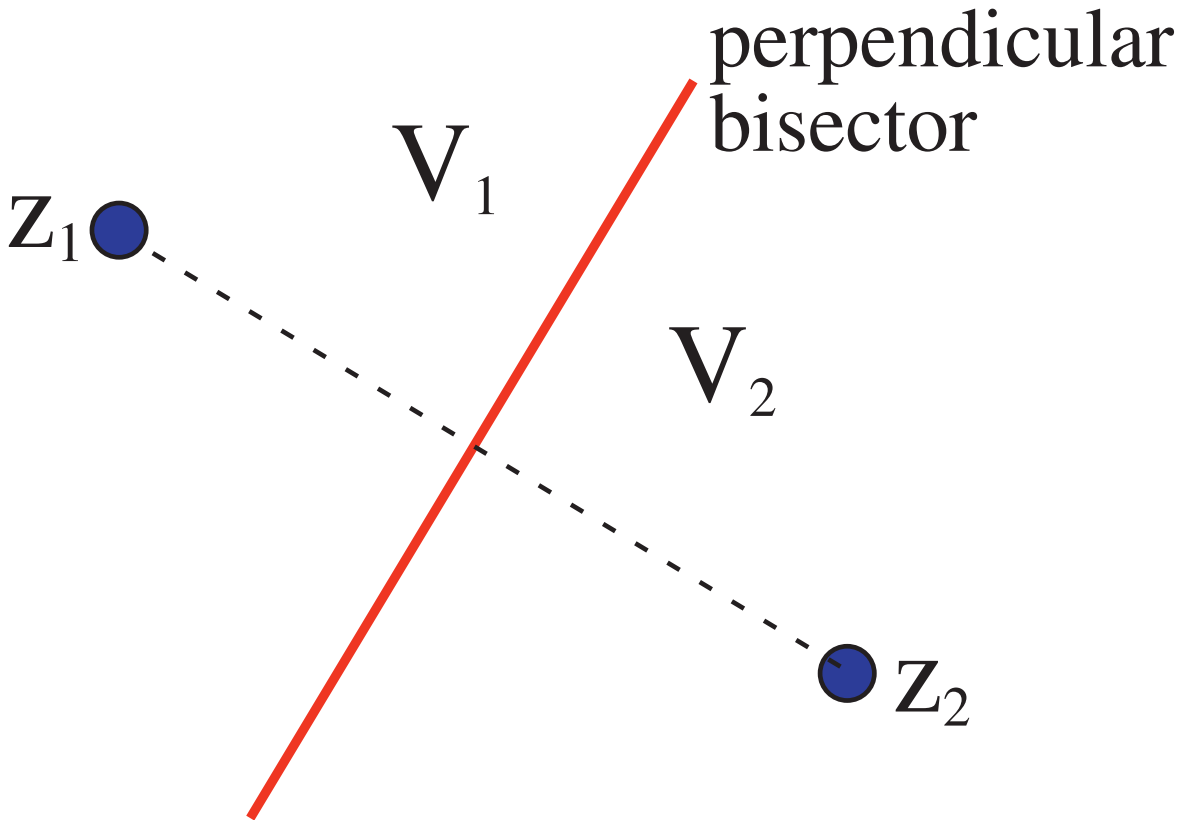
- Of course, k-means provides an example of such a subdivision in which the sets and subsets are discrete

- Here, in our discussion of tessellations, we are also concerned with continuous sets and subsets
  e.g., subsets of R^d consisting of an infinite number of points

[3] Please forgive the use of "tessellations" in this context; some may have a stricter definition of what constitutes a tessellation

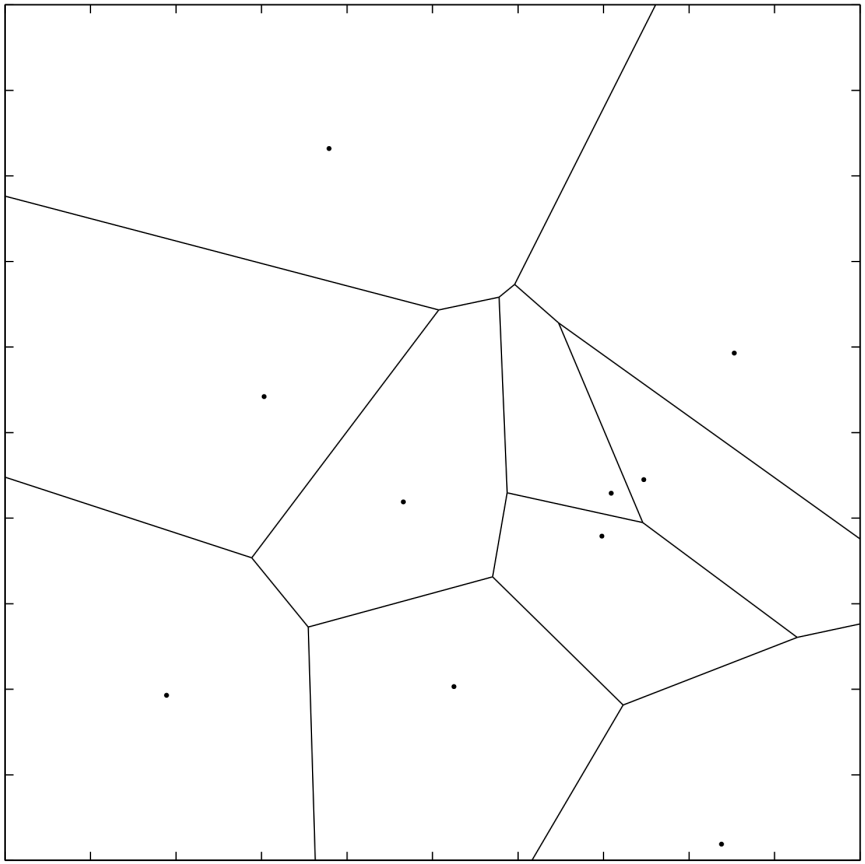
- Voronoi tessellations
  - given a set S which may consist of a finite[4] or an infinite number of objects
  - given a set of K other objects $Z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$ which do not necessarily belong to S
  - given a way to measure the distance between objects in S and the objects in Z
  - then, the Voronoi subset V_k is the set of objects in S that are closer to a $\boldsymbol{z}_k \in Z$ than to any of the other objects $\in Z$
  - we refer to the collection of Voronoi subsets $\{V_1, V_2, \ldots, V_K\}$ as a Voronoi tessellation of S or Voronoi diagram of S
  - we refer to the set of objects $Z = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_K\}$ as the generators of the Voronoi tessellation
    $\Longleftarrow$ choosing the generators determines the tessellation

[4] If it is finite, we enter the realm, e.g., of having to cluster a finite number of vectors



$S =$ the plane; Euclidean distance; $K = 2$

The Voronoi regions for two points $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ in the plane are the two regions on either side of the perpendicular bisector of the line segment joining $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$



$S =$ a square; Euclidean distance; $K = 10$

Voronoi tessellation for 10 randomly selected points in a square

- Note that our notion of distance can be very general

  - all we ask of a distance $d(\boldsymbol{z}, \boldsymbol{z}')$ between object $\boldsymbol{z}$ and object $\boldsymbol{z}'$ is that

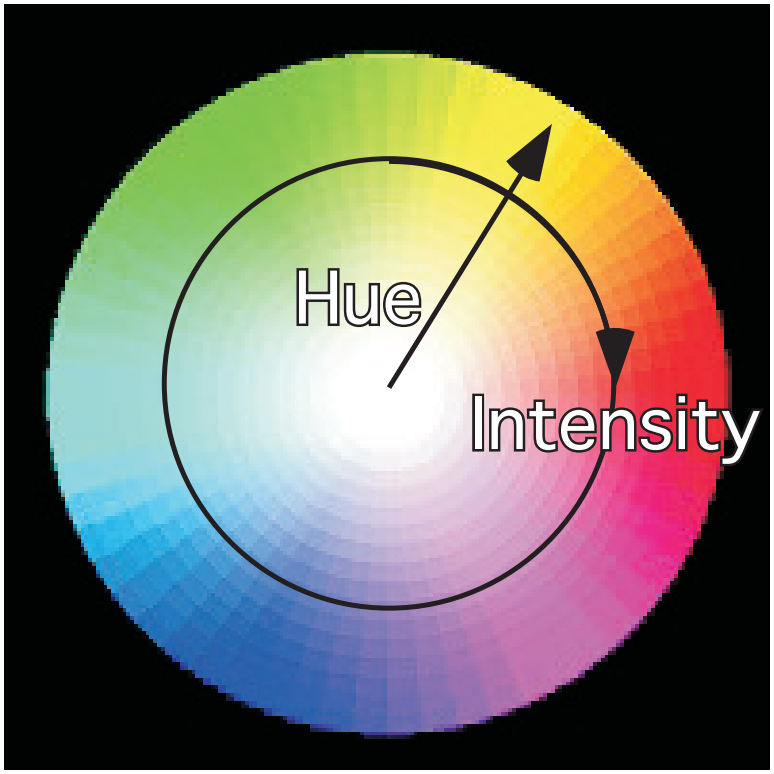
$$d(\boldsymbol{z}, \boldsymbol{z}') \geq 0$$

$$d(\boldsymbol{z}, \boldsymbol{z}') = d(\boldsymbol{z}', \boldsymbol{z})$$

$$d(\boldsymbol{z}, \boldsymbol{z}') = 0 \quad \text{only if } \boldsymbol{z} \text{ and } \boldsymbol{z}' \text{ are the same object}$$

- A distance measure for a less familiar set of objects

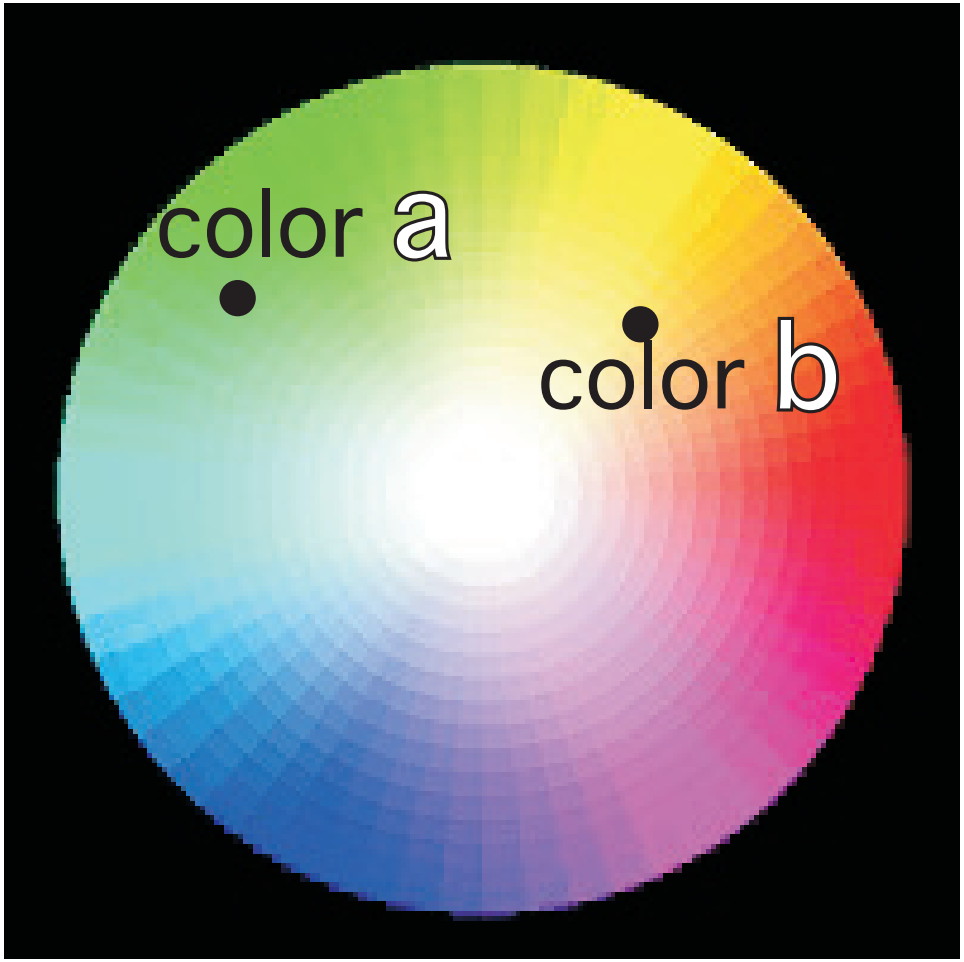
  colors in a color wheel



Points along the same radial line have the same hue

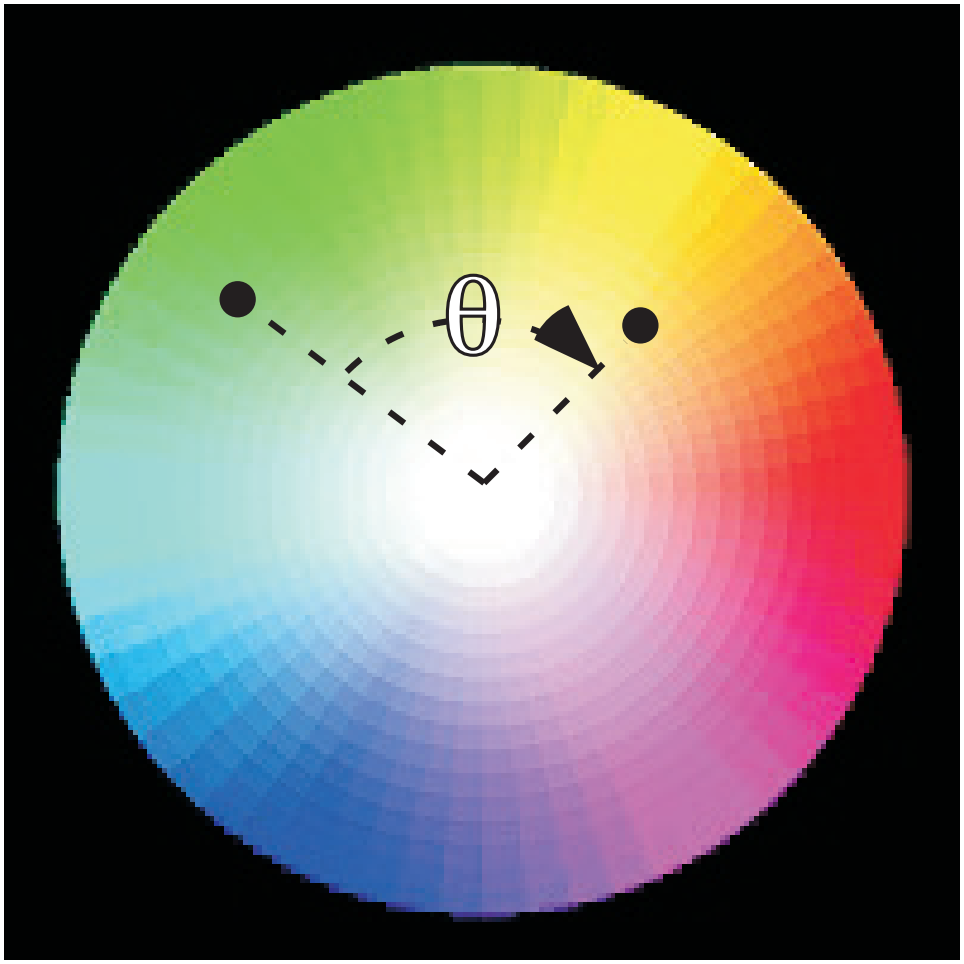
Points along the same circle have the same intensity

– how does one define a distance between two colors in the color wheel?



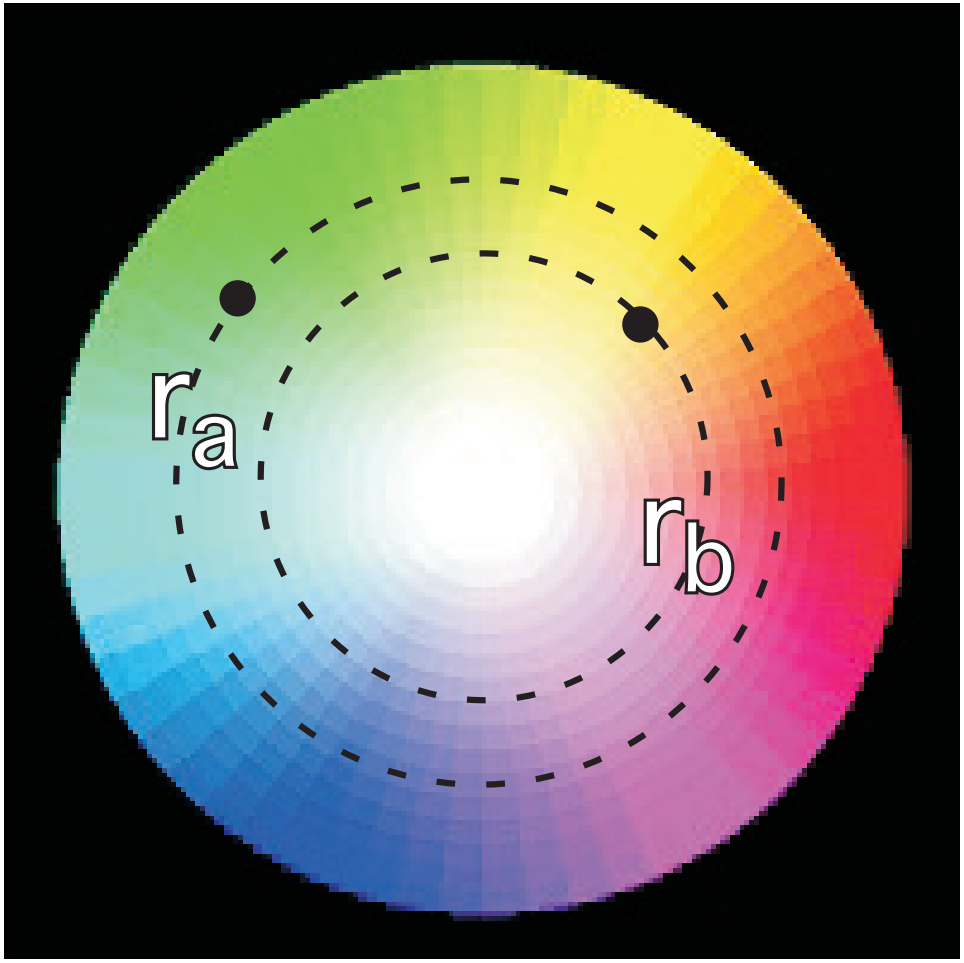
– if the distance between two colors is the smallest angle θ between them

- then, colors with the same hue (but different intensities) would be judged to be the same



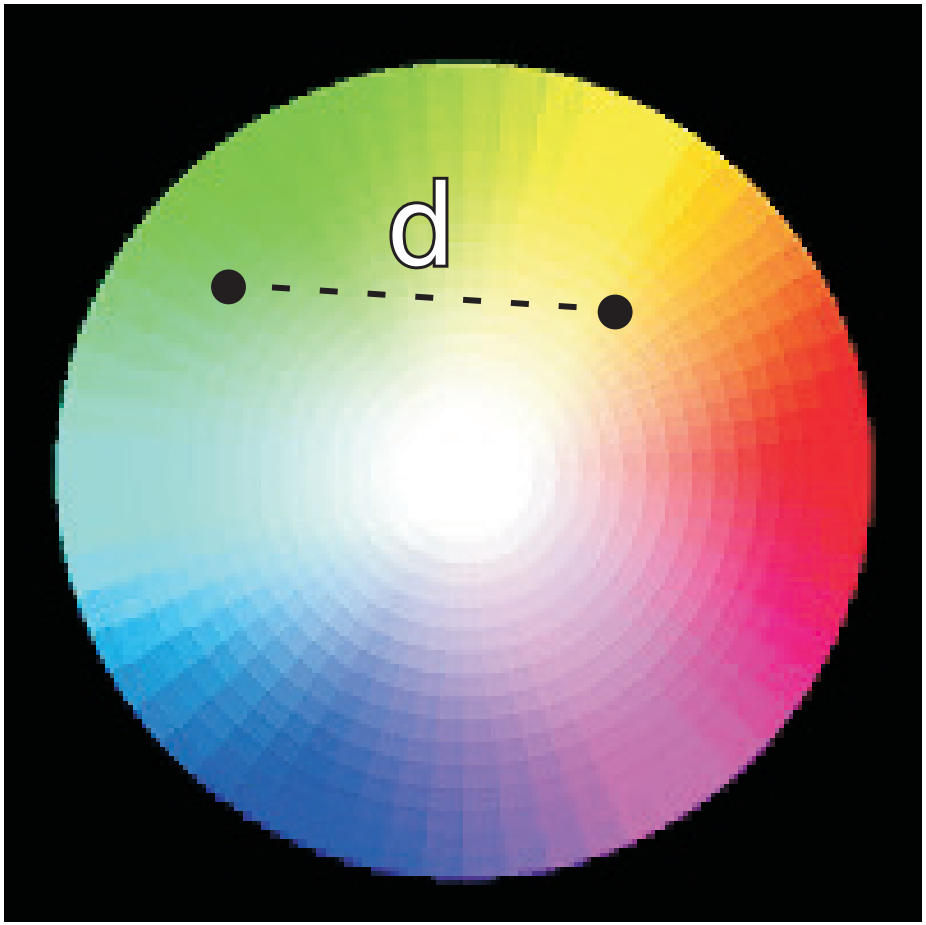
– if the distance between two colors is the difference in their radii $|r_b - r_a|$

- then points with the same intensity (but different hues) would be judged to be the same



– if the distance between two colors is the
Euclidean distance in color space

- then, only if two points have the same angle and radius
will they be judged to be the same



- Centroid
  - for example, given a region $V_{continuous}$ in Euclidean space and a weight function $w(\boldsymbol{z})$ defined for $\boldsymbol{z} \in V_{continuous}$, the centroid (or center of mass) $\boldsymbol{z}^*$ of V is given by

$$\boldsymbol{z}^* = \frac{\displaystyle\int_{V_{continuous}} \boldsymbol{z} w(\boldsymbol{z})\, d\boldsymbol{z}}{\displaystyle\int_{V_{continuous}} w(\boldsymbol{z})\, d\boldsymbol{z}}$$

  - for example, given a set of points $V_{discrete} = \{\boldsymbol{z}_m\}_{m=1}^{M}$ in Euclidean space and a weights $\{w_m\}_{m=1}^{M}$ defined for $\boldsymbol{z} \in V$, the centroid (or center of mass) $\boldsymbol{z}^*$ of $V_{discrete}$ is given by

$$\boldsymbol{z}^* = \frac{\displaystyle\sum_{\boldsymbol{z}_m \in V_{discrete}} \boldsymbol{z}_m w_m}{\displaystyle\sum_{\boldsymbol{z}_m \in V_{discrete}} w_m}$$

- We have now defined two different notions

  clustering via Voronoi tessellations

  and

  centroids

  – let's bring the two notions together

Centroidal Voronoi tessellations[5]

- Given K points $\boldsymbol{z}_k$, $k = 1, \ldots, K$, in a region
  - we can construct the associated Voronoi sets

$$V_k \quad \text{for } k = 1, \ldots, K$$

    where $\boldsymbol{z}_k$ is the generator for V_k

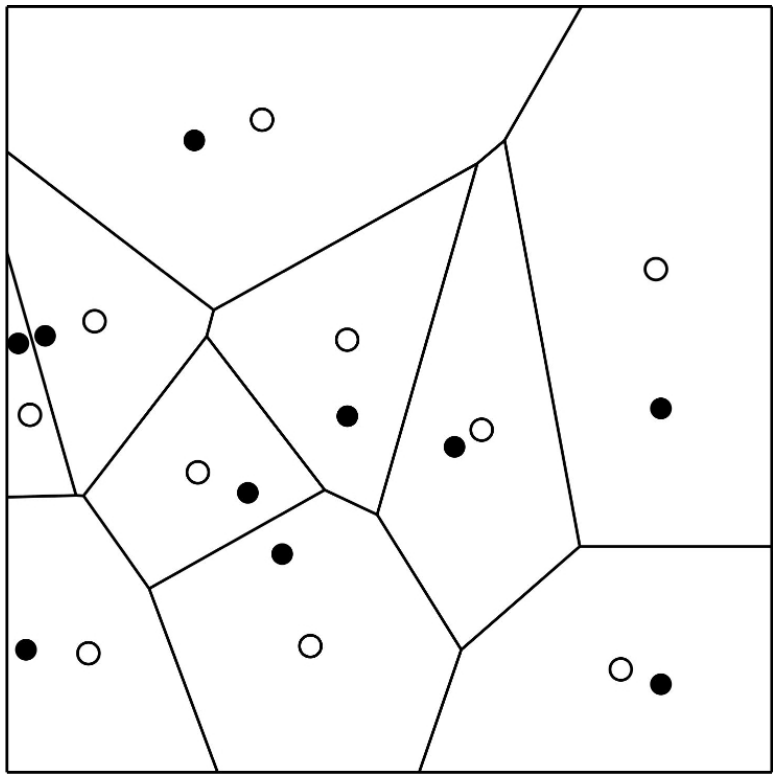
- Given the Voronoi sets V_k, $k = 1, \ldots, K$
  - we can construct the corresponding centroids

$$\boldsymbol{z}_k^* \quad \text{for } k = 1, \ldots, K$$

- In general, the centroids of the Voronoi sets do not coincide with the generators of the Voronoi sets

$$\boldsymbol{z}_k \neq z_k^* \quad \text{for } k = 1, \ldots, K$$

[5] We have been discussing CVTs in terms of points, regions, Euclidean distances, etc.
for the most part, we will continue to do so
however, lots of what we present applies to more general objects, sets of objects, and/or different distance measures



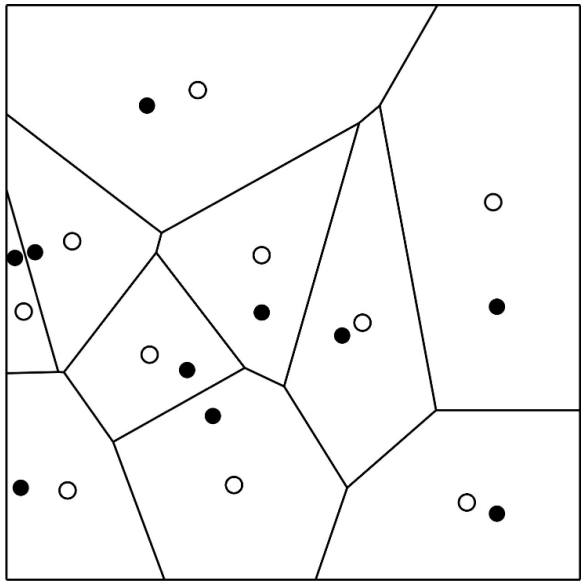
• generators of the Voronoi regions

◦ centroids of the Voronoi regions

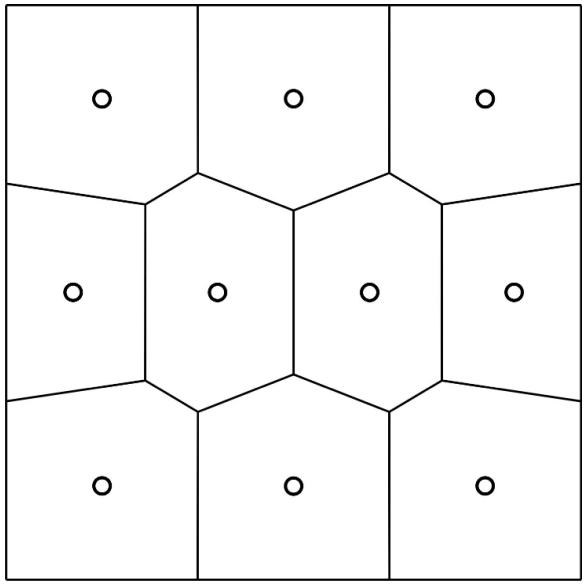
- the generators do not coincide with the centroids

- We are interested in the very special cases
  for which the generators and centroids do actually coincide

$$\boldsymbol{z}_k = \boldsymbol{z}_k^* \qquad \text{for } k = 1, \ldots, K$$



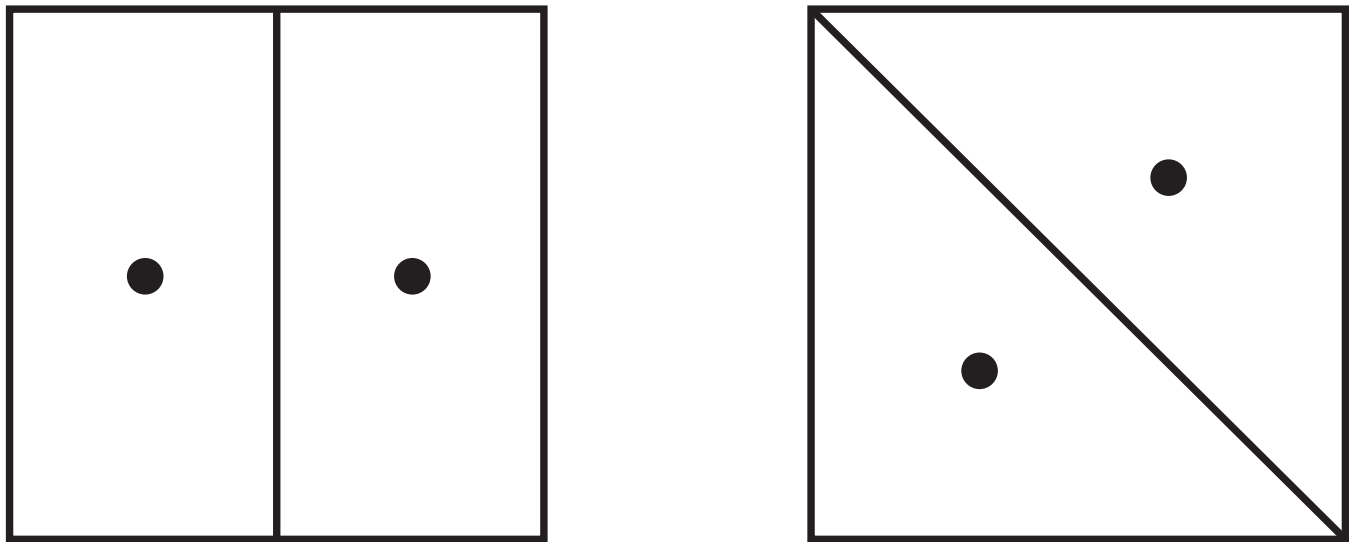
Random: centroids and generators do not coincide



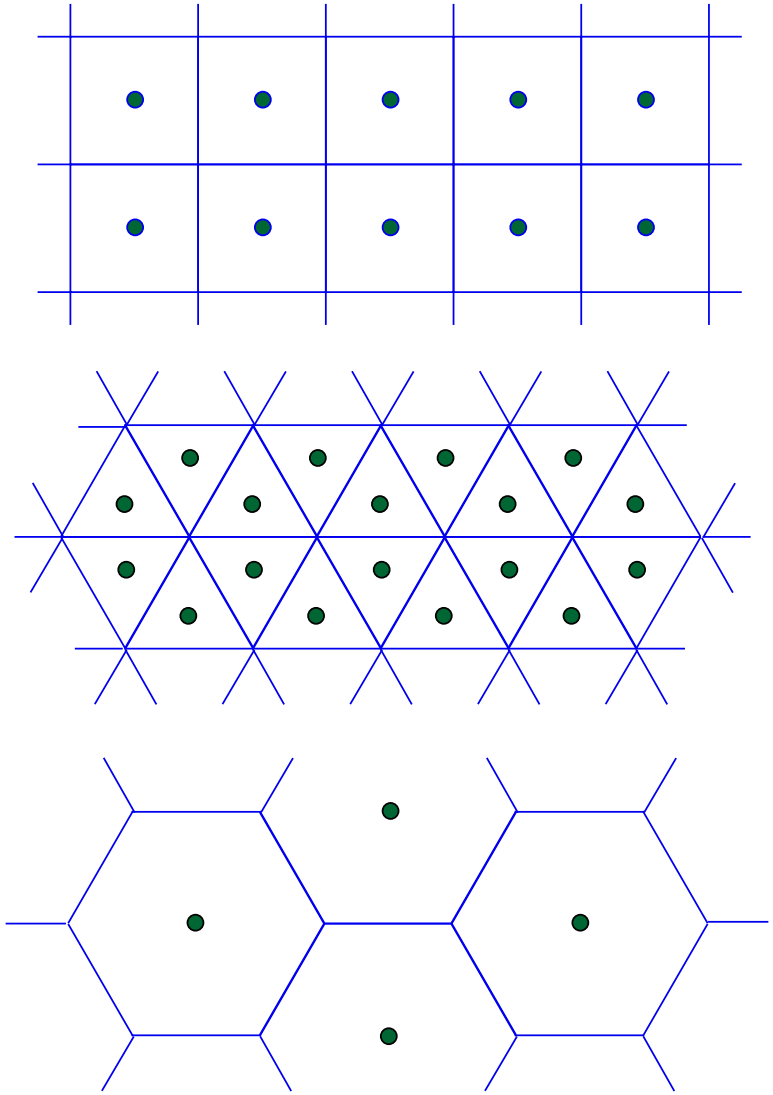
CVT: centroids and generators do coincide

– we refer to the tessellation associated with such a happenstance as a centroidal Voronoi tessellation

- Centroidal Voronoi tessellations (CVTs) do not happen by accident
  - in fact
    the probability is zero that a randomly
    selected set of points $\{\boldsymbol{z}_k\}_{k=1}^{K}$ has the CVT property
    that is
    that they are at the same time
    the generators of a Voronoi tessellation
    and
    the centroids of the Voronoi regions
  - therefore, CVTs must be constructed by some method
- Note that, in general, one does not have uniqueness
- We again note that it one is dealing with a finite set of objects, then, for the most part, CVT reduces to k-means



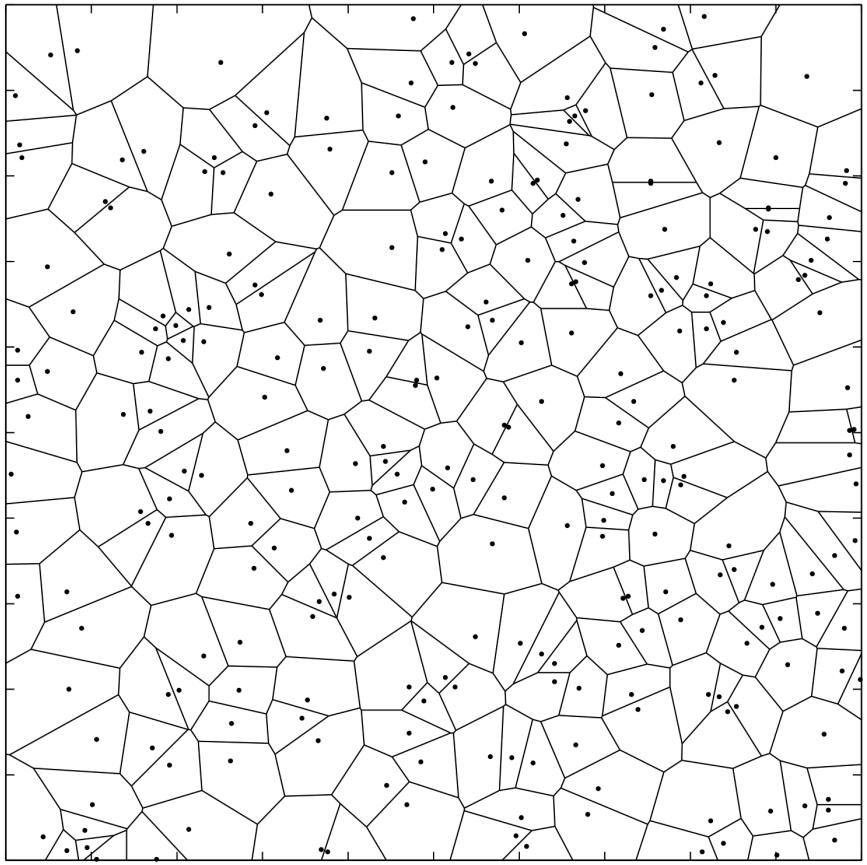
Two two-point centroidal Voronoi tessellations of a square



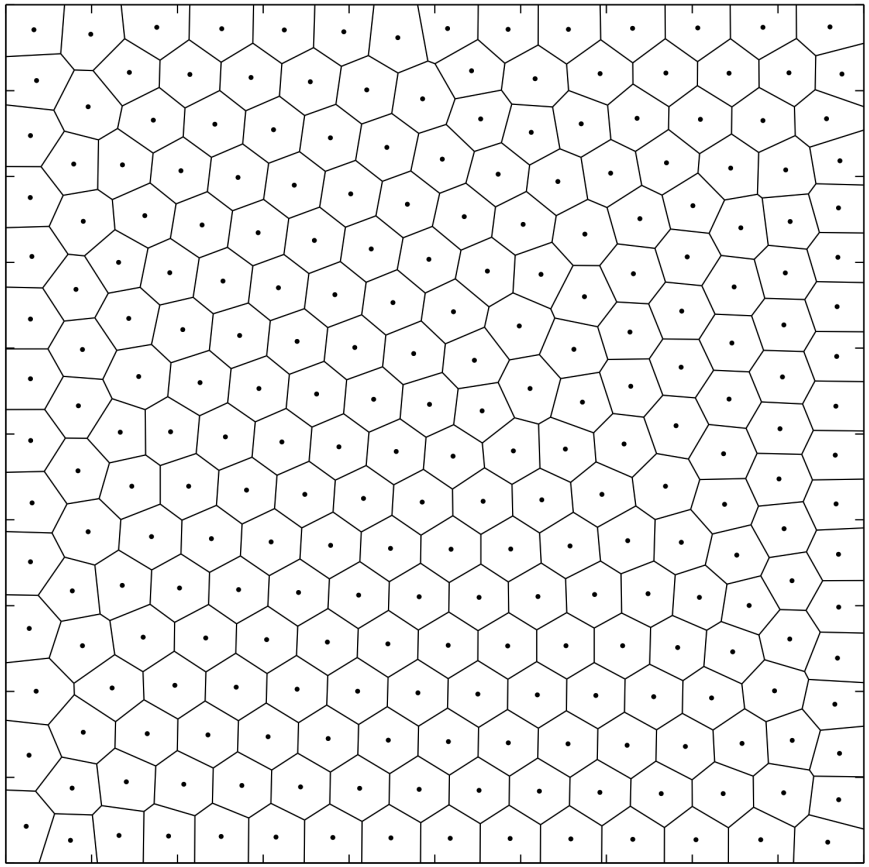
The three regular tessellations of the plane are CVTs

• Example: CVTs in the square

- uniform weight 256 generators



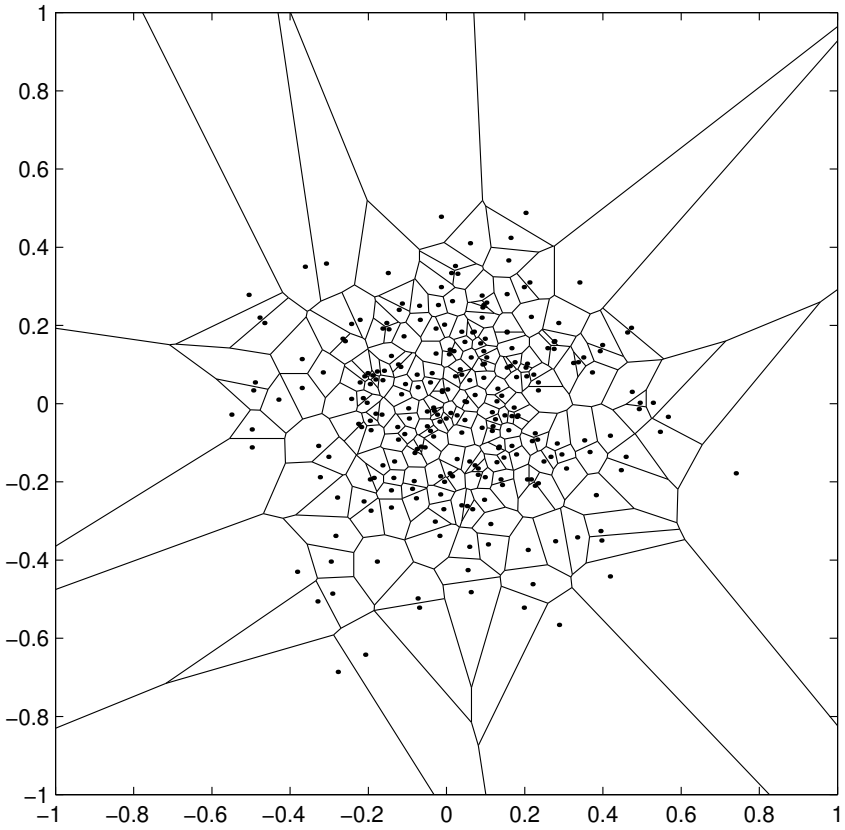
Random sampling



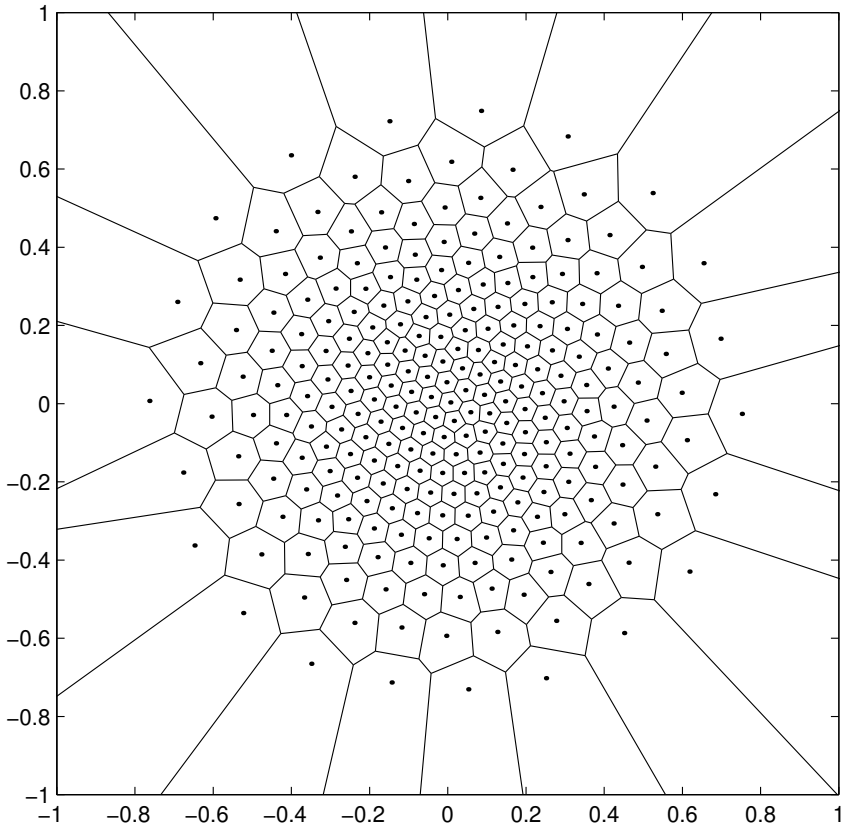
Centroidal Voronoi

- weight function with peak in middle

256 generators



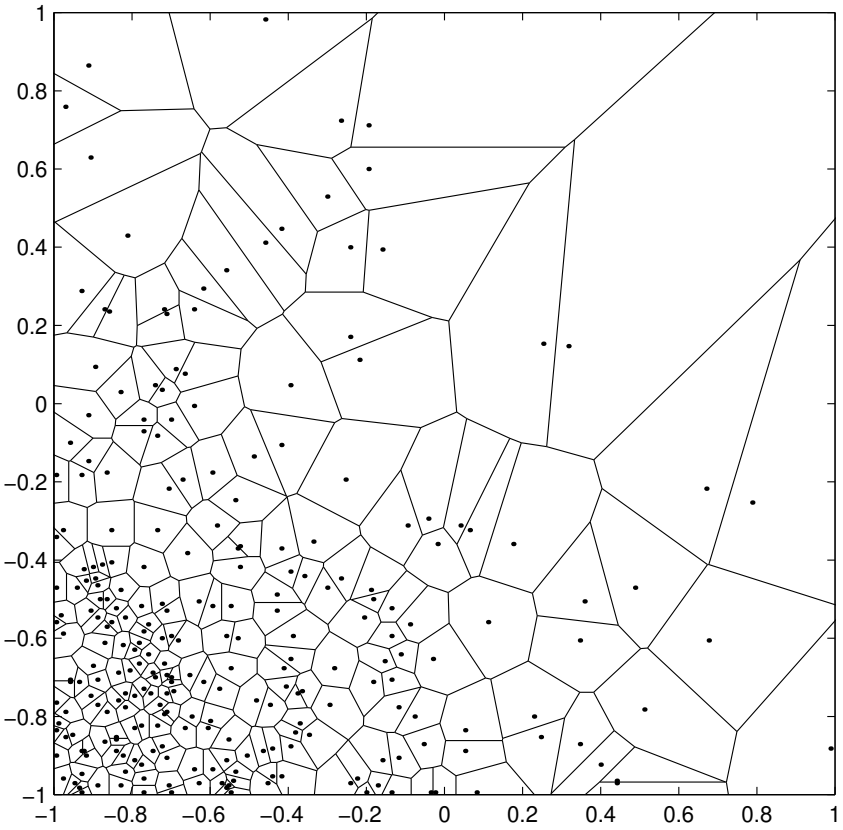
Random sampling



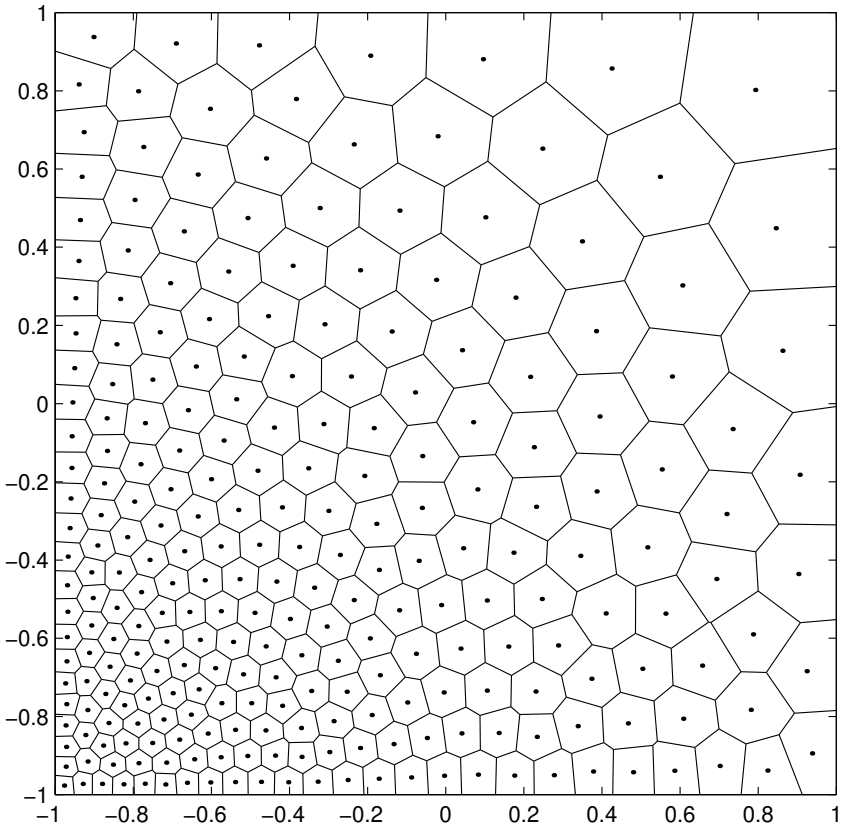
Centroidal Voronoi

- weight function with peak at a corner

256 generators

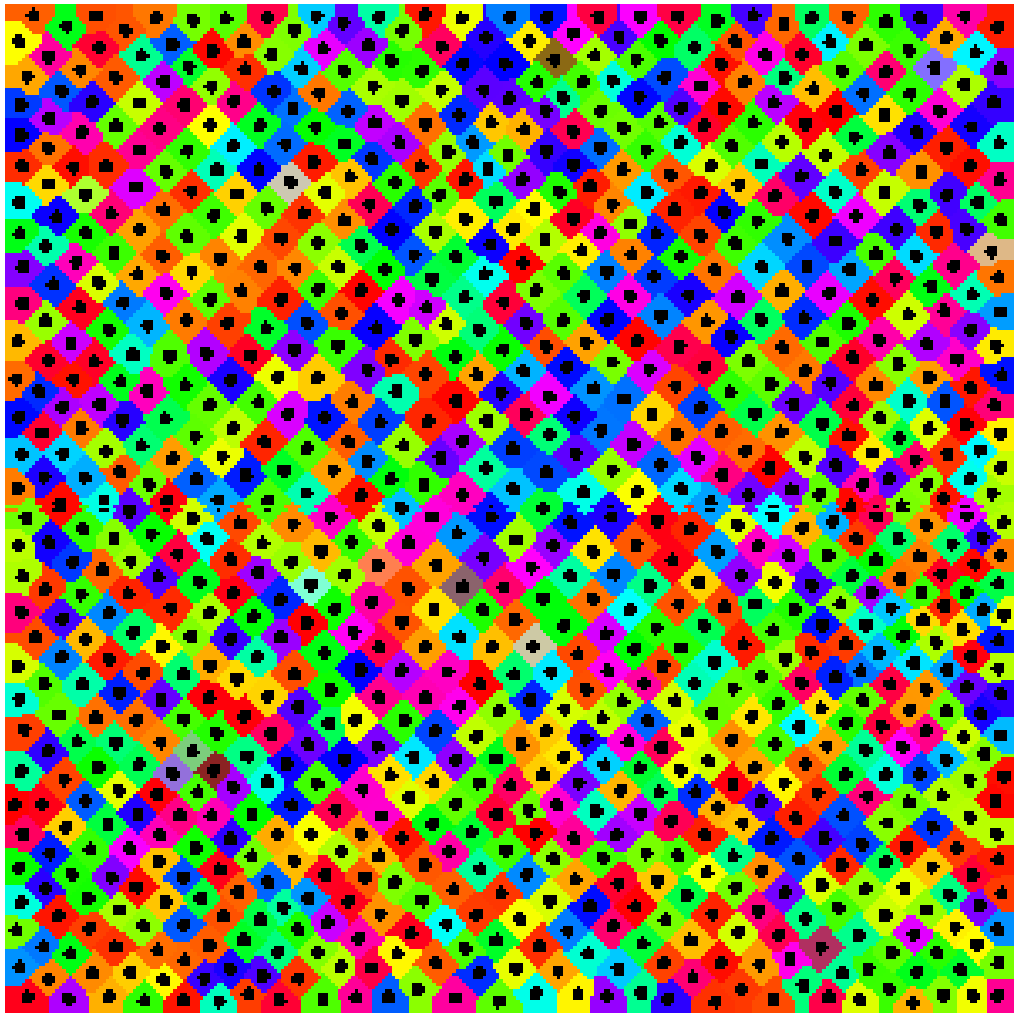


Random sampling

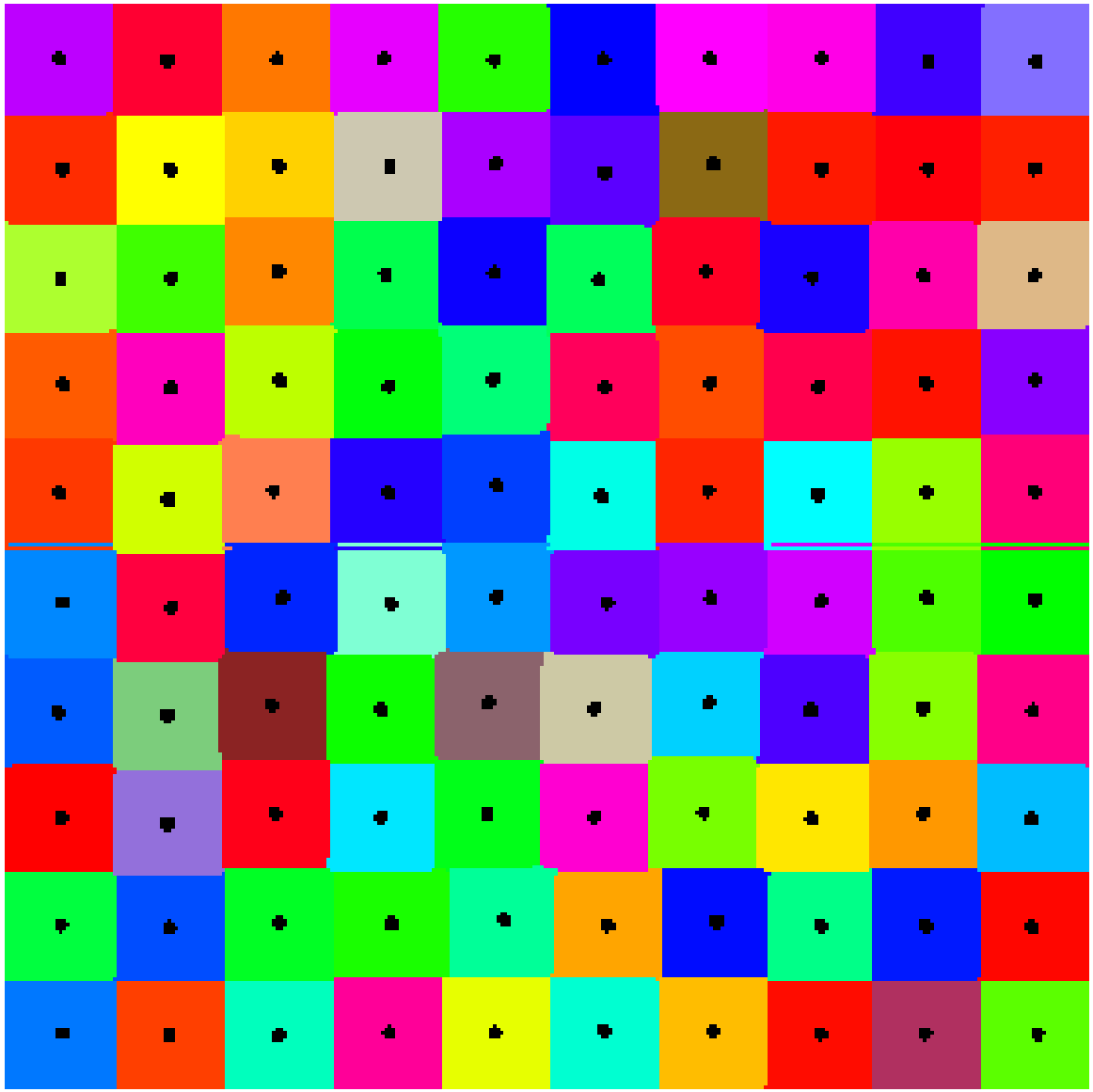


Centroidal Voronoi

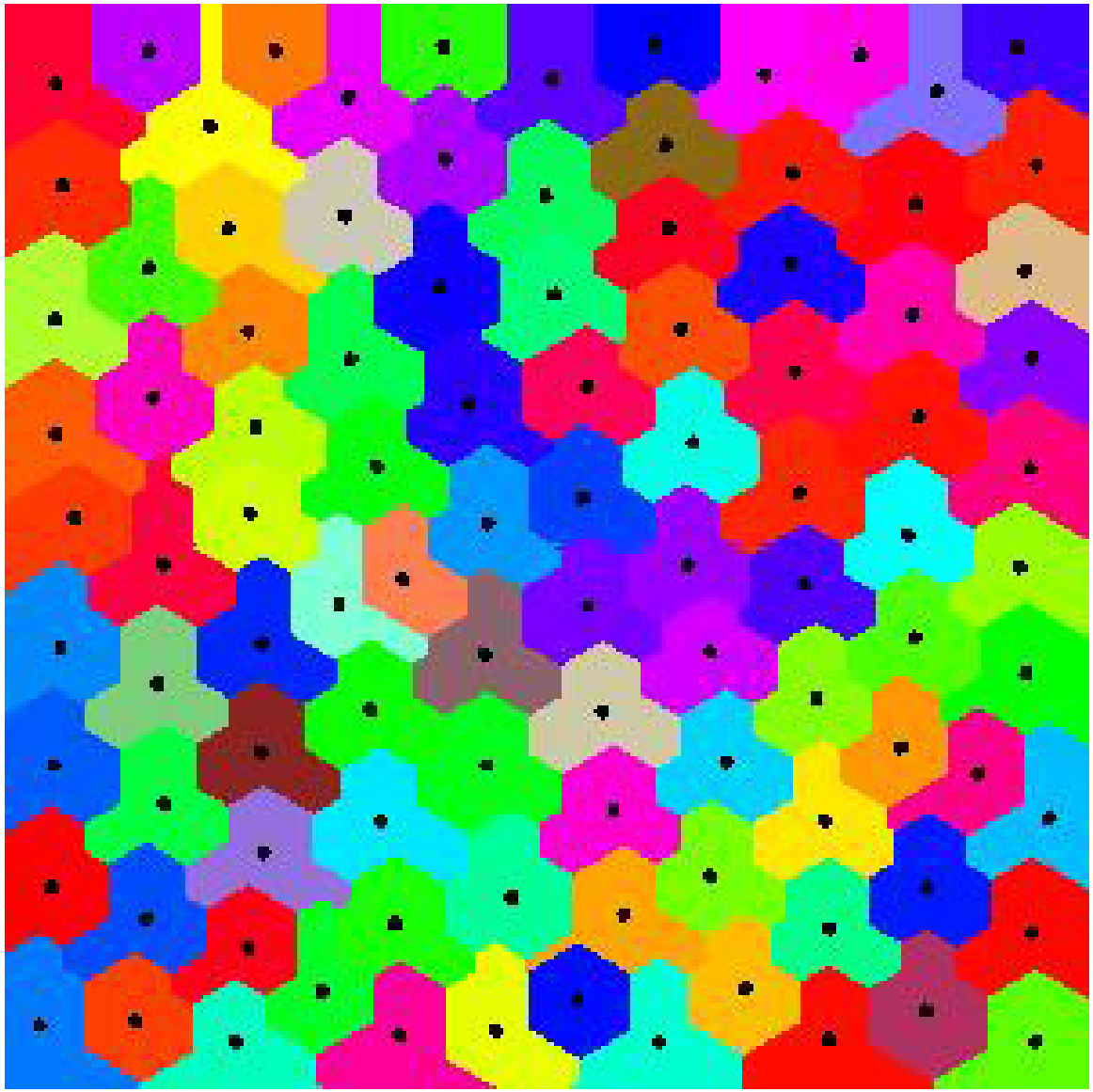
- CVTs of the square for other metrics



CVT for the ℓ^1 metric



CVT for the ℓ^{20} metric (approximating the ℓ^{∞} metric)

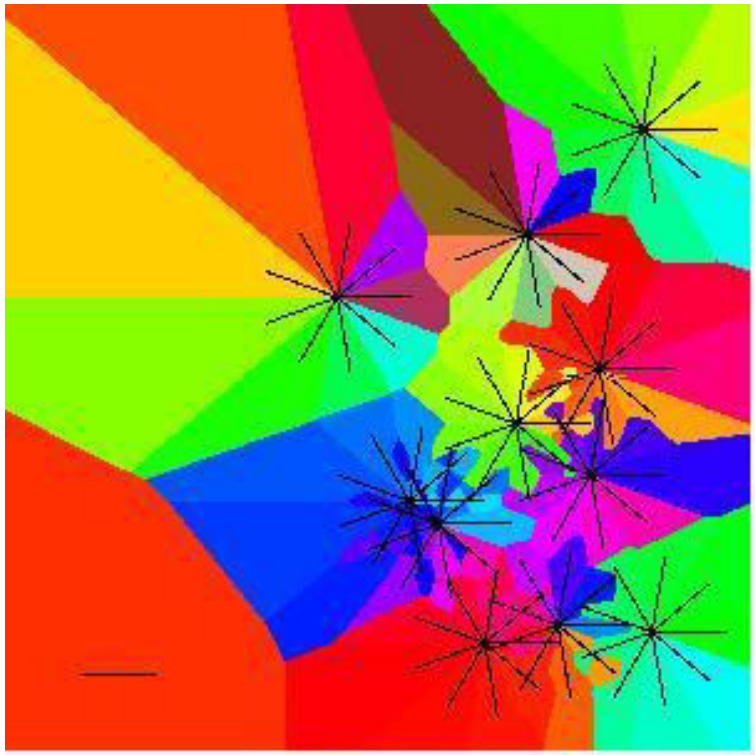


CVT for the triangle metric

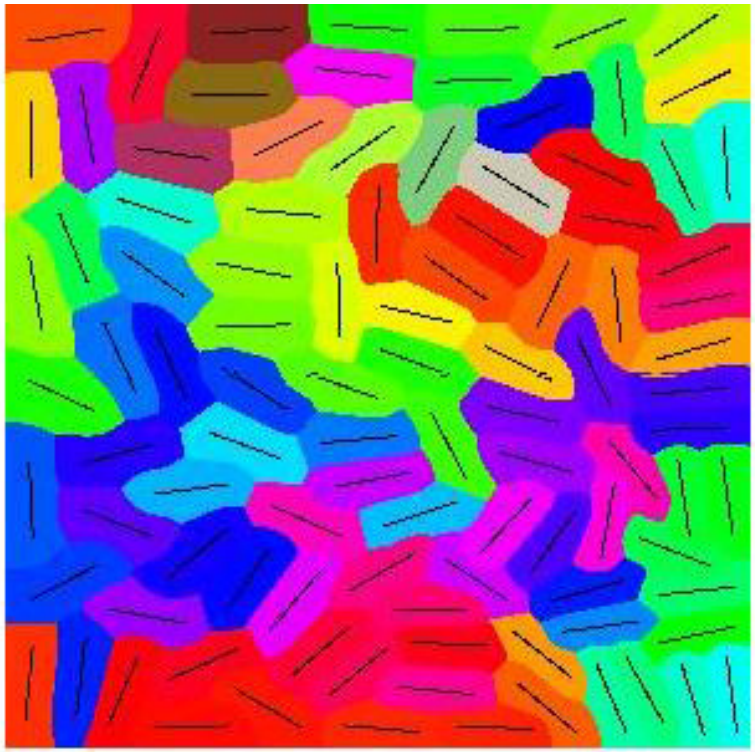
- CVTs for generalized generators

  e.g., generators are straight line segments

  Voronoi regions have points that are
  closer to one line than to any other line

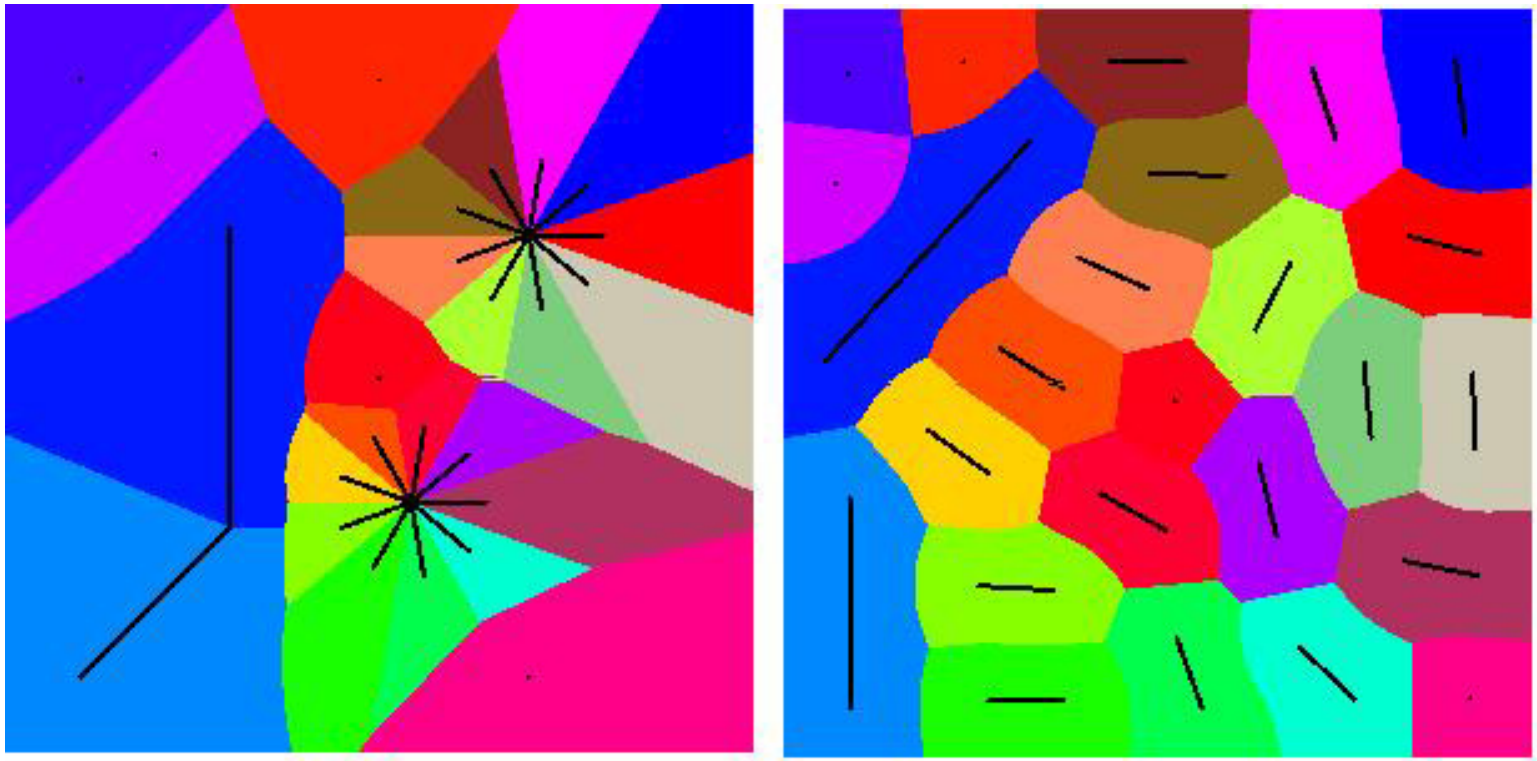


not a CVT



is a CVT

Voronoi tesselations with line generators



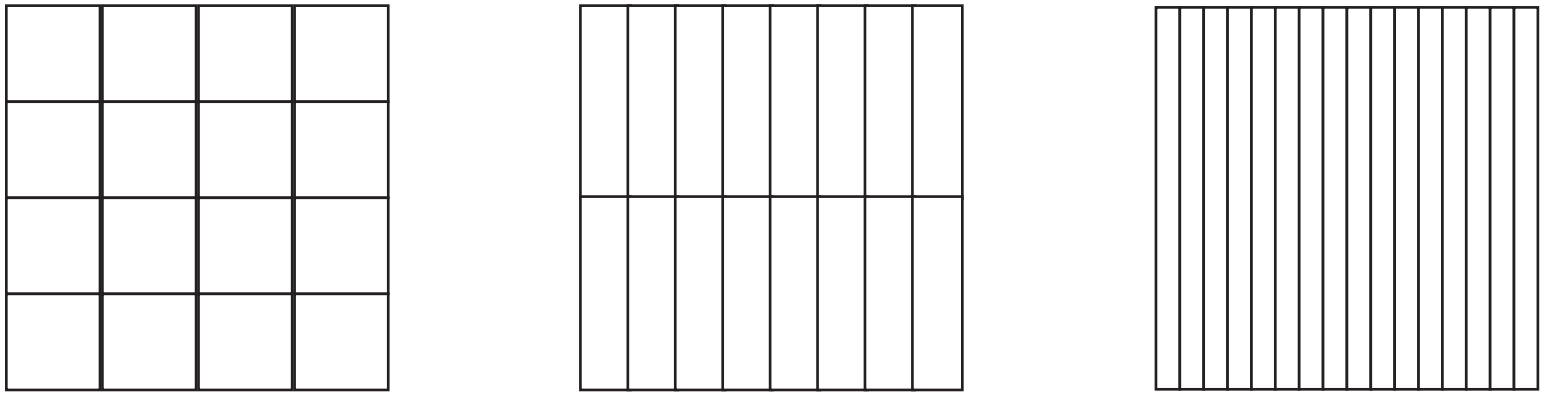
not a CVT

is a CVT

Can have both point and line generators

- Clearly, even for uniform weights and the Euclidean metric there are many possible CVT's for a square

  – for example, here are three 16-point CVT's of the square

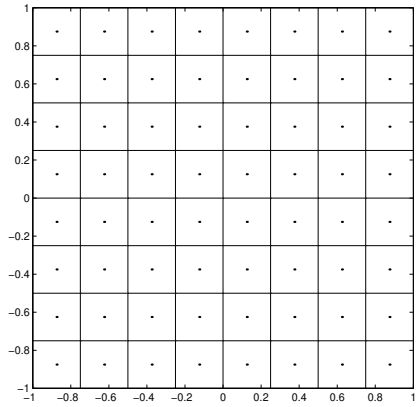


  – which CVT will the CVT construction algorithms give you?

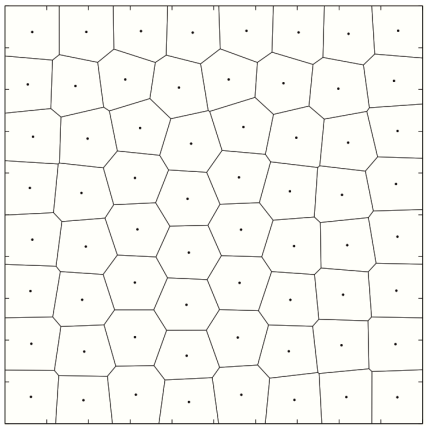
    - actually it is none of these if you use the Euclidean metric

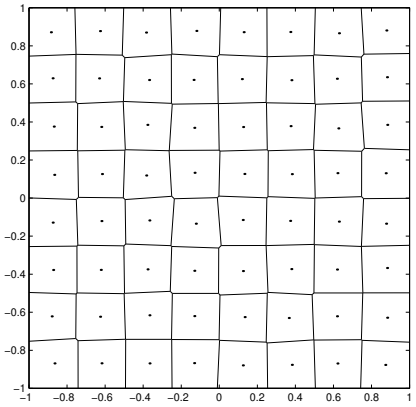
– even if you start from a CVT, if you apply any CVT construction process, you will end up with a CVT that is as hexagonal as possible

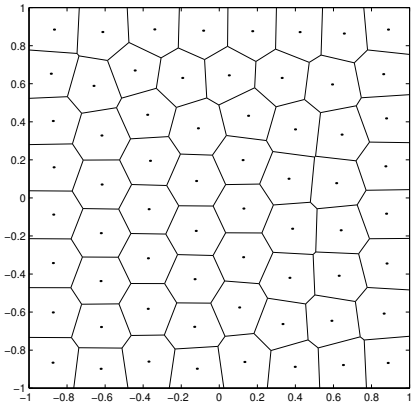
- stages in a CVT iterative construction algorithm $\Longrightarrow$

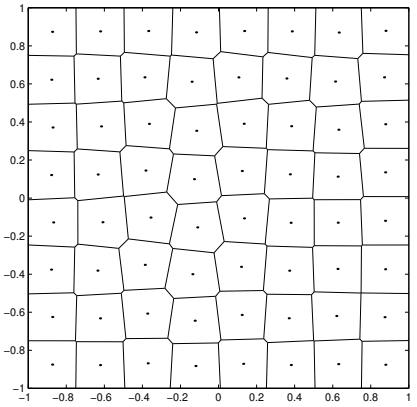


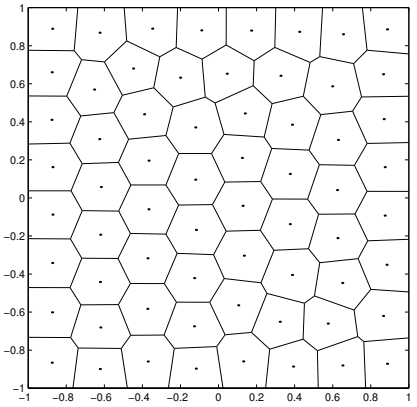
initial











final

Centroidal Voronoi tessellations as minimizers

- Given
  - a domain Ω
  - a positive integer K
  - a density function $w(\cdot)$ defined on $\overline{\Omega}$
  - let
    - $\{\boldsymbol{z}_k\}_{k=1}^K$ denote any set of K points belonging to $\overline{\Omega}$
    - $\{V_k\}_{k=1}^K$ denote any tessellation of Ω into K regions
      - the points do not have be generators of a Voronoi tessellation
      - the tessellation does not have to be a Voronoi one
  - let

$$\mathcal{F}_{cvt}\left(\{\boldsymbol{z}_k, V_k\}_{k=1}^K\right) = \sum_{k=1}^{K} \int_{\boldsymbol{z}\in V_k} w(\boldsymbol{z})|\boldsymbol{z}-\boldsymbol{z}_k|^2 d\boldsymbol{z}$$

- Then, a necessary condition for $\mathcal{F}$ to be minimized is that
  $\{V_k\}_{k=1}^K$ and $\{\boldsymbol{z}_k\}_{k=1}^K$ form a centroidal Voronoi tessellation of Ω

– if Ω is bounded, then $\mathcal{F}$ has a global minimizer

– assume that $w(\cdot)$ is positive except on a set of measure zero in Ω

- then $\boldsymbol{z}_k \neq \boldsymbol{z}_{k'}$ for $k \neq k'$

– for general metrics

- existence is proved by the compactness of the Voronoi regions

- uniqueness can also be proved under some assumptions
e.g., convexity of the metric

– there are many additional results available for the discrete (k-means) case

- many of these are in the nature of limiting results
as the number of generator increases

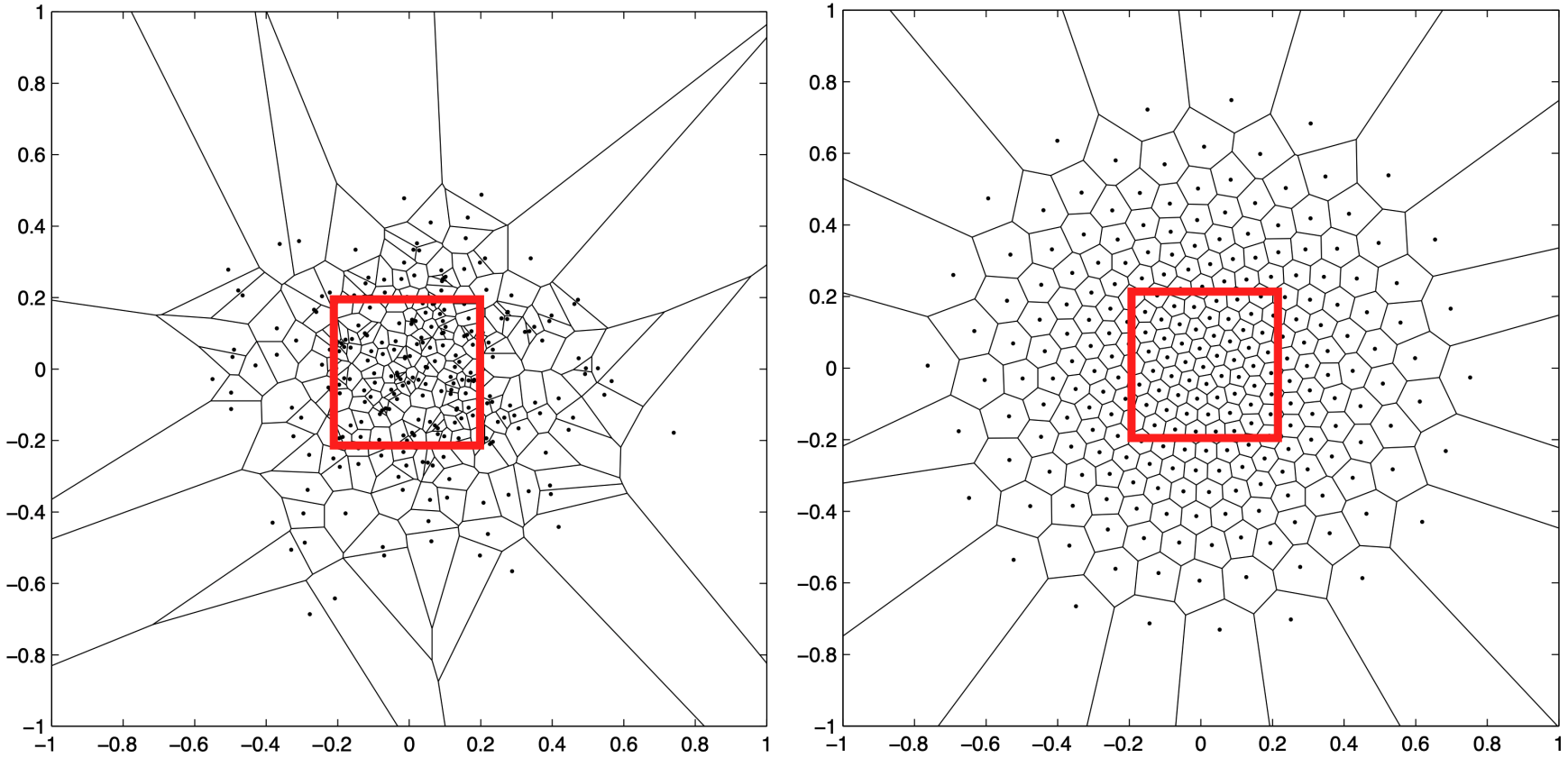
• All that was written concerning the functional $\mathcal{F}_{CVT}$ holds for the discrete k-means functional

$$\mathcal{F}_{k-means}\left(\{\boldsymbol{z}_k, S_k\}_{k=1}^{K}\right) = \sum_{k=1}^{K} \sum_{\boldsymbol{z}_m \in S_k} w_m |\boldsymbol{z}_m - \boldsymbol{z}_k|^2 d\boldsymbol{z}$$

Local uniformity of centroidal Voronoi tessellations

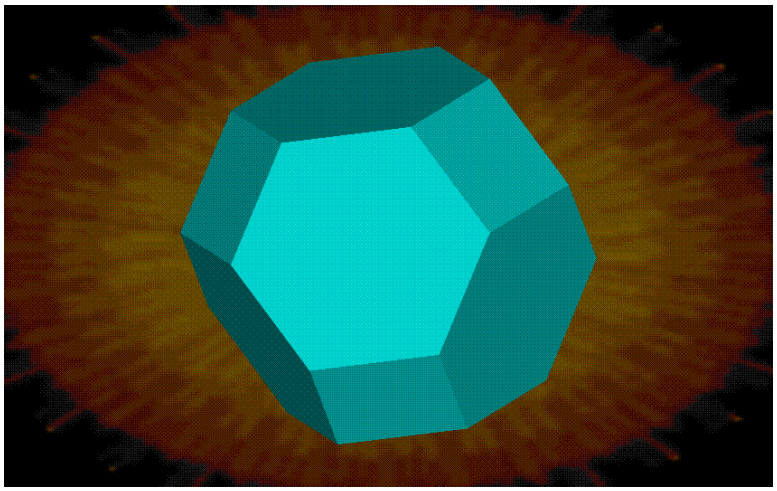
- Gersho's conjecture (proven in 2D under assumptions)
  - for any weight function, as the number of points increases
    the distribution of CVT points becomes locally uniform
    - this means that if
      one looks at small enough subdomain
      containing a large enough number of points
      a CVT point distribution will look uniform
      regardless of how nonuniform it is globally
  - locally in 2D, CVT Voronoi regions tend to be congruent regular hexagons

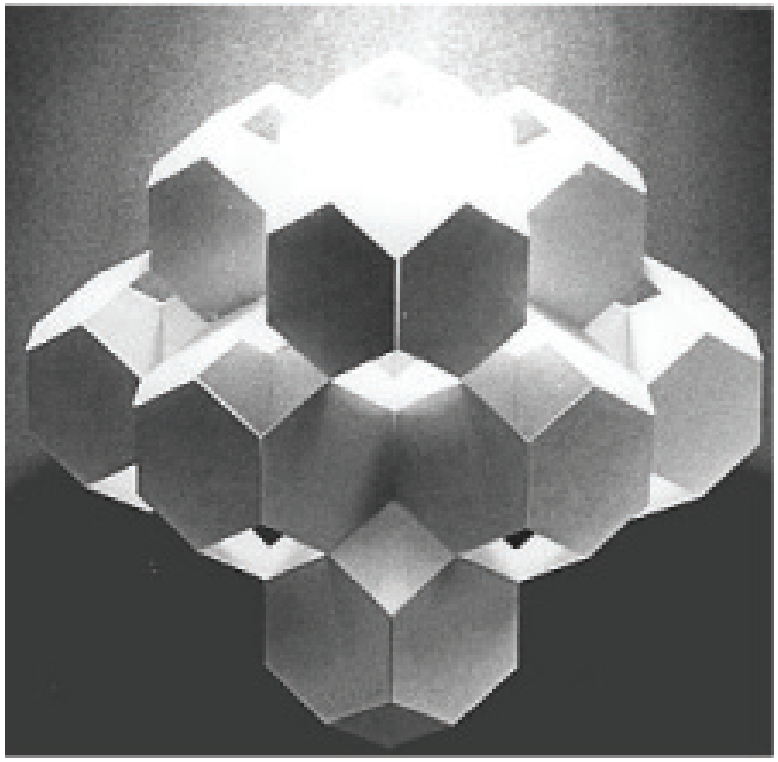
– compare the uniformity of nonuniform MC and nonuniform CVT in a region having a high density of points



– in higher dimensions, the basic cell of a CVT is not known

- in 3D, computational studies indicate that it is a truncated octahedron (one of the Platonic solids)





ALGORITHMS FOR CONSTRUCTING CVTs

Lloyd's method – iterating between generators and centroids

0. start with some initial set of K points $\{\boldsymbol{z}_k\}_{k=1}^K$ in a region Ω

1. construct the Voronoi tessellation $\{V_k\}_{k=1}^K$ of Ω
   with the points $\{\boldsymbol{z}_k\}_{k=1}^K$ being the generators

2. construct the centroids $\{\boldsymbol{z}_k^*\}_{k=1}^K$ of the Voronoi
   regions $\{V_k\}_{k=1}^K$ found in Step 1

3. set the centroids found in Step 2 to be a new set of generators

4. go back to Step 1, or, if you are happy with what you have, quit

- note that in steps 1 and 2
  one has to explicitly construct the Voronoi tessellations
  one has to explicitly determine centroids

software is available to do both tasks

McQueen's method – concurrent random sampling and weighted averaging

- Because McQueen's method involves sampling points

  it is actually an implementation of
  the weighted k-means method

  - this observation is even more relevant for the
    modified McQueen's method we consider later

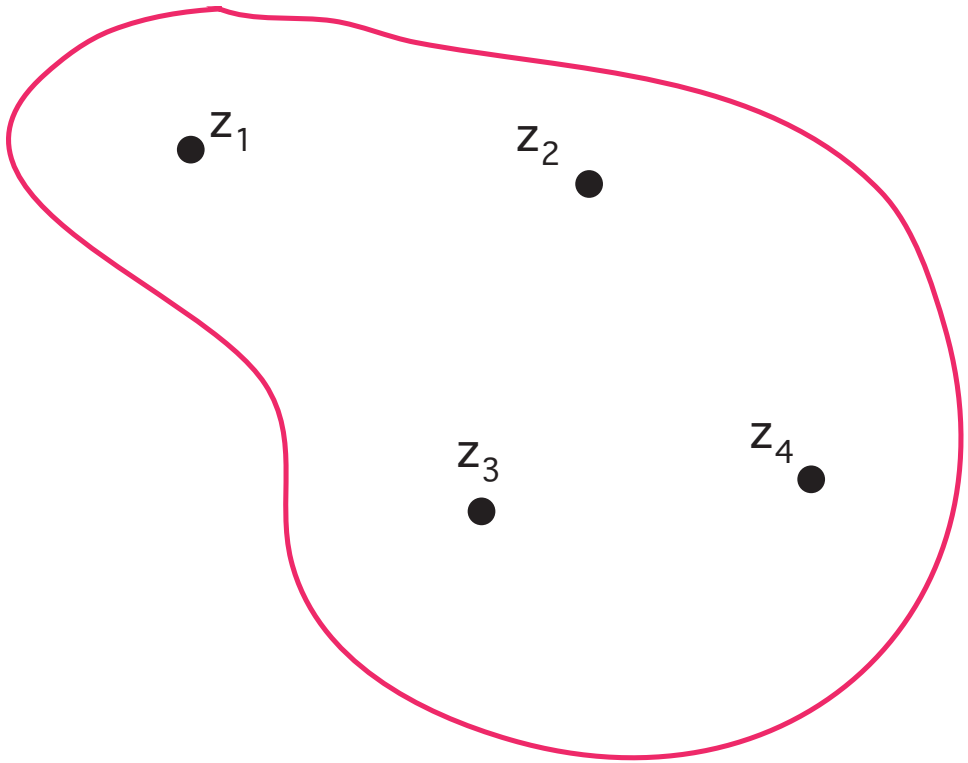
  – in the continuous setting

    - McQueen's method does not require the explicit
      construction of Voronoi tessellations or of centroids

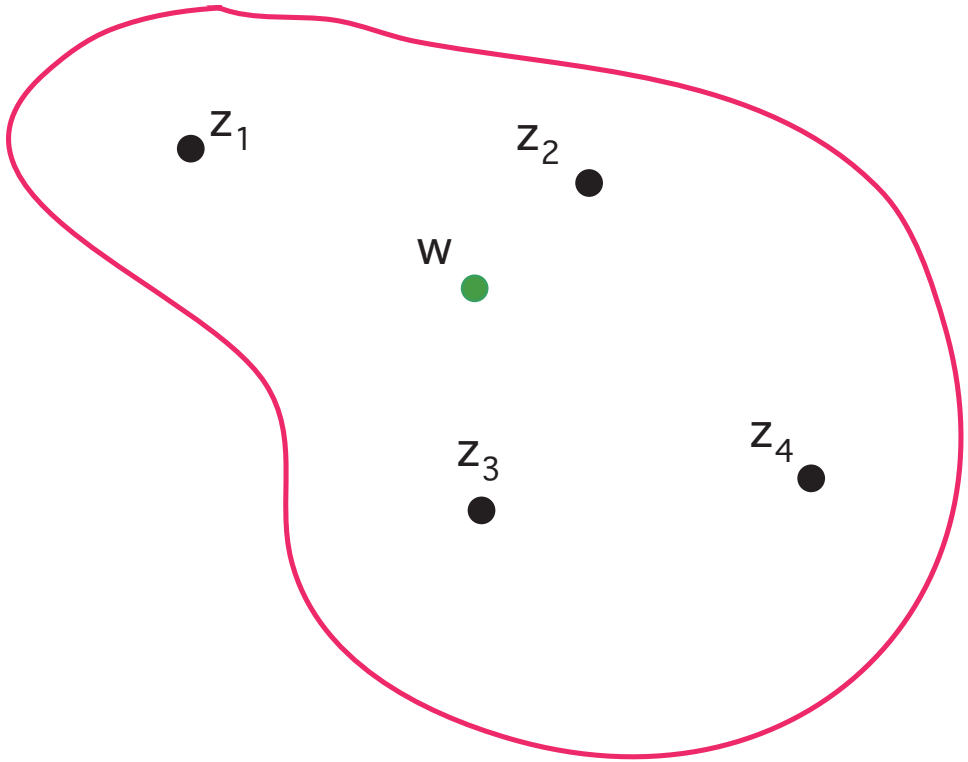
  – in the discrete setting

    - McQueen's method provides an exact k-means clustering

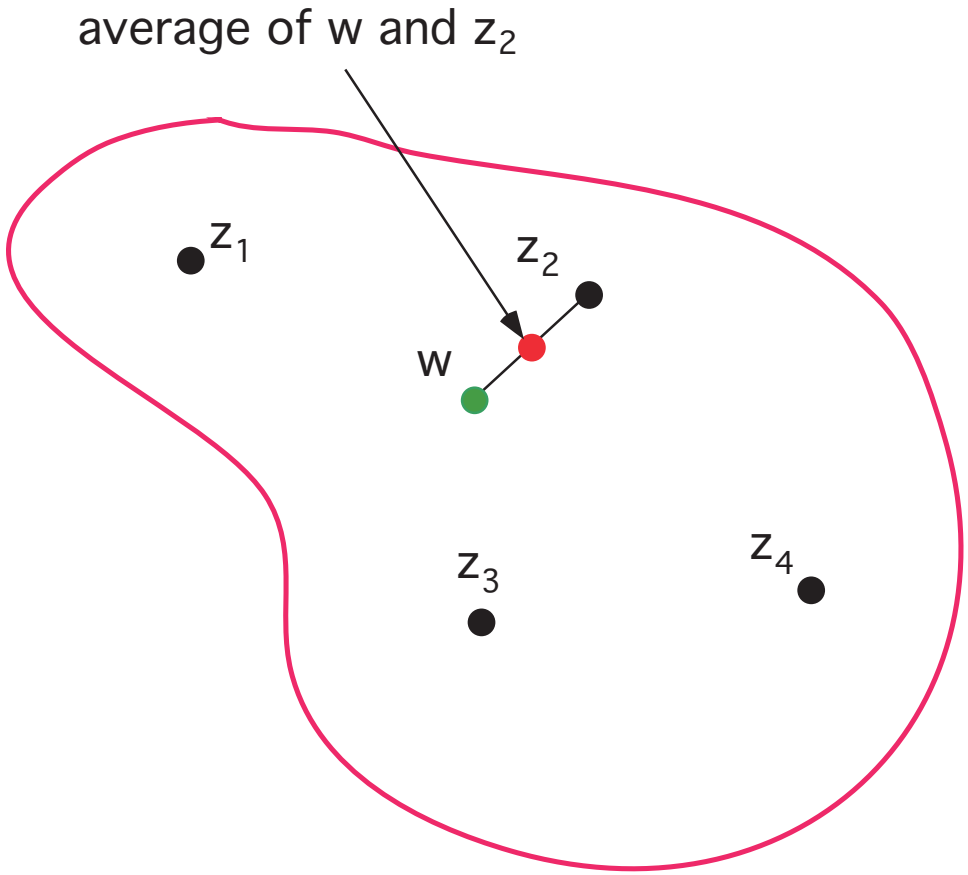
- Start with some initial set of K points $\{\boldsymbol{z}_k\}_{k=1}^{K}$ ($K = 4$ in the sketch)



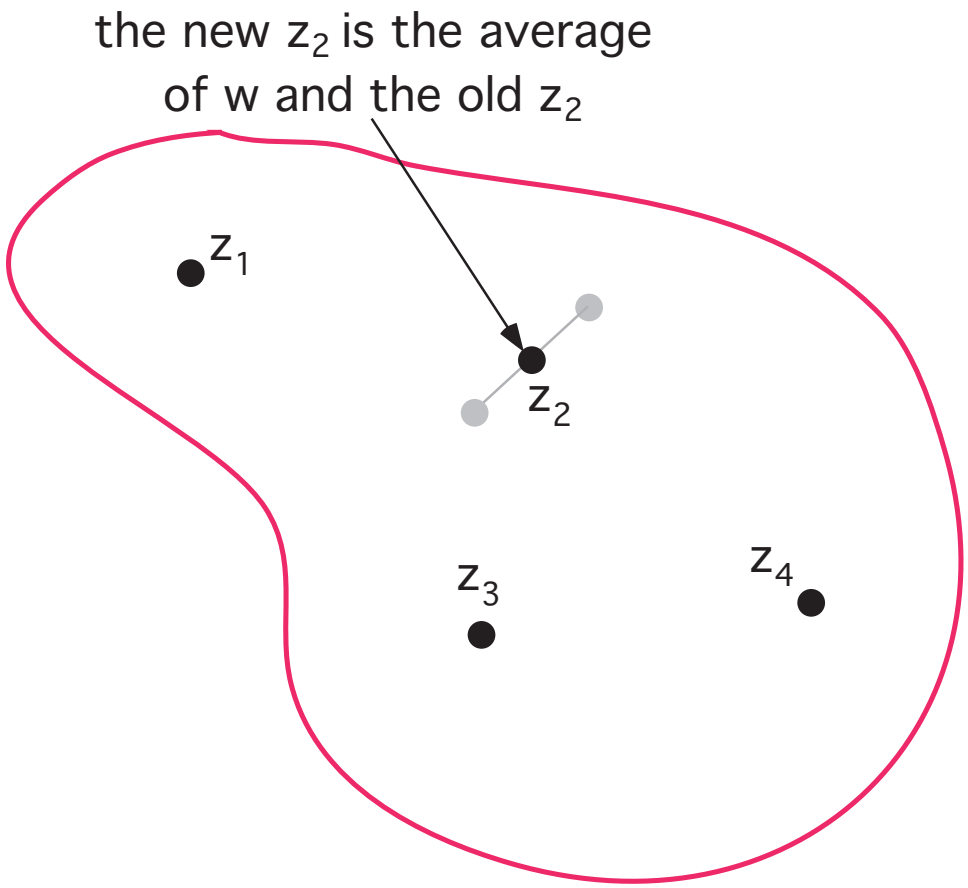
- Sample another point **w**
- Determine which of the z_k's is closest to **w** (it is z_2 in the sketch)



- Find the average of **w** and the z_k closest to it



- Replace the z_k by the average point



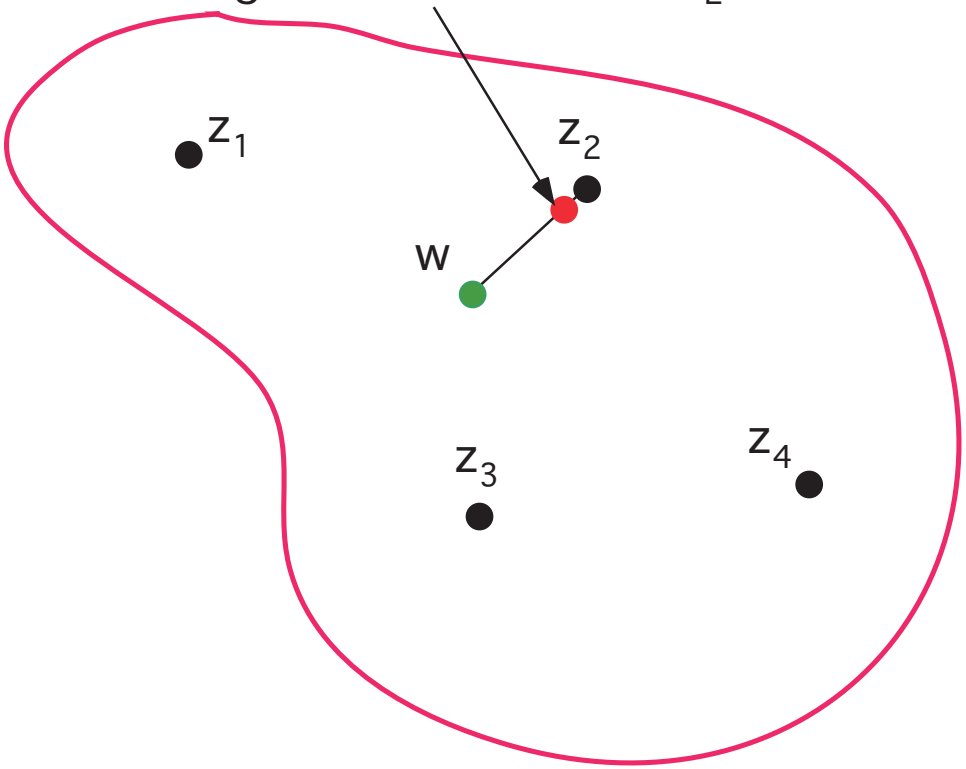
- Continue the process, that is,
  - sample points $\mathbf{w}$
  - find the closest $\boldsymbol{z}_k$
  - average $\mathbf{w}$ and that $\boldsymbol{z}_k$
  - replace that $\boldsymbol{z}_k$ by the average

except

  - we keep track of how many times a point $\boldsymbol{z}_k$ has been previously updated
  - when we do the averaging, we weight the old point according to the number of times it has been previously updated

- For example, suppose $\boldsymbol{z}_2$ had already been updated 12 times (counting the initial positions as the first update); then
  - instead of the new $\boldsymbol{z}_2 \leftarrow \dfrac{\mathbf{w} + \boldsymbol{z}_2}{2}$
  - we have the new $\boldsymbol{z}_2 \leftarrow \dfrac{\mathbf{w} + 12\boldsymbol{z}_2}{13}$

since z_2 had previously updated,
the new z_2 is the weighted
average of w and the old z_2



- As was already noted

  McQueen's method does not require the explicit construction of Voronoi sets or centroids

  – despite this

    the K points produced by McQueen's method converge towards the generators of a CVT

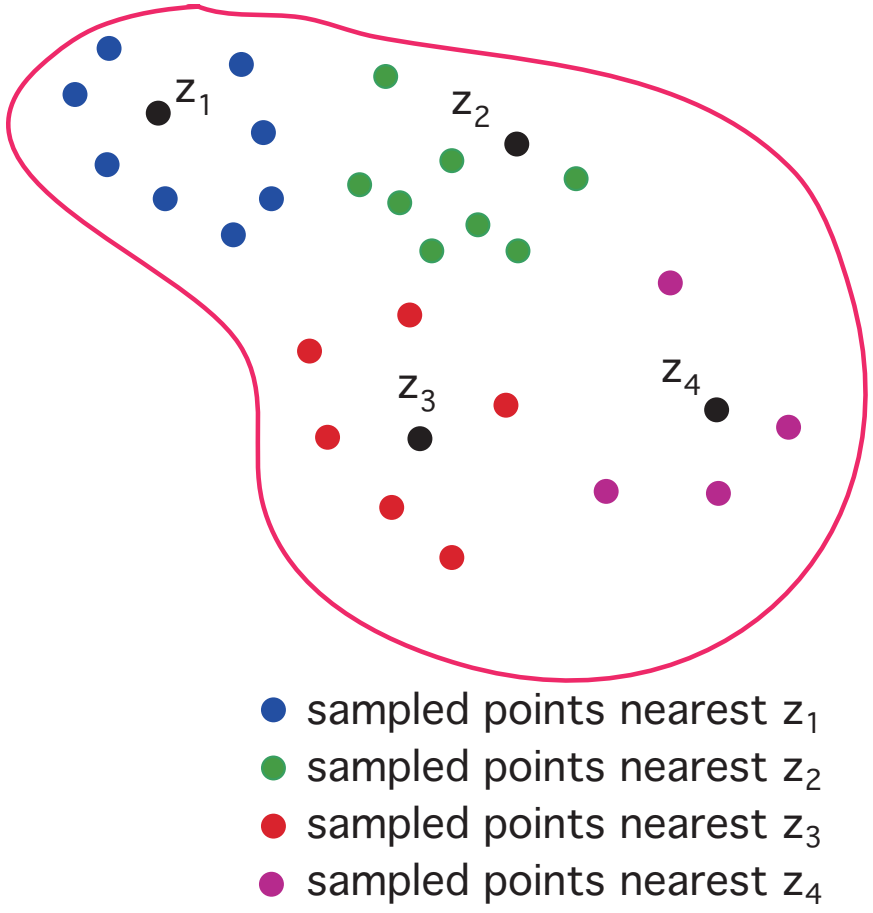
- The convergence of McQueen's method is very slow

  - it takes many steps (millions) to get close to true CVT generating points

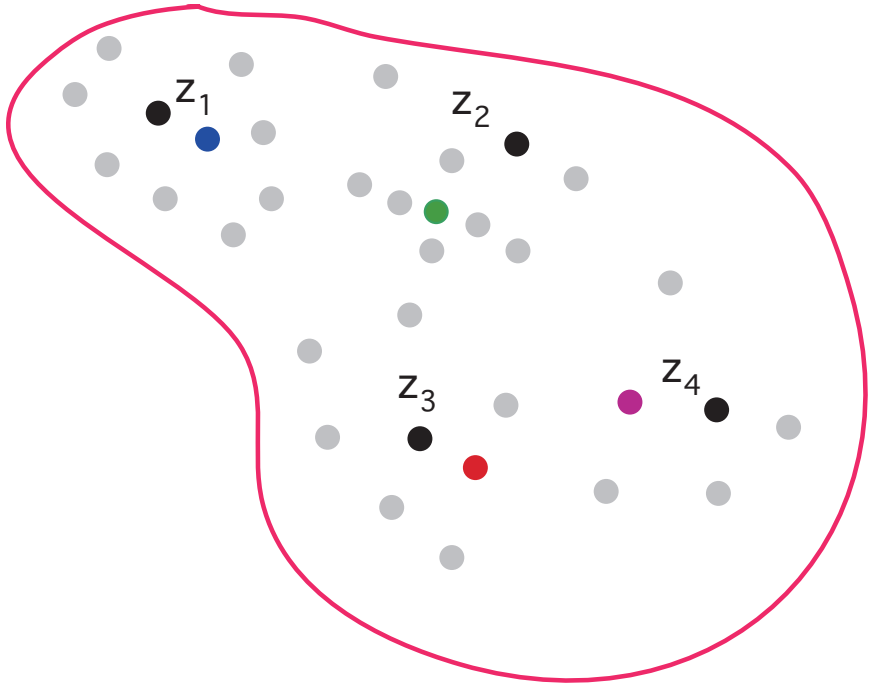
  – the problem with McQueen's method is that it samples only one point at time before it averages

Modified McQueen's method – random sampling and averaging

- Before averaging, sample lots of points (zillions) and then group (cluster) them according to which is the nearest $\boldsymbol{z}_k$

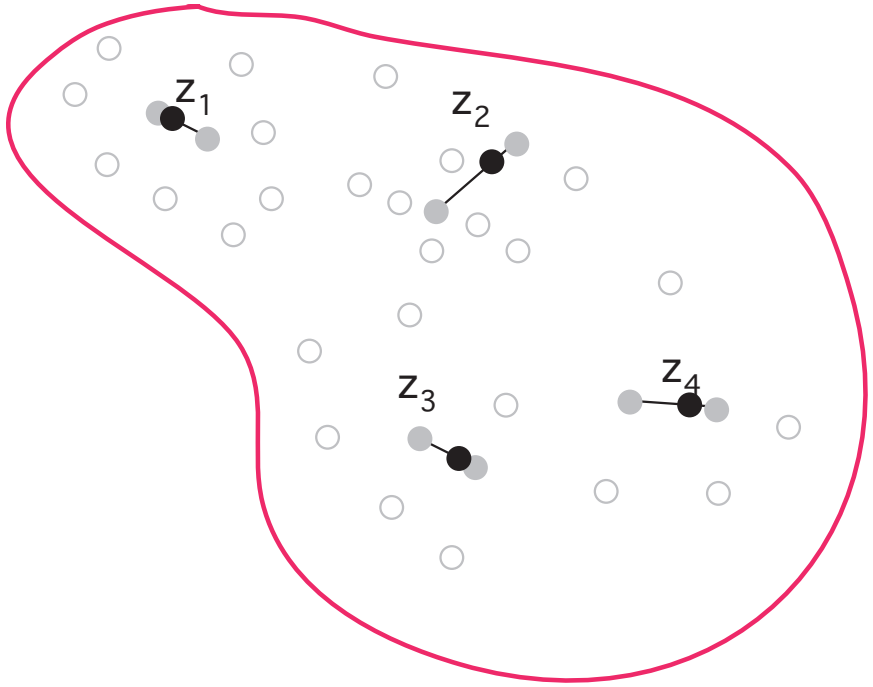


- Find the average of each of the clusters

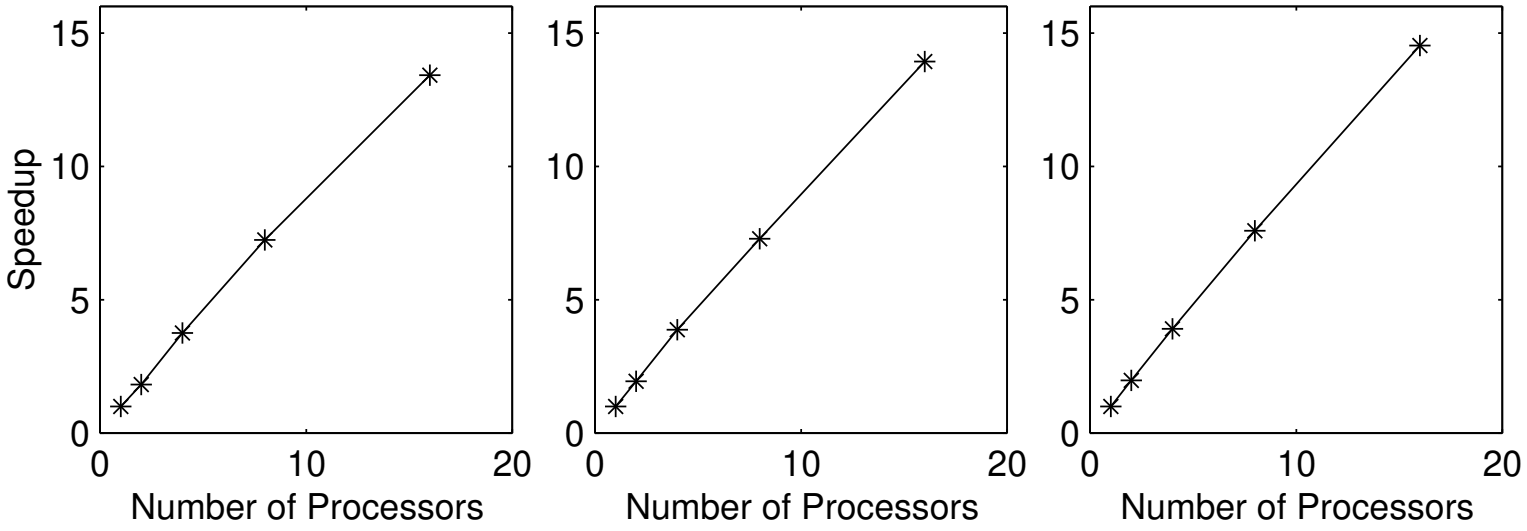


- average of points nearest z_1
- average of points nearest z_2
- average of points nearest z_3
- average of points nearest z_4

- The new $\boldsymbol{z}_k$'s are weighted averages of the old $\boldsymbol{z}_k$'s and the corresponding cluster averages



- The modified McQueen is great for parallelization
  - it exhibits near-perfect scalability



Speedup of a parallel implementation of the modified McQueen's method for three different weight functions that cause different refinements

  - there is available a two parameter family of effective, probabilistic methods for generating CVT's in general regions and with general weight distributions

Functional minimization

- CVTs can be constructed by directly minimizing the functional $\mathcal{F}_{cvt}$

  just as

  k-means clustering can be constructed by directly minimizing the functional $\mathcal{F}_{k-means}$

  – for example, one can use Newton's method for this purpose

Generaliztions of CVTs (a partial list)

in every instance, can replace "CVT" by "weighted k-means"

- Constrained CVT's for placing some or all the points on a surface
- Constrained CVT's for fixing the position of some points
- CVT's for other metrics
  - e.g., for anisotropic point distributions one can use anisotropic metrics
- CVT's for other types of generators
- Constrained CVT's so that some of the final generators are located on the boundary of the domain
  - CVT's hate boundaries; left on their own, no CVT generator would be located on the boundary
  - even if initially one places some generators on the boundary, they leave the boundary in the first step of any of the algorithms
  - something has to be done to force some generators to stay on the boundary
    such as not letting boundary generator participate in the construction process
    or, even better, constraining boundary generators to slide along the boundary

APPLICATIONS OF WEIGHTED k-MEANS and/or CVTs (a partial list)

- optimal quadrature rules
- covolume and finite difference methods for PDE's
- optimal representation, quantization, and clustering
- finite volume methods for PDE's
- optimal placement of sensors and actuators
- surrogate optimization
- particle methods
- stippling
- visualization of software metrics
- melodic structure improvement
- mosaic effects for images
- point distributions and grid generation on surfaces
- meshfree methods

We have already considered one application

- hypercube point sampling

  $\Longleftrightarrow$ the ??? case in the discussion of hypercube point sampling

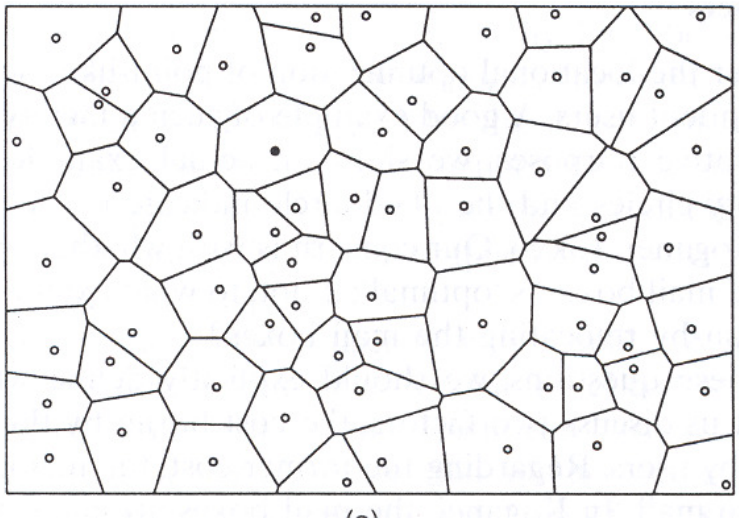
We briefly consider these applications

- optimal distribution of resources
- cell division
- territorial behavior of animals
- data compression
- image segmentation and edge detection
- multichannel reconstruction of images
- reduced-order modeling
- grid generation

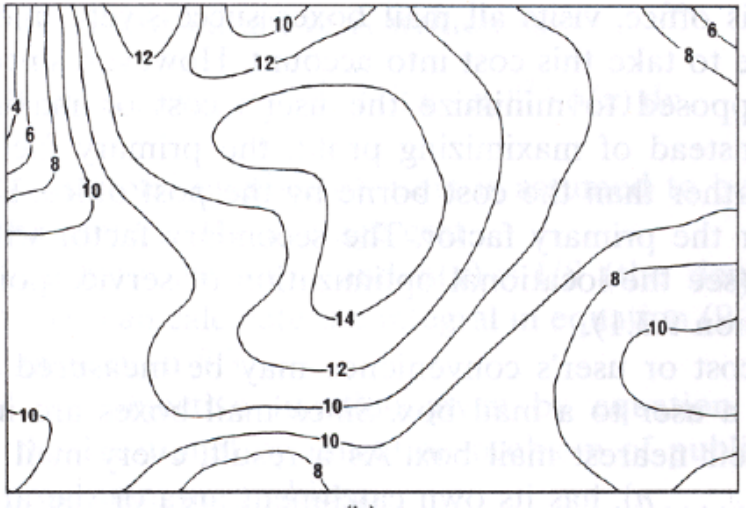
$\cdots\cdots\cdots\cdots\cdots$ and there are more

Optimal distribution of resources

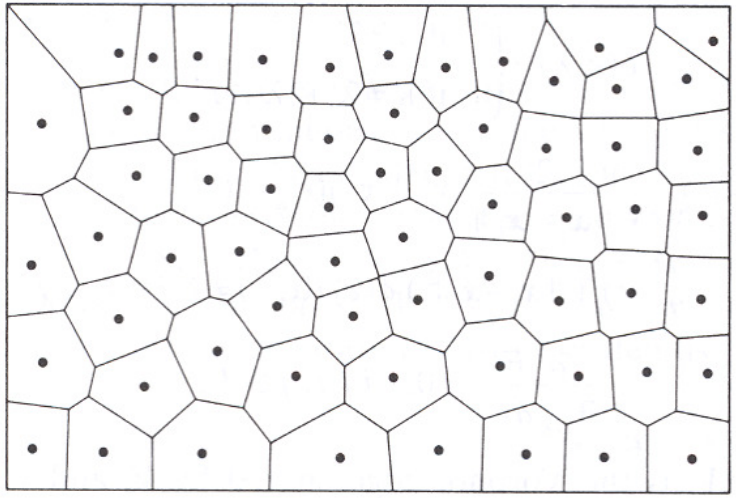
- What is the optimal placement of mailboxes in a city?
- The optimization problem
  - a user will use the mailbox nearest to their home
  - the cost (to the single user) of using a mailbox is proportional to the distance from the user's home to the mailbox
  - the total cost over a whole city of users is measured by the average distance to the nearest mailbox of all users in the city
  - the optimal placement of mailboxes is defined to be the one that minimizes the total cost
- The optimal placement of the mail boxes is at the generators of a centroidal Voronoi tessellation of the city
  - you have to use a different metric because if you use the Euclidean metric, people will be walking on other people's lawns and through other people's houses



Actual distribution of mailboxes
in a district of Tokyo



Population density
in the district



CVT distribution of mailboxes

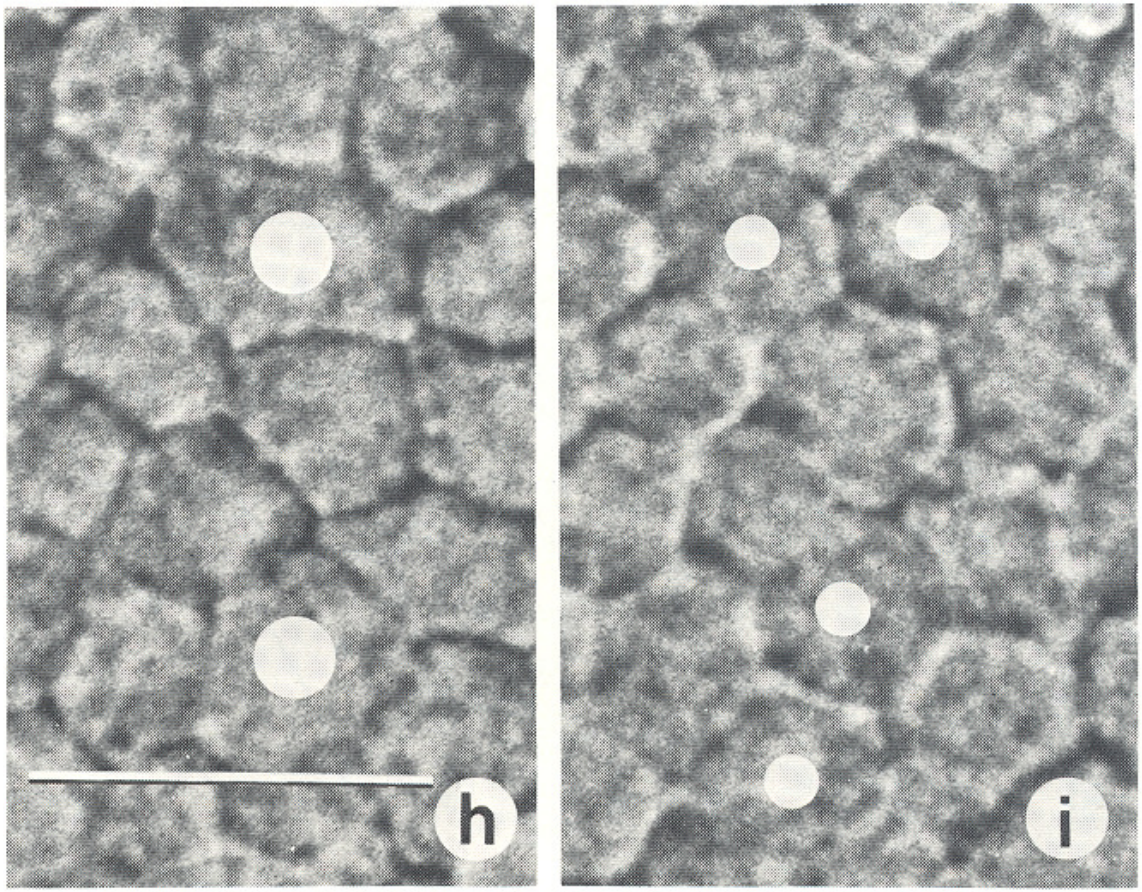
Cell division

- There are many examples of cells that are polygonal
  - often they can be identified with a Voronoi tessellation, indeed, a centroidal Voronoi tessellation
  - this is especially evident in monolayered or columnar cells
    - e.g., as in the early development of a starfish (*Asteria pectinifera*)

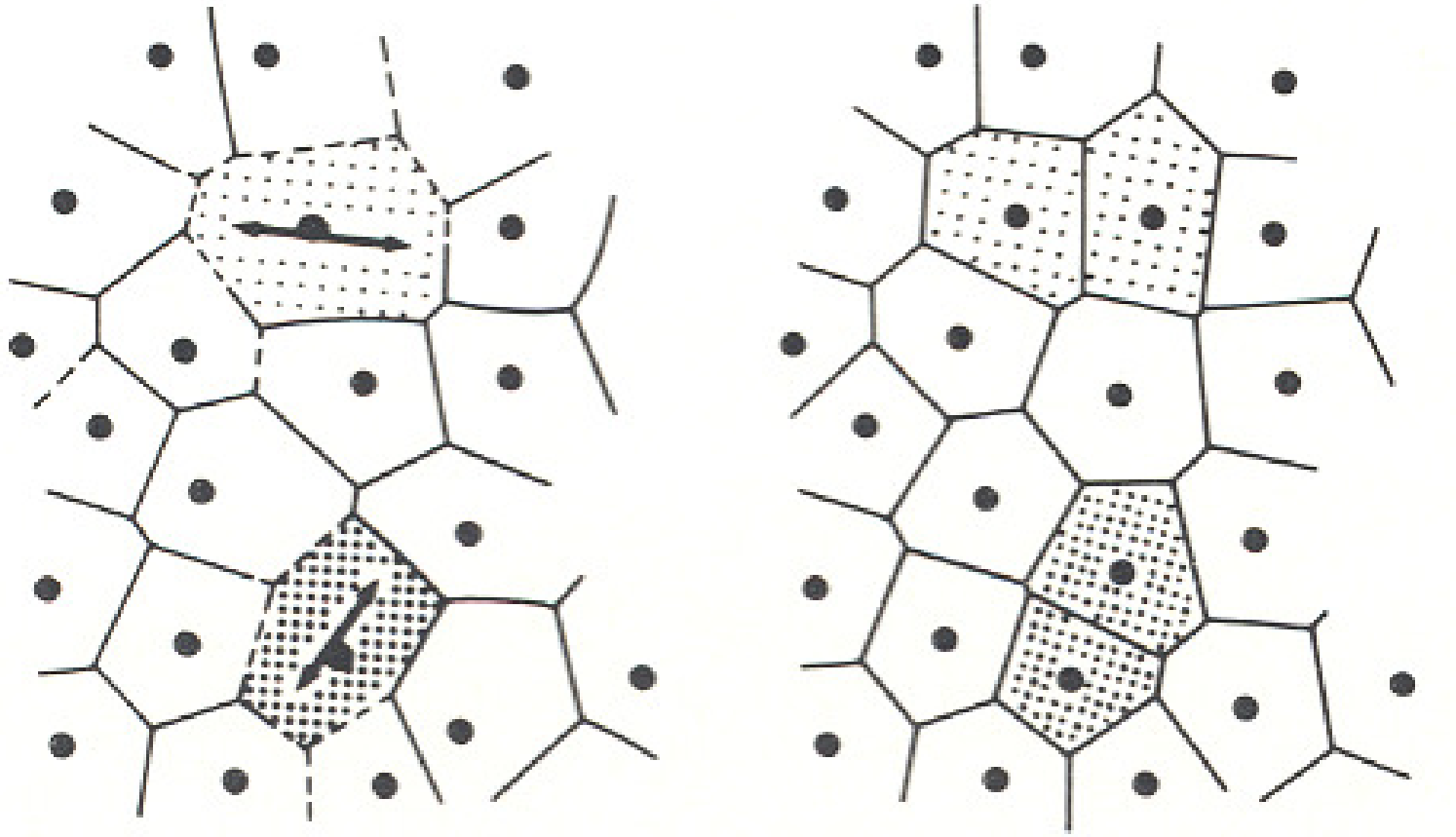
- Cell division
  - start with a configuration of cells that, by observation, form a Voronoi tessellation (this is very commonly the case)
  - after the cells divide, what is the shape of the new cell arrangement?
  - it is observed that the new cell arrangement is closely approximated by a centroidal Voronoi tessellation

- Actual cellular patterns of a starfish embryo before (left) and after (right) cell division

  the white circles on the left are the parent cells that divide
  into the four daughter cells on the right indicated by white dots



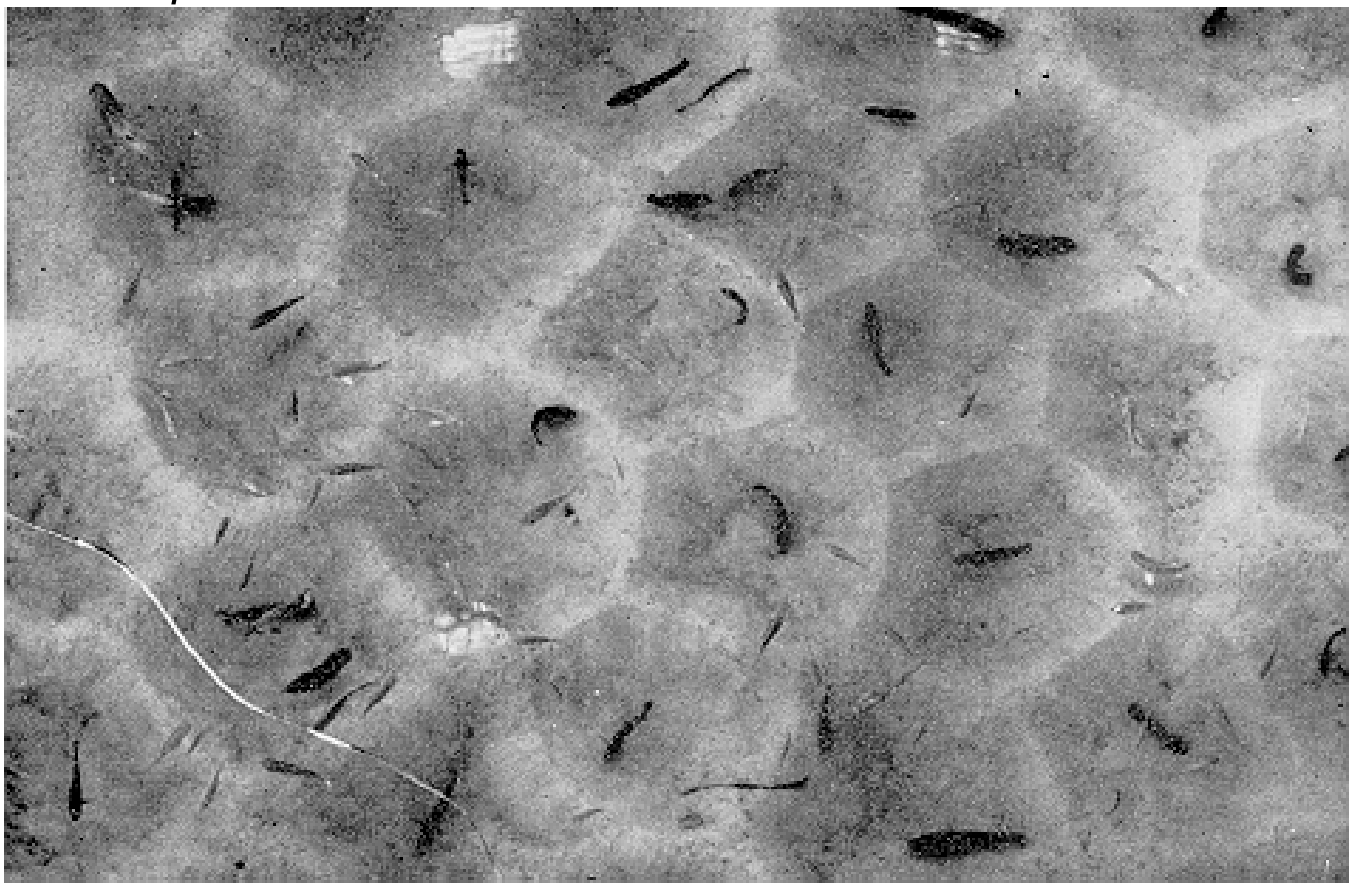
- Left: Actual cellular patterns before cell division traced from photograph.
- Right: CVT-based cellular patterns after two parent cell generators are allowed to separate. The CVT-based cellular pattern can be shown to be close to the actual cellular pattern after cell division



Territorial behavior of animals

- Male mouthbreeder fish – *Tilapia mossambica*
  - fishes dig nesting pits in sandy bottoms
  - they adjust the centers and boundaries of the pits so that the final configuration of territories is a centroidal Voronoi tessellation

– A top view photograph, using a polarizing filter, of the territories of the male *Tilapia mossambica*



– A superposition of the actual boundaries of the territories and a CVT of the nests

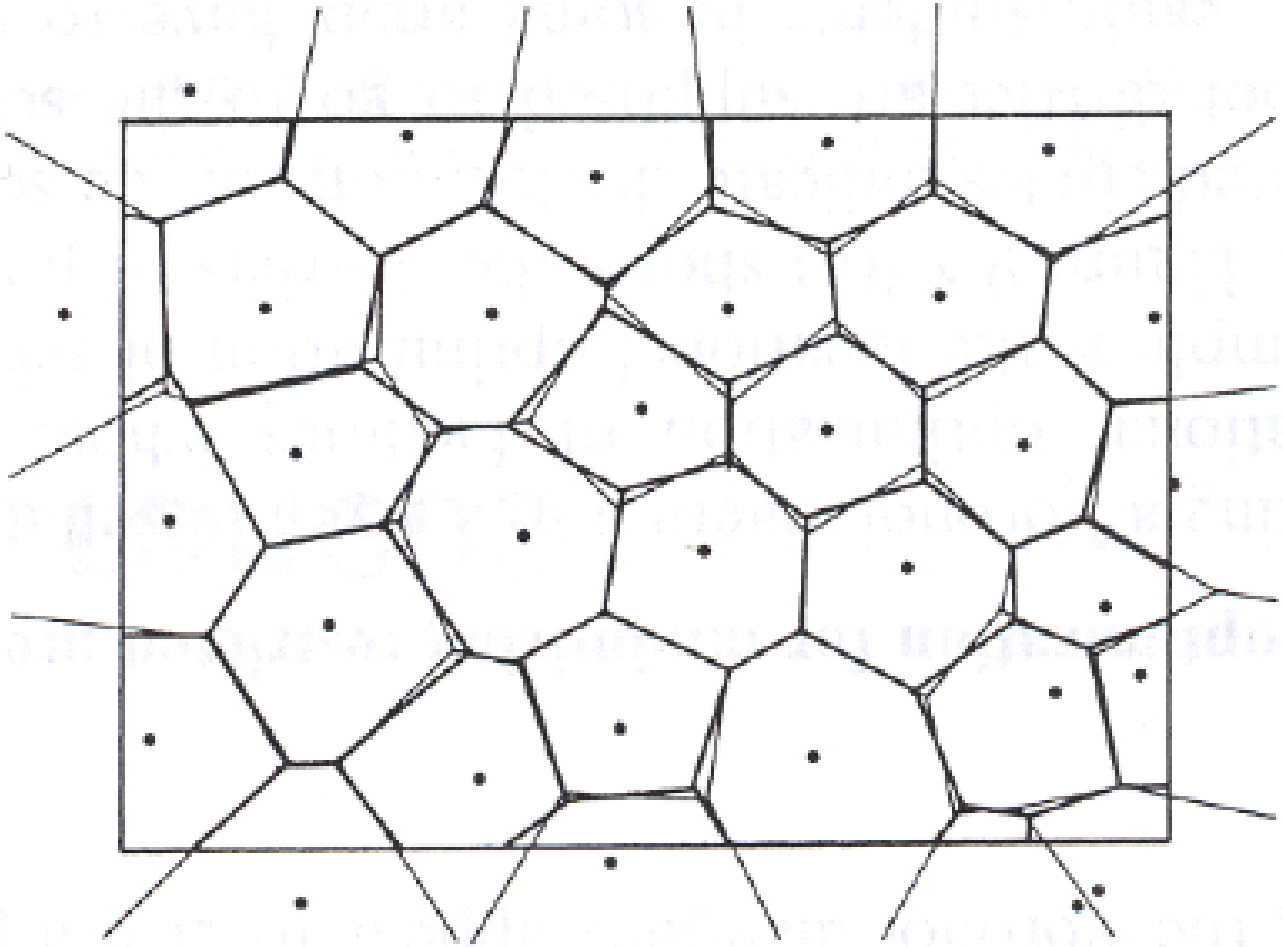


Image processing

- Compression
  - each pixel in a image has a specific color
  - each color is a combination of basic (primary or RGB or CMY) colors
  - there are zillions of different colors in a given image
  - one would like to approximate the image using just a few colors
    - note that the compression is not due to reducing the number of pixels but is due to amount of information about the colors at each pixel
  - question: how does one choose the few colors that are to be used to represent the image?
  - k-means of the image in color space does a very good job

– original image with 256 shades of gray

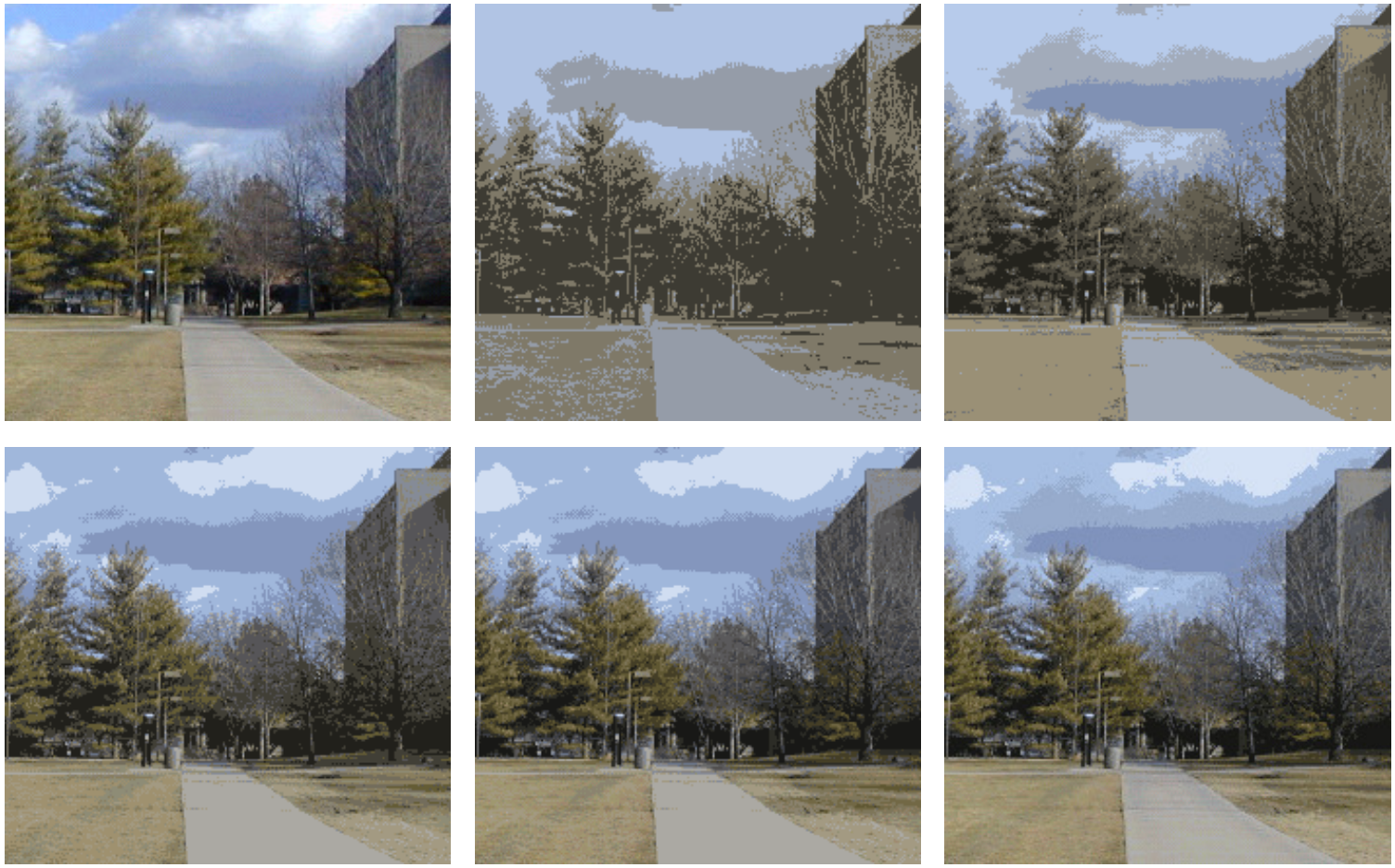
– k-means approximate image using 64 shades of gray

– k-means approximate image using 32 shades of gray

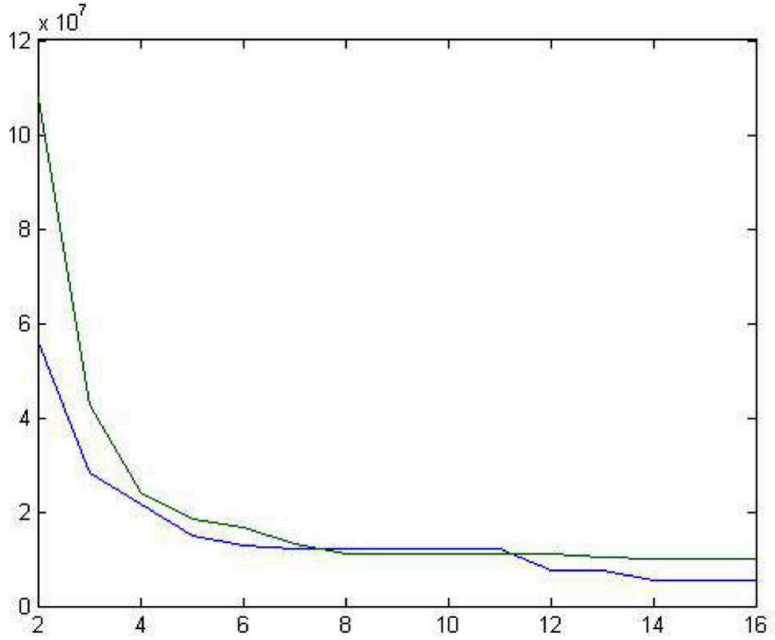
– k-means approximate image using 16 shades of gray

– k-means approximate image using 4 shades of gray

– from left to right and top to bottom: original image containing 1434 different colors and k-means-approximate images containing 4, 8, 16, 32, and 64 colors, respectively



– elbowing effect



The k-means energy of k-means approximate images
for two images vs. the number of replacement colors

- the k-means energy vs. number of generators
(the reduced set of colors) decreases rapidly at first
but then, as
the number of generators increase, reductions
in the energy plateau, i.e., become less pronounced

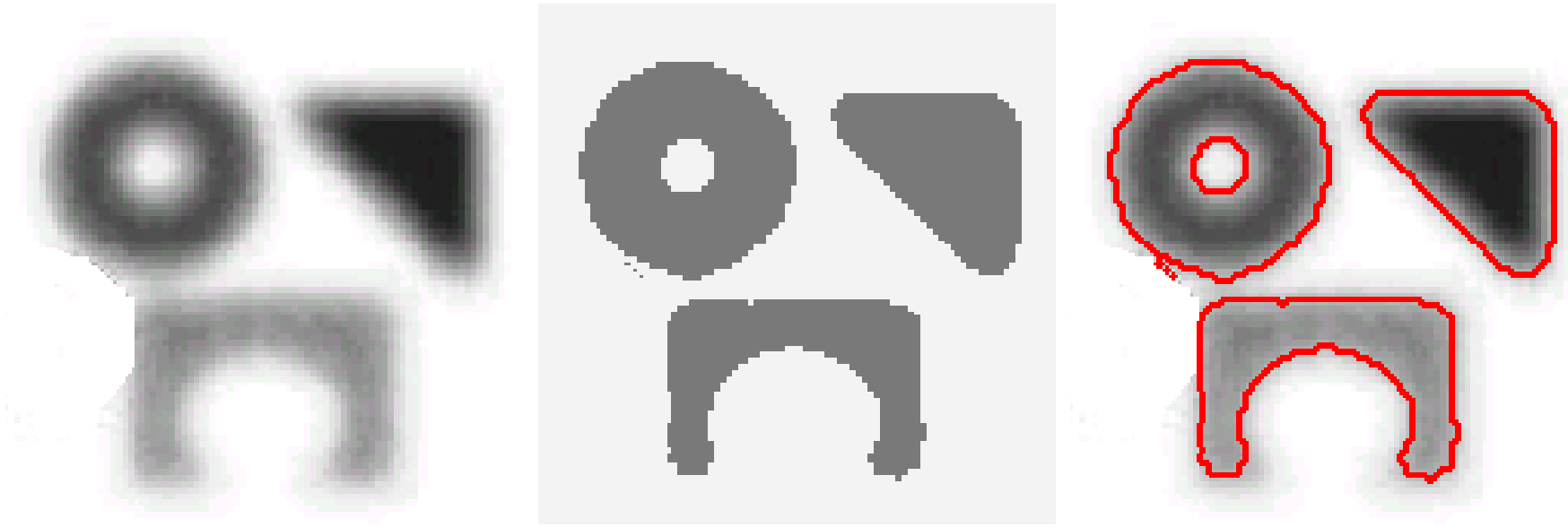
- once the plateau is reached,
  adding more generators does not help much

  so that
  the elbowing effect can be used to determine the number of generators that should be used

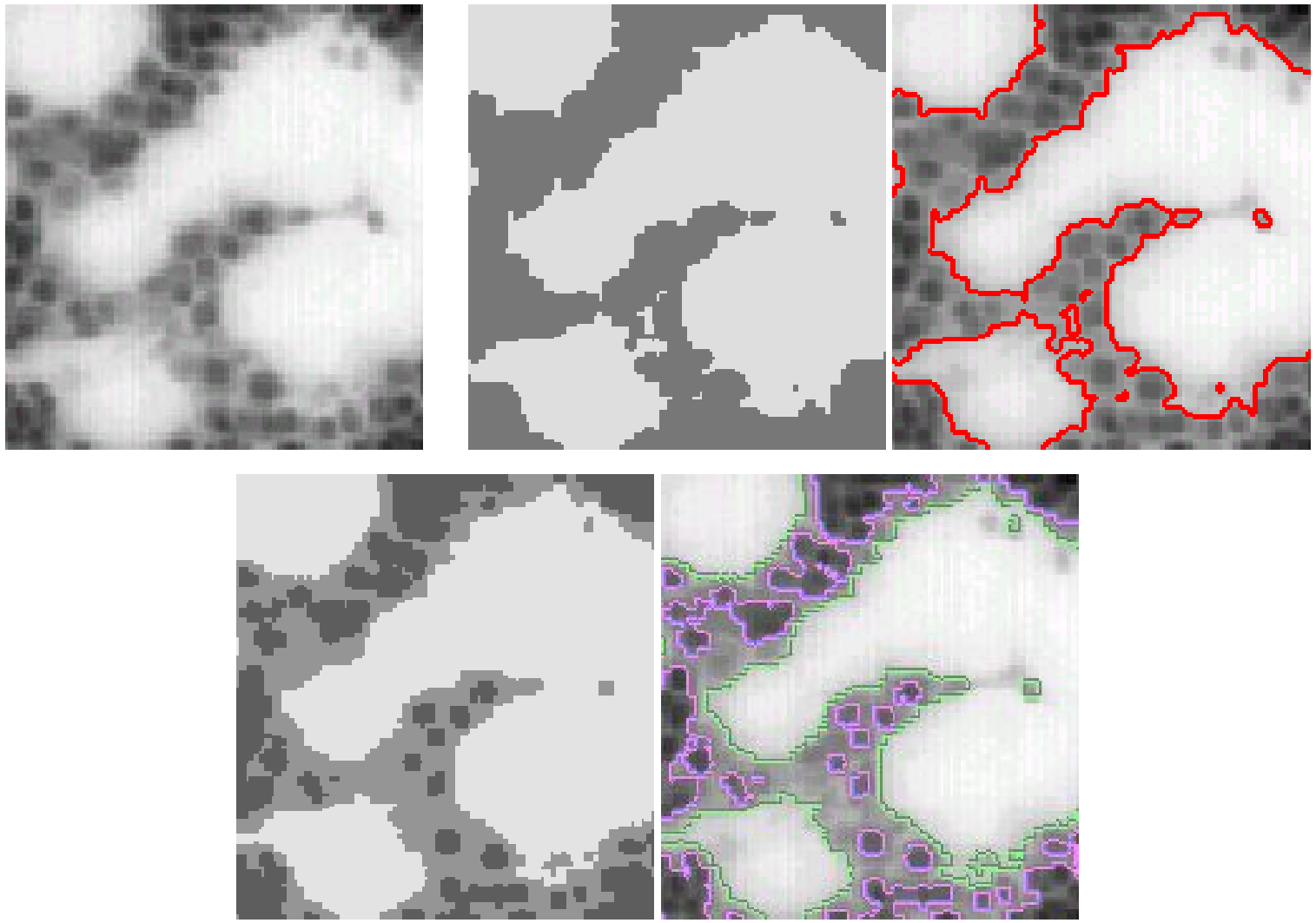
- Segmentation and edge detection
  - image segmentation is the division of an image into two or more segments such that each segment has pixels in the image that share a common attribute, e.g., hue or intensity
  - edge detection is the task of finding the boundaries between different segments in an image
  - the segmentation is not done on the physical image, but in the color space
    - so that points with the same color can be in totally disjoint in physical space but are clustered together in color space
  - k-means in color space does a very good job at segmenting images and detecting edges

– k-means-based segmentation and edge detection into two segments

- left: the original fuzzy image
- middle: k-means segmentation into two segments
- right: the edges of the segments

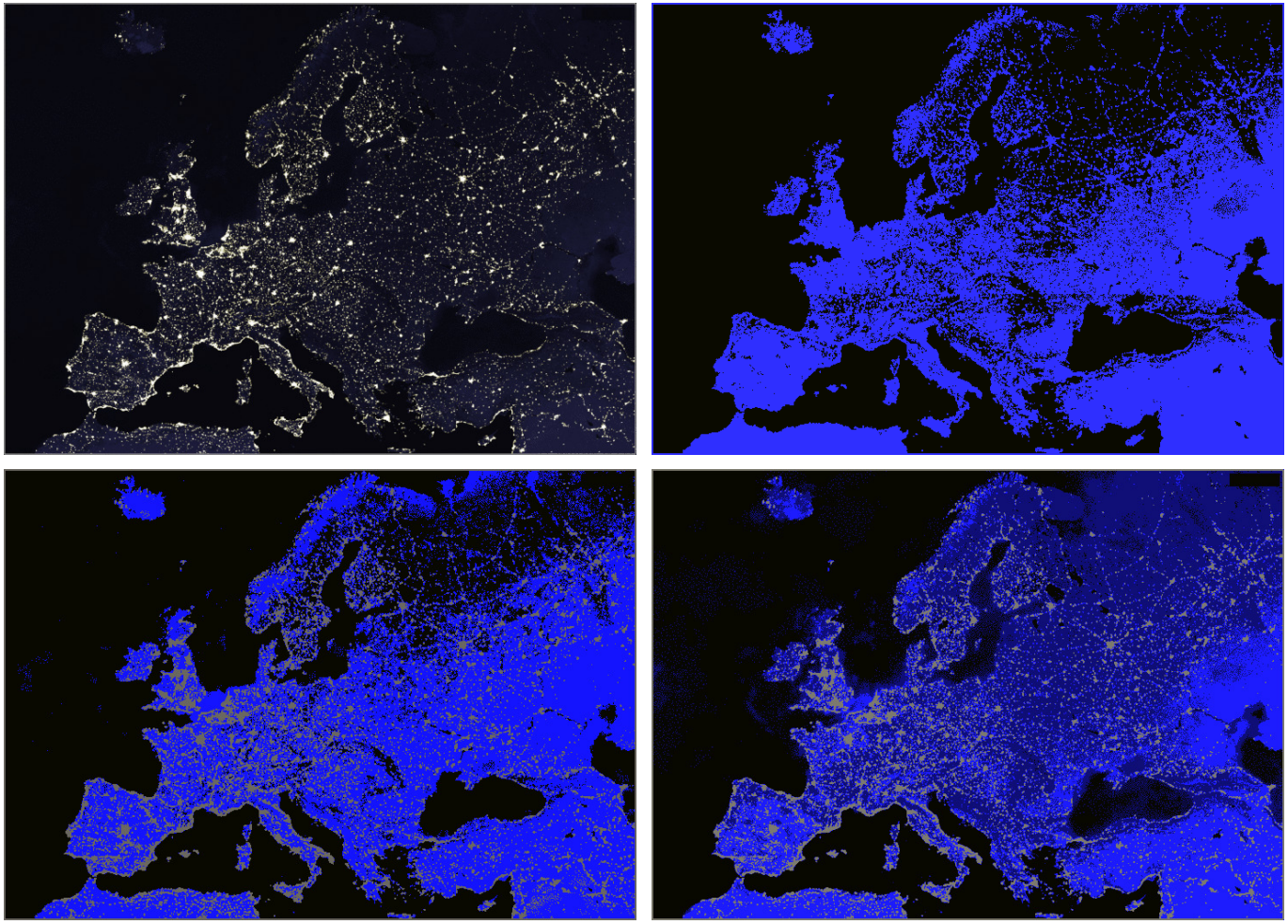


– k-means-based segmentation and edge detection of a bone tissue image (top-left) into two (top-right) and three (bottom) segments



– top left image: original "Europe-by-night" image

– other images: k-means segmentation into 2, 3, and 4 segments



- Restoration
  - suppose one has in hand
    several versions of a image
    none of which contains all the information
    necessary to recover the complete image
  - how can the information contained in the different versions be combined so as to recover the whole image?
  - this is a natural task for k-means-based image processing

– top: three noisy and incomplete versions of the same image

– bottom: k-means-based reconstructed image

k-means reduced-order modeling

- Given

  a set of M snapshots N-vectors $S = \{\boldsymbol{z}_1,\ \boldsymbol{z}_2,\ \cdots\ \boldsymbol{z}_M\}$

  a set of weights $\{w_m\}_{m=1}^{M}$

  an integer $1 < K \leq M$

  use k-means clustering to partition the snapshot
  set $S = \{S_1,\ S_2,\ \cdots\ S_K\}$ into K clusters

  – i.e., determine the partition $S = \{S_1,\ S_2,\ \cdots\ S_K\}$ by

  minimizing $\displaystyle\sum_{k=1}^{K} \sum_{\boldsymbol{z}_m \in S_k} |\boldsymbol{z}_m - \boldsymbol{\mu}_k|^2$ over all possible subsets $\{S_k\}_{k=1}^{K}$

  where $\boldsymbol{\mu}_k = \dfrac{1}{K_k} \displaystyle\sum_{\boldsymbol{z}_m \in S_k} \boldsymbol{z}_m$ = the average of the K_k vectors $\boldsymbol{z}_k \in S_k$

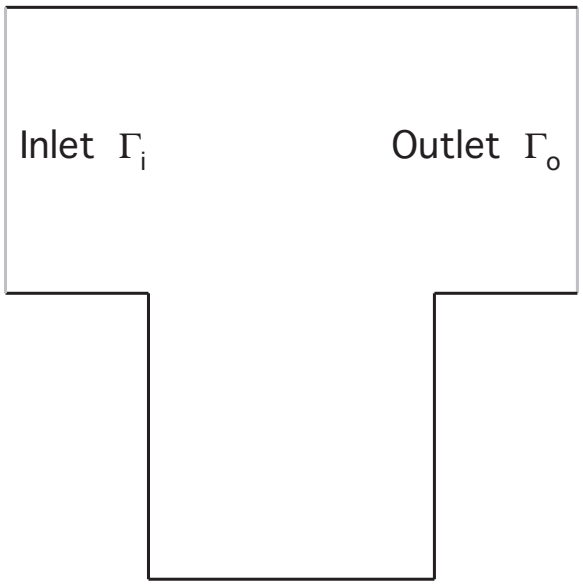
- Then, the k-means reduced basis having cardinality $K \leq M$ is simply

  the set of average vectors $\{\boldsymbol{\mu}_1,\ \boldsymbol{\mu}_2,\ \cdots\ \boldsymbol{\mu}_K\}$

A comparison of POD and k-means reduced-order modeling

Generating snapshots and reduced-order bases

- The problem setting is flow in a T-shaped domain with a fluid inlet and outlet as given in the figure

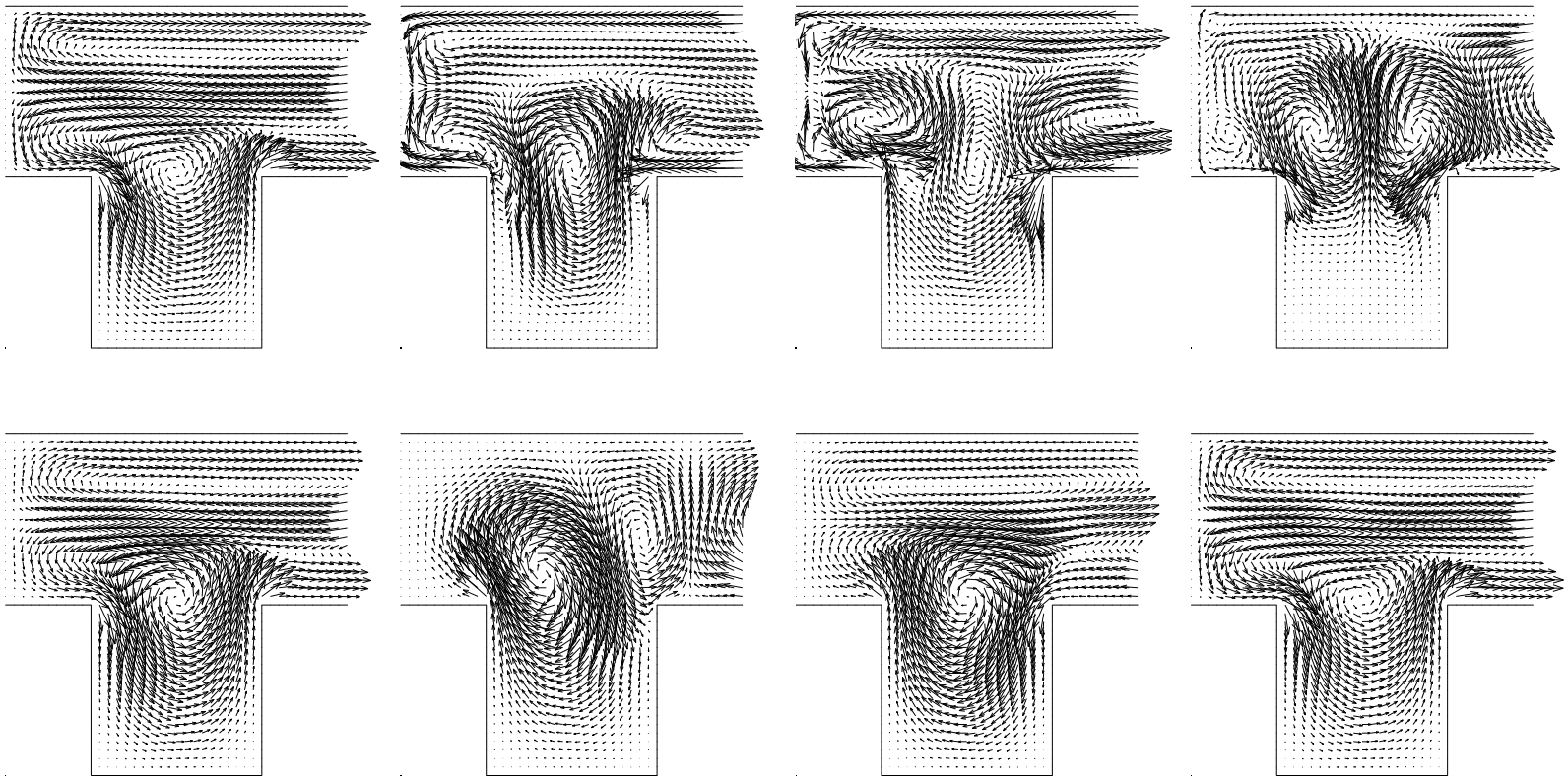


  - the parabolic inflow velocity is given $\mathbf{u}_{inlet} = \begin{pmatrix} 100\,\gamma(t)\,(1-y)\,(0.5-y) \\ 0 \end{pmatrix}$

    - the input parameter function $\gamma(t)$ chosen to generate the snapshot sets is a step function in time

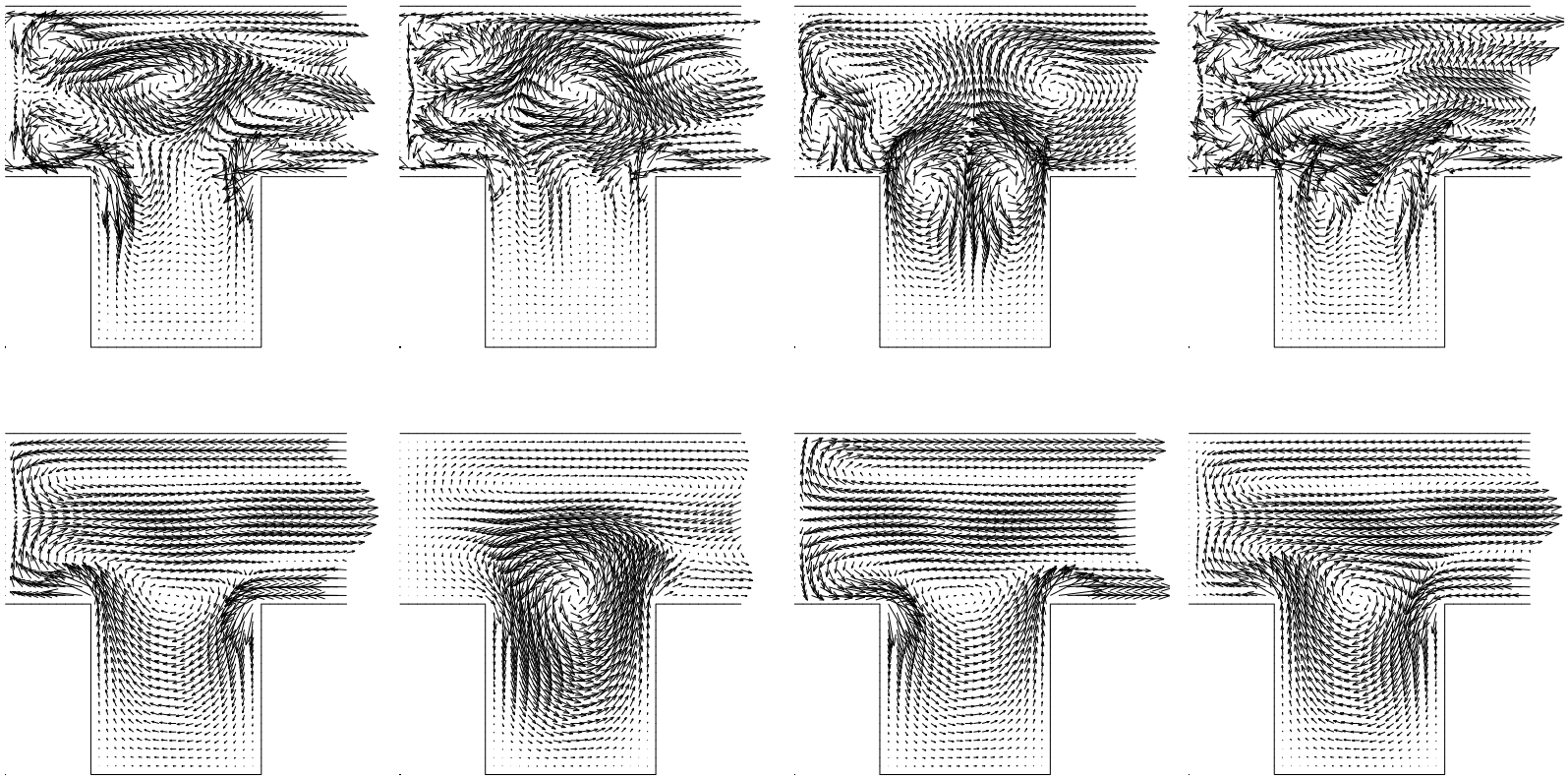
- The Navier-Stokes equations are discretized in space and time
  - full order expensive simulations are sampled in time to generate $M = 500$ snapshots
  - the snapshot vectors have dimension $N > 21000$
  - the snapshots are used to determine POD and k-means bases

The first 4 POD basis



The first 4 k-means basis

The next 4 POD basis



The next 4 k-means basis

- Interestingly, the POD and k-means bases look very different

- The singular values of the snapshot matrix decrease rapidly

1	2.7234	5	0.0866	9	0.0077	13	0.0014
2	0.6704	6	0.0349	10	0.0060	14	0.0010
3	0.2612	7	0.0168	11	0.0029	15	0.0008
4	0.1197	8	0.0151	12	0.0023	16	0.0004

The first 16 singular values of the snapshot matrix

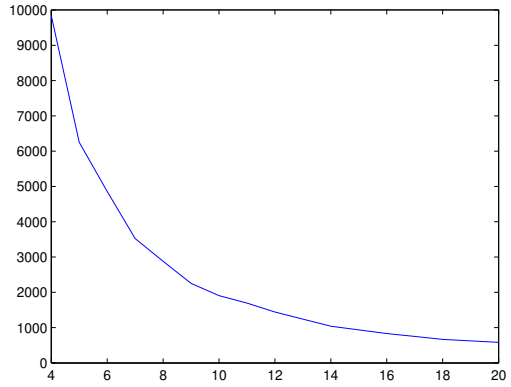
  - this indicates that for the problem we consider here

    - a "small-dimensional" POD basis can capture most of the information contained in the snapshot set

  - this benevelent behavior cannot, of course, be universal but it has been observed in many other examples

- The "k-means energy" (which is the functional $\mathcal{F}_{k-means}$) decays rapidly as the size of the k-means basis increases
  - the elbowing effect is evident

4	9837	8	2879	12	1442	16	833
5	6250	9	2254	13	1241	17	751
6	4857	10	1906	14	1041	18	669
7	3524	11	1697	15	936	19	626
						20	584

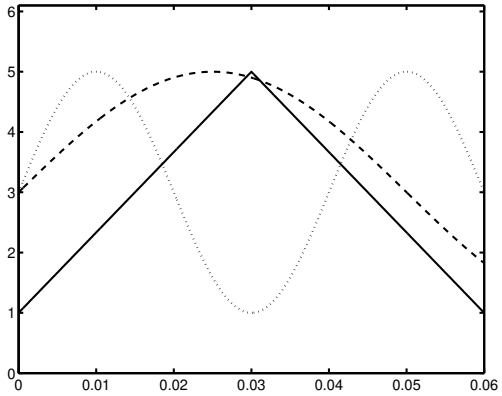


The k-means energy vs. the dimension K of the k-means reduced basis

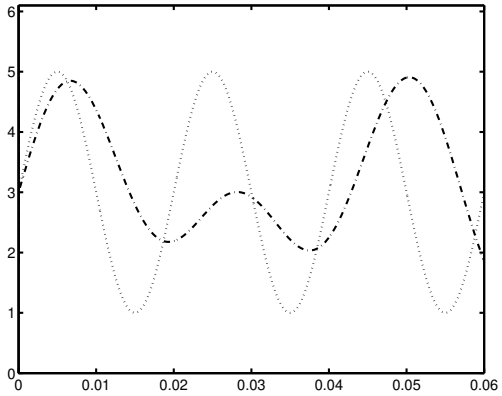
  - for very few basis functions, increasing the number of basis functions effects a large decrease in the energy
  - but as the number of basis functions increase, the reduction in the energy becomes much smaller
  - this is an indication that the snapshot set for our problem clusters well
    - again, this phenomenon has been often observed but is not universal

Performance of the reduced-basis methods

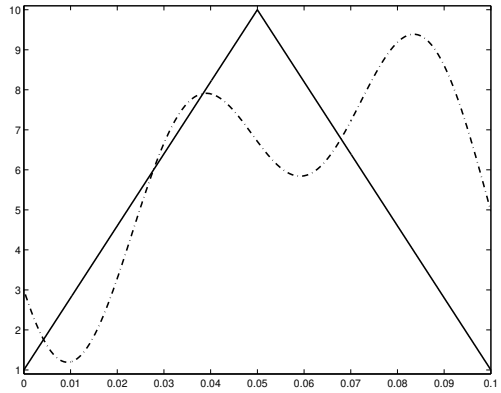
- Seven choices for the inlet input function $\gamma(t)$ are used to test the POD and k-means reduced-order model
  - all seven are different from the $\gamma(t)$ used to generate the snapshots



1. hat function
2. low-frequency sinusoid
3. middle-frequency sinusoid



4. high-frequency sinusoid
5. "general" function



6. hat function
7. "general" function

– we test the performance of the two reduced-order models in two settings

- interpolatory setting

  choices 1 through 5 feature

  inputs $\gamma(t)$ than are different from

  those used to generate snapshots

  but

  ROM simulations are over the same time

  interval as that used to generate snapshots

- extrapolatory setting

  choices 6 and 7 feature

  inputs $\gamma(t)$ that are also different from

  those used to generate snapshots

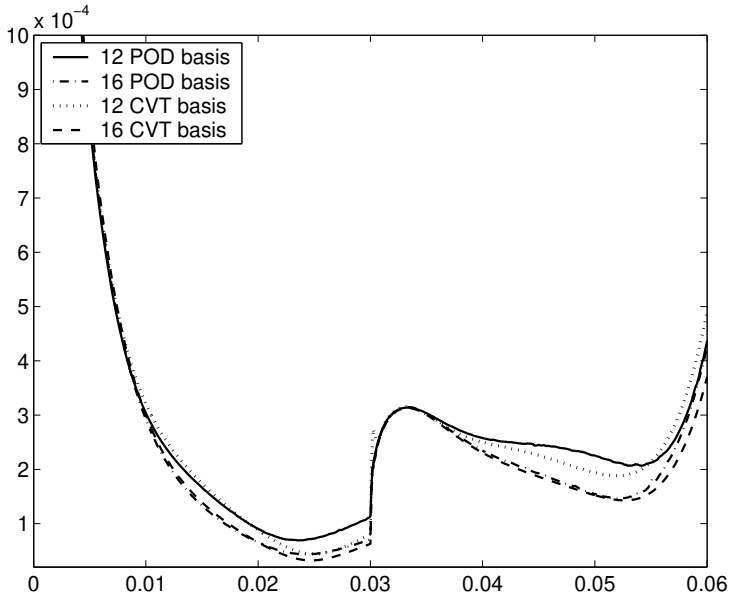
  but now

  ROM simulations are over a time interval that is

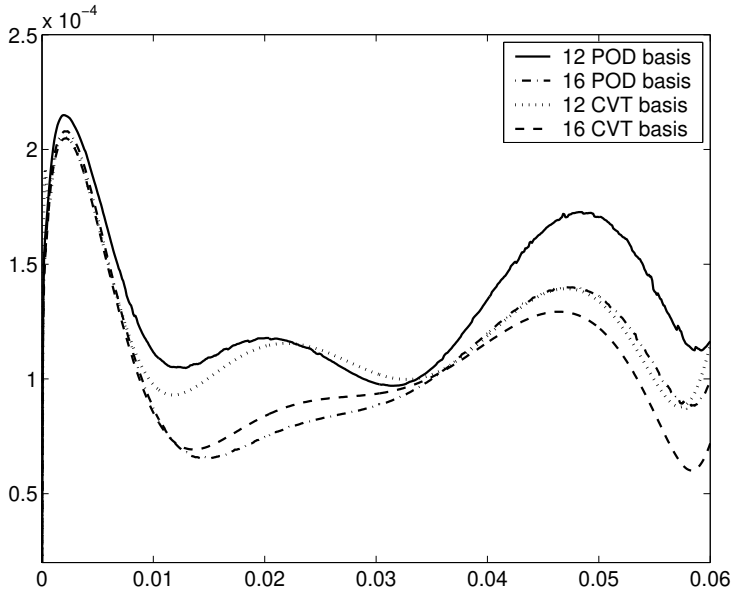
  twice as long as that used to generate snapshots

– we plot the error vs. time of the two ROM solutions
when compared to solutions of the expensive model
- here, the error is the difference between
ROM and full-order simulations

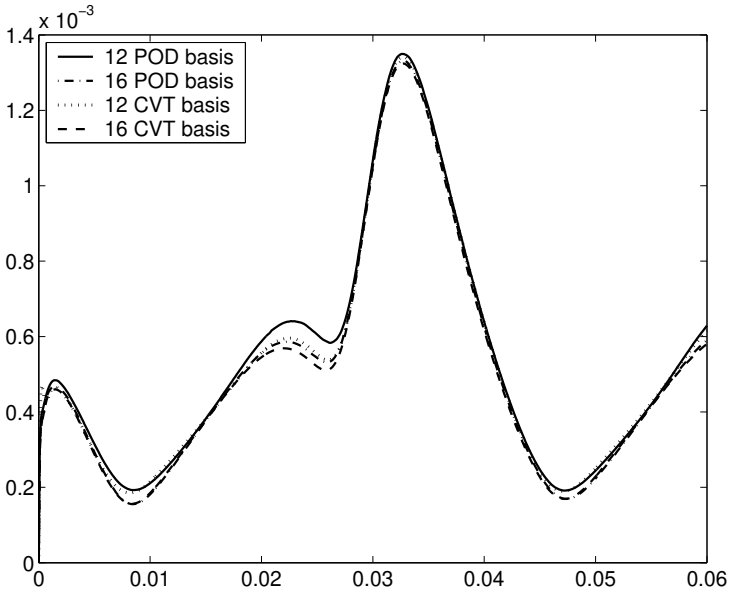
– interpolatory setting



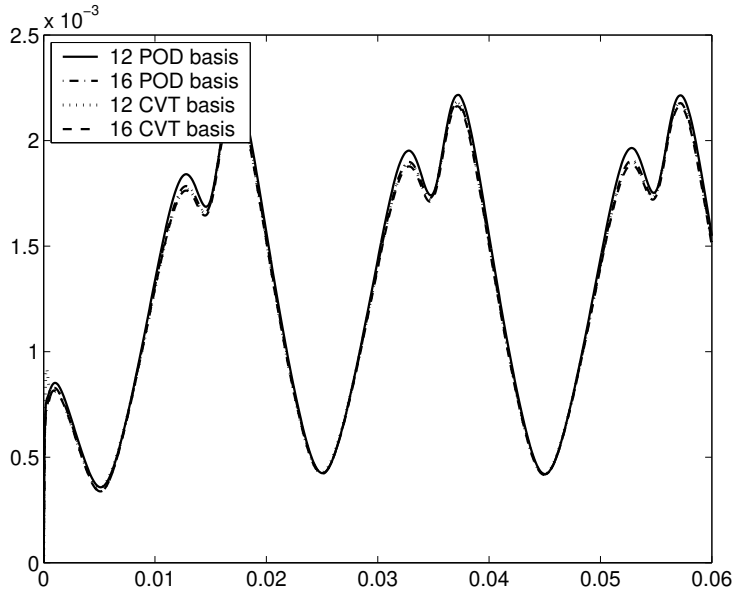
Test 1



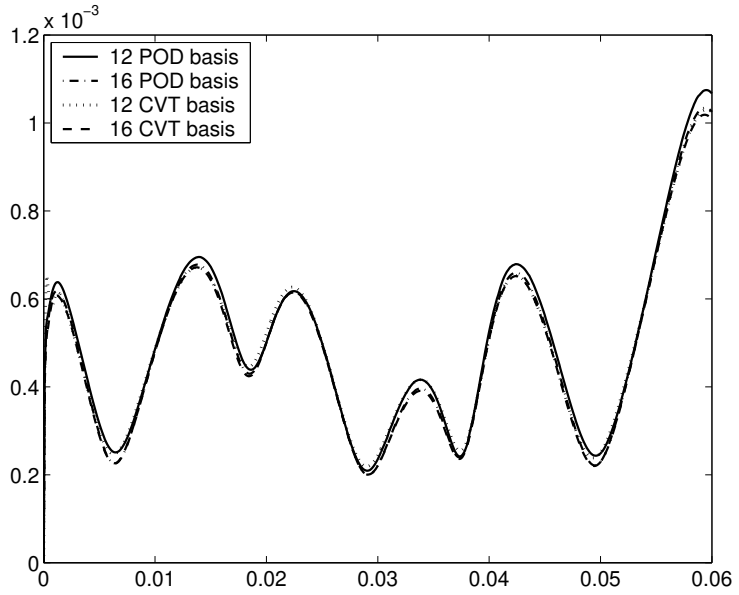
Test 2


x 10^−3
12 POD basis
16 POD basis
12 CVT basis
16 CVT basis
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.01
0.02
0.03
0.04
0.05
0.06

Test 3

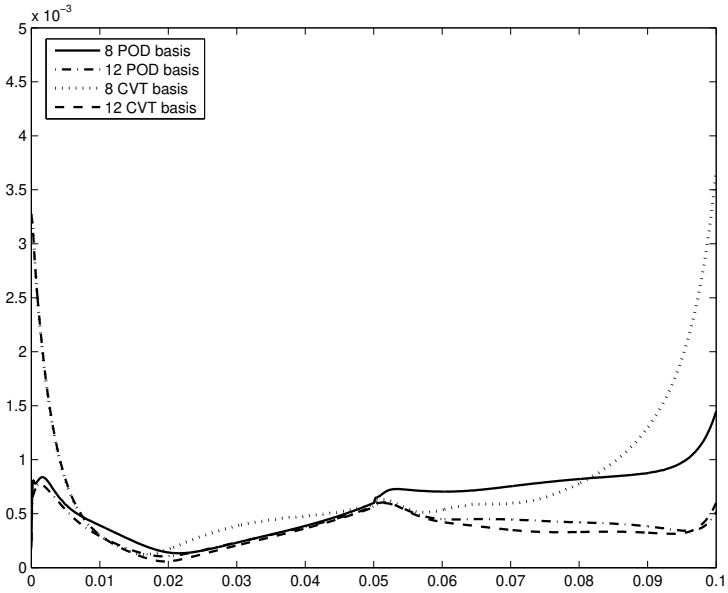

x 10^−3
12 POD basis
16 POD basis
12 CVT basis
16 CVT basis
2.5
2
1.5
1
0.5
0
0
0.01
0.02
0.03
0.04
0.05
0.06

Test 4

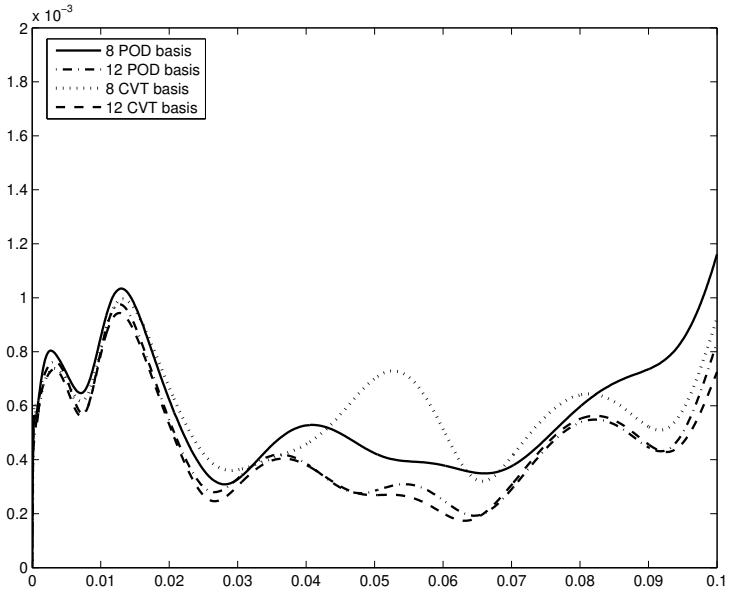

x 10^−3
12 POD basis
16 POD basis
12 CVT basis
16 CVT basis
1.2
1
0.8
0.6
0.4
0.2
0
0
0.01
0.02
0.03
0.04
0.05
0.06

Test 5

– extrapolatory setting



Test 6



Test 7

K	POD	k-means
4	6.125e-02	5.264e-02
5	3.255e-02	1.012e-01
6	2.192e-02	2.976e-02
7	2.097e-02	4.038e-02
8	1.914e-02	2.147e-02
10	1.830e-02	1.857e-02
12	1.787e-02	1.761e-02
16	1.736e-02	1.721e-02

For case 5, the error incurred by the POD and k-means reduced-order models vs. the cardinality K of the reduced bases space

- One observes that, at least for the T-cell example we consider

  very low-dimensional POD and k-means ROMs are both quite effective at approximating expensive full-order solutions

  - even for bases of cardinality less than 10, the "error" is small

  - one also observes that if the cardinality K of the basis is about 12, further increases in that cardinality effects very little improvement in the performance of the two ROMs

  - this is an indication that the two ROMs of cardinality about 12 already effectively capture all of the information contained in the snapshot set that, in our example, has cardinality 500

  - this conclusion can also be inferred from the table in which the space-time error vs. K is listed

    - we see that there is almost no reduction in the error if K is increased from 12 to 16

- We also observe that comparing the first 5 cases with the last two that
  - the errors for the extrapolatory cases are not much worse that those for the interpolatory cases
  - note, however, that for the extrapolatory cases
    - there is some deterioration evident in the performance of the ROMs near the end of the extrapolated simulation for the longer time interval
  - in more general situations
    - it is possible for extrapolatory ROMs to produce substantially worse results compared to interpolatory ROMs
    - the cause of this is that the snapshot set itself does not contain sufficient information to accurately represent the solution in the extrapolatory regime

      i.e., some snapshots should be sampled at later times

POD vs. k-means

- For the test problem we consider and as far as accuracy is concerned
  there is very little to choose between POD
  and k-means based ROMs
  to approximate full expensive solutions

  – the only cost difference between the two approaches are the costs incurred to determine a reduced basis once a snapshot set has been determined

  – any cost difference between the two approaches is miniscule compared to the cost of generating the snapshot set

    - for the example problem we present here
      the cost of determining a reduced-order basis
      from the snapshot set is less than one-half of
      one percent of the cost of determining the
      snapshot set itself

- POD does have one advantage over k-means, namely
  that the POD bases are nested
  i.e., a POD basis contains all the
  POD bases of lower cardinality
  whereas k-means bases are not nested

  – if we want to enlarge the POD basis
    one just adds column of the matrix $\mathbb{U}$
    appearing in the SVD of the snapshot set

  – if we want to enlarge the k-means basis
    one has to start the construction of the
    k-means basis from the very beginning

  – again, this advantage is not huge because
    the costs of determining an new enlarged
    k-means basis (should one be needed) from
    the snapshot set is very small compared to
    the cost of determining the snapshot set itself

POD + k-means

- Reduced-order models
  that involve both POD + k-means
  can be easily defined and
  can result in a better reduced-order model
  than either one of its components

  – for example

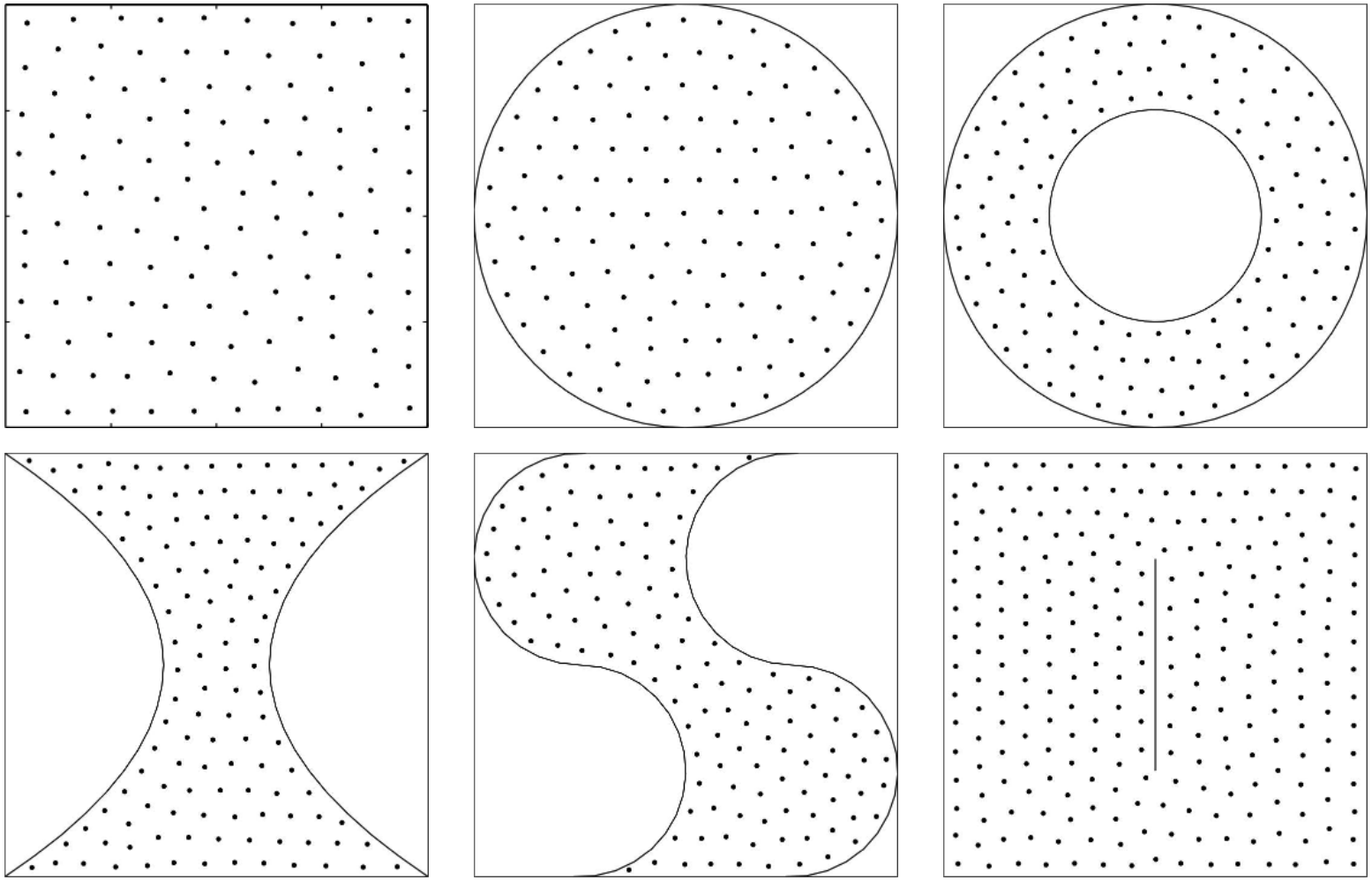
    - one can first do a coarse k-means clustering of the snapshots
      and then
      do a separate POD within each k-means cluster

  – POD + k-means ROMs could be advantageous in situations where the solution behavior in one cluster is very different from that others

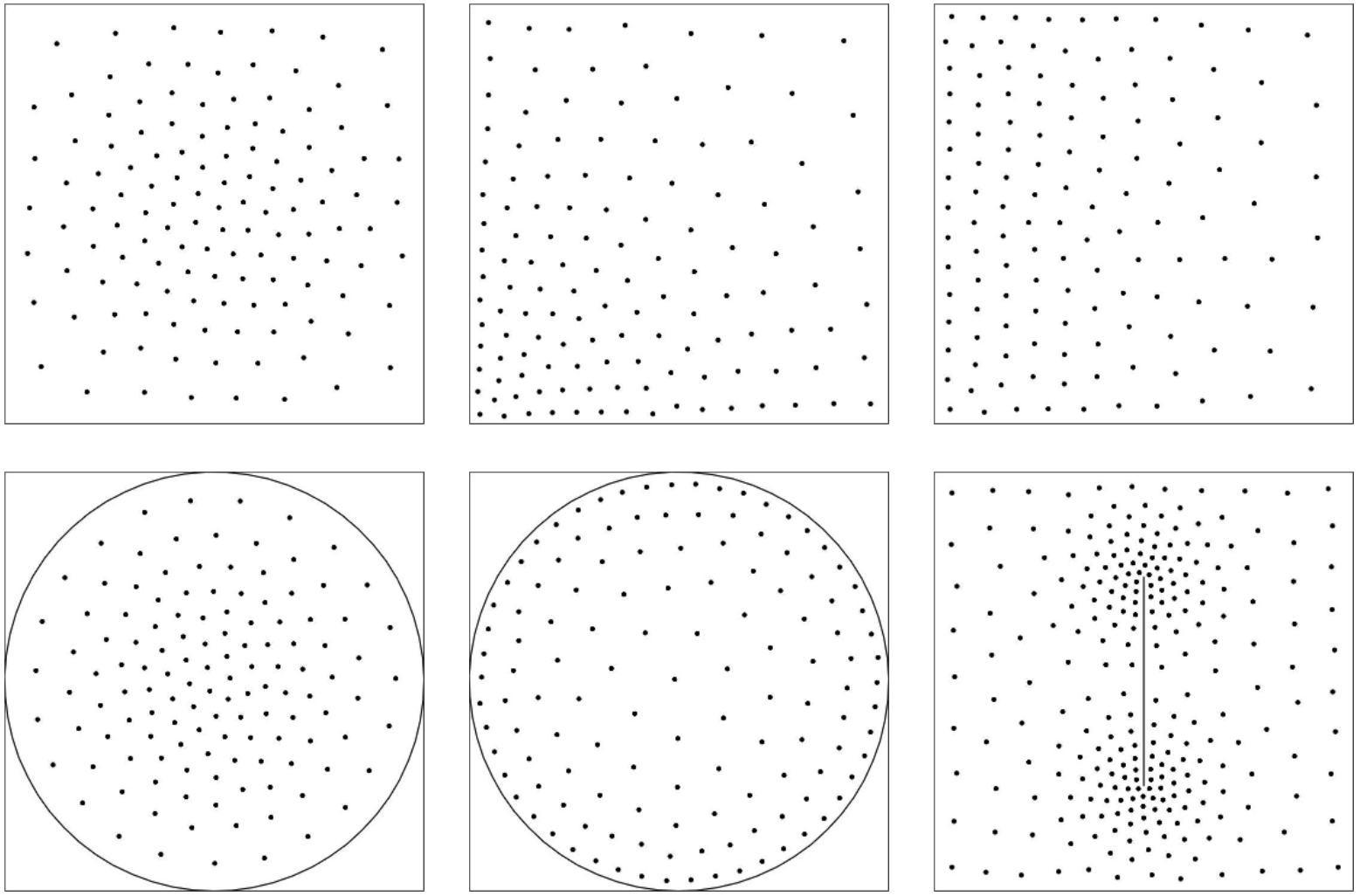
    - then, doing a POD ROM specialized to each cluster may be advantageous

Point sampling in general domains

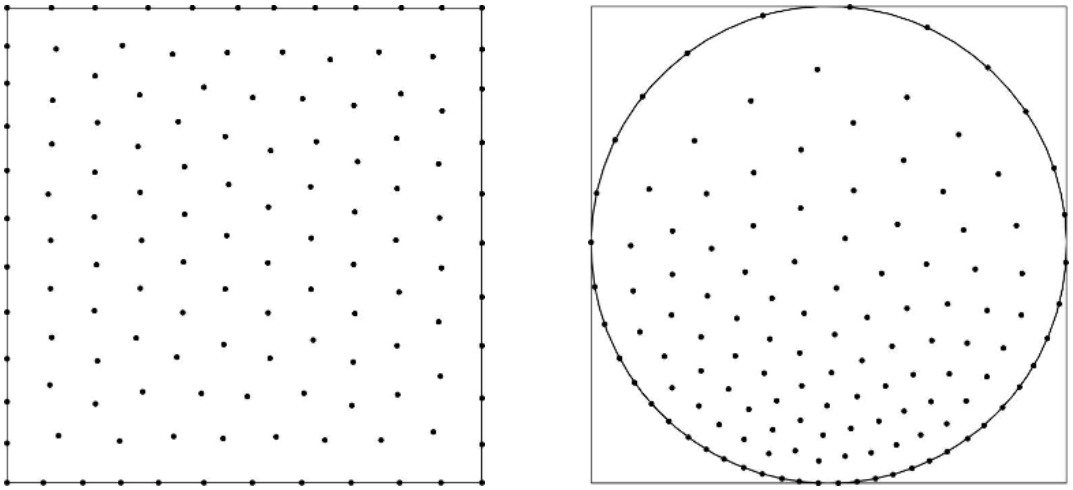
Examples of the versatility of CVT point sampling



Uniform CVT point distributions in "general" domains



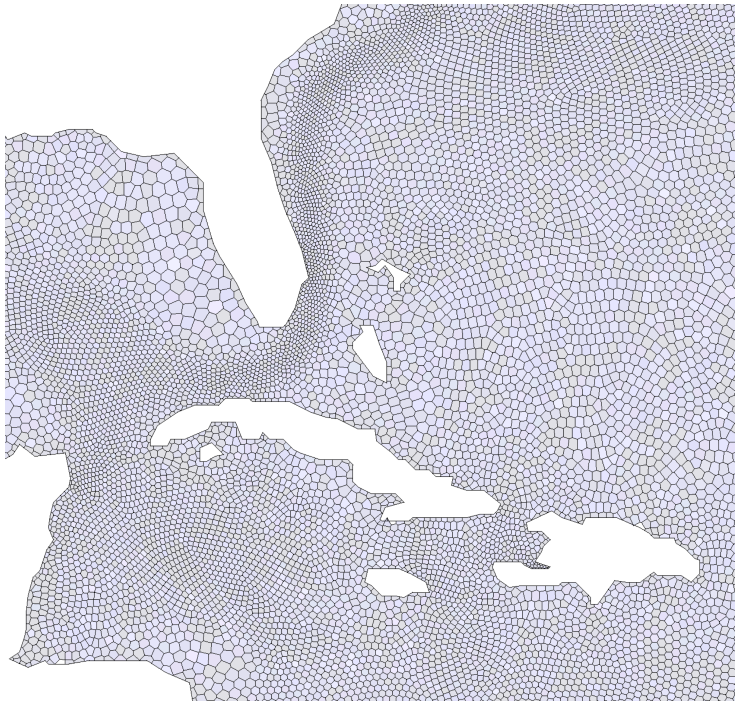
Non-uniform CVT point distributions



CVTs with points on the boundary

CVT-based grid generation

- Voronoi tessellations can be used as grids for many purposes such as histograms and notably for the discretization of PDEs
  - the figure is an example a CVT grid in actual use in practice for finite-volume ocean simulations



  - however, CVT grids are most often used in their dual form

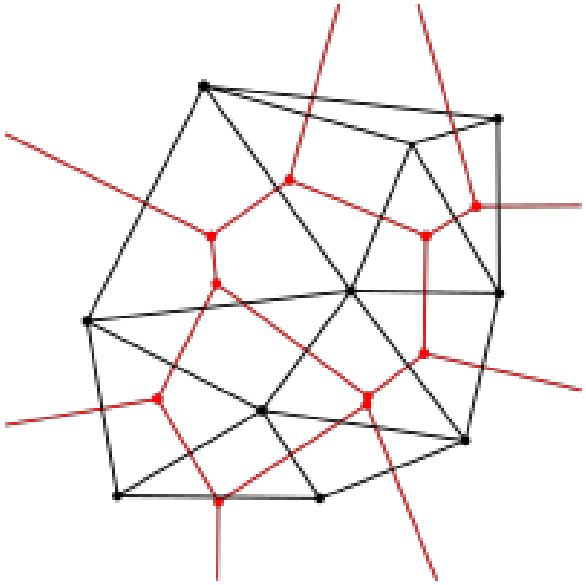
Delaunay triangulations

- There are several ways to define Delaunay grids including several that do not rely on Voronoi tessellations

  - here, however, we use the Voronoi tessellation approach

- Given a Voronoi tessellation, the associated Delaunay grid is defined by connecting Voronoi generators that share a common face

  – in two dimensions, they share a common edge



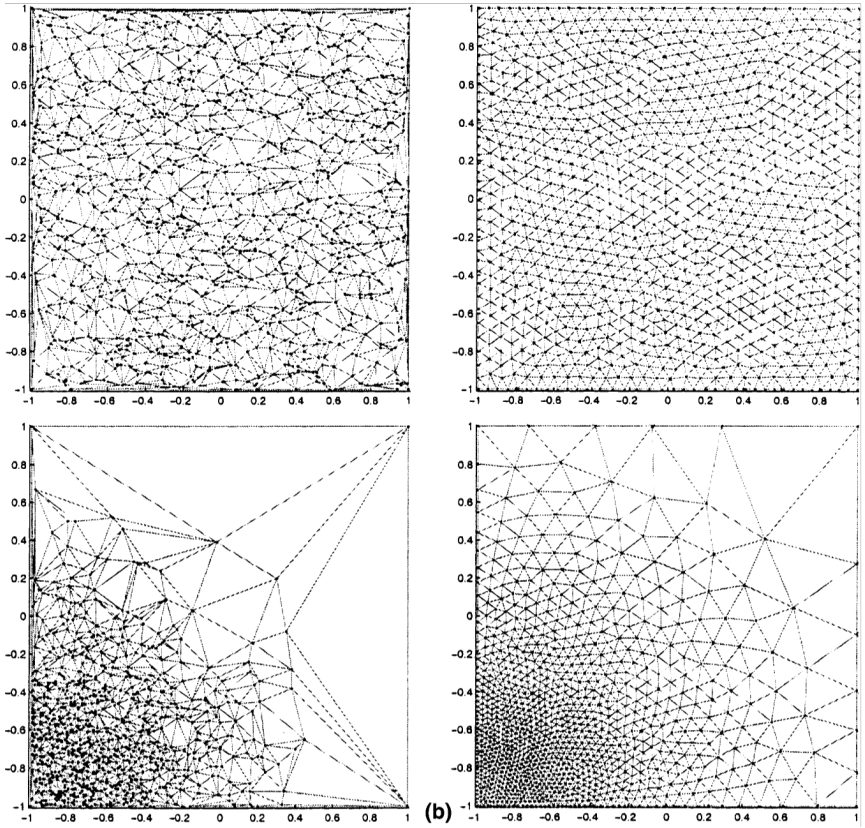
the red grid denotes a Voronoi tessellation corresponding to the red dots

the black grid denotes the corresponding Delaunay triangulation

- Delaunay grids are made up of simplices

  - triangles in two dimensions, tetrahedra in three dimensions, ...

- It is not surprising that the high quality of centroidal Voronoi tessellations induce high quality Delaunay grids



Left: Delaunay grids corresponding to Voronoi tessellations associated with uniformly and non-uniformly distributed random points

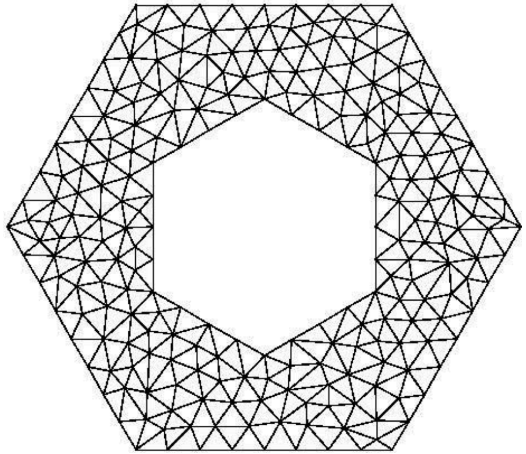
Right: Delaunay grids corresponding to uniform and non-uniform CVTs

we refer to Delaunay grids corresponding to a CVT as CVDT grids

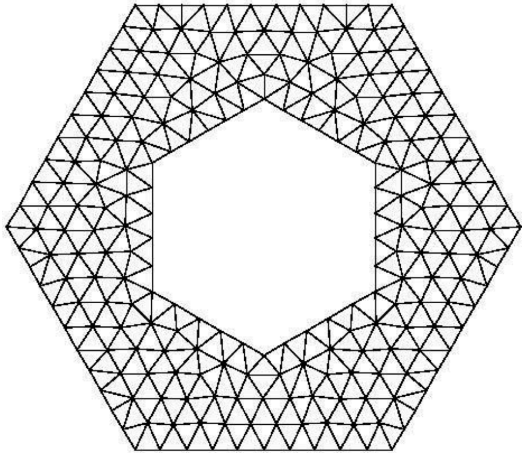
– there are many software packages available for the construction of Delaunay grids

Comparison of CVDT grids with other grid-generation software

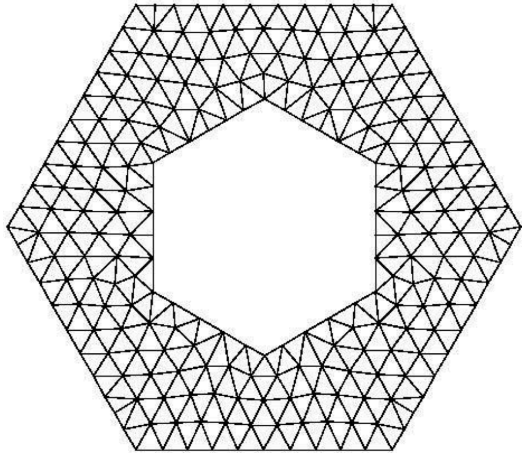
eyeball norm comparisons



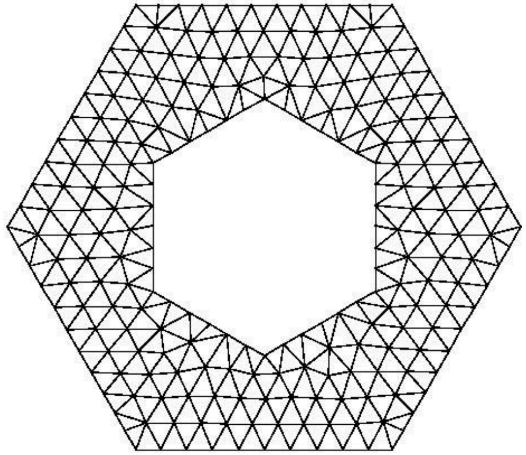
TRIANGLE



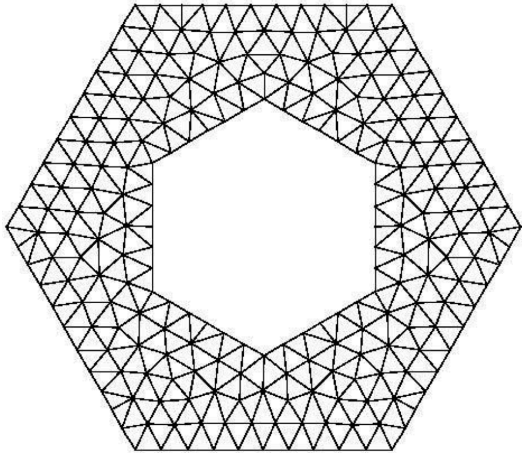
DISTMESH



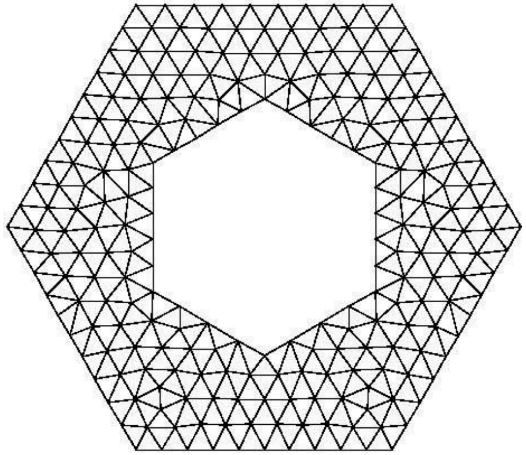
MESHGEN



VTM

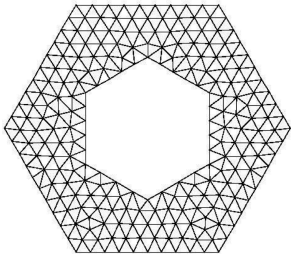


CVDT1



CVDT2

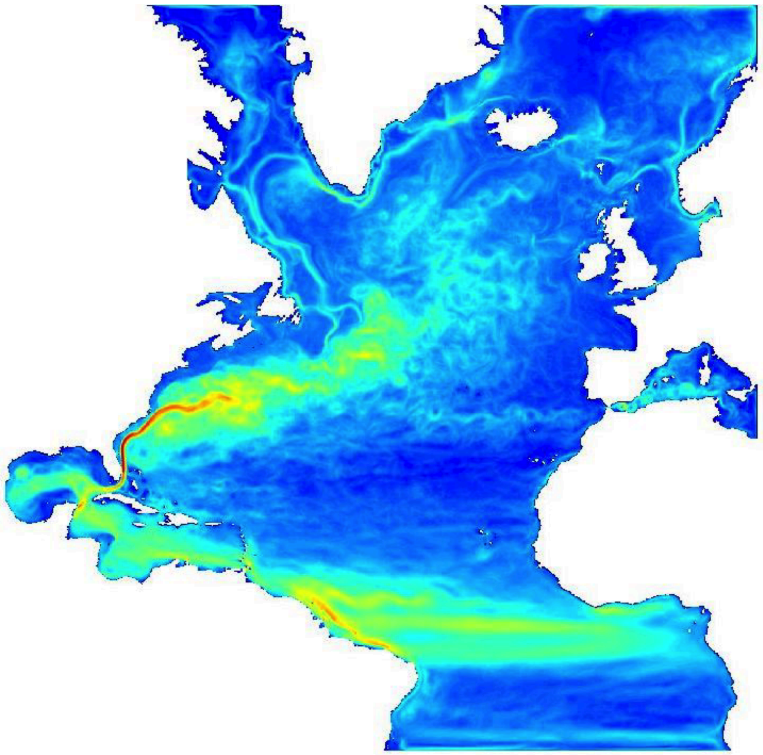
quantitative comparisons of different grids using several quality measures



	TRIANGLE	DISTMESH	MESHGEN	TVM	CVDT1	CVDT2
no. points	242	240	240	240	240	240
no. triangles	412	385	385	381	381	393
$h \times 10^2$	8.61	7.32	7.30	7.78	7.16	7.37
χ	2.25	1.75	1.85	2.02	1.80	1.72
$\tau \times 10^4$	8.83	9.25	9.19	8.98	8.94	8.47
$d \times 10^7$	3.36	1.53	1.09	0.94	0.72	1.72
$\alpha \times 10^3$	6.94	6.48	6.15	7.49	6.55	6.00
$\beta \times 10^2$	3.32	2.82	3.09	3.33	2.90	2.56
q	2.00	1.37	1.30	1.69	1.39	1.37
$p \times 10^2$	11.60	3.60	4.60	6.00	4.30	4.20

red best blue worst

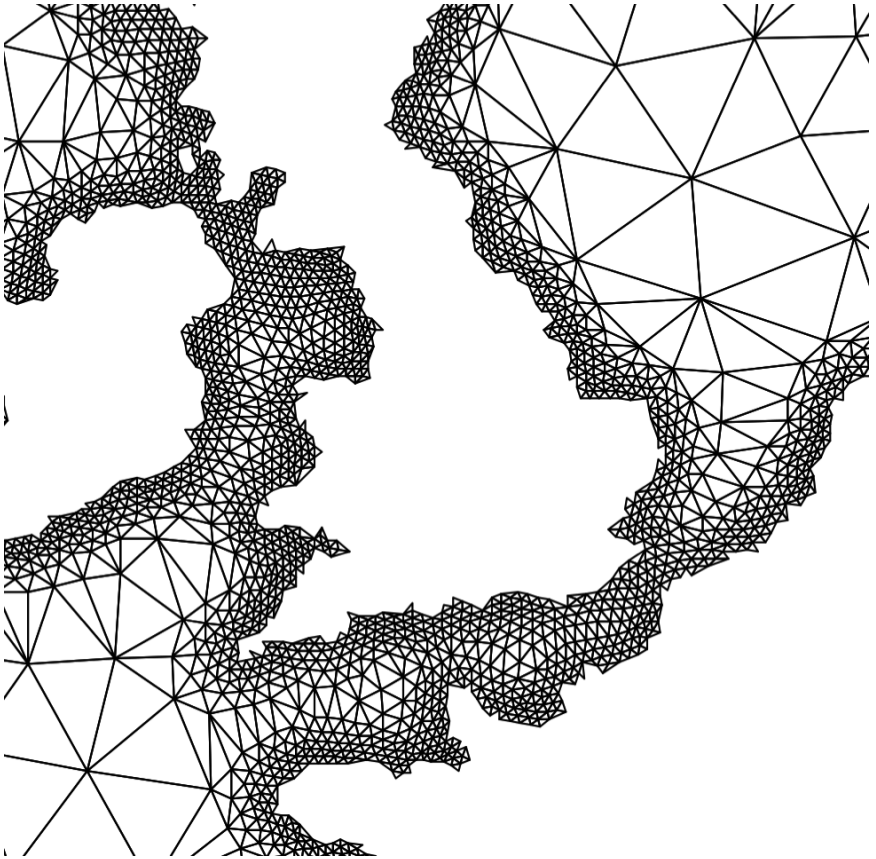
Example of CVDT grids



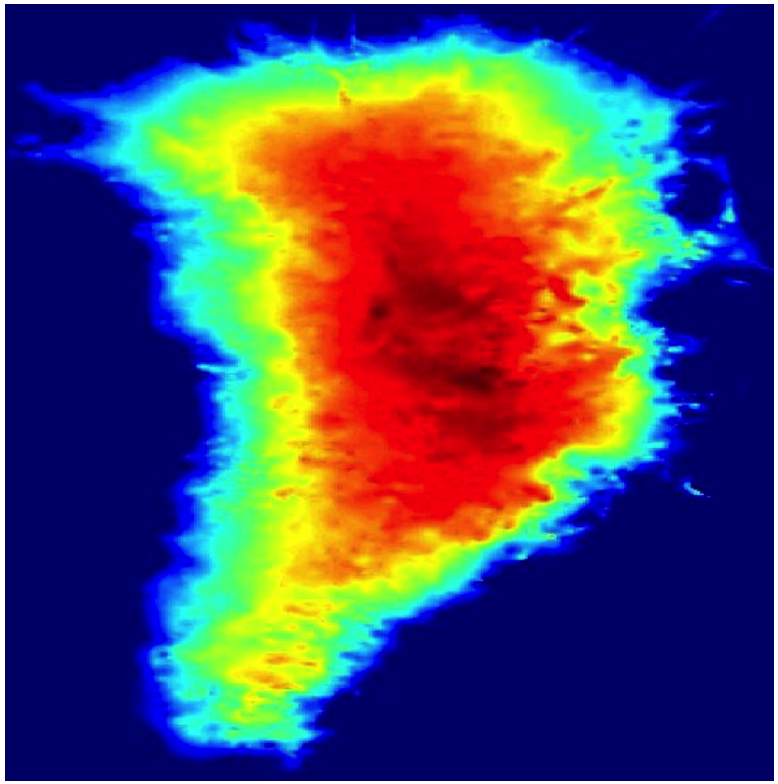
Satellite data for the kinetic energy distribution in the North Atlantic
Note that the boundary is quite complicated

- The grid is refined based on two goals
  - where the kinetic energy is relatively large (red and yellow areas)
  - to accurately resolve the boundary

Left: a CVDT of the North Atlantic



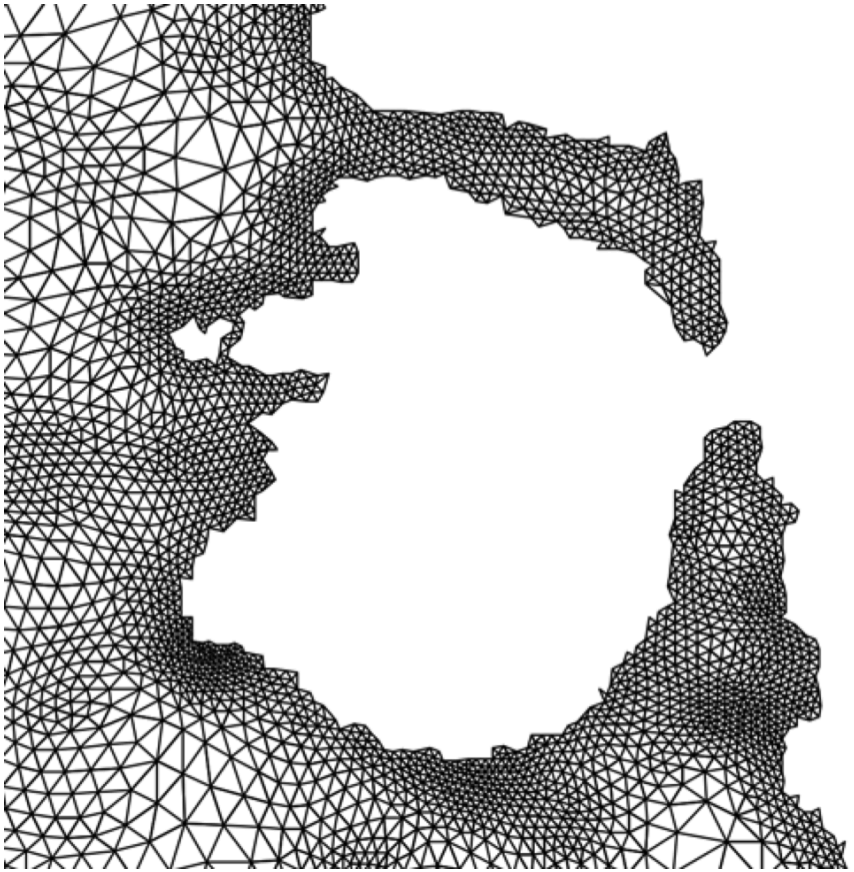
Right: a zooomed in portion of that CVDT



Measured ice thickness of Greenland; note that the boundary is quite complicated

- The grid is refined based on two goals
  - where the ice thickness is relatively small (light blue areas)
  - to resolve the boundary

A CVDT of the Greenland ice sheet



A zoomed in portion of that CVDT

RANDOM FIELDS

- A random field is a function $\eta(\boldsymbol{x}, t; \omega)$ defined for

  $\boldsymbol{x}$ belonging to a spatial domain $\mathcal{D}$

  and for

  t belonging to a time interval $[0, T]$

  whose value at any point $\boldsymbol{x} \in \mathcal{D}$ and at any time $t \in [0, T]$

  is drawn at random from a given PDF $\rho(\omega)$

  - more general definitions of random fields (i.e., definitions that involve variables other than space-time ones) are, of course, also possible

- Random fields are often used as inputs into a model

$$\mathcal{M}\Big(U(\boldsymbol{x}, t; \omega); \eta(\boldsymbol{x}, t; \omega), g(\boldsymbol{x}, t)\Big) = 0$$

  where here $U(\boldsymbol{x}, t; \omega)$ denotes the model solution

  $g(\boldsymbol{x}, t)$ denotes an input that does not depend on a sample ω

WHITE NOISE – UNCORRELATED RANDOM FIELDS

- The value of a white noise random field $\eta_{white}(\boldsymbol{x}, t; \omega)$ at every point in space and at every instant in time is independently chosen according to a Gaussian PDF

- The inclusion of ω in the argument of the random field indicates that

  - the values of the random field at any point $\boldsymbol{x}$ and any time instant t are determined by first sampling ω according to a one-dimensional PDF $\rho_g(\omega)$
    here this is the standard normal distribution $\mathcal{N}(0, 1)$

  - then adjusting the sample value taken for the point $\boldsymbol{x}$ and time instant t so that it corresponds to a sample from the normal distribution having given mean $\mu_{white}(\boldsymbol{x}, t)$ and variance $\sigma^2_{white}(\boldsymbol{x}, t)$

  - as a result, the sample value used is given by

    $$\mu_{white}(\boldsymbol{x}, t) + \sigma_{white}(\boldsymbol{x}, t)\omega$$

  - often, σ^2_{white} is chosen to be constant

- The covariance function[1] corresponding to a white noise random field $\eta_{white}(\boldsymbol{x}, t; \omega)$ is given by

$$\mathsf{Cov}_{white}(\boldsymbol{x}, t, \boldsymbol{x}', t') = \sigma_{white}(\boldsymbol{x}, t)\sigma_{white}(\boldsymbol{x}', t')\delta(\boldsymbol{x} - \boldsymbol{x}')\delta(t - t')$$

  where $\delta(\cdot)$ denotes the Dirac delta function

  - thus, the variance $\mathsf{V}_{white}(\boldsymbol{x}, t)$ of white noise is infinite[2] so that white noise cannot describe a real process

  - notwithstanding this observation, white noise random fields are a very common random input used, e.g., as inputs in the PDE setting

    - later we comment on why numerical simulations involving white noise inputs do not "see" this infinity

[1] The mean of a random field $\eta(\boldsymbol{x}, t; \omega)$ is defined by $\mu_\eta(\boldsymbol{x}, t) = \mathsf{E}\big[(\eta(\boldsymbol{x}, t; \cdot)\big]$, where E denotes the expectation with respect to the PDF $\rho_g(\omega)$. Its covariance is given by

$$\begin{aligned}\mathsf{Cov}_\eta(\boldsymbol{x}, t; \boldsymbol{x}', t') &= \mathsf{E}\Big[\big(\eta(\boldsymbol{x}, t; \cdot) - \mu_\eta(\boldsymbol{x}, t)\big)\big(\eta(\boldsymbol{x}', t'; \cdot) - \mu_\eta(\boldsymbol{x}', t')\big)\Big] \\ &= \mathsf{E}\Big[\eta(\boldsymbol{x}, t; \cdot)\big(\eta(\boldsymbol{x}', t'; \cdot)\Big] - \big(\eta_{white}(\boldsymbol{x}, t; \omega)\big)^2\end{aligned}$$

and its variance by $\mathsf{V}_\eta(\boldsymbol{x}, t) = \mathsf{Cov}_\eta(\boldsymbol{x}, t; \boldsymbol{x}, t)$

[2] One should not confuse the variance of the white noise field $\eta(\boldsymbol{x}, t; \cdot)$, which is infinite, with the variance $\sigma^2_{white}(\boldsymbol{x}, t)$ of the Gaussian PDF from which sample values of the random field are drawn, which is finite

- Obviously, in any computer simulation, one cannot sample the Gaussian distribution at every point of the spatial domain and at every instant of time so that white noise is replaced by discretized white noise
  - among the means available for discretizing white noise in the PDE setting, grid-based methods are the most popular
  - truncated expansions in terms of Hermite polynomials are another means used to discretized white noise; we don't have time to discuss this approach

Grid-based methods for discretizing white noise

- For simplicity, we consider time-independent white noise random fields
- To define a single realization of grid-based discretized white noise
  - we first subdivide the spatial domain $\mathcal{D}$ into N non-overlapping, covering subdomains $\{\mathcal{D}_n\}_{n=1}^N$
  - then, for each $n = 1, \ldots, N$, we let

    $|\mathcal{D}_n|$ denote the volume of the subdomain $\mathcal{D}_n$

    $\chi_n(\boldsymbol{x})$ denote the characteristic (or indicator) function corresponding to the subdomain $\mathcal{D}_n$

    $y_n(\omega)$ denotes an $\mathcal{N}(0,1)$ i.i.d. random number

    so that the value of $y_n(\omega)$ is independent of the value of $y_{n'}(\omega)$ for all $n' \neq n$ but both $y_n(\omega)$ and $y_{n'}(\omega)$ are drawn from the same PDF $\rho_g(\omega)$

- Then, for some constants $\{a_n\}_{n=1}^N$, we seek an approximation of the white noise random field $\eta_{white}(\boldsymbol{x};\omega)$ of the form[3]

$$\eta_{white}^N(\boldsymbol{x};\boldsymbol{y}) = \mu_{white}(\boldsymbol{x}) + \sum_{n=1}^N a_n \chi_n(\boldsymbol{x}) y_n(\omega) \approx \eta_{white}(\boldsymbol{x};\omega)$$

  where $\boldsymbol{y} \in \mathbb{R}^N$ denotes the vector having components y_n, $n = 1, \ldots, N$

  - note that this is a piecewise constant approximation (with respect to the subdivision $\{\mathcal{D}_n\}_{n=1}^N$ of the spatial domain $\mathcal{D}$) of the random field $\eta_{white}(\boldsymbol{x};\omega) - \mu_{white}(\boldsymbol{x})$

- The covariance of the discrete random field $\eta_{white}^N(\boldsymbol{x};\omega)$ is given by

$$\mathrm{Cov}_{white}^N(\boldsymbol{x},\boldsymbol{x}') = \begin{cases} a_n^2 & \text{if both } \boldsymbol{x},\boldsymbol{x}' \in \mathcal{D}_n \\ 0 & \text{otherwise} \end{cases}$$

[3] Usually the mean function $\mu_{white}(\boldsymbol{x})$ is also approximated by a function $\mu_{white}^N(\boldsymbol{x})$ defined with respect to the grid. For example, one could use the piecewise constant approximation $\mu_{white}^N(\boldsymbol{x}) = \sum_{n=1}^N \mu(\boldsymbol{x}_n^*)\chi_n(\boldsymbol{x})$, where $\boldsymbol{x}_n^*$ denotes the centroid of the subdomain $\mathcal{D}_n$.

- Because pointwise values of the covariance

$$\mathrm{Cov}_{white}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_{white}(\boldsymbol{x})\sigma_{white}(\boldsymbol{x}')\delta(\boldsymbol{x} - \boldsymbol{x}')$$

of a spatially dependent white noise random field are not defined, we determine, for $n = 1, \ldots, N$, the coefficients a_n by matching the averages over $\mathcal{D}_n$ of the covariance $\mathrm{Cov}_{white}(\boldsymbol{x}, \boldsymbol{x}')$ and its approximation $\mathrm{Cov}^N_{white}(\boldsymbol{x}, \boldsymbol{x}')$

  - in this manner we obtain that

$$a_n^2 = \frac{1}{|\mathcal{D}_n|^2} \int_{\mathcal{D}_n} \sigma^2_{white}(\boldsymbol{x})\, d\boldsymbol{x}$$

  - approximating $\sigma_{white}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{D}_n$ by its value at the centroid $\boldsymbol{x}_n^*$ of $\mathcal{D}_n$, we then have $a_n = \frac{\sigma_{white}(\boldsymbol{x}_n^*)}{\sqrt{|\mathcal{D}_n|}}$

  - so that the discretized white noise random field is given by

$$\eta^N_{white}(\boldsymbol{x}; \boldsymbol{y}) = \mu_{white}(\boldsymbol{x}) + \sum_{n=1}^{N} \frac{\sigma_{white}(\boldsymbol{x}_n^*)}{\sqrt{|\mathcal{D}_n|}} \chi_n(\boldsymbol{x}) y_n(\omega) \approx \eta_{white}(\boldsymbol{x}; \omega)$$

- Thus, via discretization, white noise has been reduced to the case of N random parameters.

- If $\sigma_{white}(\boldsymbol{x}) = \sigma_{white} =$ constant, as is often the case, then

$$\eta^N_{white}(\boldsymbol{x};\boldsymbol{y}) = \mu_{white}(\boldsymbol{x}) + \sigma_{white}\sum_{n=1}^{N}\frac{1}{\sqrt{|\mathcal{D}_n|}}\chi_n(\boldsymbol{x})y_n(\omega) \approx \eta_{white}(\boldsymbol{x};\omega)$$

- In one dimension, for constant variance σ^2_{white} and zero expected value and for a uniform grid of size h, we have the well-known formula

$$\eta^N_{white}(x;\boldsymbol{y}) = \frac{\sigma_{white}}{\sqrt{h}}\sum_{n=1}^{N}\chi_n(x)y_n(\omega) \approx \eta_{white}(x;\omega)$$

  for approximating a white noise random field

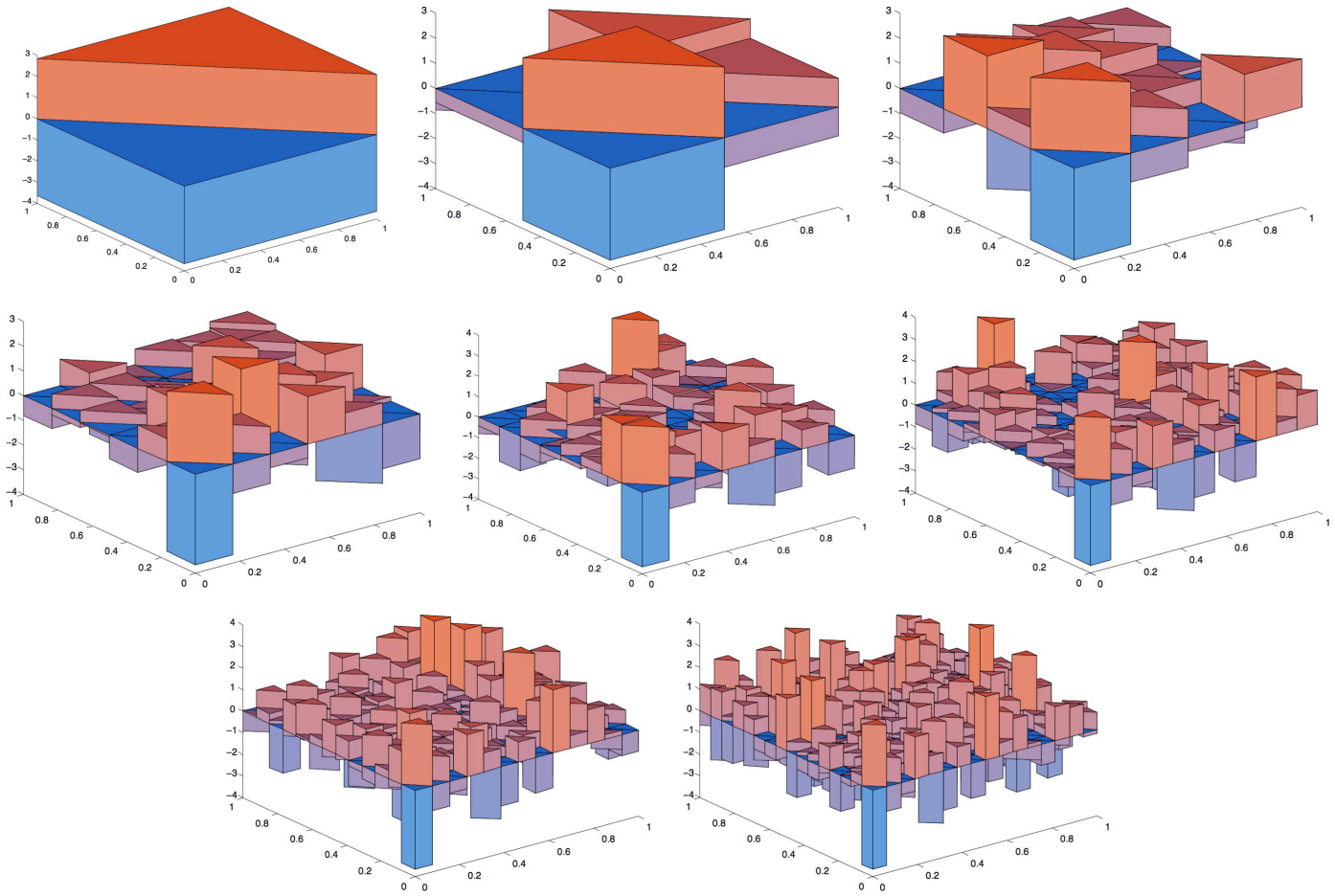
- We have constructed an approximate white noise field based on sampling in subdomains
  - this is convenient for finite element spatial discretization schemes
  - for other spatial discretization schemes, e.g., finite difference methods, it may be more convenient to sample at grid points; the construction process we have discussed is easily amended to this case

- It is important to note that
  the variance of the discretized white noise field is finite
  - in fact, we have, for $n = 1, \ldots, N$,

$$\mathsf{V}^N_{white}(\boldsymbol{x}) = \mathsf{Cov}^N_{white}(\boldsymbol{x}; \boldsymbol{x}) = \frac{\sigma^2_{white}(\boldsymbol{x}^*_n)}{|\mathcal{D}_n|} \qquad \boldsymbol{x} \in \mathcal{D}_n$$

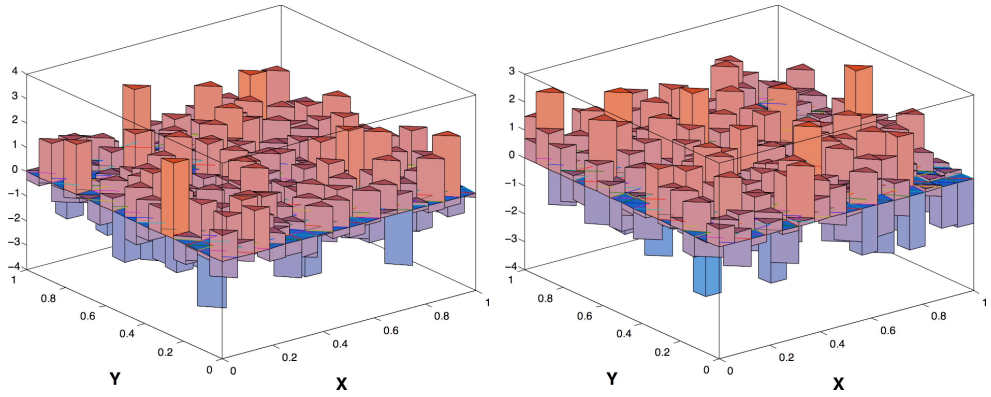
    which clearly is finite for all $\boldsymbol{x} \in \mathcal{D}$
  - this is one reason why, in simulations, the fact that the variance of a white noise random field is infinite does not cause codes to fail
  - however, note also that as the spatial grid size goes to zero, i.e., as $|\mathcal{D}_n| \to 0$, that the variance of discretized white noise goes to infinity

- Time-dependent white noise fields can be treated in an entirely similar manner



Realizations of grid-based discretized white noise over the same time interval in a square subdivided into 2, 8, 32, 72, 238, 242, 338, and 512 triangles

Independent random constants are sampled for each triangle



Realizations of grid-based discretized white noise over two different time intervals in a square subdivided into the same number of triangles

- What about non-Gaussian white noise fields?
  - it seems an easy matter to sample according to a different PDF instead of the normal distribution; indeed, this is done in practice
  - however, there are issues that arise when doing so, not the least of which is that the Gaussian PDF is the only one which is completely defined by its mean and variance

COLORED NOISE – CORRELATED RANDOM FIELDS

- Suppose that you are standing next to a friend and it starts raining hard where you are standing
  - if rainfall were a white noise random field, then, it is possible that it is not raining, or perhaps just sprinkling, over the friend standing next to you; in fact, it may raining even harder over your friend
  - of course, in reality, it is highly unlikely that it is not raining, or even sprinkling, over you friend
  - instead, it is highly likely that it is indeed raining hard over your friend because rainfall is (a well) correlated random field

- A correlated random fields $\eta(\boldsymbol{x}, t; \omega)$ is such that at each point $\boldsymbol{x}$ in a spatial domain $\overline{\mathcal{D}}$ and at each instant t in a time interval $[t_0, t_1]$, the value of η is determined by a random variable ω whose values are drawn from a given probability density function $\rho(\omega)$
  - however, unlike the white noise case, these draws are not independent

    $\Longleftarrow$ the covariance function of a correlated random field $\eta(\boldsymbol{x}, t; \omega)$ does not reduce to delta functions

- In rare cases, a formula for the random field is "known"
  - more often, only the mean $\mu_\eta(\boldsymbol{x}, t)$ and covariance function $\text{Cov}_\eta(\boldsymbol{x}, t; \boldsymbol{x}', t')$ are known at points $\boldsymbol{x}$ and $\boldsymbol{x}'$ in $\overline{\mathcal{D}}$ and time instants t and t' in $[t_0, t_1]$
  - in this case, we do not have a formula for $\eta(\boldsymbol{x}, t; \omega)$ so that we cannot evaluate $\eta(\boldsymbol{x}, t; \omega)$ when we need to

- Actually, we usually do not even know the covariance function so that instead, it is often guessed; common guesses are

$$\begin{aligned}
\text{exponential covariance} \qquad \mathrm{Cov}(\boldsymbol{x}, t; \boldsymbol{x}', t') &= e^{-\frac{|\boldsymbol{x}-\boldsymbol{x}'|}{L} - \frac{|t-t'|}{T}} \\
\text{(multivariate) Gaussian covariance} \qquad \mathrm{Cov}(\boldsymbol{x}, t; \boldsymbol{x}', t') &= e^{-\frac{|\boldsymbol{x}-\boldsymbol{x}'|^2}{L^2} - \frac{|t-t'|^2}{T^2}} \\
\text{(general) Gaussian covariance} \qquad \mathrm{Cov}(\boldsymbol{x}; \boldsymbol{x}') &= e^{-(\boldsymbol{x}-\boldsymbol{x}')^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{x}')}
\end{aligned}$$

  where

  L denotes the correlation length

  T the correlation time

  Σ a symmetric positive definite matrix

  - large L and T correspond to long-range correlation whereas small L and T correspond to short-range correlation

- Note that, in general, covariance functions are symmetric and non-negative
  - for the sake of simplicity, we will assume they are positive

- Examples of known covariance functions
  - one-dimensional Weiner process or Brownian motion W_t for which

$$\mu(t) = 0 \qquad \text{and} \qquad \text{Cov}(t, t') = \min(t, t')$$

  - the Ornstein-Uhlenbeck (or mean-reverting) process defined by the stochastic ordinary differential equation

$$d\eta = \theta(\overline{\mu} - \eta)dt + \sigma dW_t$$

where W_t is a Wiener process so that we have that dW_t is Gaussian white noise

$\theta =$ the deterministic speed of reversion

$\overline{\mu} =$ the long-run equilibrium level or mean respectively

$\sigma^2 =$ the variance

- the solution of this stochastic ODE is given by

$$\eta(t;\omega) = \eta_0 e^{-\theta t} + \overline{\mu}(1 - e^{-\theta t}) + \sigma e^{-\theta t} \int_0^t e^{\theta s} dW_s$$

- η_0 denotes the deterministic initial condition

- the corresponding mean is given by

$$\mu_\eta(t) = \mathsf{E}\big(\eta(t;\cdot)\big) = \eta_0 e^{-\theta t} + \overline{\mu}\big(1 - e^{-\theta t}\big)$$

$\Longrightarrow$ hence the process is "mean reverting"
as $t \to \infty$, $\mu_\eta(t) \to \overline{\mu}$ for any initial data η_0

- the covariance function is given by

$$\mathsf{Cov}_\eta(t;t') = \sigma^2 e^{-\theta(t+t')} \mathsf{E}\Big(\int_0^t e^{\theta s} dW_s \Big) \Big(\int_0^{t'} e^{\theta u} dW_u \Big)$$

$$= \frac{\sigma^2}{2\theta} e^{-\theta(t+t')} \big(e^{2\theta \min(t,t')} - 1 \big)$$

– the Matérn covariance

$$\text{Cov}(\boldsymbol{x};\boldsymbol{x}') = \sigma^2 \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu}\frac{|\boldsymbol{x}-\boldsymbol{x}'|}{L}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{|\boldsymbol{x}-\boldsymbol{x}'|}{L}\right)$$

$\sigma^2 =$ the variance

$\Gamma(\cdot) =$ the Gamma function

$K_\nu(\cdot) =$ the modified Bessel function of the second kind

$\nu =$ a parameter $\qquad \nu = \frac{1}{2} \longrightarrow$ Matérn reduces to exponential

- In other cases, Fourier spectral information about a correlated random field is known
  - an example is pink noise that has, in one dimension, a $1/f$ power spectrum, in contrast to the $1/f^0$ and $1/f^2$ power spectra for white noise and Brownian noise, respectively; here f denotes the frequency
  - again, we do not have a formula for the random field that we can evaluate at spatial and temporal points

- Here, we consider the case for which the mean and covariance functions of a random field are known or guessed
  - in this case, we would like to find a simple formula depending on only a few parameters whose mean and covariance functions are approximately the same as the given mean and covariance functions
  - expansions in terms of orthogonal polynomials provide a means for approximating correlated random fields
  - there also exist grid-based methods for this purpose
  - we consider perhaps the most popular approach: the Karhunen-Loève (KL) expansion of a correlated random field $\eta(\boldsymbol{x}, t; \omega)$

- Given the mean and covariance functions for a random field $\eta(\boldsymbol{x}, t; \omega)$, the KL expansion provides a simple formula for the random field in terms of an infinite number of uncorrelated random parameters $\{y_n\}_{n=1}^{\infty}$
  - in principle, a truncated KL expansion can be used whenever one needs a value of $\eta(\boldsymbol{x}, t; \omega)$ at a point $\boldsymbol{x}$ and a time t

Karhunen-Loève expansions of Guassian random fields

- To keep things simple, we discuss KL expansions for the case of time-independent random fields; extension to the case of time-dependent fields is straightforward

- Let $\eta(\boldsymbol{x};\omega)$ denote a Gaussian random field, i.e., at each point $\boldsymbol{x}$, we draw a sample from a standard Gaussian PDF

- Given the mean $\mu_\eta(\boldsymbol{x})$ and covariance $\mathrm{Cov}_\eta(\boldsymbol{x},\boldsymbol{x}')$ of a random field $\eta(\boldsymbol{x};\omega)$, the eigenpairs $\{\lambda_n, b_n(\boldsymbol{x})\}_{n=1}^{\infty}$ satisfy the eigenvalue problem

$$\int_{\mathcal{D}} \mathrm{Cov}_\eta(\boldsymbol{x},\boldsymbol{x}')\, b(\boldsymbol{x}')\, d\boldsymbol{x}' = \lambda b(\boldsymbol{x}) \qquad \forall\, \boldsymbol{x} \in \mathcal{D}$$

  - due to the symmetry of $\mathrm{Cov}_\eta(\cdot;\cdot)$, the eigenvalues λ_n are real and the eigenfunctions $b_n(\boldsymbol{x})$ can be chosen to be real and orthonormal, i.e.,

$$\int_{\mathcal{D}} b_n(\boldsymbol{x})\, b_{n'}(\boldsymbol{x})\, d\boldsymbol{x} = \delta_{nn'}.$$

– due to the positivity[4] of $\mathrm{Cov}_\eta(\cdot;\cdot)$, the eigenvalues are all positive

- without loss of generality, they may be ordered in non-increasing order $\lambda_1 \geq \lambda_2 \geq \cdots$

- Then, the random field $\eta(\boldsymbol{x};\omega)$ admits the KL expansion

$$\eta(\boldsymbol{x};\omega) = \mu_\eta(\boldsymbol{x}) + \sum_{n=1}^{\infty} \sqrt{\lambda_n}\, b_n(\boldsymbol{x})\, y_n(\omega)$$

where $\{y_n(\omega)\}_{n=1}^{\infty}$ are sampled independently from the standard normal distribution

[4] By non-negativity we mean that

$$\int_{\mathcal{D}} \int_{\mathcal{D}} \mathrm{Cov}_\eta(\boldsymbol{x},\boldsymbol{x}')\, b(\boldsymbol{x}) b(\boldsymbol{x}')\, d\boldsymbol{x}' d\boldsymbol{x} \geq 0$$

for all suitable functions $b(\boldsymbol{x})$; in general, covariance functions are non-negative, but, for the sake of simplicity, we assume positivity, i.e., the above relation holds strictly

- Thus, the KL expansion accomplishes two important things
  - it provides a formula (albeit one involving and infinite number of parameters) for the correlated random field $\eta(\boldsymbol{x};\omega)$ in terms of random parameters
  - it expresses the correlated random field in terms of uncorrelated parameters

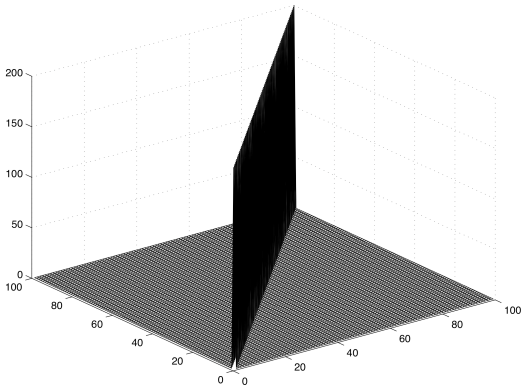
- As an example of KL expansions, consider Brownian motion which has the KL eigenpairs

$$\lambda_n = \frac{1}{(n-\frac{1}{2})^2\pi^2} \qquad \text{and} \qquad b_n(t) = \sqrt{2}\sin\left((n-\frac{1}{2})\pi t\right)$$

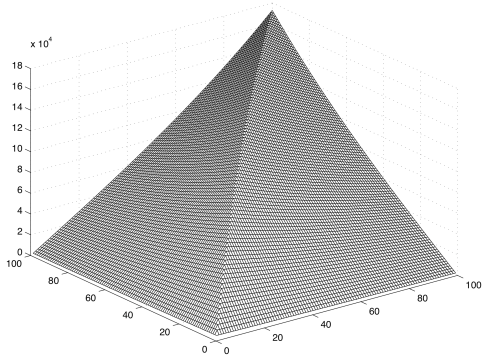
and the KL expansion

$$W_t = \sqrt{2}\sum_{n=1}^{\infty}\frac{\sin\left((n-\frac{1}{2})\pi t\right)}{(n-\frac{1}{2})\pi}y_n(\omega)$$

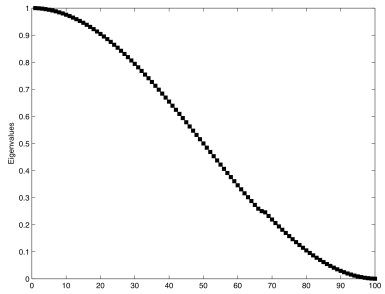
- The usefulness of the KL expansion results from the fact that the eigenvalues $\{\lambda_n\}_{n=1}^{\infty}$ decay as n increases
  - how fast they decay depends on the smoothness of the covariance function $\mathrm{Cov}_\eta(\boldsymbol{x}, \boldsymbol{x}')$ and on the correlation length L



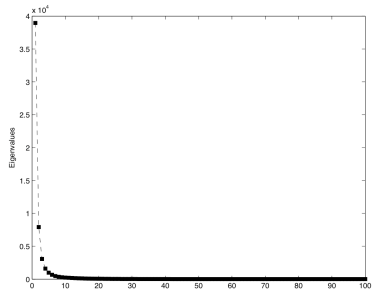
Peaked covariance function with very small correlation length



Slowly decaying covariance function with large correlation length



Corresponding KL eigenvalues



Corresponding KL eigenvalues

- The decay of the eigenvalues implies that truncated KL expansions

$$\eta_N(\boldsymbol{x};\boldsymbol{y}) = \mu(\boldsymbol{x}) + \sum_{n=1}^{N} \sqrt{\lambda_n} b_n(\boldsymbol{x}) y_n(\omega)$$

  can be accurate approximations to the exact expansions

  $\boldsymbol{y} \in \mathbb{R}^N$ is the vector with components y_n, $n = 1, \ldots, N$

- Note that often in practice, $\mu(\boldsymbol{x})$ and $b_n(\boldsymbol{x})$ are only approximately known, so that one is left with a spatial approximation to the truncated KL expansion

$$\eta_N^h(\boldsymbol{x};\boldsymbol{y}) = \mu^h(\boldsymbol{x}) + \sum_{n=1}^{N} \sqrt{\lambda_n} b_n^h(\boldsymbol{x}) y_n(\omega) \approx \eta_N(\boldsymbol{x};\boldsymbol{y})$$

  where $\mu^h(\boldsymbol{x}) \approx \mu(\boldsymbol{x})$ and $b_n^h(\boldsymbol{x}) \approx b_n(\boldsymbol{x})$

  – thus we have an error induced by the truncation of the KL expansion and an another error induced by, e.g., the spatial approximation of the coefficients

    - one endeavors to balance these two errors

– if one wishes for the relative error of the truncated expansion to be less than a prescribed tolerance ε, i.e., if one wants[5]

$$\frac{\mathsf{E}[\|\eta_N - \eta\|]}{\mathsf{E}[\|\eta\|]} \leq \varepsilon,$$

where here $\|\cdot\|$ denotes the $L^2(\mathcal{D})$ norm, one should choose N to be the smallest integer such that

$$\frac{\sum_{n=N+1}^{\infty} \lambda_n}{\sum_{n=1}^{\infty} \lambda_n} \leq \varepsilon^2 \qquad \text{or, equivalently,} \qquad \frac{\sum_{n=1}^{N} \lambda_n}{\sum_{n=1}^{\infty} \lambda_n} \geq 1 - \varepsilon^2$$

– these estimates easily follow because the orthonormality of the eigenfunctions $\{b_n\}$ imply that, e.g., $\|\eta\|^2 = \sum_{n=1}^{\infty} \lambda_n$

– of course, in practice, the infinite sums appearing in these estimates would themselves have to be truncated, with the number of retained terms being considerably larger than N

– this requires computing more eigenpairs than the N one ends up using in the truncated KL expansion

[5] For the sake of simplicity, for the time being we ignore the mean function $\mu(\boldsymbol{x})$; being a deterministic quantity, its role in error estimation is straightforward to analyze

- It is easily seen that the covariance function for the truncated KL expansion is given by

$$\mathrm{Cov}_{\eta}^{N}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{n=1}^{N} \lambda_n b_n(\boldsymbol{x}) b_n(\boldsymbol{x}')$$

  - then, a theorem due to Mercer shows the convergence of the covariance function
  - specifically, if $\mathrm{Cov}_{\eta}(\boldsymbol{x}, \boldsymbol{x}')$ is continuous, symmetric, non-negative definite, and square integrable on $\mathcal{D} \times \mathcal{D}$, we then have

$$\lim_{N \to \infty} \max_{\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{D}} \left| \mathrm{Cov}_{\eta}(\boldsymbol{x}, \boldsymbol{x}') - \mathrm{Cov}_{\eta}^{N}(\boldsymbol{x}, \boldsymbol{x}') \right| = 0$$

- There are also estimates available that relate the decay of the eigenvalues λ_n to the smoothness of the covariance function
  - for example, if $\mathrm{Cov}_\eta(\boldsymbol{x}, \boldsymbol{x}') \in L^2(\mathcal{D} \times \mathcal{D})$ is piecewise analytic on $\mathcal{D} \times \mathcal{D}$, then there exist constants c_1 and c_2 independent of n such that

$$0 \leq \lambda_n \leq c_1 e^{-c_2 n^{1/d}} \quad \text{for all integers } n \geq 1$$

    i.e., we have exponential decay in the eigenvalues; here, d denotes the spatial dimension

  - if instead $\mathrm{Cov}_\eta(\boldsymbol{x}, \boldsymbol{x}')$ is merely piecewise $H_0^k(\mathcal{D} \times \mathcal{D})$, where $H_0^k(\mathcal{D} \times \mathcal{D})$ denotes the Sobolev space of functions having square integrable derivatives of order up to k that also vanish on the boundary of $\mathcal{D} \times \mathcal{D}$, we have the algebraic decay estimate

$$0 \leq \lambda_n \leq c_2 n^{-k/d} \quad \text{for all integers } n \geq 1$$

- To summarize
  - we approximate a Gaussian random field $\eta(\boldsymbol{x};\omega)$ by its N-term truncated KL expansion
  - the parameters $\{y_n\}_{n=1}^{N}$ are uncorrelated
  - because we are considering multivariate Gaussian random variables, the parameters are also independent
  - thus, we now have a formula for an approximation to a correlated Gaussian random field that involves a finite number of independent random parameters
  - we can then use any of the methods involving models with a finite number of random parameters to solve problems defined in terms of Gaussian random fields

Approximating general non-Gaussian correlated random fields

- In many applications, random field inputs are assumed to be not normally distributed

- We can still use the KL expansion to express the field in terms of uncorrelated random parameters $\{y_n\}_{n=1}^{\infty}$
  - formally, one only need draw samples y_n, $n = 1, 2, \ldots,$ from a desired non-Gaussian PDF

- There is a problem with doing this, which is often ignored in practice
  - although the independence of random variables implies that they are uncorrelated,
    the converse is not in general true, i.e.,
    uncorrelated does not imply independence

- a classic example is to
  - let $y_1 \in [-1, 1]$ denote a uniformly distributed random variable
  - then let $y_2 = y_1^2$
  - clearly, y_1 and y_2 are not independent
  - however, it is easy to show that $\text{Cov}(y_1, y_2) = 0$
    i.e., y_1 and y_2 are uncorrelated

- In fact, uncorrelated implies independence only if the components of $\boldsymbol{y}$ are multivariate Gaussian variables
  i.e., if the components of $\boldsymbol{y}$ are jointly Gaussian

- So, for general non-Gaussian random fields, our only recourses are to
  - assume that the KL parameters are independent (what is usually done), in which case we simply express the random field in terms of its KL expansion

  or
  - using CDFs and inverse CDFs, express the non-Gaussian random field in terms of a Gaussian random field and then use the KL expansion for the latter