Announcements

- Second assignment solutions available on Friday.

- Third assignment available on Saturday.

- Project progress meetings.

- Thursday: no class.

- Optional reading on splines in commercial modelers.

- Tuesday: quaternions. *CIS*.

The solutions to the assignment will be made available on Friday, after all students have turned in their work (including those students who got permission to turn in their work late). The graded assignments will be handed out next Tuesday.

The third assignment is in the making. `SoftImage`, the commercial modeler (and more than just a modeler) you will use to complete the assignment, has been installed on the raptors. Instructions on configuring your account to use `SoftImage` will be posted on the class WWW pages soon.

Project progress meetings will take place next week, and you must be well into your projects by then; see the WWW for a detailed list of my expectations from you during these meetings. Also, you have to sign up for a time slot by Friday; again, see the WWW.

Happy Turkey Day!

The optional reading is a well-written article reviewing a commercial modeler that uses splines and only splines. The reviewer discusses the strengths and shortcomings of such a system. No, I am not endorsing the particular product by handing out this review. Leftover copies will be left on my desk.

Today

- Surfaces:
  - Total degree patches.
  - Total degree splines.

- Wavelets:
  - Scaling functions.
  - Haar wavelets.
  - Multiresolution analysis.
  - Filter banks.
  - Applications.

Total Degree Patches

Extend Bézier curves to surfaces in two steps:

1. Substitute
   - maps of domain triangles to surface patches for
   - maps of real line segments to curve segments.
2. Substitute
   - barycentric coordinates on plane for
   - linear interpolation.

Total degree patches are also called *Bézier triangles*.

Recall that linear interpolation is a special case of using barycentric coordinates: the coordinate of a point t on the real line is a single number w, such that

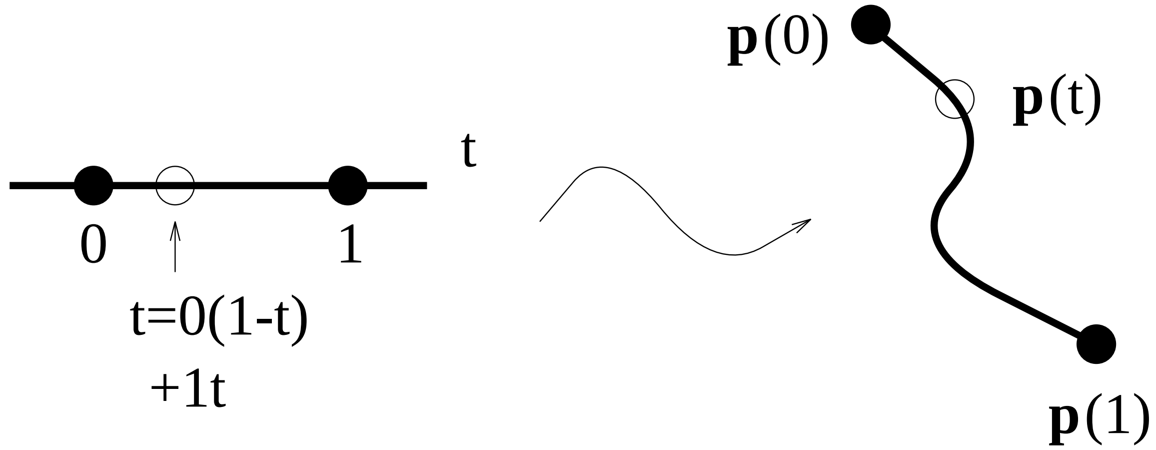
$$t = (1 - w)r + ws,$$

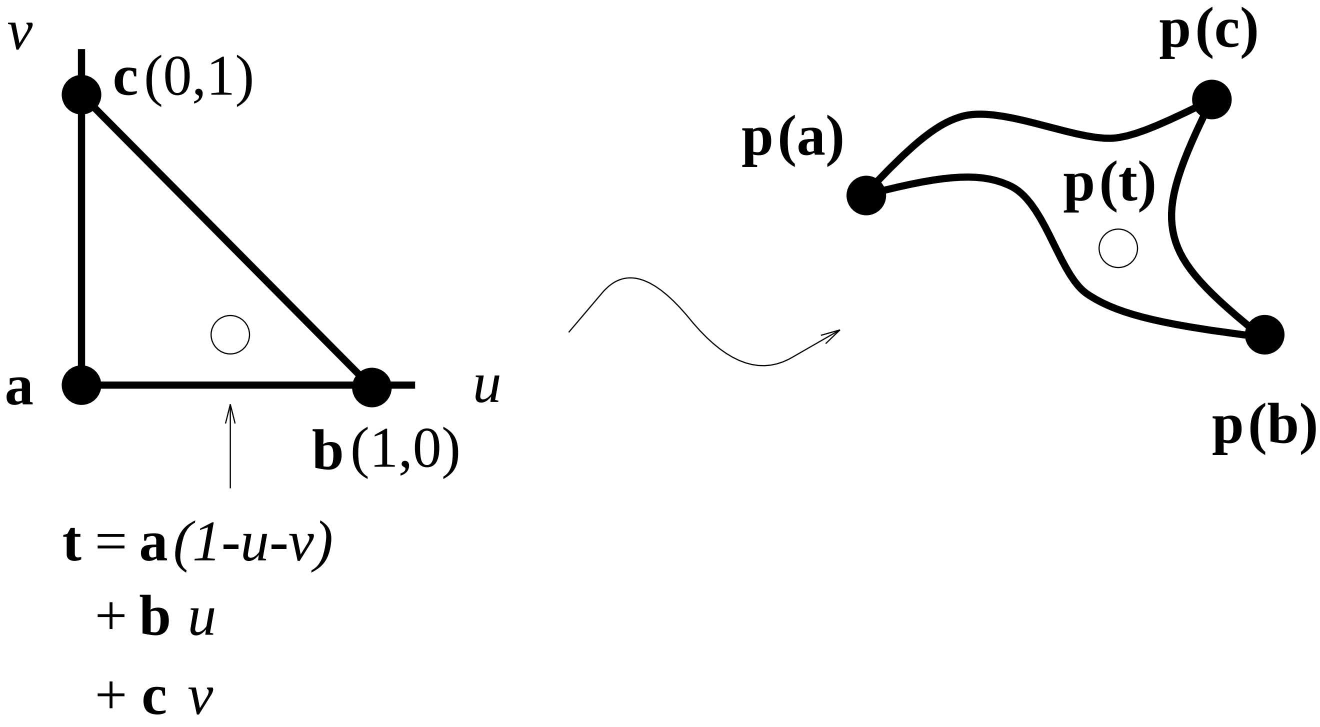
where s and r are fixed (reference) points on the line. Solving for w yields

$$w = \frac{t - r}{s - r}$$

for the barycentric coordinate of t with respect to s and r.

Total Degree Patches: Mapping





This slide illustrates the analogy we are attempting to draw between curve segments, defined over segments of the real line, and surface patches, defined over triangles on the plane. As usual, we assume that our parameters lie is some convenient interval.

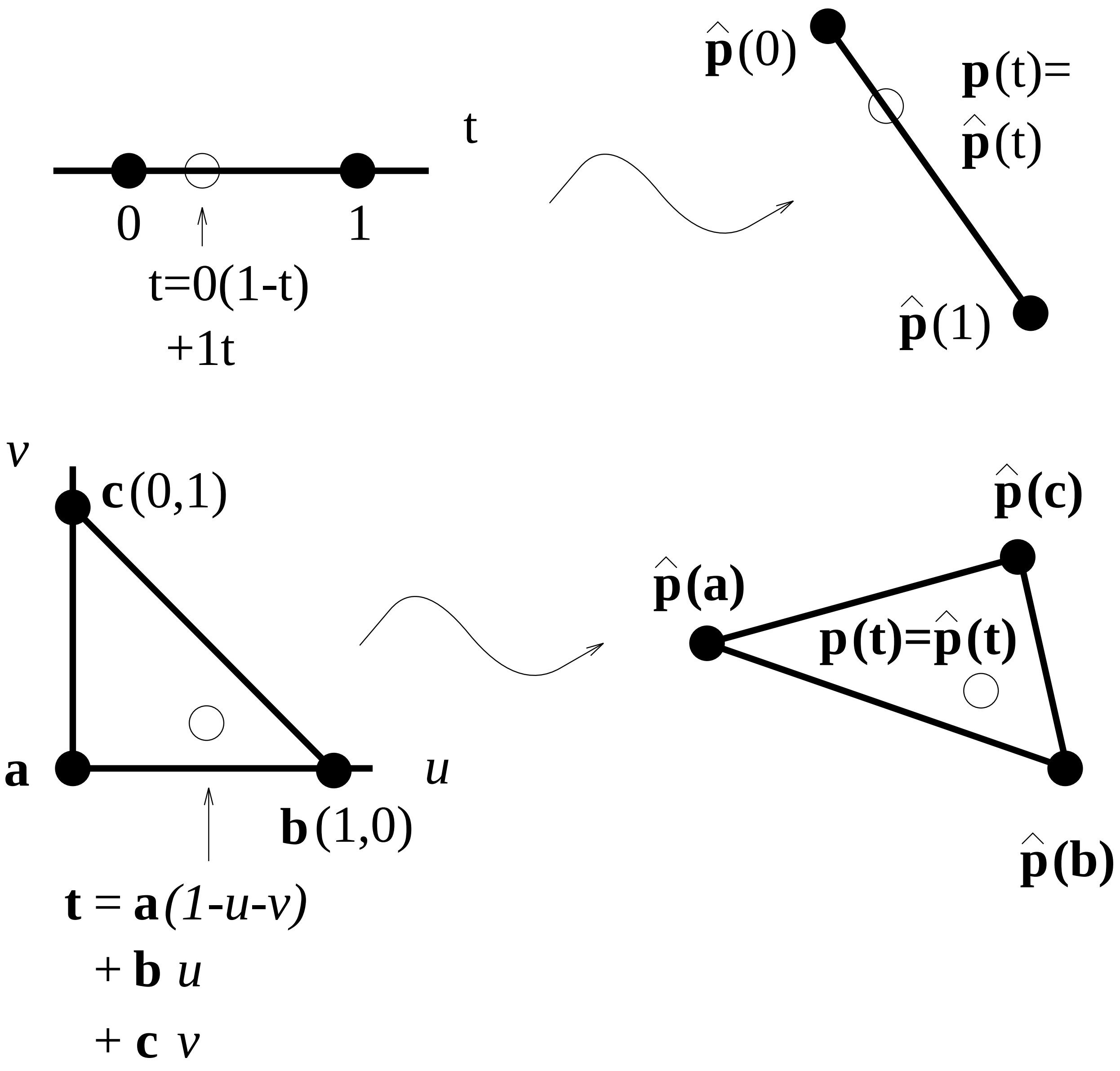
In the case of curves, our parameter t, a point on the real line, is expressed in terms of its barycentric coordinate $w = t$ with respect to the reference points 1 and 0; this is a special case of the general formula for the barycentric coordinate w, with $s = 1$ and $r = 0$.

In the case of patches, it is convenient to use the triangle whose vertices are

$$\begin{aligned} \mathbf{a} &= (0,0) \\ \mathbf{b} &= (1,0) \\ \mathbf{c} &= (0,1) \end{aligned}$$

since, in this case, the barycentric coordinates of an arbitrary point $\mathbf{t}$ on the domain are directly given by $1-u-v$, u, and v. Remember, though, that this simplifying assumption is made only for the sake of convenience: total degree patches may be defined over arbitrary domain triangles (that is, for arbitrarily points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$).

Total Degree Patches: Affine

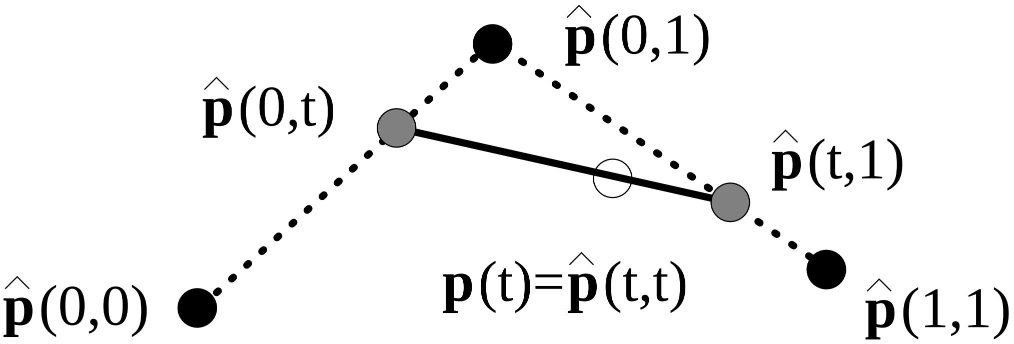


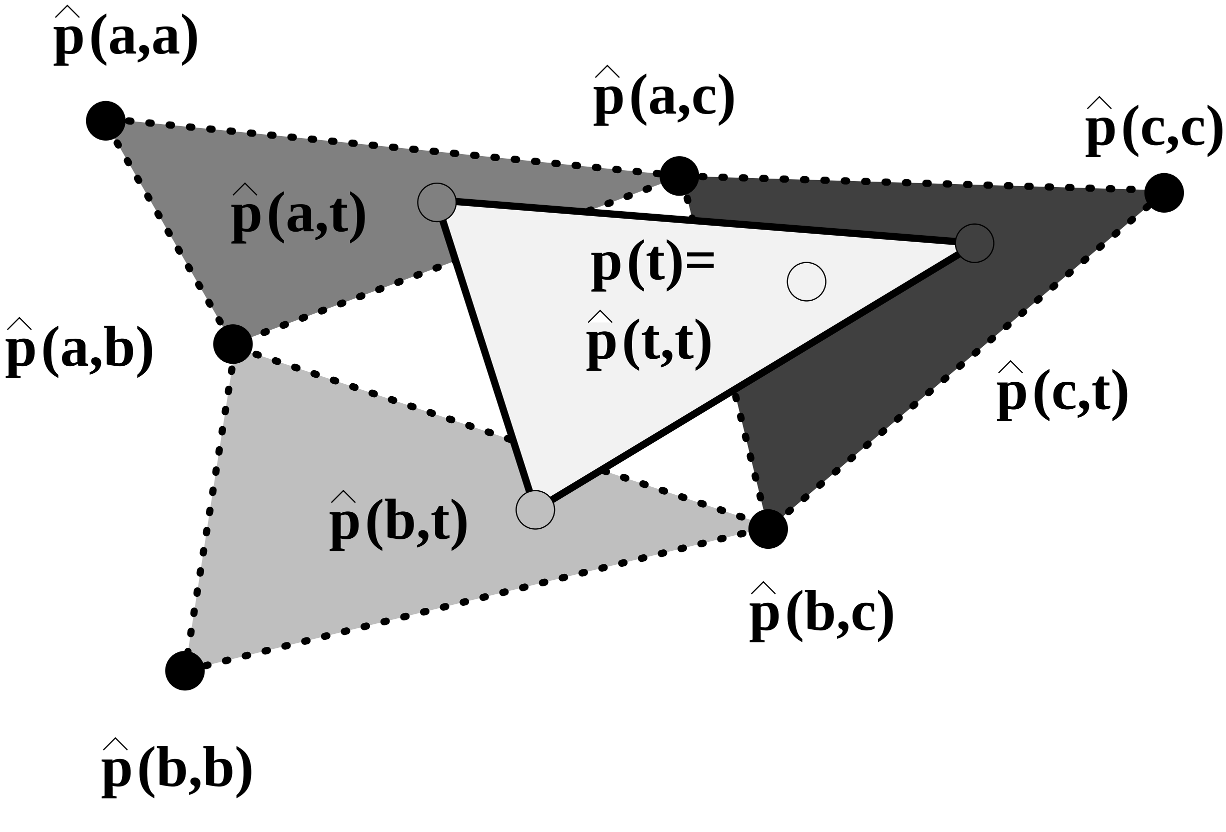
On top is drawn the regular procedure we follow to determine the point $\mathbf{p}(t) = \widehat{\mathbf{p}}(t)$ on an affine curve segment: we first find the barycentric coordinate t of point t with respect to points 1 and 0 on the real line; then, $\widehat{\mathbf{p}}(t)$ (drawn as a hollow circle) is the point with barycentric coordinate t but relative to the user-specified Bézier points $\widehat{\mathbf{p}}(1)$ and $\widehat{\mathbf{p}}(0)$ (drawn as solid disks).

The corresponding procedure for finding the point $\mathbf{p}(\mathbf{t}) = \widehat{\mathbf{p}}(\mathbf{t})$ on an affine patch is similar: we first find the barycentric coordinates $1 - u - v$, u, v of the point $\mathbf{t}$ with respect to points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$ on the plane; then, $\widehat{\mathbf{p}}(\mathbf{t})$ (drawn as a hollow circle) is the point with barycentric coordinates $1 - u - v$, u and v but relative to the user-specified *Bézier points* $\widehat{\mathbf{p}}(\mathbf{a})$, $\widehat{\mathbf{p}}(\mathbf{b})$, and $\widehat{\mathbf{p}}(\mathbf{c})$ (drawn as solid disks).

Observe that, as $\widehat{\mathbf{p}}(t)$ lies on the line defined by $\widehat{\mathbf{p}}(1)$ and $\widehat{\mathbf{p}}(0)$, so does $\widehat{\mathbf{p}}(\mathbf{t})$ lie on the plane through $\widehat{\mathbf{p}}(\mathbf{a})$, $\widehat{\mathbf{p}}(\mathbf{b})$, and $\widehat{\mathbf{p}}(\mathbf{c})$.

Total Degree Patches: Quadratic





This slide shows the procedure we follow to determine a point on a quadratic curve or patch.

In the case of a curve, the given input comprises the Bézier points

$$\widehat{\mathbf{p}}(0,0) \quad \widehat{\mathbf{p}}(0,1) \quad \widehat{\mathbf{p}}(1,1),$$

marked as solid disks. We find

$$\widehat{\mathbf{p}}(0,t) \qquad \widehat{\mathbf{p}}(t,1)$$

in the first level of interpolations, marked with dotted lines and gray disks; the second level, marked with solid lines and hollow circles, yields $\mathbf{p}(t) = \widehat{\mathbf{p}}(t,t)$. We used t as the weight of all these linear interpolations.

Using the same visualization key, we have drawn the corresponding interpolations for determining a point on a quadratic patch. The input consists of the Bézier points

$$\begin{array}{cc} \hat{\mathbf{p}}(\mathbf{a},\mathbf{a}) & \hat{\mathbf{p}}(\mathbf{a},\mathbf{b}), \\ \hat{\mathbf{p}}(\mathbf{b},\mathbf{b}) & \hat{\mathbf{p}}(\mathbf{b},\mathbf{c}), \\ \hat{\mathbf{p}}(\mathbf{c},\mathbf{c}) & \hat{\mathbf{p}}(\mathbf{a},\mathbf{c}), \end{array}$$

forming the *control net* of the patch.

In the first level of interpolations, we find

1. $\hat{\mathbf{p}}(\mathbf{a},\mathbf{t})$ as a barycentric combination of
$$\hat{\mathbf{p}}(\mathbf{a},\mathbf{a}) \quad \hat{\mathbf{p}}(\mathbf{a},\mathbf{b}) \quad \hat{\mathbf{p}}(\mathbf{a},\mathbf{c}),$$
2. $\hat{\mathbf{p}}(\mathbf{b},\mathbf{t})$ as a barycentric combination of
$$\hat{\mathbf{p}}(\mathbf{a},\mathbf{b}) \quad \hat{\mathbf{p}}(\mathbf{b},\mathbf{b}) \quad \hat{\mathbf{p}}(\mathbf{b},\mathbf{c}),$$
3. $\hat{\mathbf{p}}(\mathbf{c},\mathbf{t})$ as a barycentric combination of
$$\hat{\mathbf{p}}(\mathbf{a},\mathbf{c}) \quad \hat{\mathbf{p}}(\mathbf{b},\mathbf{c}) \quad \hat{\mathbf{p}}(\mathbf{c},\mathbf{c}).$$

In the second level, we find $\mathbf{p}(\mathbf{t}) = \hat{\mathbf{p}}(\mathbf{t},\mathbf{t})$ as a barycentric combination of

$$\hat{\mathbf{p}}(\mathbf{a},\mathbf{t}) \quad \hat{\mathbf{p}}(\mathbf{b},\mathbf{t}) \quad \hat{\mathbf{p}}(\mathbf{c},\mathbf{t}).$$

All these barycentric combinations were calculated using the barycentric coordinates $1-u-v$, u and v. In the drawing, we used different shades of gray to shade distinct planes on which barycentric combinations were computed.

Total Degree Patches: Polar Form

Polar form of patch $\mathbf{p}(\mathbf{t})$ is symmetric, multiaffine $\hat{\mathbf{p}}(\mathbf{t}_1, \mathbf{t}_2)$ with

$$\hat{\mathbf{p}}(\mathbf{t}, \mathbf{t}) = \mathbf{p}(\mathbf{t}).$$

$\hat{\mathbf{p}}(\mathbf{t}_1, \mathbf{t}_2)$ multiaffine means that

- if all polar arguments but one are fixed, then,
- as the single varying argument moves around on the domain, the resulting polar values lie on a 3D plane.

We are assuming quadratic patches here, in order to make our discussion simpler. However, very similar definitions apply to higher degree patches.

As in the case of curves, non-diagonal polar values are found by performing the two levels of interpolations using different parameters $\mathbf{t}_1$ and $\mathbf{t}_2$. That is, in the first level, we use the barycentric coordinates of $\mathbf{t}_1$ with respect to points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$; in the second level, we use the barycentric coordinates of $\mathbf{t}_2$ with respect to the same points $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$.

In the previous diagrams, we have used polar values to label all points.

Total Degree Patches: Domain Lines

Image of arbitrary domain line

$$\mathbf{t} = (1-w)\mathbf{d} + w\mathbf{e}$$

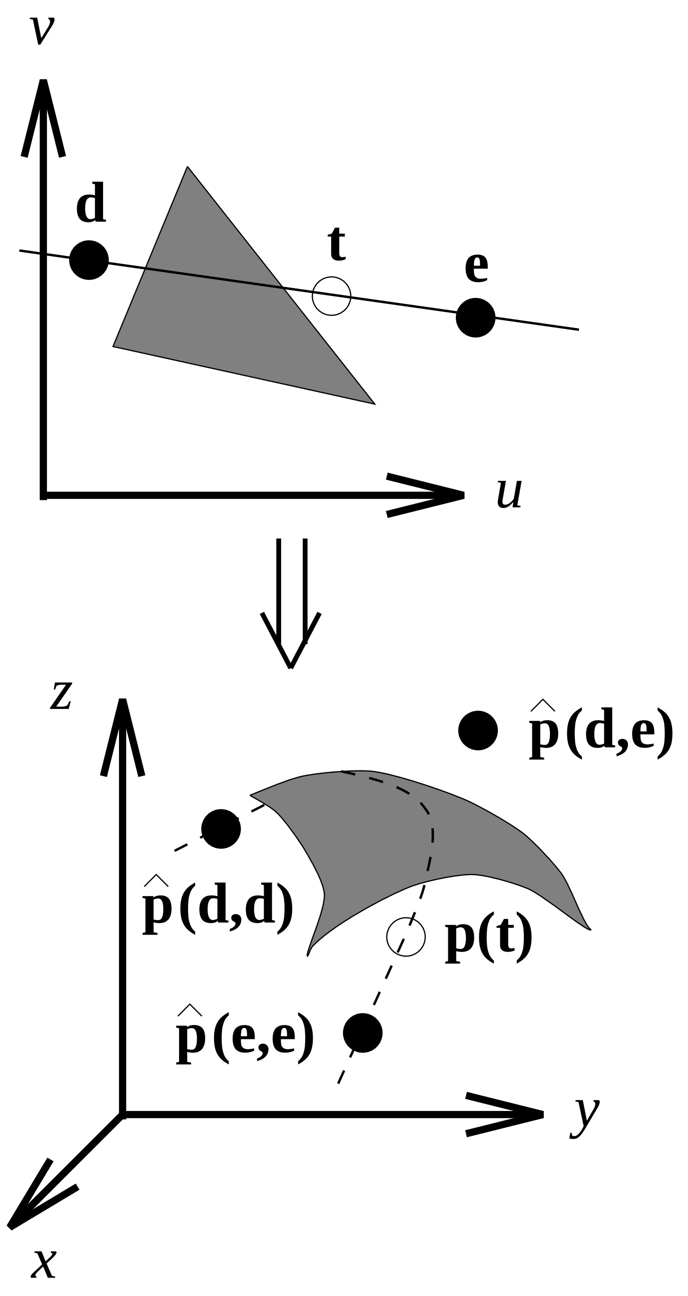
is

$$\begin{aligned}
\mathbf{p}(\mathbf{t}) &= \hat{\mathbf{p}}(\mathbf{t}, \mathbf{t}) \\
&= \hat{\mathbf{p}}((1-w)\mathbf{d} + w\mathbf{e}, (1-w)\mathbf{d} + w\mathbf{e}) \\
&= (1-w)\hat{\mathbf{p}}(\mathbf{d}, (1-w)\mathbf{d} + w\mathbf{e}) \\
&\quad + w\hat{\mathbf{p}}(\mathbf{e}, (1-w)\mathbf{d} + w\mathbf{e}) \\
&= (1-w)^2\hat{\mathbf{p}}(\mathbf{d}, \mathbf{d}) \\
&\quad + 2(1-w)w\hat{\mathbf{p}}(\mathbf{d}, \mathbf{e}) \\
&\quad + w^2\hat{\mathbf{p}}(\mathbf{e}, \mathbf{e}),
\end{aligned}$$

which is quadratic curve with Bézier points

$$\hat{\mathbf{p}}(\mathbf{d}, \mathbf{d}) \quad \hat{\mathbf{p}}(\mathbf{d}, \mathbf{e}) \quad \hat{\mathbf{p}}(\mathbf{e}, \mathbf{e}).$$

Total Degree Patches: Domain Lines



domain
line

embedded
quadratic curve

In this derivation, we took advantage of the symmetric and multiaffine character of polar forms. For example, symmetry was used to freely swap the arguments of the polar form in the fourth step of the derivation of $\mathbf{p}(\mathbf{t})$: in part, this step follows from the third by combining the terms $(1-w)w\widehat{\mathbf{p}}(\mathbf{d},\mathbf{e})$ and $w(1-w)\widehat{\mathbf{p}}(\mathbf{e},\mathbf{d})$ into

$$2(1-w)w\widehat{\mathbf{p}}(\mathbf{d},\mathbf{e}).$$

It is easy to see that the coefficients of the Bézier points are, as should be for a Bézier curve, the Bernstein polynomials for quadratics, namely

$$B_1(w) = (1-w)^2 \quad B_2(w) = 2(1-w)w \quad B_3(w) = w^2;$$

simply observe that, as we stated in a past lecture, the Bernstein polynomials are the summands in the expansion of

$$[(1-w)+w]^2.$$

This example illustrates the superiority of total degree patches compared to NURBS patches in terms of arbitrary domain lines: unlike in the case of NURBS, such lines map to curves of the same degree as the boundary curves of the total degree patch; moreover, the control points of the embedded curve are found trivially, by inspection.

Total Degree Patches: Remarks

Properties follow from Bézier curves:

- Affine invariance.
- Convex hull.
- And more...

Algorithms derived by writing curve algorithms in polar form and applying them to polar form of patch:

- Degree elevation.
- Subdivision.
- And more...

For example, the convex hull property holds: a total degree patch is fully contained in the convex hull of its control net (i.e. the set of its Bézier points).

Total Degree Splines

Join patches with some parametric continuity along patch boundaries. Continuity guaranteed by following theorem:

Two patches $\mathbf{p}(\mathbf{t})$ and $\mathbf{q}(\mathbf{t})$ join with C^n continuity along their boundary $\mathbf{ab}$ if and only if their polar forms agree on all sequences of polar arguments that include

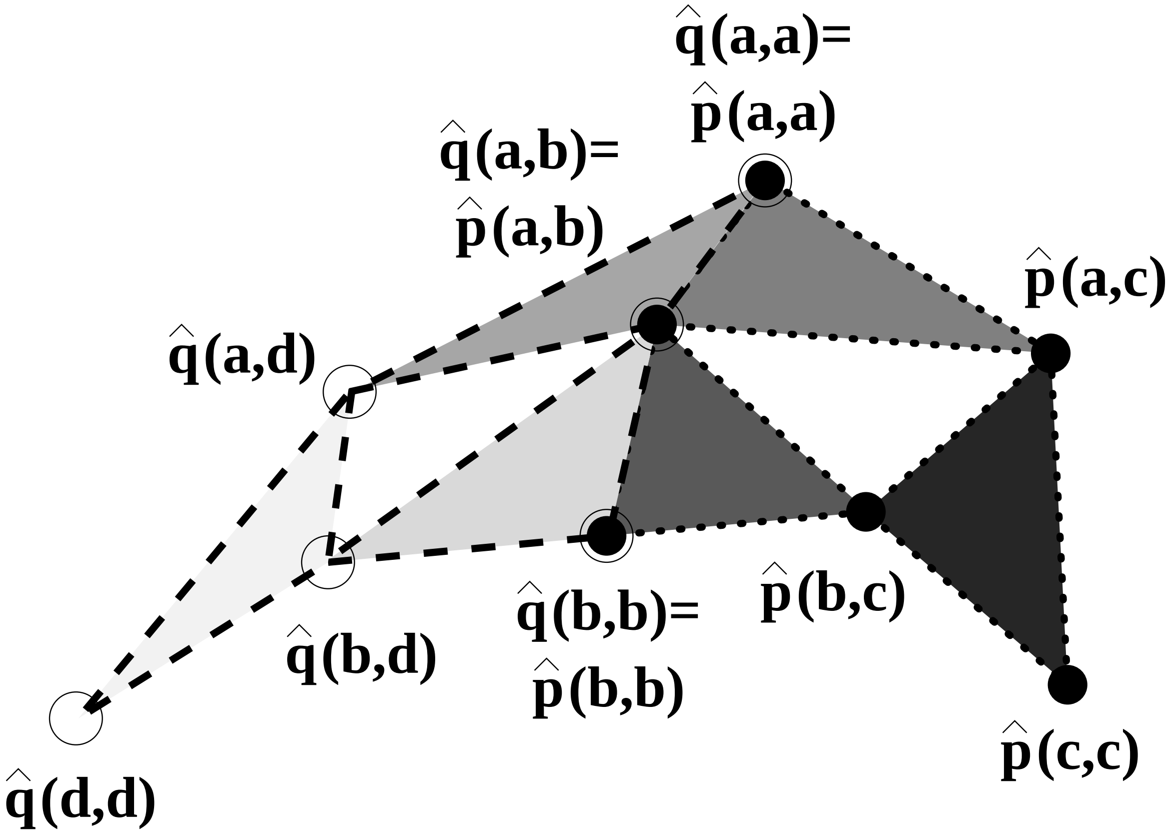
- *at most* n arguments different from...
- all combinations of $\mathbf{a}$ and $\mathbf{b}$, which comprise the rest of the arguments.

Total degree splines are also called *triangular splines*, or *true surfaces*.

Total Degree Splines: Continuity

C^0 requires

$$\begin{aligned}\hat{p}(a,a) &= \hat{q}(a,a)\\ \hat{p}(a,b) &= \hat{q}(a,b)\\ \hat{p}(b,b) &= \hat{q}(b,b).\end{aligned}$$



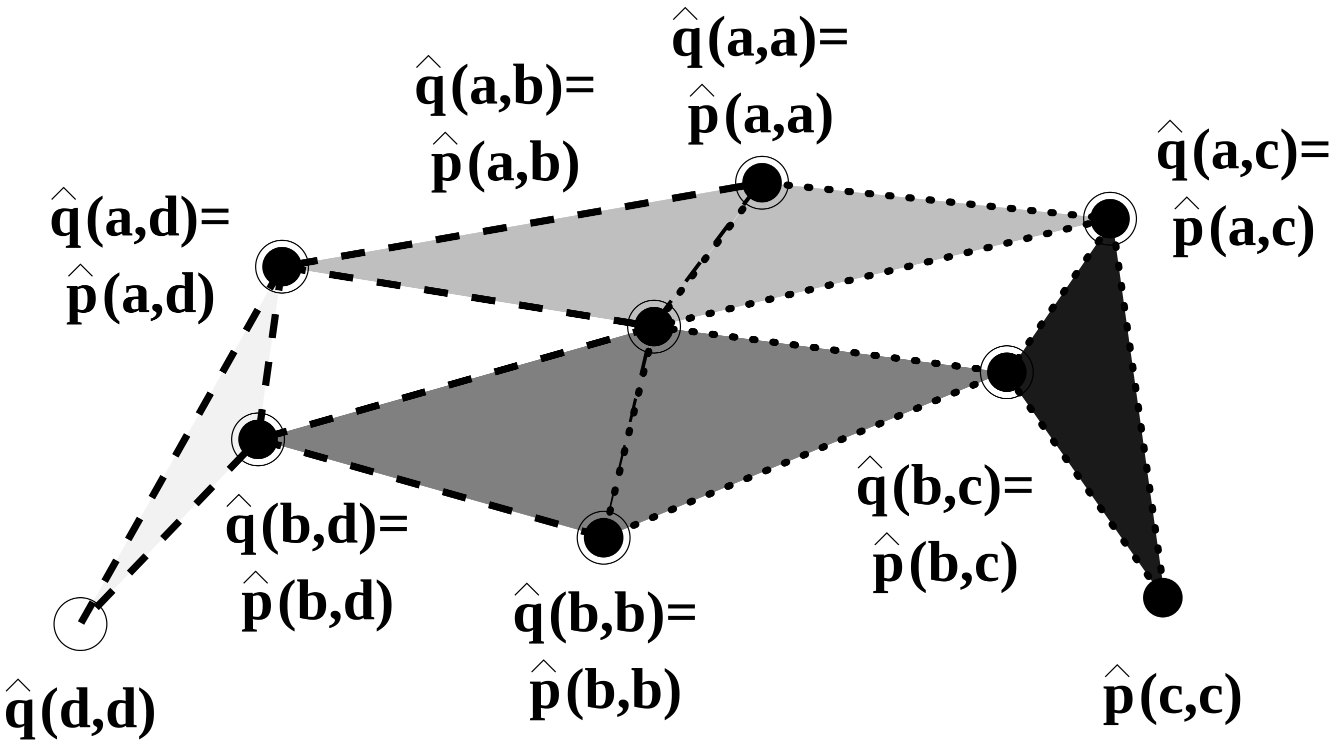
If $n = 0$, the theorem simplifies to the following statement: two patches $\mathbf{p(t)}$ and $\mathbf{q(t)}$ join with C^0 continuity along their boundary $\mathbf{ab}$ if and only if their polar forms agree on all sequences of polar arguments that include all combinations of $\mathbf{a}$ and $\mathbf{b}$.

In this drawing, as well as in the next two, we have shaded distinct planes with different shades of gray. Also, the polar values of the patch $\mathbf{q(t)}$ are drawn as hollow circles, while polar values of $\mathbf{p(t)}$ are drawn as solid disks. Finally, planes on which we did polar interpolations for $\mathbf{q(t)}$ are outlined with dashed line style, while interpolation planes for $\mathbf{p(t)}$ are outlined with dotted line style.

Total Degree Splines: Continuity

C^1 requires that, for all t,

$$\hat{\mathrm{p}}(\mathrm{t},\mathrm{a}) = \hat{\mathrm{q}}(\mathrm{t},\mathrm{a}) \qquad \hat{\mathrm{p}}(\mathrm{t},\mathrm{b}) = \hat{\mathrm{q}}(\mathrm{t},\mathrm{b}).$$



In this drawing, we calculate $\widehat{\mathbf{q}}(\mathbf{a}, \mathbf{c})$ and $\widehat{\mathbf{q}}(\mathbf{b}, \mathbf{c})$ and observe that they overlap respectively $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{c})$ and $\widehat{\mathbf{p}}(\mathbf{b}, \mathbf{c})$. In a similar fashion as in the case of curves, testing just these two equalities, and assuming C^0 continuity, guarantees C^1 continuity.

Note that, alternatively, we can calculate $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{d})$ and $\widehat{\mathbf{p}}(\mathbf{b}, \mathbf{d})$ and observe that they overlap respectively $\widehat{\mathbf{q}}(\mathbf{a}, \mathbf{d})$ and $\widehat{\mathbf{q}}(\mathbf{b}, \mathbf{d})$.

In the previous example, the drawn patches did not meet with C^1 continuity, since, for example, $\widehat{\mathbf{q}}(\mathbf{a}, \mathbf{c})$ does not even lie on the same plane as $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{c})$, $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{b})$, and $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{a})$. Just as in the case of curves, where collinearity is necessary but not sufficient, in the case of patches, even if

- $\widehat{\mathbf{q}}(\mathbf{a}, \mathbf{c})$ lied on the same plane as $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{c})$, $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{b})$, and $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{a})$, and

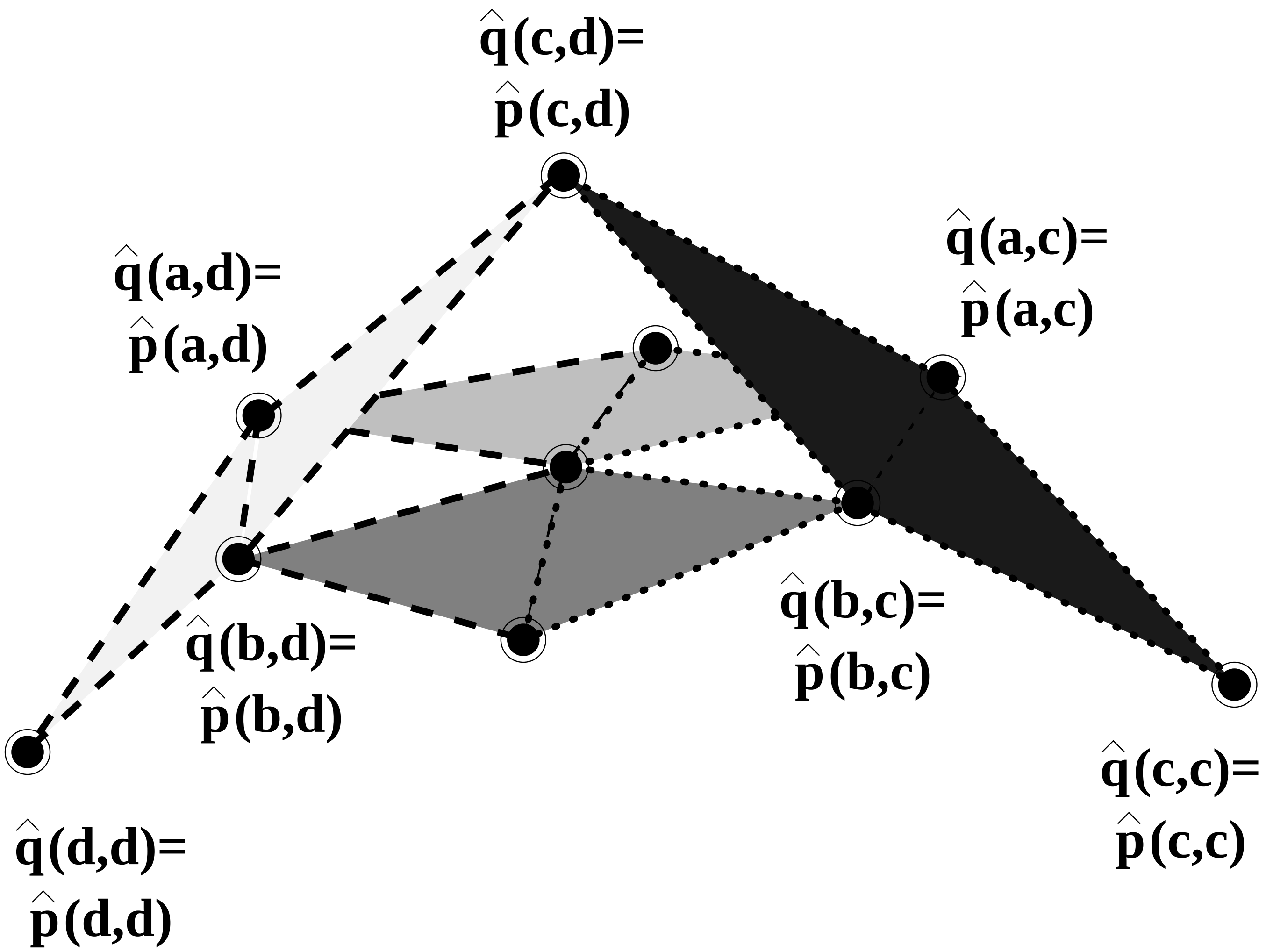
- $\widehat{\mathbf{q}}(\mathbf{b}, \mathbf{c})$ lied on the same plane as $\widehat{\mathbf{p}}(\mathbf{b}, \mathbf{c})$, $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{b})$, and $\widehat{\mathbf{p}}(\mathbf{b}, \mathbf{b})$,

this fact would not guarantee C^1 continuity: $\widehat{\mathbf{q}}(\mathbf{a}, \mathbf{c})$ and $\widehat{\mathbf{q}}(\mathbf{b}, \mathbf{c})$ must *overlap* respectively $\widehat{\mathbf{p}}(\mathbf{a}, \mathbf{c})$ and $\widehat{\mathbf{p}}(\mathbf{b}, \mathbf{c})$. Also, unlike the case of curves, the above condition cannot even guarantee geometric continuity (G^1).

Total Degree Splines: Continuity

C^2 requires that, for all t_1, t_2,

$$\hat{\mathrm{p}}(\mathrm{t}_1, \mathrm{t}_2) = \hat{\mathrm{q}}(\mathrm{t}_1, \mathrm{t}_2).$$



In this drawing, we calculate $\hat{\mathbf{q}}(\mathbf{c}, \mathbf{d})$ and $\hat{\mathbf{p}}(\mathbf{c}, \mathbf{d})$, and observe that they overlap. In a similar fashion as in the case of curves, testing just this equality, as assuming C^1 continuity, guarantees C^2 continuity.

To prevent cluttering the diagram, we have not explicitly labeled some of the polar values. As the unlabeled points did not move, consult the previous drawings to find their associated polar arguments.

Of course, C^2 continuity between quadratic patches means that the two patches are just different pieces of the *same* quadratic surface.

Total Degree Splines: Remarks

Trivially generalizing B-splines, can define total degree splines from de Boor points. Most properties (e.g. convex hull) and algorithms (e.g. knot insertion) generalize very easily.

If rational splines desired for quadrics, just define total degree patches with weights!

Wavelets: Function Space

A subset of all possible functions. Examples:

- All polynomials.
- All planar cubic curves.
- All grayscale images of size 100×100.
- All 1D "images" with 4 pixels...

1D image with n pixels is $I(x)$ where

- $x \in [0, 1)$, and
- $I(x) \in \Re$ is piecewise constant. $(i+1)$st *pixel* covers $x \in \left[\frac{i}{n}, \frac{i+1}{n}\right)$, $i = 0, \ldots, n-1$.

In the ensuing presentation, we'll use the notation that Stollnitz uses in his wavelet tutorial, reproduced in the course reader. While there is a possibility that the non-standard notation will confuse you, using it will make it easier for you to follow the paper when you read it. And I strongly suggest that you read it — wavelets are a hot topic these days, and this tutorial is one of the best I have come across.

A peculiar feature of this notation is the dual use of superscripts. Most of the time, they play the role of qualifiers, much like subscripts. Sometimes, however, they are used to denote exponentiation, especially when they are applied to constants. The context usually allows the reader to distinguish between the two uses.

Our presentation of wavelets will be highly focused on 1D images. In their context, we will introduce wavelet notation, discuss theorems, and present examples. Most of our discussion generalizes to other function spaces, general forms of wavelets, as well as the whole field of multiresolution analysis. We will only mention such generalizations in the very end, when we address applications of wavelets; until then, although we won't repeat this, keep in mind that our presentation sets the basis for a much broader investigation of wavelets, far beyond the context of 1D images.

A function space, precisely defined, is more than just any subset. It is a subset, along with some operations defined on it, like addition; also, the sum of any two elements in a function space should be a member of the function space, too. For such tedious details, you may consult a linear algebra primer.

A 100×100 grayscale image is a map between the set containing pairs of integers between 1 and 100 — each pair corresponding to a pixel — and the subset $[0, 1]$ of the real line, denoting pixel intensity.

To illustrate 1D images, consider the case where $n = 2$. In this case, i is 0 or 1. The first pixel of the image, corresponding to $i = 0$, covers

$$x \in \left[\frac{0}{2}, \frac{1}{2}\right) = \left[0, \frac{1}{2}\right),$$

while the second pixel, corresponding to $i = 1$, covers

$$x \in \left[\frac{1}{2}, \frac{2}{2}\right) = \left[\frac{1}{2}, 1\right).$$

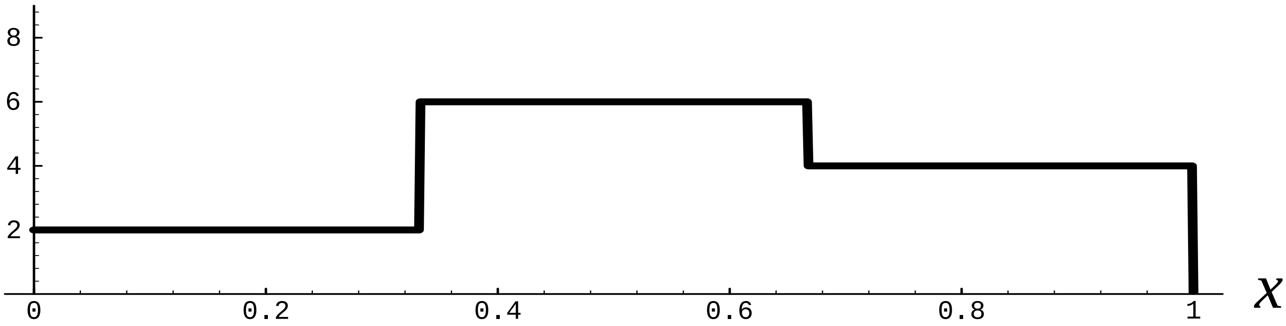
In general, since

$$\frac{i+1}{n} - \frac{i}{n} = \frac{1}{n}$$

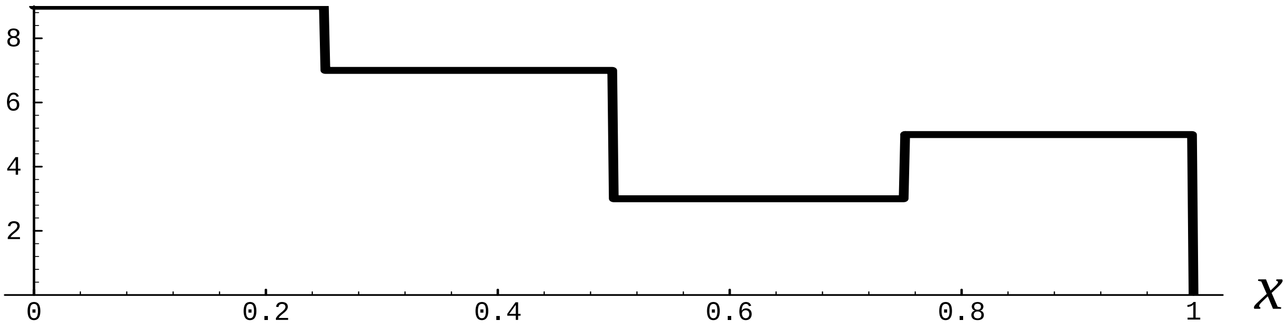
for all i, an n-pixel image contains n adjacent pixels, each of width $1/n$.

Wavelets: 1D Images

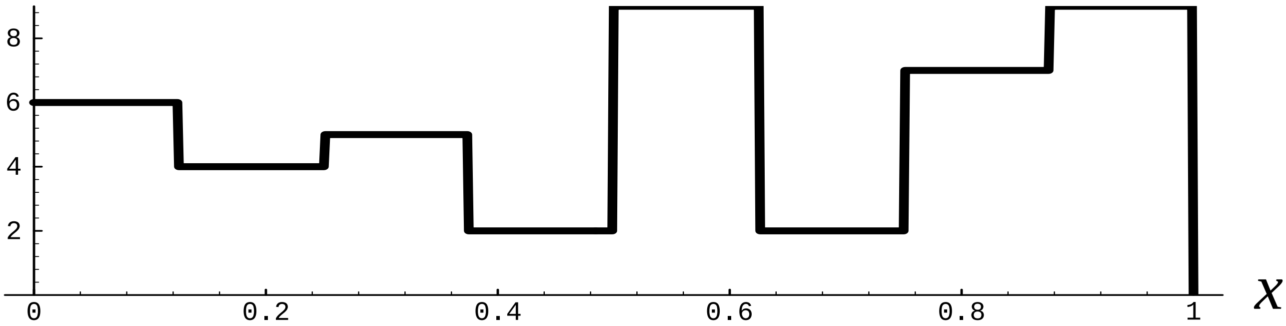
3-pixel image



4-pixel image



8-pixel image



Wavelets: Nested Spaces

A sequence of function spaces V^j where one is contained in the next:

$$V^0 \subset V^1 \subset \cdots$$

Can see elements of V^j as elements of V^{j+1}.

Examples: V^j contains

- curves of degree j, or
- splines with j knots, or
- 1D images with $n = 2^j$ pixels.

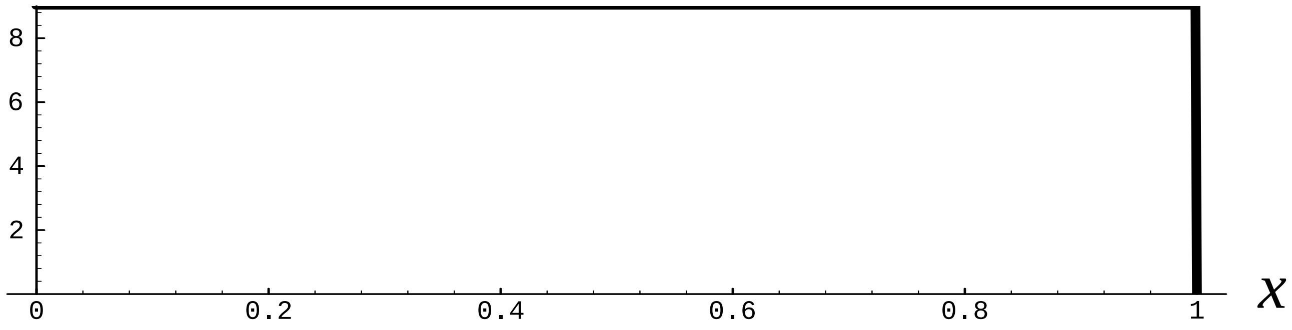
Counterexample: V^j contains 1D images with j pixels.

If V^j contains 1D images with 2^j pixels, then we can view an image in V^j as an image in V^{j+1}: just split each pixel of the image (perceived as an element of V^j) in two!

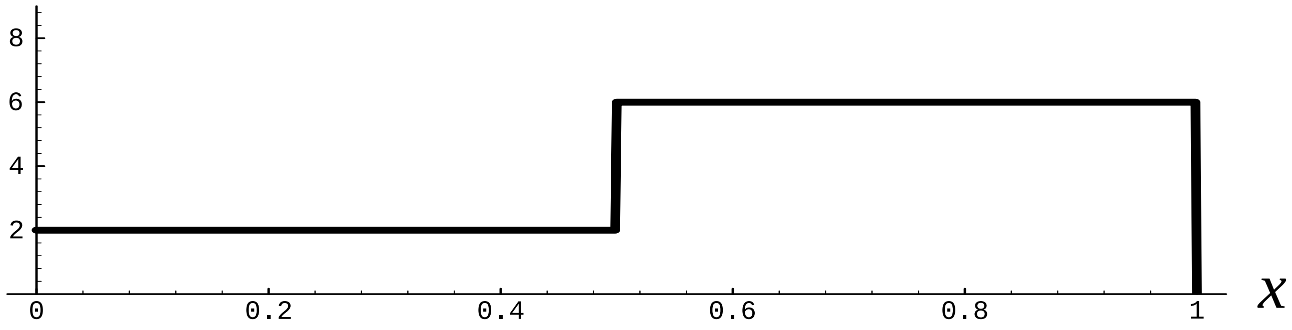
If, instead, V^j contains 1D images with j pixels, then consider $j = 2$ to see why containment fails: an arbitrary 2-pixel image cannot be viewed as a 3-pixel image since the pixel boundaries of 2-pixel images fall *within* the pixels of 3-pixel images.

Wavelets: Nested 1D Images

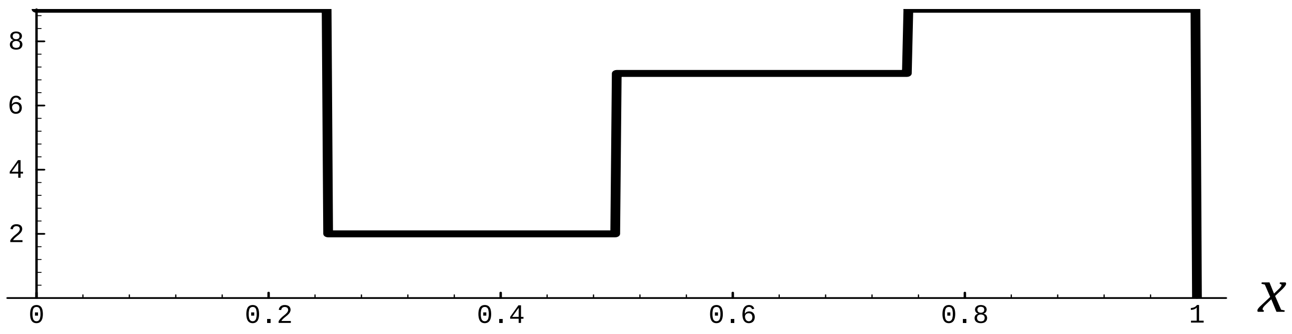
1-pixel image



2-pixel image



4-pixel image



Scaling Functions

A *basis* is a minimal collection of members $\phi_0(x), \ldots, \phi_m(t)$ of a function space which can produce all other members as linear combinations:

$$I(x) = c_0\phi_0(x) + \cdots + c_m\phi_m(x).$$

Examples:

- Cubics: $m = 3$, $\phi_i(x) = x^i$.

- 1D images with $n = 2^j$ pixels: $m = n - 1$,

  $$\phi_i^j(x) = \phi(nx - i),$$

  where

  $$\phi(x) = \begin{cases} 1 & \text{for } x \in [0, 1) \\ 0 & \text{otherwise.} \end{cases}$$

  $\phi_i^j(x)$ is *scaling function*: box with pixel $i + 1$ as *support*.

By a *minimal* collection we mean that we have just as many functions in the basis as we need, and no more. This requirement can be stated in two other, equivalent, ways:

- $\phi_0(x), \ldots, \phi_m(x)$ are linearly independent; that is, none of them can be expressed as a linear combination of the others.

- Every $I(x)$ has one and only one set of coefficients; that is, there is a one-to-one correspondence between images $I(x)$ and value assignments for the c_i's.

The support of a function is the set of x's for which the function is non-zero.

A way to visualize $\phi_i^j(x)$ is to observe that $\phi(x)$ is a box with unit height, and non-zero for $x \in [0, 1)$. Hence, $\phi_i^j(x)$ is also a box of unit height but it is non-zero for $nx - i \in [0, 1)$, which is equivalent to $x \in \left[\frac{i}{n}, \frac{i+1}{n}\right)$, i.e. the $(i + 1)$st pixel.

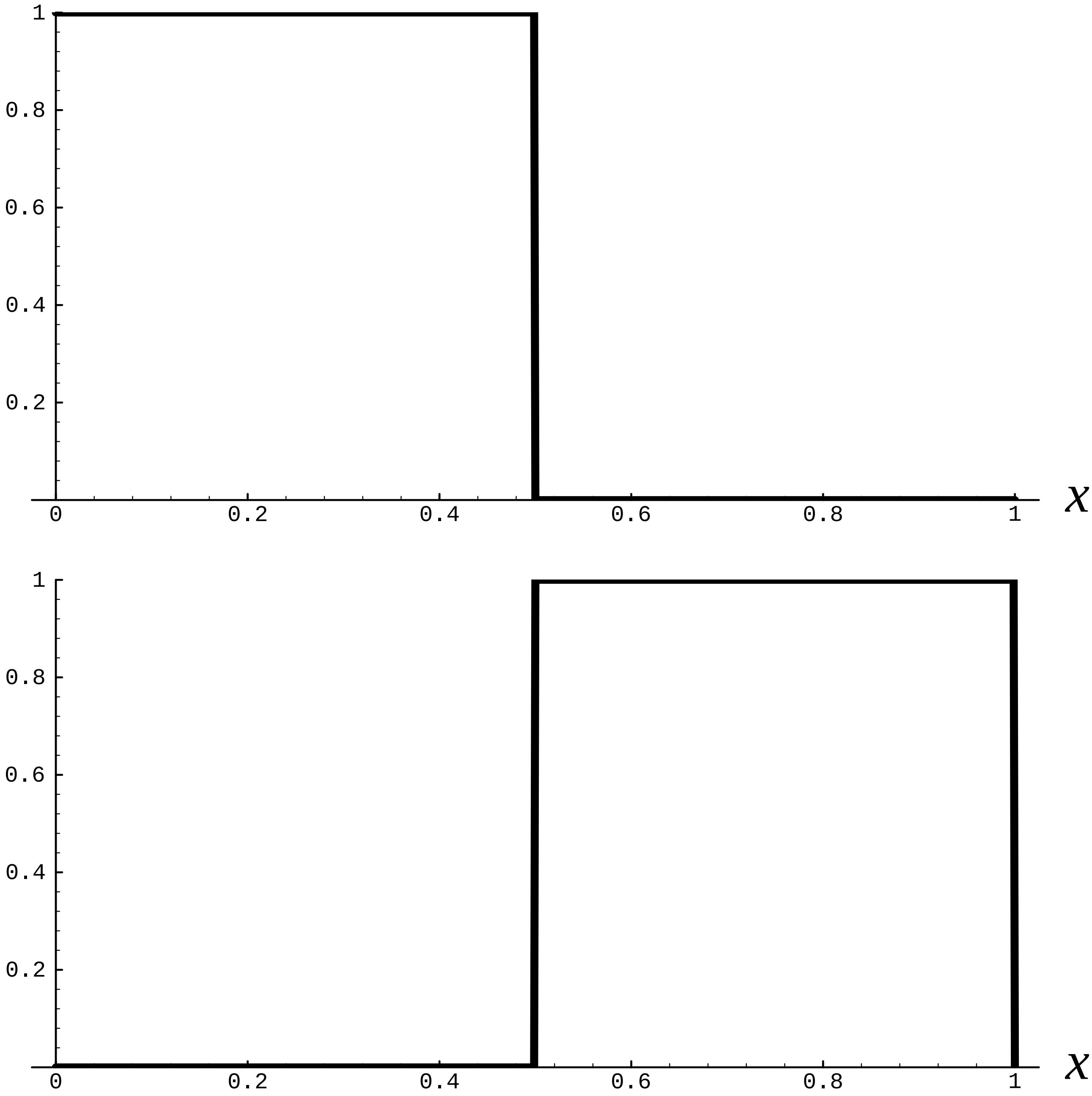
Since we will be using these functions quite a bit, it is important that you become familiar with the notation:

- The superscript j specifies the relevant image size. That is, $\phi_i^j(x)$ is one of the basis functions of the function space V^j of 2^j-pixel images.

- The subscript i specifies the relevant pixel of a 2^j-pixel image. That is, as we saw above, $\phi_i^j(x)$ has support the pixel $i + 1$. Note that the subscript varies from 0 to $m = 2^j - 1$, and thus can exceed the superscript j.

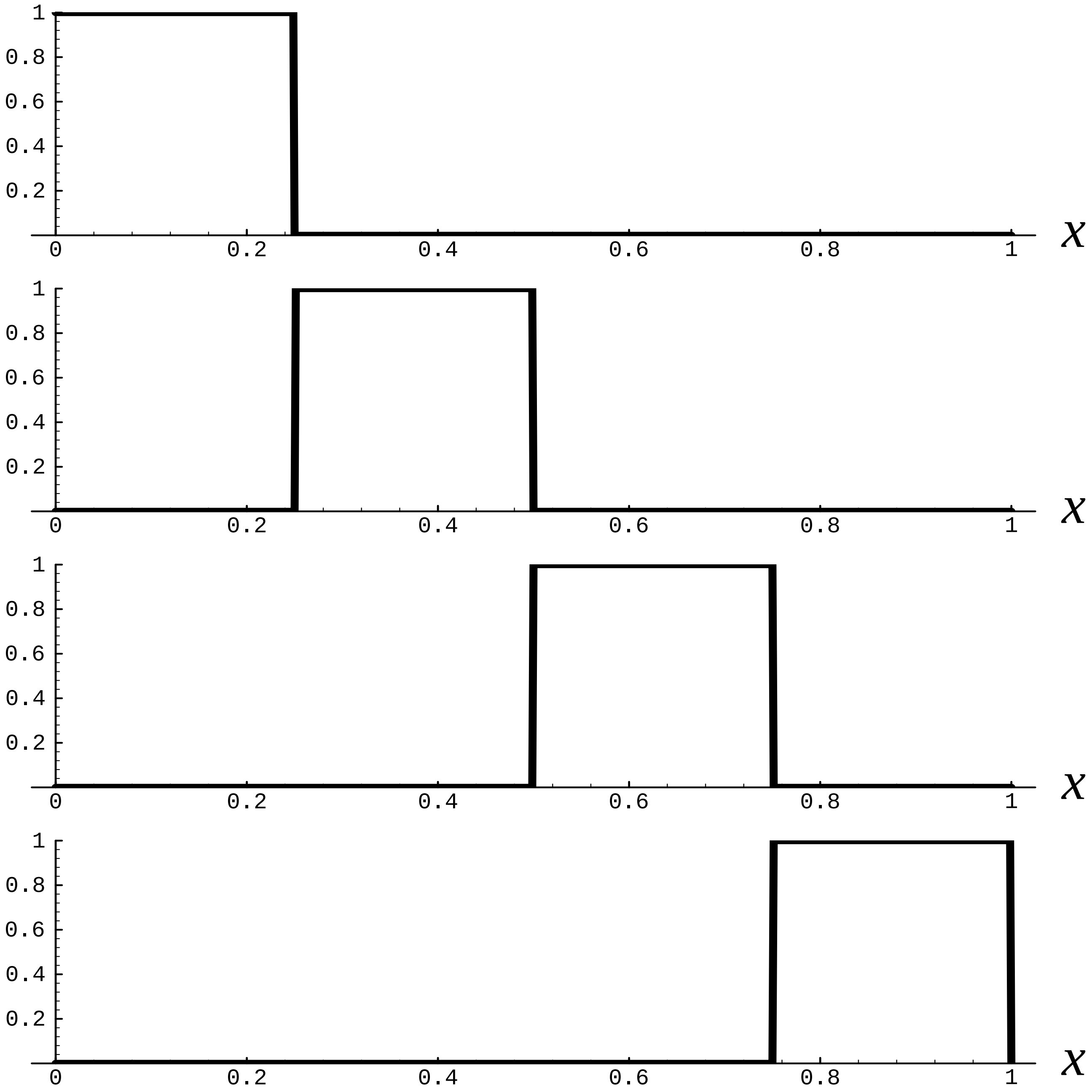
From now on, we will always use n for the number of pixels of an image in V^j, i.e. a 2^j-pixel image. Also, m will always stand for $n - 1$, i.e. the subscript of the last scaling function in the basis of V^j.

Scaling Functions: $n = 2$



$\phi_0^1(x)$ and $\phi_1^1(x)$, the scaling functions of 2-pixel images, are plotted in this order, from top to bottom.

Scaling Functions: $n = 4$



$\phi_0^2(x)$, $\phi_1^2(x)$, $\phi_2^2(x)$, and $\phi_3^2(x)$, the scaling functions of 4-pixel images, are plotted in this order, from top to bottom.

Scaling Functions: Matrix Notation

Let

$$
\begin{aligned}
\Phi^j &= \begin{bmatrix} \phi_0^j(x) & \dots & \phi_m^j(x) \end{bmatrix} \\
C^j &= \begin{bmatrix} c_0^j \\ \vdots \\ c_m^j \end{bmatrix}.
\end{aligned}
$$

Then

$$
\begin{aligned}
I(x) &= c_0^j \phi_0^j(x) + \cdots + c_m^j \phi_m^j(x) \\
&= \Phi^j C^j,
\end{aligned}
$$

where $I(x) \in V^j$.

As a last reminder, V^j is the set of 2^j-pixel images. while $m = 2^j - 1$ is one less than the pixel count of any image $I(x)$ in V^j.

Unlike our customary notation, Stollnitz uses brackets to denote matrices. As we stated earlier, we will employ his notation in today's presentation.

Scaling Functions: Refinement

For all $i = 0, \ldots, m$,

$$\phi_i^j(x) = \phi_{2i}^{j+1}(x) + \phi_{2i+1}^{j+1}(x),$$

since

- support of $\phi_{2i}^{j+1}(x)$ is pixel $2i+1$ of 2^{j+1}-pixel image,

- support of $\phi_{2i+1}^{j+1}(x)$ is pixel $2i+2$ of 2^{j+1}-pixel image, and

- pixels $2i+1$ and $2i+2$ of 2^{j+1}-pixel image correspond to left and right halves of pixel $i+1$ of 2^j-pixel image.

Scaling functions are *refinable*.

Scaling Functions: Synthesis

For $j = 1$ and $i = 0, 1$,

$$\begin{aligned}\phi_0^1(x) &= \phi_0^2(x) + \phi_1^2(x) \\ \phi_1^1(x) &= \phi_2^2(x) + \phi_3^2(x),\end{aligned}$$

which imply that

$$\begin{aligned}\mathbf{\Phi}^1 &= \left[\begin{array}{cc} \phi_0^1(x) & \phi_1^1(x) \end{array}\right] \\ &= \left[\begin{array}{cc} \phi_0^2(x) + \phi_1^2(x) & \phi_2^2(x) + \phi_3^2(x) \end{array}\right] \\ &= \left[\begin{array}{cccc} \phi_0^2(x) & \phi_1^2(x) & \phi_2^2(x) & \phi_3^2(x) \end{array}\right] P^2 \\ &= \mathbf{\Phi}^2 P^2,\end{aligned}$$

where

$$P^2 = \left[\begin{array}{cc} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{array}\right].$$

In this example, we are expressing the scaling functions of 2-pixel images (since the superscript is 1 on the left-hand side) in terms of the scaling functions of 4-pixel images (since the superscript is 2 on the right-hand side).

Scaling Functions: Synthesis

In general,

$$\Phi^{j-1} = \Phi^j P^j,$$

where *synthesis matrix* is

$$P^j = \begin{bmatrix} 1 & 0 & & 0 \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & 1 & & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & & 1 \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

with dimensions $2^j \times 2^{j-1}$.

In words, we have expressed the scaling functions of 2^{j-1}-pixel images in terms of the scaling functions of 2^j-pixel images.

Our previous slide proved the special case of this formula for $j = 2$.

The dimensions of the matrix P^j follow from the sizes of the row vectors Φ^j and Φ^{j-1}. For example, Φ^j has size 2^j since it contains the 2^j scaling functions which form the basis of V^j, the function space of 2^j-pixel images.

Haar Wavelets

Define

$$\psi_i^j(x) = \phi_{2i}^{j+1}(x) - \phi_{2i+1}^{j+1}(x)$$

as 1D *Haar wavelet*. Properties:

- Support is pixel $i+1$ of 2^j-pixel image.
- $\psi_i^j(x) = 1$ for left half of support.
- $\psi_i^j(x) = -1$ for right half of support.

The properties of the Haar wavelets listed above follow from the correspondence between pixels in 2^j-pixel images and 2^{j+1}-pixel images. This correspondence was investigated earlier when the refinement of the scaling functions was covered.

As in the case of scaling functions, we will be using wavelets a lot, so it is important that you become familiar with the notation:

- The superscript j specifies the relevant image size. That is, $\psi_i^j(x)$ is a wavelet associated with 2^j-pixel images.

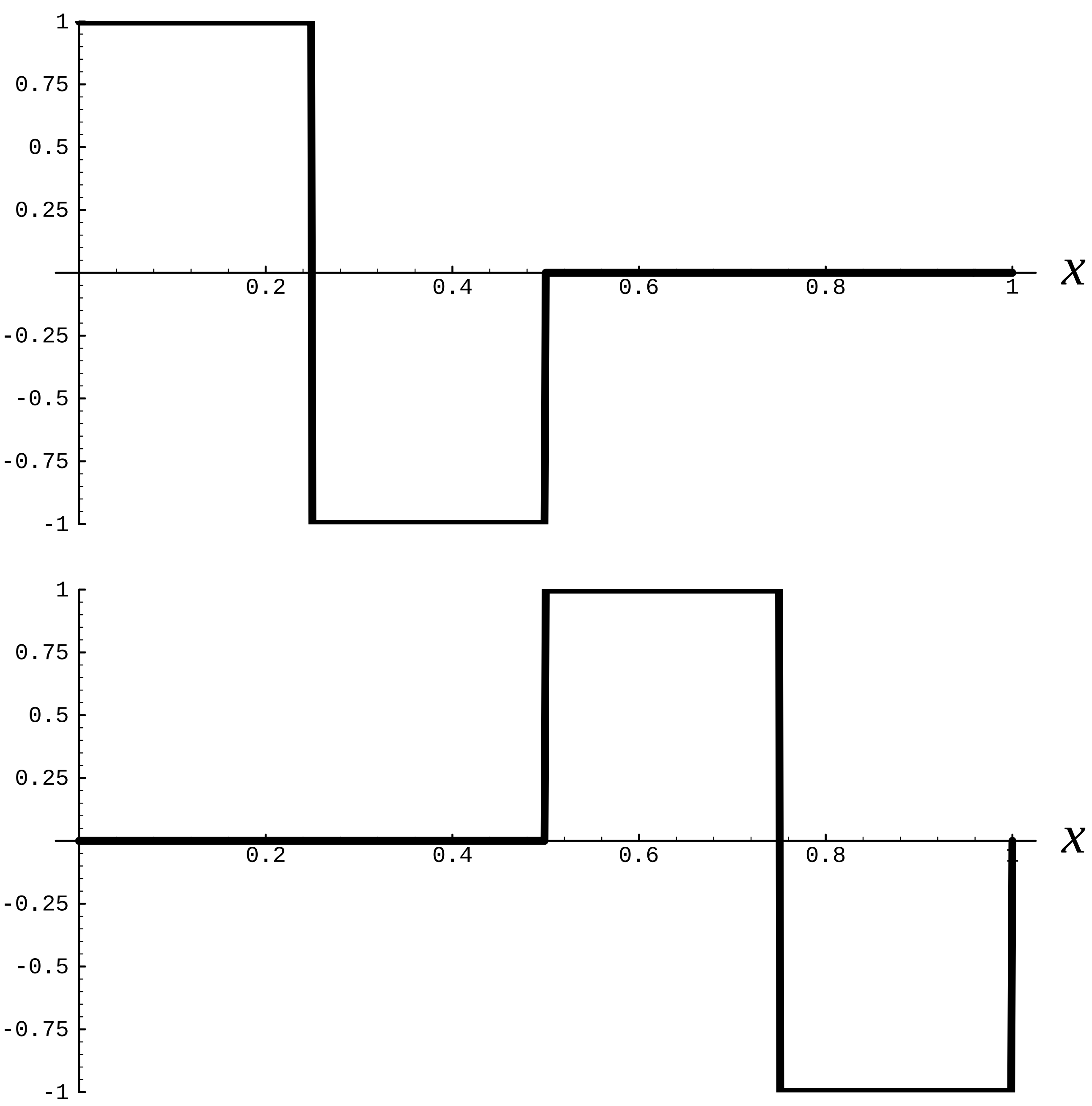
- The subscript i specifies the relevant pixel of the 2^j-pixel image. That is, as we saw above, $\psi_i^j(x)$ has support the pixel $i+1$.

However, do not let the similarity of notation between $\phi_i^j(x)$ and $\psi_i^j(x)$ confuse you into believing that

$$\psi_0^j(x), \ldots, \psi_m^j(x)$$

is a basis of V^j; it is not! We will relate wavelets and bases shortly.

Haar Wavelets: $n = 2$



$\psi_0^1(x)$ and $\psi_1^1(x)$, the wavelets for 2-pixel images, are plotted in this order, from top to bottom.

Haar Wavelets: Generation

Alternate definition of $\psi_i^j(x)$:

$$\psi_i^j(x) = \psi(nx - i),$$

where

$$\psi(x) = \begin{cases} 1 & \text{for } x \in [0, \frac{1}{2}) \\ -1 & \text{for } x \in [\frac{1}{2}, 1) \\ 0 & \text{otherwise.} \end{cases}$$

$\psi(x)$ is *mother wavelet*. $\psi_i^j(x)$ are *translations and dilations* of $\psi(x)$.

A "dilation" of a function is the result of stretching the function along the horizontal axis.

The property shown on the slide follows by analogy to the definition of the scaling functions. $\psi_i^j(x)$ and $\phi_i^j(x)$ are very similar, differing only in that the former is a flipped version of the latter for half of its support. Hence, the definition of $\psi(x)$ contains the lines

$$\begin{array}{ll} 1 & \text{for } x \in [0, \frac{1}{2}) \\ -1 & \text{for } x \in [\frac{1}{2}, 1), \end{array}$$

in place of

$$1 \quad \text{for } x \in [0, 1),$$

which appeared in the definition of $\phi(x)$.

Haar Wavelets: Synthesis

Let

$$\Psi^j = \left[\begin{array}{ccc} \psi_0^j(x) & \ldots & \psi_m^j(x) \end{array} \right].$$

Then

$$\begin{aligned} \Psi^1 &= \left[\begin{array}{cc} \psi_0^1(x) & \psi_1^1(x) \end{array} \right] \\ &= \left[\begin{array}{cc} \phi_0^2(x) - \phi_1^2(x) & \phi_2^2(x) - \phi_3^2(x) \end{array} \right] \\ &= \left[\begin{array}{cccc} \phi_0^2(x) & \phi_1^2(x) & \phi_2^2(x) & \phi_3^2(x) \end{array} \right] Q^2 \\ &= \Phi^2 Q^2, \end{aligned}$$

where

$$Q^2 = \left[\begin{array}{cc} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{array} \right].$$

In this example, we set $j = 1$ in the first formula, and thus we investigated the wavelets of 2-pixel images. On the right-hand side, we have used the scaling functions of 4-pixel images, as the superscript 2 indicates.

Haar Wavelets: Synthesis

In general,

$$\Psi^{j-1} = \Phi^j Q^j,$$

where synthesis matrix is

$$Q^j = \begin{bmatrix} 1 & 0 & & 0 \\ -1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & -1 & & 0 \\ & \vdots & \ddots & \vdots \\ 0 & 0 & & 1 \\ 0 & 0 & \cdots & -1 \end{bmatrix},$$

with dimensions $2^j \times 2^{j-1}$.

In words, we have expressed the wavelets of 2^{j-1} pixel images in terms of the *scaling functions* (not the wavelets) of 2^j-pixel images. This formula is similar to our earlier result

$$\Phi^{j-1} = \Phi^j P^j;$$

we will soon combine our results into a single formula.

The dimensions of the matrix Q^j follow from the sizes of the row vectors Φ^j and Ψ^{j-1}, in a manner similar to our earlier result on scaling functions.

Our previous slide proved the special case of this formula for $j = 2$. (Do not be confused by the previous note that stated $j = 1$ — it was referring to the first formula of the previous slide, not this one.)

Multiresolution Analysis: Bases

From

$$\begin{aligned} \phi_i^j(x) &= \phi_{2i}^{j+1}(x) + \phi_{2i+1}^{j+1}(x) \\ \psi_i^j(x) &= \phi_{2i}^{j+1}(x) - \phi_{2i+1}^{j+1}(x), \end{aligned}$$

we deduce that

$$\begin{aligned} \phi_{2i}^{j+1}(x) &= \frac{1}{2}(\phi_i^j(x) + \psi_i^j(x)) \\ \phi_{2i+1}^{j+1}(x) &= \frac{1}{2}(\phi_i^j(x) - \psi_i^j(x)). \end{aligned}$$

Hence basis of images with $k = 2^{j+1}$ pixels,

$$\phi_0^{j+1}(x), \ldots, \phi_{k-1}^{j+1}(x),$$

can be replaced by

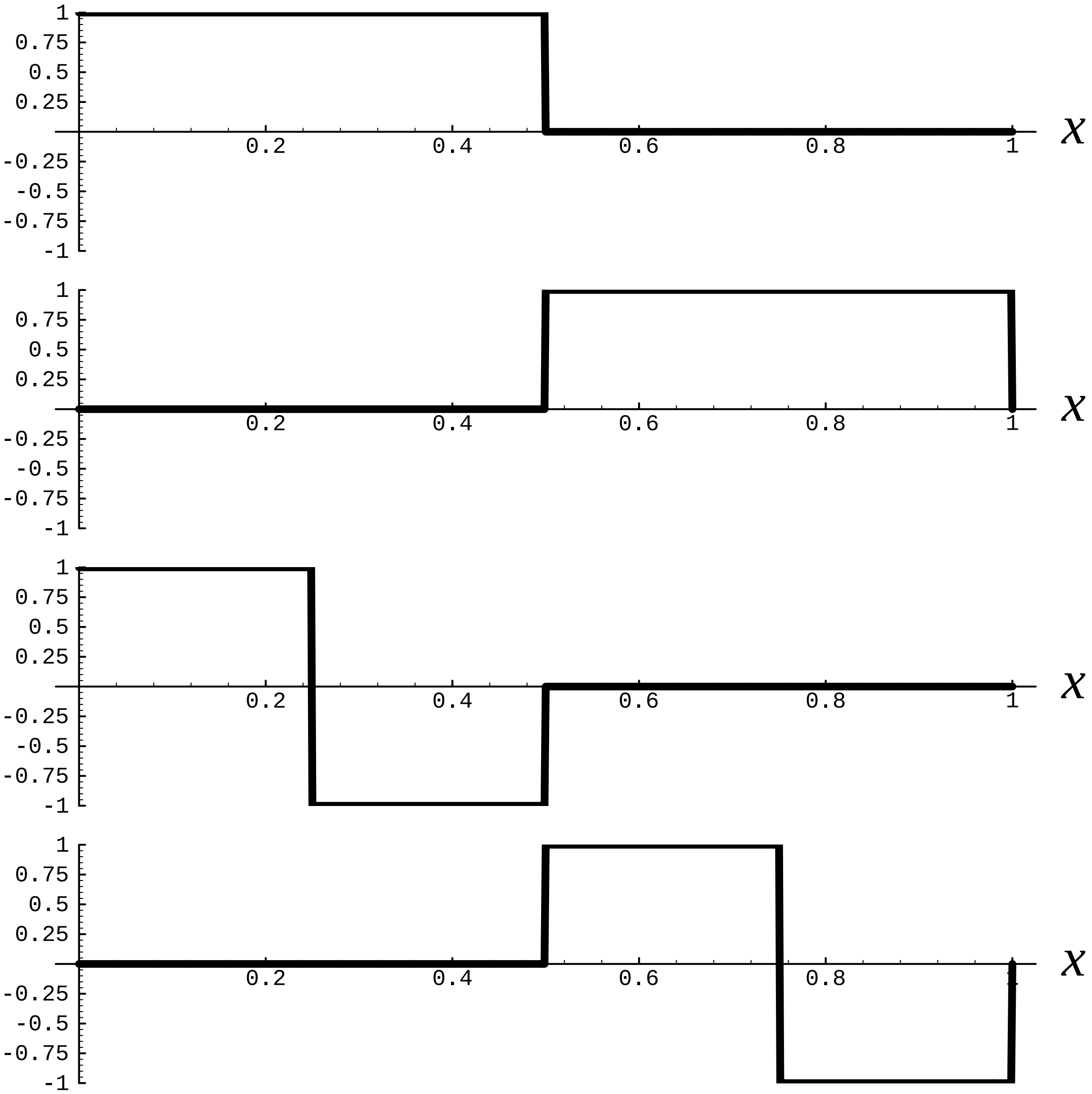
$$\phi_i^j(x), \psi_i^j(x).$$

In words, the first two equations express the wavelets and scaling functions of 2^j pixel images in terms of the scaling functions of 2^{j+1}-pixel images. Both of these equations appeared on earlier slides: when we considered the refinement of scaling functions, and when we defined wavelets.

We obtained the new basis by reasoning as follows: the regular basis for images with n pixels allows us to represent any n-pixel image $I(x)$ as a linear combination of these functions. But the expressions on the slide show how we can express each of these scaling functions as linear combinations of $\phi_i^j(x)$ and $\psi_i^j(x)$. So, $I(x)$ can be represented by the latter functions instead of the original basis. It follows that the $\phi_i^j(x)$'s and $\psi_i^j(x)$'s form a basis of V^{j+1}.

In summary, we have found a new basis of V^{j+1}, the set of 2^{j+1}-pixel images. This basis comprises the scaling functions and wavelets of 2^j-pixel images. To reduce mathematical notation, we will use "higher-level" and "lower-level" to qualify "scaling functions" and "wavelets": these qualifiers refer respectively to context-dependent higher or lower superscript values.

Multiresolution Analysis: $n = 4$



$\phi_0^1(x)$, $\phi_1^1(x)$, $\psi_0^1(x)$, and $\psi_1^1(x)$, the scaling functions and wavelets of 2-pixel images, are plotted in this order, from top to bottom. These four functions together form a new basis of V^2, the function space of 4-pixel images.

Multiresolution Analysis: Synthesis

Recall

$$
\begin{aligned}
\Phi^{j-1} &= \Phi^j P^j \\
\Psi^{j-1} &= \Phi^j Q^j \\
I(x) &= \Phi^j C^j
\end{aligned}
$$

for $I(x) \in V^j$. Write $I(x)$ using new basis

$$
\begin{aligned}
I(x) &= \Phi^{j-1} C^{j-1} + \Psi^{j-1} D^{j-1} \\
&= \Phi^j P^j C^{j-1} + \Phi^j Q^j D^{j-1},
\end{aligned}
$$

for some coefficients

$$
C^{j-1} = \begin{bmatrix} c_0^{j-1} \\ \vdots \\ c_m^{j-1} \end{bmatrix} \qquad D^{j-1} = \begin{bmatrix} d_0^{j-1} \\ \vdots \\ d_m^{j-1} \end{bmatrix}.
$$

Equating coefficients of Φ^j yields

$$
C^j = P^j C^{j-1} + Q^j D^{j-1}.
$$

We first expressed $I(x)$, a 2^j-pixel image, in terms of the scaling functions and wavelets of 2^{j-1}-pixel images. We then used the equations

$$\begin{aligned}\Phi^{j-1} &= \Phi^j P^j \\ \Psi^{j-1} &= \Phi^j Q^j\end{aligned}$$

to express $I(x)$ in terms of the scaling functions of 2^j-pixel images, i.e. Φ^j, while retaining in the resulting formula the coefficients of the lower-level scaling functions and wavelets. Finally, we also expressed $I(x)$ as a direct linear combination of Φ^j, that is,

$$I(x) = \Phi^j C^j,$$

to obtain

$$\Phi^j C^j = \Phi^j P^j C^{j-1} + \Phi^j Q^j D^{j-1},$$

from which we equated the coefficients of Φ^j on the two sides on the equation; the reason that we can equate coefficients is that, from the definition of a basis, we know that $I(x)$ has a unique representation in terms of the scaling functions Φ^j.

This is called a synthesis or *reconstruction* formula, because it allows us to combine (synthesize) the coefficients of the lower-level scaling functions and wavelets in order to find the coefficients of the higher-level scaling functions. How does one reverse this process? That is, how can we decompose (*analyze*) the coefficients of the higher-level scaling functions into the coefficients of the lower-level scaling functions and wavelets? We address this issue next...

Multiresolution Analysis: Analysis

Find *analysis matrices* A^j and B^j such that

$$\begin{aligned} C^{j-1} &= A^j C^j \\ D^{j-1} &= B^j C^j. \end{aligned}$$

A^j and B^j must satisfy

$$\left[\Phi^{j-1} | \Psi^{j-1}\right] \left[\frac{A^j}{B^j}\right] = \Phi^j.$$

So, from

$$\left[\Phi^{j-1} | \Psi^{j-1}\right] = \Phi^j \left[P^j | Q^j\right],$$

we conclude

$$\left[\frac{A^j}{B^j}\right] = \left[P^j | Q^j\right]^{-1}.$$

We use bar-delimited matrices within brackets to denote block matrices, which are simply formed by placing elements of two (or more, in general) matrices side by side or on top of each other. For example, using the block matrix notation, we can write

$$\left[\Phi^{j-1} | \Psi^{j-1}\right] = \Phi^j \left[P^j | Q^j\right],$$

instead of

$$\begin{aligned} \Phi^{j-1} &= \Phi^j P^j \\ \Psi^{j-1} &= \Phi^j Q^j. \end{aligned}$$

The reason A^j and B^j must satisfy the equality

$$\left[\Phi^{j-1} | \Psi^{j-1}\right] \left[\frac{A^j}{B^j}\right] = \Phi^j.$$

is that, if we multiply both sides of this equality by C^j, we must obtain the same answer, namely $I(x)$. The right-hand side yields

$$\Phi^j C^j,$$

which is $I(x)$ in the basis containing only scaling functions, while the left-hand side yields

$$\left[\Phi^{j-1} | \Psi^{j-1}\right] \left[\frac{A^j C^j}{B^j C^j}\right] = \left[\Phi^{j-1} | \Psi^{j-1}\right] \left[\frac{C^{j-1}}{D^{j-1}}\right],$$

which is $I(x)$ in the basis containing lower-level scaling functions and wavelets.

The inverse operation, used above, is well-defined in terms of matrix dimensions: $\left[P^j|Q^j\right]$ is a square matrix since it contains

- 2^j rows (as many as P^j and Q^j individually), and

- 2^j columns (as many as P^j and Q^j together, i.e. twice 2^{j-1}).

Filter Banks

From basis of V^j,

$$\phi_i^j(x), \psi_i^j(x),$$

applying

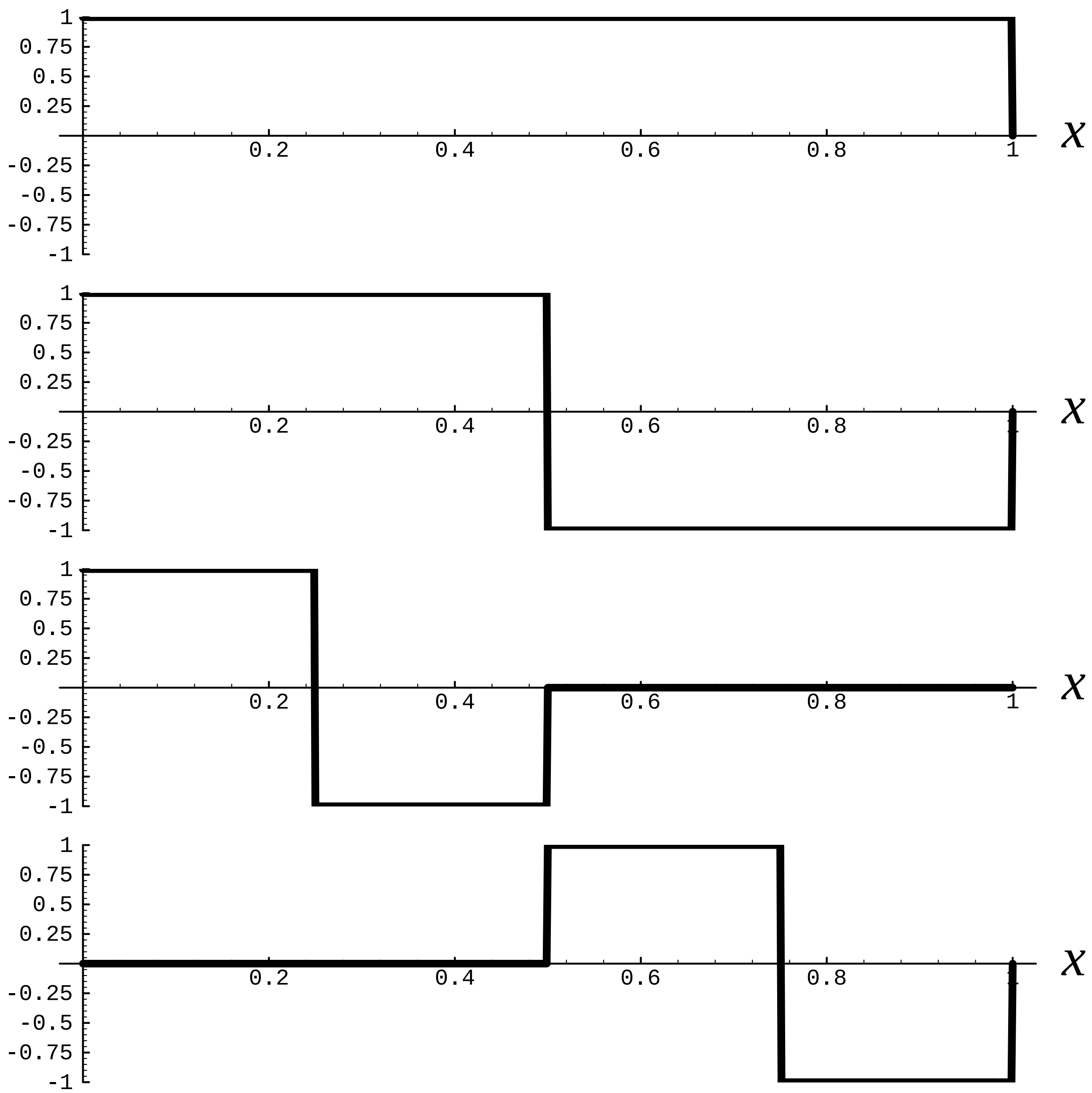
$$\begin{aligned}
\phi_{2i}^{j+1}(x) &= \frac{1}{2}(\phi_i^j(x) + \psi_i^j(x)) \\
\phi_{2i+1}^{j+1}(x) &= \frac{1}{2}(\phi_i^j(x) - \psi_i^j(x))
\end{aligned}$$

repeatedly, we eliminate all but one $\phi_i^j(x)$. Resulting basis for V^{j+1} is

$$\phi_0^0(x), \psi_i^k(x),$$

with $k = 0, \ldots, j$.

Filter Banks: $n = 4$



$\phi_0^0(x)$, $\psi_0^0(x)$, $\psi_0^1(x)$, and $\psi_1^1(x)$ are plotted in this order, from top to bottom. These four functions together form yet another basis of V^2, the function space of 4-pixel images.

Filter Banks: Bases

Basis of V^0 (1-pixel images):

$$\phi_0^0(x).$$

Basis of V^1 (2-pixel images):

$$\phi_0^0(x), \psi_0^0(x).$$

Basis of V^2 (4-pixel images):

$$\phi_0^0(x), \psi_0^0(x), \psi_0^1(x), \psi_1^1(x).$$

etc.

- $\psi_i^j(x)$'s add *detail*.

- *Orthogonal complement:* function space W^j whose elements, when added to elements of V^j, produce all elements of V^{j+1}. Basis of W^j is the $\psi_i^j(x)$'s.

For example, the function space W^1 contains the detail which, when added to all 2-pixel images (elements of V^1), produces all 4-pixel images (elements of V^2).

Orthogonal complements are named after the function spaces involving 3D vectors. If V^2 contains all the vectors on the xy plane, and V^3 contains all vectors in 3D space, then V^2 is a subset of V^3. Also, what vectors in V^2 need to produce vectors in V^3 is a component along the z axis, which is what W^2 offers. The usual choice for W^2 is all vectors perpendicular to the xy plane, hence the name orthogonal complement.

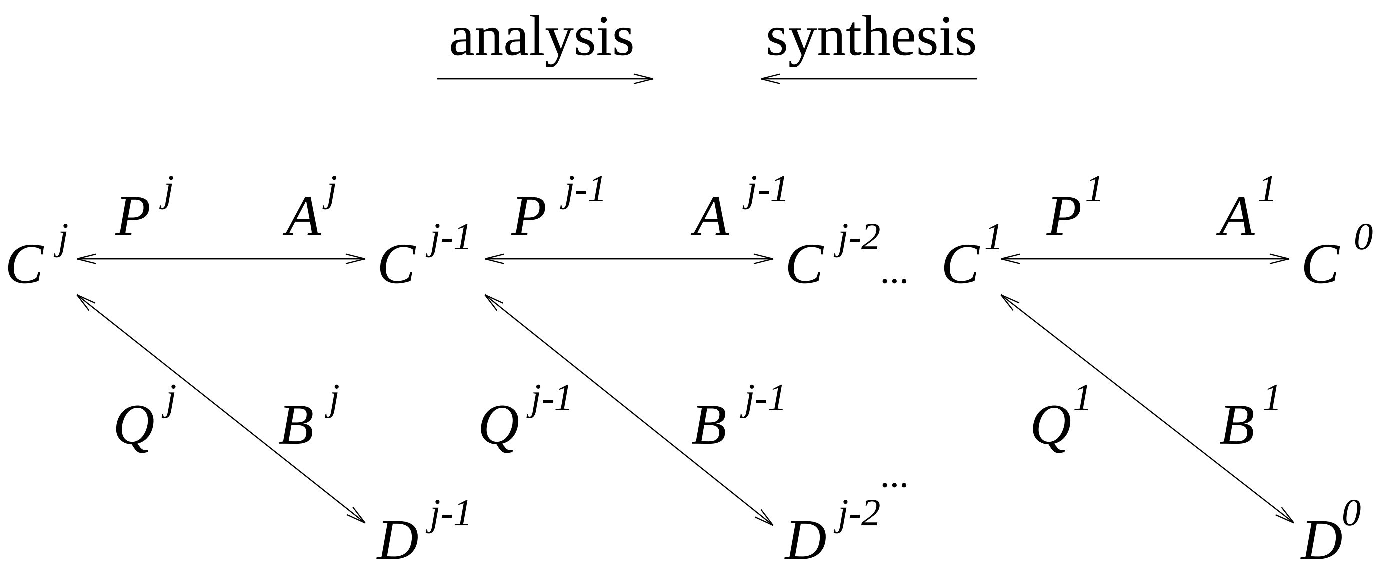
It is important to avoid a common misunderstanding here: the W^j do not contain the elements of V^{j+1} which are not in V^j; that is, V^{j+1} is not the union of W^j and V^j. In the case of vector spaces, for example, W^2 is not the set of all vectors which do not lie on the xy plane. Instead, it is (usually) the set of vectors along the z axis; it is only by *adding*, by vector addition, the elements of W^2 and V^2 that we get all elements of V^3.

The sequence of basis functions shown on the slide illustrates the containment of the V^j's: as j increases, the basis of V^j is expanded by the basis of W^j to yield a basis of V^{j+1}.

In general, wavelets are *defined* as the basis functions of the orthogonal complements of a sequence of nested spaces. The 1D Haar wavelets are just one of many choices for bases of the orthogonal complements of the 1D image spaces. Similarly, in the case of the space of 3D vectors, there are infinite ways to choose W^2: any set of vectors along a line which does not lie on the xy plane will do!

Since the wavelets add Detail, it is customary to use D^j to denote the column vector of the coefficients of Ψ^j, as we did earlier.

Filter Banks: General



$C^0, D^0, \ldots, D^{j-1}$ are *wavelet transform* of image in V^j, where

- C^0 is coefficient of $\Phi^0 = \left[\phi_0^0(x)\right] = [\phi(x)]$,
- $D^0, \ldots, D^{j-1}$ are coefficients of $\Psi^0, \ldots, \Psi^{j-1}$, respectively.

A *transform*, in general, is an alternate, equally expressive, representation of an object. The reason that $C^0, D^0, \ldots, D^{j-1}$ are a transform is that they fully represent an image in V^j in terms of the basis functions

$$\phi_0^0(x), \psi_i^k(x),$$

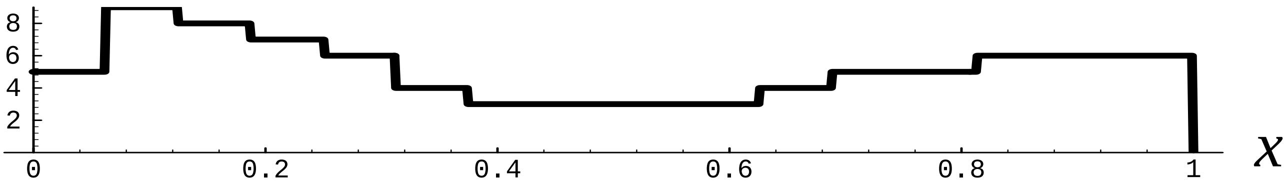
with $k = 0, \ldots, j-1$.

The diagram illustrates that, by repeatedly applying synthesis, we can find C^j, which is the image representation in terms of the basis functions Φ^j. And, going the other way, repeated analysis produces $C^0, D^0, \ldots, D^{j-1}$ from C^j.

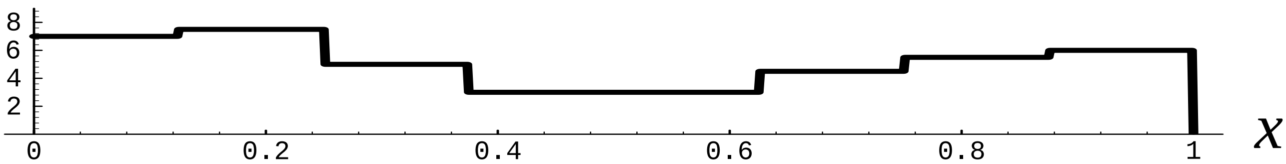
The set of analysis and synthesis matrices is also called the set of analysis and synthesis *filters*, collectively known as a *filter bank*. *Multiresolution analysis* is the study of filter banks, not necessarily wavelet-based ones, used to create transforms of objects which intuitively correspond to our notion of "levels of detail".

Filter Banks: $\Phi^j C^j$'s

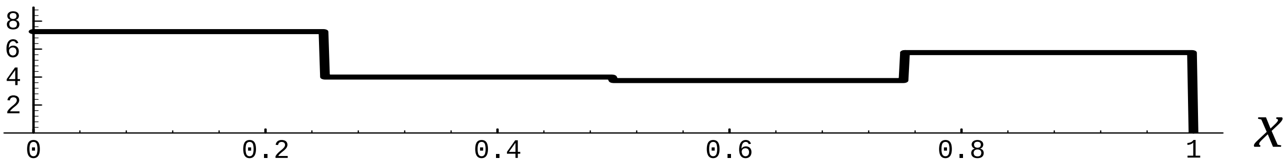
full resolution



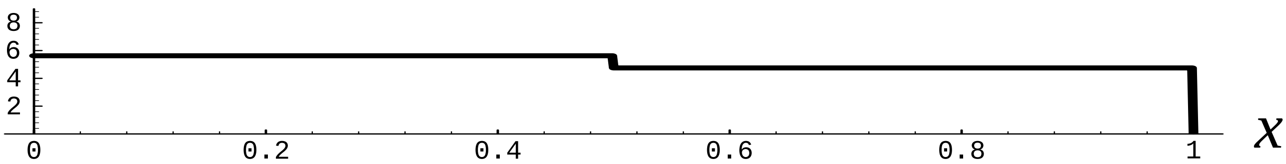
first approximation



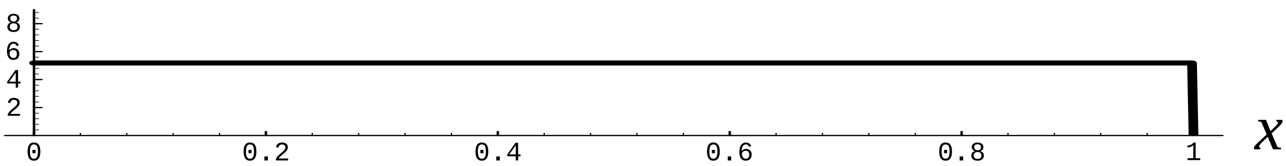
second approximation



third approximation



fourth approximation

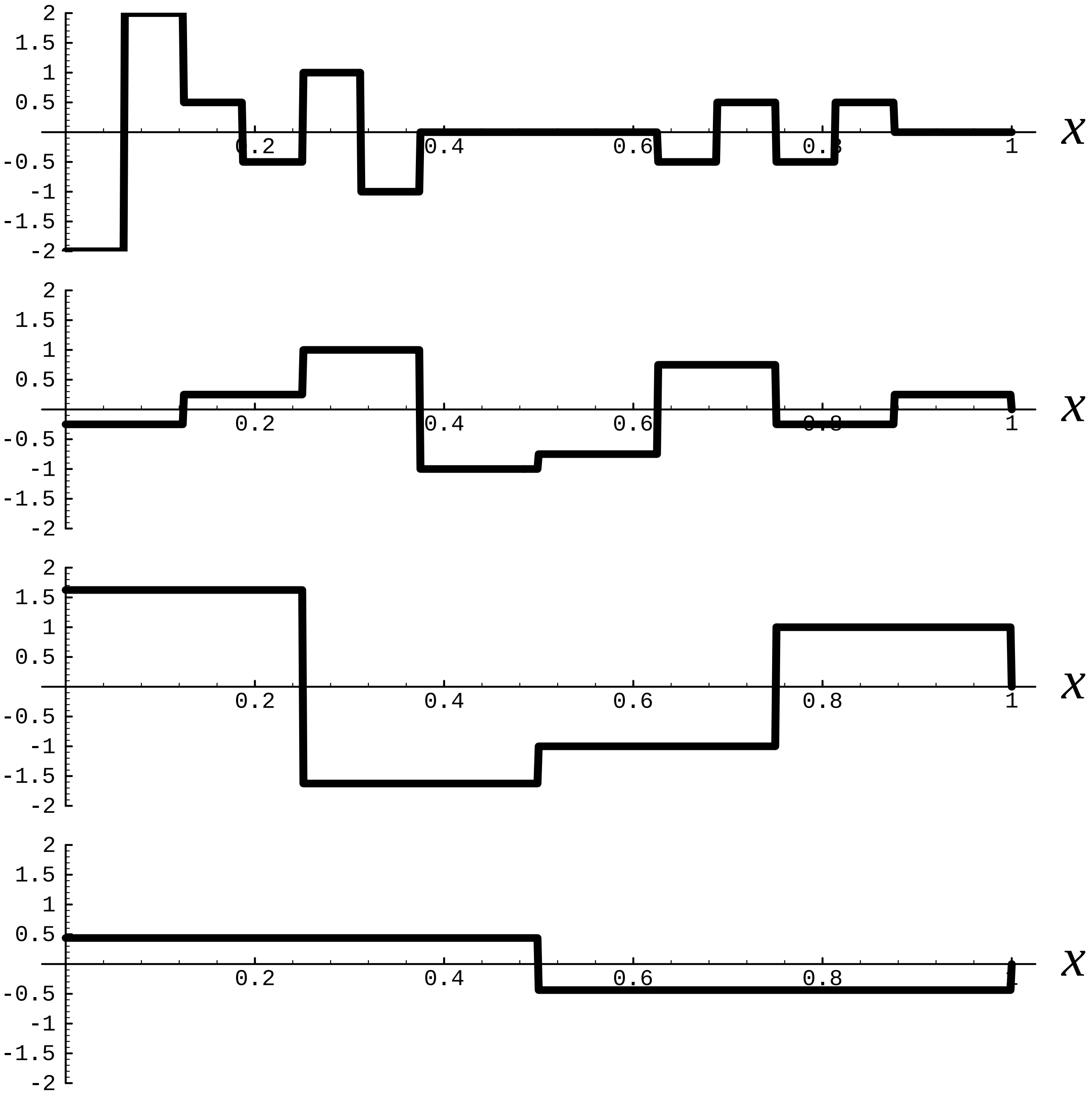


The topmost of these plots shows a 16-pixel image in its full form, namely $\Phi^4 C^4$. Below the full resolution version, we plotted successive approximations to this image, given by

$$\Phi^3 C^3, \Phi^2 C^2, \Phi^1 C^1, \Phi^0 C^0,$$

in order from top to bottom.

Filter Banks: $\Psi^j D^j$'s



These plots show the detail which has been lost by making the approximations shown in the previous slide. This detail is given by

$$\Psi^3 D^3, \Psi^2 D^2, \Psi^1 D^1, \Psi^0 D^0,$$

plotted in this order, from top to bottom. Observe that the range of the vertical axes is only $[-2, 2]$, in contrast to $[0, 9]$ for the previous slide.

Applications: Compression

Steps:

1. Represent image by wavelet transform.

2. If D^k is small, assume little detail is added to image by Ψ^k. So, discard D^k: don't store and assume zero.

Examples:

1D: Signal compression, e.g. audio.

2D: Generalizing scaling functions and wavelets, obtain raster picture compression, e.g. textures.

The paper in the reader contains pictures of compressed images for different thresholds, i.e. values for which D^k is considered small enough to be discarded. Unfortunately, the reproduction process destroyed all fine detail. However, I brought the original paper to class to see the pictures, and I also found on the WWW some of the pictures; see the class WWW pages.

Image compression is usually used for efficient storage and/or transmission. An unusual application which uses compressed image representations is image matching. That is, in order to find similar images, one ignores high detail (D^k for large k) and focuses on comparing coarse features (C^0 and D^k for small k).

Applications: Multiresolution Curves

Steps:

- V^j contains *uniform* cubic B-splines with 2^j knots.
- Scaling functions found by expressing in closed form polar interpolations from de Boor points.
- Wavelets chosen with *local* support.

Using B-spline surfaces, obtain *multiresolution surfaces*!

Uniform knot spacing is needed to guarantee containment of the V^j's, i.e.

$$V^0 \subset V^1 \subset \cdots$$

Also, for reasons of simplicity, we usually assume that all interior knots have multiplicity 1.

Recall that we found the Bernstein polynomials, representing the curve in terms of its Bézier points, by expressing polar interpolations in closed form. Similarly, the polar interpolations we do during the de Boor algorithm can be reduced in order to yield the shape of the B-spline in closed form, i.e. in terms of the desired scaling functions.

We want the wavelets to have narrow support because this forces the coefficient of each wavelet to have a local effect on the curve.

We will now see a short excerpt from a video accompanying a 1995 paper by Steven J. Gortler and Michael F. Cohen, presented at the 1995 Symposium on Interactive 3D Graphics. It illustrates the use of wavelets for geometric modeling, as well as for efficiently solving geometric optimization problems (variational modeling, a.k.a. modeling using constraints).

References

- Böhm: sections 43, 50.
- Seidel: pages 43–45.
- Ramshaw: section 15.
- Farin: pages 321–329, 343.
- Stollnitz (part 1): 76–80.
- Stollnitz (part 2): 75–80.
- Finkelstein: sections 2, 4.
- DeRose: section 1.

Böhm, Seidel, Ramshaw, Stollnitz, Finkelstein and DeRose are in the course reader, while Farin is on reserve at the MCS library.

The reading list, on the class WWW pages, has a hyperlink to wavelet-related research at the University of Washington (click on the title of any wavelet paper from the readings). If you start surfing from this link, you will find on-line copies of papers on wavelets and their applications, sample software, on-line demos, and much more...