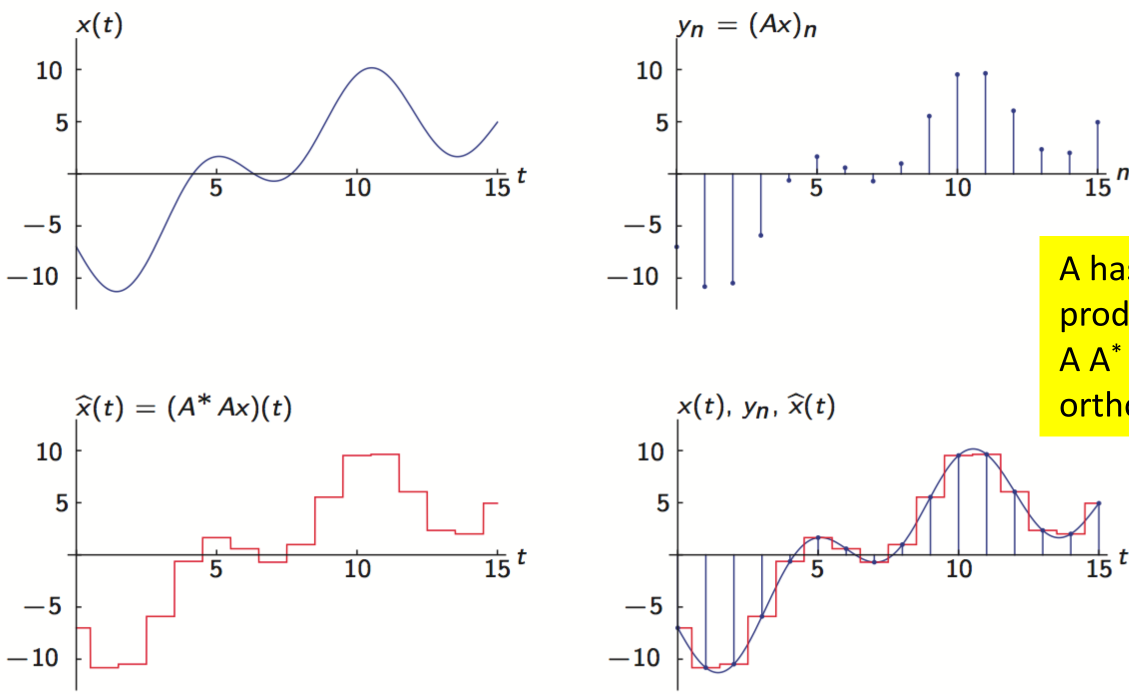
Adjoint operators: Local averaging

$$A : \mathcal{L}^2(\mathbb{R}) \to \ell^2(\mathbb{Z}) \qquad (Ax)_k = \int_{k-\frac{1}{2}}^{k+\frac{1}{2}} x(t)dt$$

$$\begin{aligned}
\langle Ax,\, y\rangle_{\ell^2} &= \sum_{n\in\mathbb{Z}} (Ax)_n\, y_n^* = \sum_{n\in\mathbb{Z}} \left(\int_{n-1/2}^{n+1/2} x(t)\, dt\right) y_n^* = \sum_{n\in\mathbb{Z}} \int_{n-1/2}^{n+1/2} x(t) y_n^*\, dt \\
&= \sum_{n\in\mathbb{Z}} \int_{n-1/2}^{n+1/2} x(t)\widehat{x}^*(t)\, dt = \int_{-\infty}^{\infty} x(t)\widehat{x}^*(t)\, dt = \langle x,\, \widehat{x}\rangle_{\mathcal{L}^2} = \langle x,\, A^* y\rangle_{\mathcal{L}^2}
\end{aligned}$$



A has adjoint A^* : $l^2(Z) \to L^2(R)$ that produces staircase functions

A A^* is identity, so A^* A is orthogonal projection

```matlab
%% Original signal definition

dt = 0.005;
t = 0:dt:25 - dt;

% useful for sinusoids
nPeriods = 2;
f0 = nPeriods/(max(t));

% x = 2*cos(2*pi*t*f0);

% signal made noisy with random integers, good for visualization and
understanding
% of the "averaging" function of the operator
 x = cos(2*pi*t*f0).*sin(2*pi*t*3*f0).*exp(-t/10).*randi([1, 2], 1, length(t));

% PAM process => perfect reconstruction if freq.s constraint are respected
% pamPulses = 50;
% pulseDuration = length(t)/pamPulses; % in samples of t
% x = [];
%
% for k = 1:pamPulses
%    x = [x, 2*randi([0, 1],1, 1)*ones(1, pulseDuration)];
% end
```

```
%% Sampling and reconstruction phase (application of A and its adjoint)

% definition of the number of samples
nSamples = 50;

% of course the sampling frequency will be max(t) + dt (e.g. 25 [seconds])
% divided by the number of samples "nSamples"(e.g. 25 => 1 Hz,
% 1 sample per second)

% in order to create the sampling operator, we can use a vector made
% entirely by 1/interval length and 0s, where the interval length is the
% number of samples of the original signal (defined in time t) taken into
% consideration in order to compute the first sample of the sampled signal

% since the number of samples choice is left to the user, we just compute
% the operator matrix by repeating a circular shift of a vector v, that we
% compute keeping in mind that given a certain number of samples,
% each and every one of them takes into account intervalLength samples
% of the ideally continuous signal. That is because:
```

```
intervalLength = length(t)/nSamples;

% is the number of samples of the original signal that will be associated
% with one sample of the sampled signal (division in nSamples
% non-overlapping intervals).

v = [(1/intervalLength)*ones(1, intervalLength), zeros(1, length(t) - intervalLength)];

% we build the operator matrix by repeating nSamples time the vector v,
% each time applying a shift of intervalLength samples
for n = 0:nSamples - 1
   A(n + 1, :) = circshift(v', intervalLength*n)' ;
end

% we obtain a matrix that has nSamples rows and length(t) columns
% the matrix product between the op. matrix and the original signal is then
% possible, since we are left with:
% A * x' = y => [nSamples x length(t) * length(t) x 1 = nSamples x 1]
y = A * x';
```

```
% the "new time step" (useful only for comparison plots..) can be computed
% as follows:
dn = (max(t) + dt)/nSamples;

% We used max(t) + dt since we made t stretch from 0 to for instance
% 25 - dt...)

% the adjoint operator is simply conj(A.') = A' = ctranspose(A);
xr = A'*y*intervalLength;

% we of course must multiply for intervalLength, otherwise each sample of
% the reconstruction would have an amplitude of x(..) * 1/intervalLength
% (by definition of matrix product, rows times columns..)
```

```
%% Plot handling

figure('units','normalized','outerposition',[0 0 1 1]);
    plot(t, x, 'Linewidth', 2); hold on;
    stem(dn/2:dn:max(t)+dn/2, y', 'Linewidth', 1.5);
    plot(t, xr, 'Linewidth', 1.5); hold off;
    title(sprintf('Sampling ($%d$ samples) and reconstruction', nSamples), 'interpreter',
'latex', 'fontsize', 13);
    grid on;
    ylim([min(x)-abs(min(x)/2), max(x)+abs(max(x)/2)]);
    xlabel('$t$', 'interpreter', 'latex');
    ylabel('$x(t)$', 'interpreter', 'latex', 'fontsize', 13);
    leg = legend('Original signal $\underline{x}$',...
        'Sampled signal $\underline{x}_s = \underline{\underline{A}} \ \underline{x}$',...
        'Reconstructed signal $\underline{x}_r =
\underline{\underline{A}}^*\underline{\underline{A}} \ \underline{x}$');
    set(leg, 'interpreter', 'latex', 'fontsize', 14);
```