First we need to recall how to find the matrix of a linear transformation.

So if \( L : V \to W \) satisfies

\[
L(\alpha \vec{v} + \beta \vec{w}) = \alpha L\vec{v} + \beta L\vec{w}
\]

What is the matrix \( M \) so that

\[
L(\vec{v}) = M\vec{v}
\]

The trick is to write \( \vec{v} \) in terms of the standard basis \( \vec{e}_1, \vec{e}_2, \ldots, \vec{e}_n \) with \( \vec{e}_j = \begin{bmatrix} 0 & \cdots & 1 & \cdots & 0 \\ 1 & \cdots & 0 & \cdots & 0 \end{bmatrix} \).

\[
\vec{v} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}
\]
\( \vec{v} = \alpha_1 \vec{e}_1 + \alpha_2 \vec{e}_2 + \ldots + \alpha_N \vec{e}_N \) so \( \vec{v} = \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{bmatrix} \)

So using the properties of \( L \)
\( L(\vec{v}) = \alpha_1 L(\vec{e}_1) + \alpha_2 L(\vec{e}_2) + \ldots + \alpha_N L(\vec{e}_N) \)
so if \( \vec{v} = [\alpha_1 \alpha_2 \ldots \alpha_N]^T \) then
\( L(\vec{v}) = \begin{bmatrix} L(\vec{e}_1) & L(\vec{e}_2) & \ldots & L(\vec{e}_N) \end{bmatrix} \vec{v} \)

**Example**
if \( L(\vec{e}_1) = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \), \( L(\vec{e}_2) = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \), \( L(\vec{e}_3) = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix} \)

Then \( L(\vec{v}) = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 0 \\ 3 & 3 & 2 \end{bmatrix} \vec{v} \)
- Now \( x \) is a data stream. A linear filter is a process which takes \( x \) and yields \( Lx \) with \( L \) a linear transformation.

- We would like the filter to work the same no matter what point we designate as the point.

- The shift of a vector pushes element to be right by one and brings the last element to the front:

\[
S \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} = \begin{bmatrix} d_N \\ d_1 \\ \vdots \\ d_{N-1} \end{bmatrix}
\]
Thus the shift is represented by the matrix

\[
\begin{bmatrix}
S_1 & S_2 & \cdots & S_p
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
\vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A linear filter is shift invariant if Shifting the input shifts the output by the same amount (also called translation invariant).

In symbols \( LS = SL \) (*).

A LTI filter is one that is linear and translation invariant. They have special matrix representations which are connected to the DFT.
Let's say \( \vec{e}_1 = \vec{g} = \begin{bmatrix} g_0 \\ \vdots \\ g_{N-1} \end{bmatrix} \)

Then since \( SE_1 = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \vec{e}_2 \)

Using equation (a)
\[
L \vec{e}_2 = LSE_1 = SL \vec{e}_1 = S \vec{g} = \begin{bmatrix} g_{N-1} \\ g_0 \\ \vdots \\ g_{N-2} \end{bmatrix}
\]

For any \( j \), let \( S^j = SS \cdots S \) (\( j \)-times)
\[
L \vec{e}_j = LSE_1 = S^j \vec{L} \vec{e}_1 = S^j \vec{g} = \begin{bmatrix} g_{N-j} \\ g_{N-j+1} \\ \vdots \\ g_0 \end{bmatrix}
\]
So the matrix representing the LTI Filter is

\[
M = \begin{bmatrix}
g & Sg & \ldots & S^{N-1}g \\
g & Sg & \ldots & S^{N-1}g \\
\vdots & \ddots & \ddots & \ddots \\
g & Sg & \ldots & S^{N-1}g
\end{bmatrix}
\]

This kind of matrix is called Toeplitz or Circulant.

\( g \) is called the impulse response since it is the output from a pulse of magnitude \( 1 \) at time zero \( \mathbb{E}_0 \). It determines the LTI Filter completely.
Thus far is another $N$-dimensional vector. Where we always work with subscripts mod $N$.

\[
\text{cyclic } \mathbf{f} \mathbf{g}^n = \begin{cases} 
0 & n = 0 \\
\mathbf{f} & n = 1 \\
\mathbf{g} - n & n \neq 1
\end{cases}
\]

Indeed, if the convolution \( \mathbf{f} \ast \mathbf{g} \) is

\[
\mathbf{f} \ast \mathbf{g} = \begin{cases} 
0 & n = 0 \\
\mathbf{f} & n = 1 \\
\mathbf{g} - n & n \neq 1
\end{cases}
\]

If \( \mathbf{f} \) and \( \mathbf{g} \) are both \( N \)-dimensional vectors, the initially unrequired notion of convolution is automatically LTI if \( \mathbf{f} \) and \( \mathbf{g} \) are understood to refer to matrices, work on the understood how to integrate matrices work on.
Example: Let \( n = 3 \)

\[
(f * g)_0 = \sum_{j=0}^{2} f_j g_{-j} = f_0 g_0 + f_1 g_{-1} + f_2 g_{-2} = f_0 g_0 + f_1 g_2 + f_2 g_1
\]

\[
(f * g)_1 = \sum_{j=0}^{2} f_j g_{j+1} = f_0 g_1 + f_1 g_0 + f_2 g_{-1} = f_0 g_1 + f_1 g_0 + f_2 g_2
\]

\[
(f * g)_2 = \sum_{j=0}^{2} f_j g_{j-2} = f_0 g_{-2} + f_1 g_1 + f_2 g_0 \quad \text{- visualize}
\]

Note that \( g \) is reversed and rotated
Now notice that convolution against $g$ is linear.

\[
L(\hat{v}) = \hat{v} * \hat{g}
\]

\[
L(\hat{v} + \hat{w}) = (\hat{v} + \hat{w}) * \hat{g} = \hat{v} * \hat{g} + \hat{w} * \hat{g} = L(\hat{v}) + L(\hat{w})
\]

\[
L(\alpha \hat{v}) = (\alpha \hat{v}) * \hat{g} = \alpha (\hat{v} * \hat{g}) = \alpha L(\hat{v})
\]

What is its matrix? Using the formulas on the previous page.

\[
L(\hat{v}) = \begin{bmatrix}
L(e_1) & L(e_2) & L(e_3)
\end{bmatrix} \hat{v} = \begin{bmatrix}
g_0 & g_2 & g_1 \\
g_1 & g_0 & g_2 \\
g_2 & g_1 & g_0
\end{bmatrix} \hat{v}
\]

The circulant matrix determined by $\hat{g}$, the same as the LTI filter with impulse $\hat{g}$. 

Theorem: If \( L : \mathbb{V} \to \mathbb{V} \) is a LTI Filter then it is determined by an impulse response vector \( \tilde{g} \) and \( L \) acts by convolution
\[
L(\tilde{v}) = \tilde{v} \ast g.
\]

LTI Filters have many uses but let's consider how they are computed. For large \( N \), \( \tilde{v} \ast \tilde{g} \) is large to compute as is \( M \tilde{v} \). The fast way is via the DFT as implemented by the FFT.
First an example, let the impulse "function" \( g \) be \( g_0 = \frac{1}{2}, \ g_1 = \frac{1}{2}, \ g_j = 0 \ for \ j = 2, \ldots, n - 1 \).

Then \((f \ast g)_n = \sum_{j=0}^{n-1} f_j g_{n-j} = f_n g_0 + f_{n-1} g_1 = \frac{1}{2} (f_n + f_{n-1}). \)

So this filter replaces the point at place \( n \) with its average with the point to its left.
The pointwise product of two vectors is

\[ \begin{bmatrix} u_1v_1 \\ u_2v_2 \\ \vdots \\ u_nv_n \end{bmatrix} \]

The main theorem says that the DFT turns convolutions into pointwise products.

\[ x*y = \frac{1}{N} \text{DFT}(x) \ast \text{DFT}(y) \]

or

\[ \text{DFT}(x*y) = \frac{1}{N} \text{DFT}(x) \ast \text{DFT}(y) \]
This gives a way to compute $\ast$ using the DFT and its inverse IDFT

$$X \ast y = \frac{1}{N} \text{IDFT} \left( \text{DFT}(x) \ast \text{DFT}(y) \right)$$

- Note in Matlab since $\overline{X_j} = \sum_{k=1}^{N} x_k \omega^{-j(k-1)}$

- In Matlab notation
  $$x \ast y = \text{ifft} \left( \text{fft}(x) \ast \text{fft}(y) \right)$$

- The proof of the convolution theorem is a calculation.
Proof:

\[ \hat{x}_j \hat{y}_j = \left( \sum_{k=0}^{\nu-1} x_k \omega^{-jk} \right) \left( \sum_{\ell=0}^{\nu-1} y_\ell \omega^{-j\ell} \right) \]

\[ = \frac{1}{\nu} \sum_{k=0}^{\nu-1} \sum_{\ell=0}^{\nu-1} x_k y_{\ell-k} \omega^{-j(k+\ell)} \]

Let \( n = k + \ell \) so \( \ell = n - k \)

\[ = \frac{1}{\nu} \sum_{n=0}^{\nu-1} \sum_{k=0}^{\nu-1} x_k y_{n-k} \omega^{-jn} \]

\[ = \frac{1}{\nu} \sum_{n=0}^{\nu-1} \left( \sum_{k=0}^{\nu-1} x_k y_{n-k} \right) \omega^{-jn} \]

\[ = \frac{1}{\nu} \sum_{n=0}^{\nu-1} (x \ast y)_n \omega^{-jn} = \frac{1}{\nu} \left( \sum_{n=0}^{\nu-1} (x \ast y)_n \right) \]

\[ \boxed{\text{DEMO}} \]