Stima & Filtraggio: Lab 3

Giacomo Baggio

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Padova

baggio@dei.unipd.it

baggio.dei.unipd.it/~teaching

May 24, 2017

Today's Lab

Wiener Filtering & Applications

1. Stochastic processes in **MATLAB**®
2. **MATLAB**® tools for Wiener filtering

Today's Lab

Wiener Filtering & Applications

(30 min) 1. Stochastic processes in **MATLAB**®

(60 min) 2. **MATLAB**® tools for Wiener filtering

Today's Lab

Wiener Filtering & Applications

1. Stochastic processes in **MATLAB®**

   - Generating white noise
   - Generating filtered noise
   - Useful **MATLAB®** commands

(weakly stationary)
white noise processes

$$\{e(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[e(t)] = \mu,\ \mathbb{E}[e(t)e(s)] = \sigma^2\delta(t-s) + \mu^2$$

(weakly stationary)
white noise processes

$$\{e(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[e(t)] = \mu,\ \mathbb{E}[e(t)e(s)] = \sigma^2\delta(t-s) + \mu^2$$

$$\bar{e}(t) := e(t) - \mu,\ \bar{e}(t) \perp \bar{e}(s),\ s \neq t$$

(weakly stationary)
white noise processes

$$\{e(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[e(t)] = \mu,\ \mathbb{E}[e(t)e(s)] = \sigma^2\delta(t-s) + \mu^2$$

100 samples of Gaussian WN with mean μ and var. σ^2:

```
>> rvE = dMu + dSigma*randn(1,100)
```

μ σ

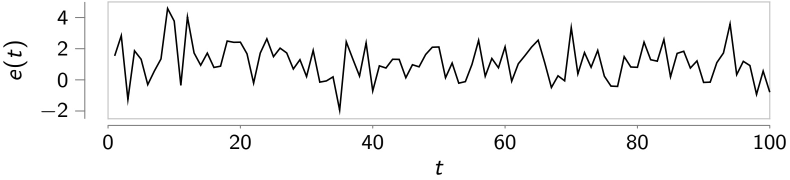
(weakly stationary)
white noise processes

$$\{e(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[e(t)] = \mu,\ \mathbb{E}[e(t)e(s)] = \sigma^2\delta(t-s) + \mu^2$$

100 samples of Gaussian white noise with mean μ and var. σ^2:

```
>> rvE = dMu + dSigma*randn(1,100)
```

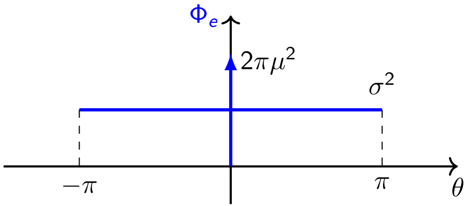
$\mu = 1$ $\sigma = 1$



(weakly stationary)
white noise processes

$$\{e(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[e(t)] = \mu,\ \mathbb{E}[e(t)e(s)] = \sigma^2\delta(t-s) + \mu^2$$

Spectral density: $\Phi_e(e^{j\theta}) = 2\pi\mu^2\,\delta(\theta) + \sigma^2,\ \theta \in [-\pi, \pi]$



(weakly stationary)
white noise processes

– multivariate case –

$$\{\boldsymbol{e}(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[\boldsymbol{e}(t)] = \boldsymbol{\mu},\ \mathbb{E}[\boldsymbol{e}(t)\boldsymbol{e}^\top(s)] = \Sigma\,\delta(t-s) + \boldsymbol{\mu}\boldsymbol{\mu}^\top$$

$$\Sigma = \Sigma^\top \geq 0$$

(weakly stationary)
white noise processes

– multivariate case –

$$\{\boldsymbol{e}(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[\boldsymbol{e}(t)] = \boldsymbol{\mu},\ \mathbb{E}[\boldsymbol{e}(t)\boldsymbol{e}^\top(s)] = \Sigma\,\delta(t-s) + \boldsymbol{\mu}\boldsymbol{\mu}^\top$$

1 sample of 2-dim. Gaussian white noise with mean $\boldsymbol{\mu} \in \mathbb{R}^2$ and cov. $\Sigma = \Sigma^\top \in \mathbb{R}^{2\times 2}$, $\Sigma \geq 0$:

```
>> mE = rvMu + sqrtm(dSigma)*randn(2,100)
```

$\boldsymbol{\mu}$

$\Sigma^{\frac{1}{2}}$

symmetric matrix

square root

(weakly stationary)
white noise processes

– multivariate case –

$$\{\boldsymbol{e}(t)\}_{t\in\mathbb{Z}} \text{ s.t. } \mathbb{E}[\boldsymbol{e}(t)] = \boldsymbol{\mu},\ \mathbb{E}[\boldsymbol{e}(t)\boldsymbol{e}^\top(s)] = \Sigma\,\delta(t-s) + \boldsymbol{\mu}\boldsymbol{\mu}^\top$$

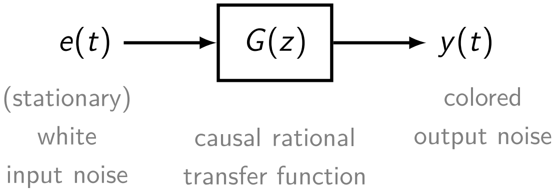
1 sample of 2-dim. Gaussian white noise with
mean $\boldsymbol{\mu} \in \mathbb{R}^2$ and cov. $\Sigma = \Sigma^\top \in \mathbb{R}^{2\times 2}$, $\Sigma \geq 0$:

```
>> mE = rvMu + chol(dSigma,'lower')*randn(2,100)
```

$\boldsymbol{\mu}$ $\qquad$ $L\ :\ \Sigma = LL^\top$

Cholesky factor

Filtered noise processes



Filtered noise processes

$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

(stationary) white input noise

stable causal rational transfer function

stationary colored output noise

Filtered noise processes

$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

$$G(z) = \frac{N(z)}{D(z)}, \qquad N(z), D(z) \text{ polynomials}$$

$$\Big\downarrow$$

$$D(z^{-1})y(t) = N(z^{-1})e(t) \qquad \text{(ARMA representation)}$$

z^{-1} = delay operator

Filtered noise processes

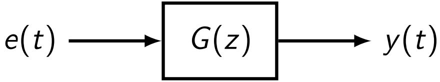
$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

Task. Generate 100 samples of the process

$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

Filtered noise processes



Task. Generate 100 samples of the process

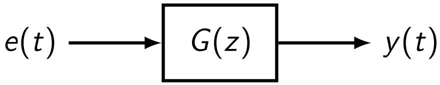
$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

Note that $N(z^{-1}) = 1 - 0.2z^{-2}$, $D(z^{-1}) = 1 - 0.1z^{-1}$

$$\Longrightarrow \; G(z) = \frac{z^2-0.2}{z^2-0.1z} \quad \text{(stable!)}$$

Filtered noise processes



Task. Generate 100 samples of the process

$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

```
>> rvN = [1 0 -0.2]
>> rvD = [1 -0.1]
```

define the numerator and denominator polynomials in z^{-1}

Filtered noise processes

$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

Task. Generate 100 samples of the process

$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

`>> rvE = randn(1,100)` create white noise sequence

Filtered noise processes

$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

Task. Generate 100 samples of the process

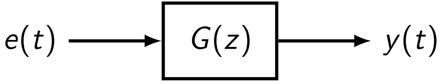
$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

```
>> rvY = filter(rvN,rvD,rvE)
```
generate the process!

N.B. Initial conditions set to 0

Filtered noise processes



Task. Generate 100 samples of the process

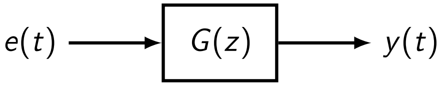
$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

---

an alternative procedure...

Filtered noise processes



Task. Generate 100 samples of the process

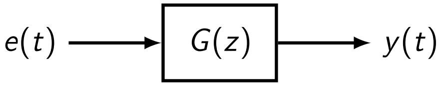
$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

```
>> z = tf('z')
```
define the transfer function

```
>> tfG = (z^2-0.2)/(z^2-0.1*z)
```

Filtered noise processes



Task. Generate 100 samples of the process

$$y(t) = 0.1y(t-1) + e(t) - 0.2e(t-2)$$

where $\{e(t)\}_{t\in\mathbb{Z}}$ is a Gaussian WN process ($\mu = 0$, $\sigma = 1$)

```
>> cvY = lsim(tfG,rvE)
```

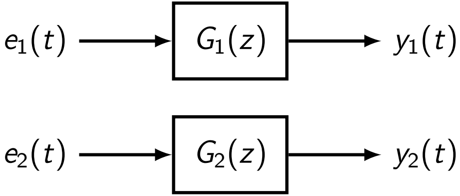
generate the process!

N.B. Initial conditions set to 0

Filtered noise processes

– an important remark –

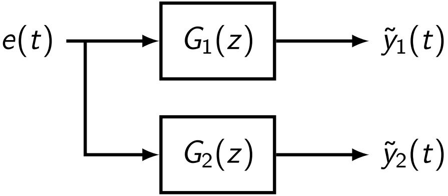
$\{e_1\}$, $\{e_2\}$ WNs, $\{e_1\} \perp \{e_2\}$



$\boldsymbol{y} := [y_1, y_2]^\top$

$\{e_1\}$, $\{e_2\}$, $\{e\}$ same mean and variance

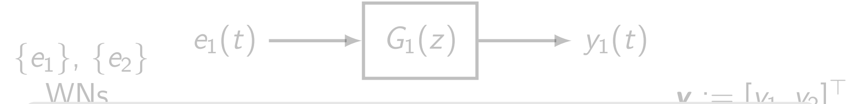
$\{e\}$ WN



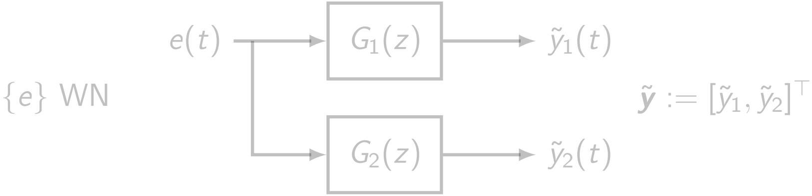
$\tilde{\boldsymbol{y}} := [\tilde{y}_1, \tilde{y}_2]^\top$

Filtered noise processes

– an important remark –

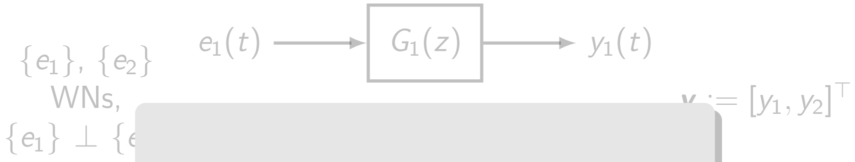


Question: Do $\boldsymbol{y}$ and $\tilde{\boldsymbol{y}}$ define the same process?

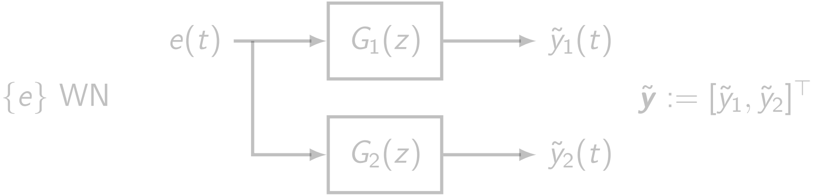


Filtered noise processes

– an important remark –



Answer: No, since $\{e_1\} \perp \{e_2\}$!



Useful MATLAB® commands

Create TF $\frac{N(z)}{D(z)}$	`>> tfG = tf(rvN,rvD,-1)`
Filter $\{e(t)\}_{t=t_1}^{t_2}$ by $\frac{N(z)}{D(z)}$	`>> rvY = filter(rvN,rvD,rvE)`
Filter $\{e(t)\}_{t=t_1}^{t_2}$ by $G(z)$	`>> cvY = lsim(tfG,rvE)`
Recover $N(z)$, $D(z)$	`>> [rvN,rvD] = tfdata(tfG,'v')`
From TF to SS	`>> ssG = ss(tfG)`
From TF to ZPK	`>> zpkG = zpk(tfG)`

✎ Practice time 1!

Ex 1.1. Create a function

```
[bS,bMP] = checkTFStability(tfG)
```

that has as input a causal scalar discrete-time transfer function object `tfG`. The function returns

- boolean `bS = true` if any coprime representation of `tfG` is (strictly) ***stable***, and `bS = false` otherwise,
- boolean `bMP = true` if any coprime representation of `tfG` is ***minimum phase***, i.e. it is stable with marginally stable inverse, and `bMP = false` otherwise.

Then, test the function on the following TFs:

$$G_1(z) = \frac{z(z+1)}{z^2+0.4z-0.45},\ G_2(z) = \frac{z^2-0.7z+1}{z^2+0.4z-0.45},\ G_3(z) = \frac{z+2}{z^2+2.5z+1}.$$

Practice time 1!

Ex 1.2. Create a function

`plotSpectrum(tfG)`

that has as input a causal scalar discrete-time transfer function `tfG`. The function plots the ***spectral density*** in the frequency interval $\theta \in [-\pi, \pi]$ of the process generated by filtering a Gaussian WN process ($\mu = 0$, $\sigma = 1$) through `tfG`. If `tfG` has poles on the unit circle, the function throws an error and displays an error message.

$$e(t) \longrightarrow \boxed{G(z)} \longrightarrow y(t)$$

Then, test the function on the TFs of Ex. 1.1.

Extra question. What happens if $\sigma \neq 1$?

Today's Lab

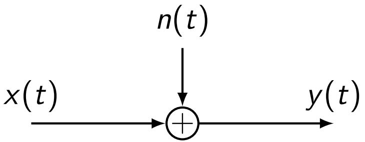
Wiener Filtering & Applications

2. **MATLAB®** tools for Wiener filtering

   - Quick recap
   - Computing integrals via residues
   - Computing causal and anticausal parts

Wiener filtering

Setup



$\{x(t)\}_{t\in\mathbb{Z}}$: stationary input process

$\{n(t)\}_{t\in\mathbb{Z}}$: stationary external noise

$\{y(t)\}_{t\in\mathbb{Z}}$: stationary output process

$$\begin{bmatrix} \Phi_x & \Phi_{xy} \\ \Phi_{xy}^* & \Phi_y \end{bmatrix}$$

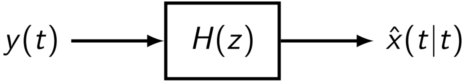
Joint spectrum

Wiener filtering

Wiener estimate

$$H(z) := \left[\frac{\Phi_{xy}(z)}{W_y(z^{-1})}\right]_+ \frac{1}{W_y(z)}$$

$$\Phi_y(e^{j\theta}) = W_y(e^{j\theta})W_y(e^{-j\theta})$$

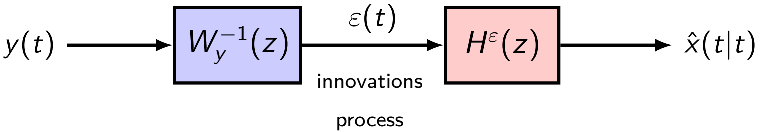


Wiener filtering

Wiener estimate

$$H(z) := \left[\frac{\Phi_{xy}(z)}{W_y(z^{-1})}\right]_+ \frac{1}{W_y(z)}$$

$$\Phi_y(e^{j\theta}) = W_y(e^{j\theta})W_y(e^{-j\theta})$$



Wiener filtering

Wiener estimate

$$H(z) := \left[\frac{\Phi_{xy}(z)}{W_y(z^{-1})}\right]_+ \frac{1}{W_y(z)}$$

$$\Phi_y(e^{j\theta}) = W_y(e^{j\theta})W_y(e^{-j\theta})$$

$$\mathrm{Var}\,\tilde{x}(t|t) = \int_{-\pi}^{\pi} \left(\Phi_x(e^{j\theta}) - H^{\varepsilon}(e^{j\theta})H^{\varepsilon}(e^{-j\theta})\right) \frac{\mathrm{d}\theta}{2\pi}$$

estimation error variance

Wiener filtering *in practice*

How to perform spectral factorization?

How to compute the causal and (strictly) anticausal part of a rational function?

How to compute the integral of a rational function upon the unit circle?

Wiener filtering *in practice*

How to perform ral factorization?

Next Lab

How to compute the causal and (strictly) anticausal part of a rational function?

How to compute the integral of a rational function upon the unit circle?

Computing integrals via residues

Let $G(z) = N(z)/D(z)$ be a scalar rational function. We can decompose $G(z)$ using partial fraction expansion

$$G(z) = \sum_{i=1}^{n_p} \sum_{j=1}^{\mu_i} \frac{R_{ij}}{(z - p_i)^j} + k_{n_\infty} z^{n_\infty} + \cdots + k_1 z + k_0$$

- $\{p_i\}_{i=1}^{n_p}$ are the poles of $G(z)$ and $\{\mu_i\}_{i=1}^{n_p}$ the corresponding multiplicities
- R_{ij} are (in general, complex) coefficients corresponding to the term of multiplicity μ_j of the pole p_i
- $n_\infty = \max\{\deg N(z) - \deg D(z), 0\}$

Computing integrals via residues

The coefficients R_{ij} are called ***residues**** and are of crucial importance in complex analysis.

If p_i has multiplicity $\mu_i = 1$, then

$$R_{i1} = \lim_{z \to p_i} G(z)(z - p_i)$$

If p_i has multiplicity $\mu_i > 1$, then

$$R_{ij} = \frac{1}{(\mu_i - j)!} \lim_{z \to p_i} \left[\frac{\mathrm{d}^{\mu_i - j}}{\mathrm{d}z^{\mu_i - j}} G(z)(z - p_i)^{\mu_i} \right], \; j = 1, \ldots, \mu_i$$

*Mathematically speaking, the residues are only the coefficients R_{i1}'s.

Computing integrals via residues

Let $F(z)$ be a rational function analytic on the unit circle. The computation of the integral of $F(z)$ upon the unit circle boils down to

$$\frac{1}{2\pi}\int_{-\pi}^{\pi} F(e^{j\theta})\,\mathrm{d}\theta = \frac{1}{2\pi j}\oint_{|z|=1} \frac{F(z)}{z}\,\mathrm{d}z = \sum_{i\,:\,|p_i|<1} R_{i1}$$

Hence, it suffices to

1. Compute the partial fraction expansion of $G(z) = F(z)/z$
2. Sum the (one-multiplicity) residues R_{i1} of poles p_i s.t. $|p_i| < 1$

Computing integrals via residues

...in MATLAB®

```
>> [cvR,cvP,rvK] = residue(rvN,rvD)
```

- $\texttt{cvR} = [R_{11}\ R_{12}\ \dots\ R_{1\mu_1}\ R_{21}\ \dots]^\top$
- $\texttt{cvP} = [p_1\ p_1\ \dots\ p_1\ p_2\ \dots]^\top$ *(every p_i is repeated μ_i times!)*
- $\texttt{rvK} = [k_{n_\infty}\ \dots\ k_1\ k_0]$

Computing integrals via residues

...in MATLAB®

```
>> [cvR,cvP,rvK] = residue(rvN,rvD)
>> [rvN,rvD] = residue(cvR,cvP,rvK)
```

Inverse operation!

- $\texttt{cvR} = [R_{11}\ R_{12}\ \ldots\ R_{1\mu_1}\ R_{21}\ \ldots]^\top$
- $\texttt{cvP} = [p_1\ p_1\ \ldots\ p_1\ p_2\ \ldots]^\top$ *(every p_i is repeated μ_i times!)*
- $\texttt{rvK} = [k_{n_\infty}\ \ldots\ k_1\ k_0]$

Computing causal and anticausal parts

Let $G(z) = N(z)/D(z)$ be a scalar rational function analytic on the unit circle. We can decompose $G(z)$ as

$$G(z) = \sum_{i=1}^{n_p} \sum_{j=1}^{\mu_i} \frac{R_{ij}}{(z-p_i)^j} + k_{n_\infty} z^{n_\infty} + \cdots + k_1 z + k_0$$

$$= \sum_{i\,:\,|p_i|<1} \mathrm{Res}_i(z) + k_0^+ + k_0^- + \sum_{i\,:\,|p_i|>1} \mathrm{Res}_i(z) + \sum_{i=1}^{n_\infty} k_i z^i$$

where $\mathrm{Res}_i(z) := \sum_{j=1}^{\mu_i} \frac{R_{ij}}{(z-p_i)^j}$, $k_0 = k_0^+ + k_0^-$ with

$$k_0^- := -\sum_{|p_i|>1} \mathrm{Res}_i(0)$$

Computing causal and anticausal parts

Let $G(z) = N(z)/D(z)$ be a scalar rational function analytic on the unit circle. We can decompose $G(z)$ as

$$G(z) = \sum_{i=1}^{n_p} \sum_{j=1}^{\mu_i} \frac{R_{ij}}{(z-p_i)^j} + k_{n_\infty} z^{n_\infty} + \cdots + k_1 z + k_0$$

$$= \underbrace{\sum_{i\,:\,|p_i|<1} \mathrm{Res}_i(z) + k_0^+}_{[G(z)]_+} + \underbrace{k_0^- + \sum_{i\,:\,|p_i|>1} \mathrm{Res}_i(z) + \sum_{i=1}^{n_\infty} k_i z^i}_{[[G(z)]]_-}$$

$[G(z)]_+$
causal part

$[[G(z)]]_-$
strictly anticausal part

Computing causal and anticausal parts

Let $G(z) = N(z)/D(z)$ be a scalar rational function analytic on the unit circle. We can decompose $G(z)$ as

$$\begin{aligned} G(z) &= \sum_{i=1}^{n_p}\sum_{j=1}^{\mu_i} \frac{R_{ij}}{(z-p_i)^j} + k_{n_\infty} z^{n_\infty} + \cdots + k_1 z + k_0 \\ &= \underbrace{\sum_{i\,:\,|p_i|<1} \mathrm{Res}_i(z) + k_0^+}_{[G(z)]_+ \text{ causal part}} + \underbrace{k_0^- + \sum_{i\,:\,|p_i|>1} \mathrm{Res}_i(z) + \sum_{i=1}^{n_\infty} k_i z^i}_{[[G(z)]]_- \text{ strictly anticausal part}} \end{aligned}$$

Moral: $[G(z)]_+$ and $[[G(z)]]_-$ can be computed using residues!

✎ *Practice time 2!*

Ex 2.1. Create a function

```
dInt = resIntegral(tfF)
```

that has as input a rational transfer function object `tfF`. The function returns the integral `dInt` upon the unit circle of `tfF` computed via the residue method.

Recall:
$$\frac{1}{2\pi}\int_{-\pi}^{\pi} F(e^{j\theta})\,\mathrm{d}\theta = \frac{1}{2\pi j}\oint_{|z|=1} \frac{F(z)}{z}\,\mathrm{d}z = \sum_{i\,:\,|p_i|<1} R_{i1}$$

✎ *Practice time 2!*

Ex 2.2. Create a function

```
[tfCaus,tfACaus] = causalPart(tfG)
```

that has as input a rational transfer function object `tfG`. The function returns the causal part `tfCaus` and the strictly anticausal part `tfACaus` of `tfG`.

Recall:
$$G(z) = \underbrace{\sum_{i\,:\,|p_i|<1} \mathrm{Res}_i(z) + k_0^+}_{[G(z)]_+} + \underbrace{k_0^- + \sum_{i\,:\,|p_i|>1} \mathrm{Res}_i(z) + \sum_{i=1}^{n_\infty} k_i z^i}_{[[G(z)]]_-}$$