

Verlässliche Echtzeitsysteme

Übungen zur Vorlesung

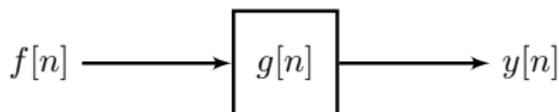
Hinweise zur Aufgabe: Filter

Phillip Raffeck, Tim Rheinfels, Simon Schuster, Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg
Lehrstuhl Informatik 4 (Verteilte Systeme und Betriebssysteme)
<https://sys.cs.fau.de>

Wintersemester 2022





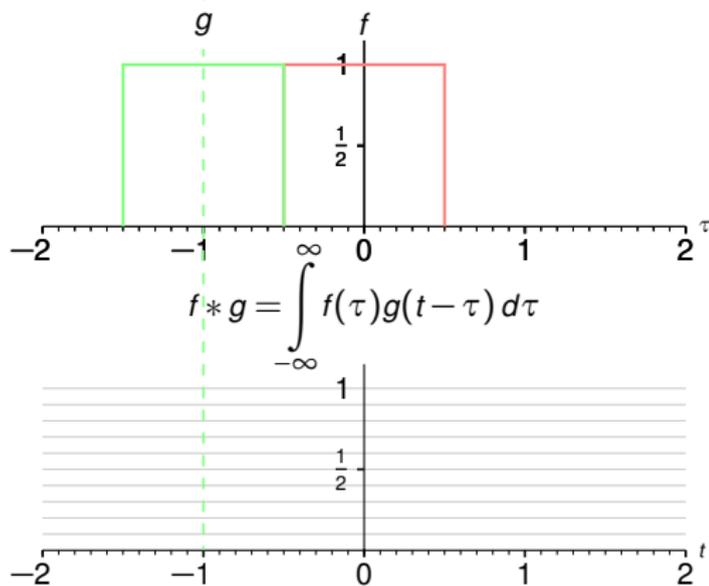
- Objekte identifizieren (z.B. Eingaben)
- Implementierung der Filterung durch **Faltung** (engl. convolution) mit Impulsantwort
 - f Signalwerte, g Filterwerte

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m] \quad (1)$$

- Impulsantwort: Reaktion des Systems auf Dirac-Impuls
- Zunächst Verwendung von `float`, anschließend **Festkommaformat**

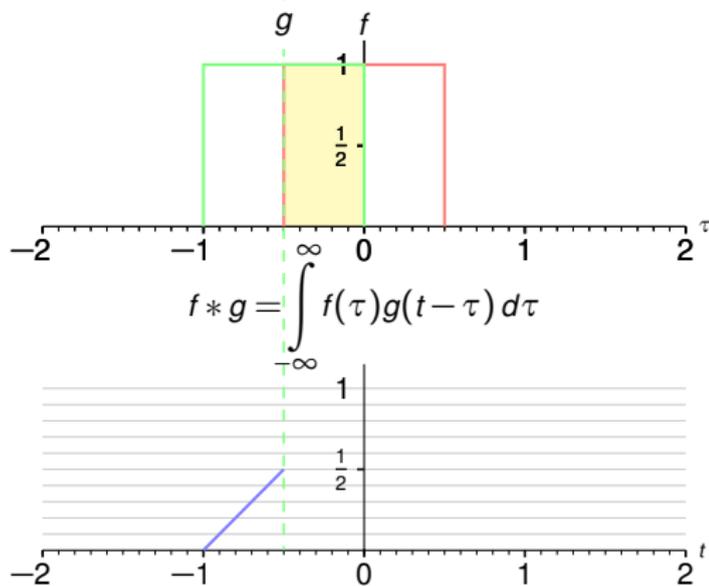


Beispiel: Faltung



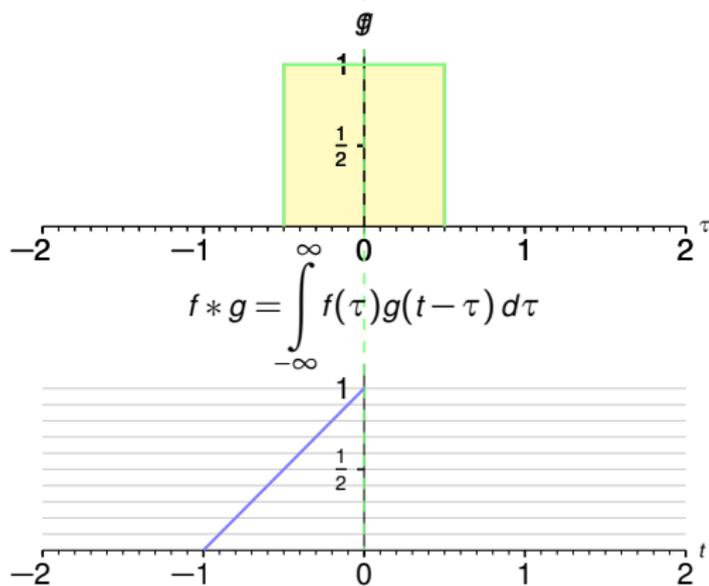
Beispiel: Faltung

$$\text{■ } f(\tau)g(-0.5-\tau)$$



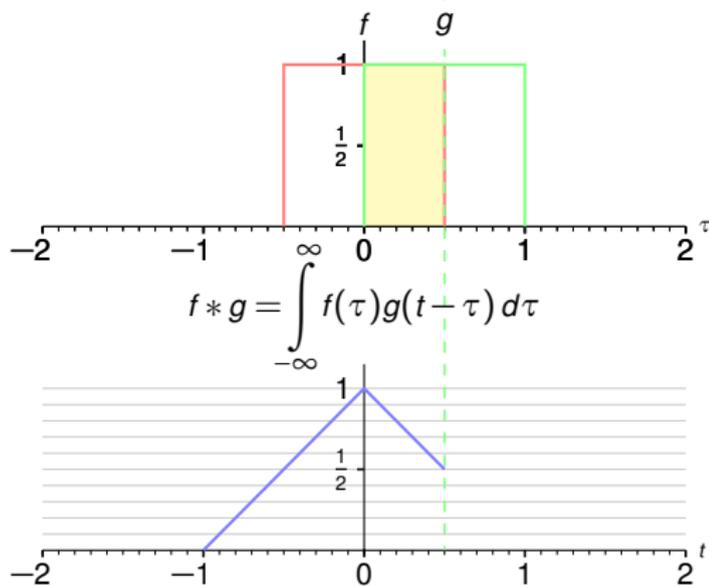
Beispiel: Faltung

$$\text{■ } f(\tau)g(0-\tau)$$



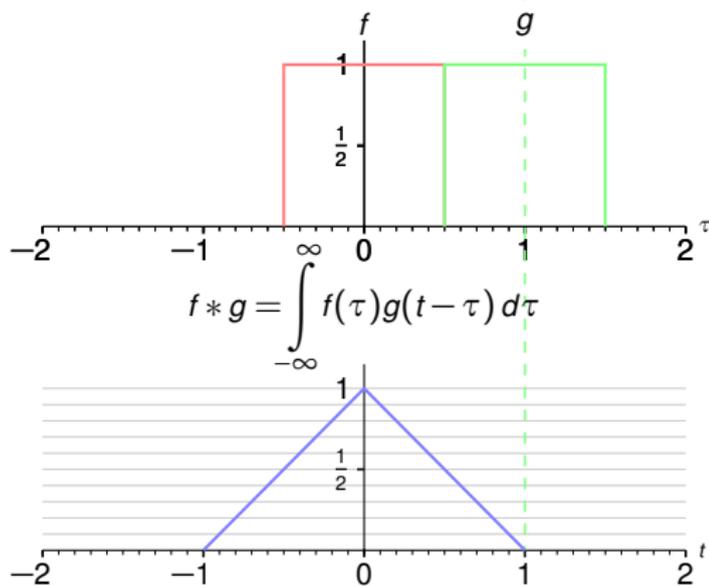
Beispiel: Faltung

$$\text{■ } f(\tau)g(0.5-\tau)$$

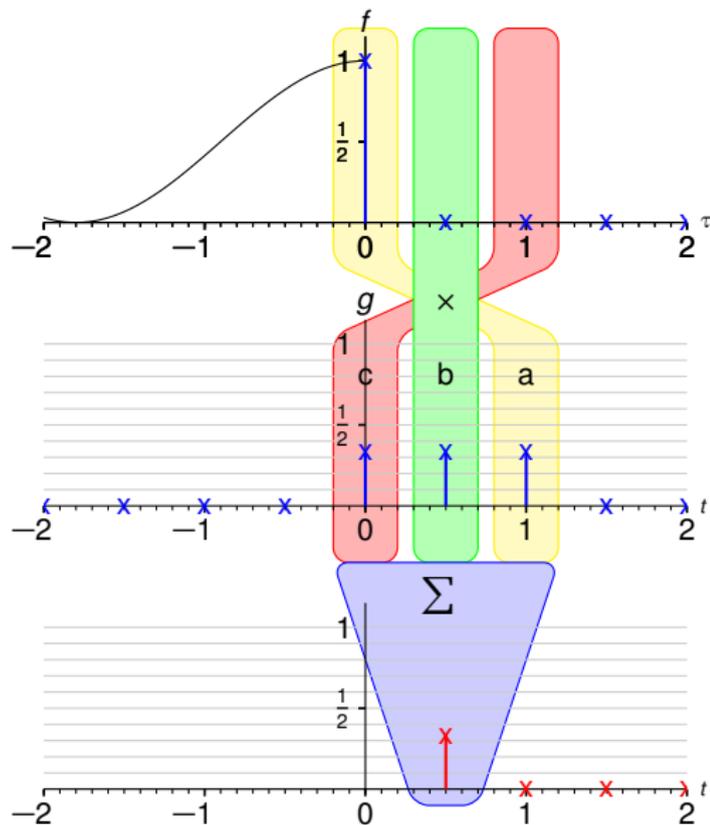


Beispiel: Faltung

■ $f(\tau)g(1-\tau)$

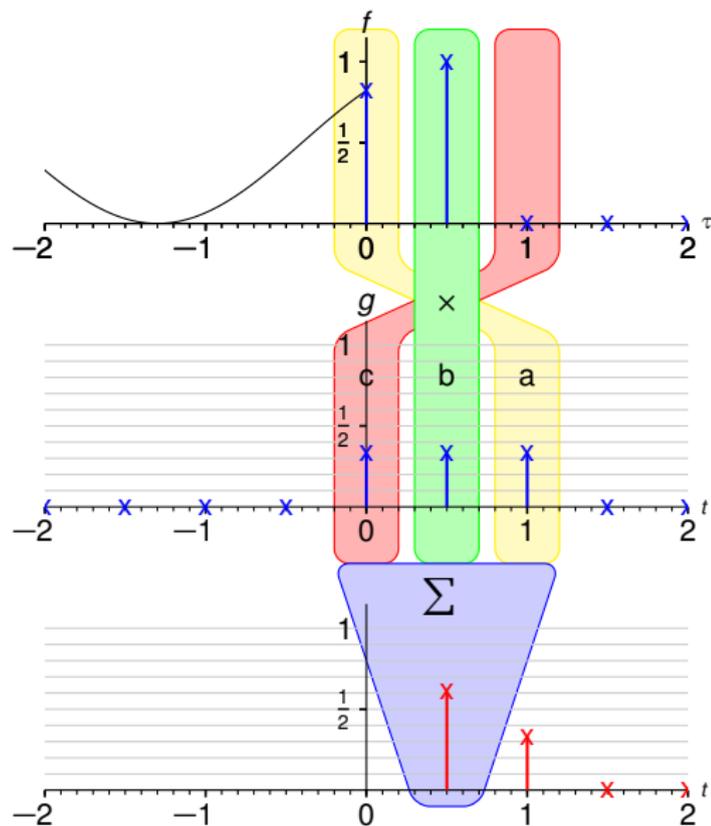


Beispiel: Diskrete Faltung



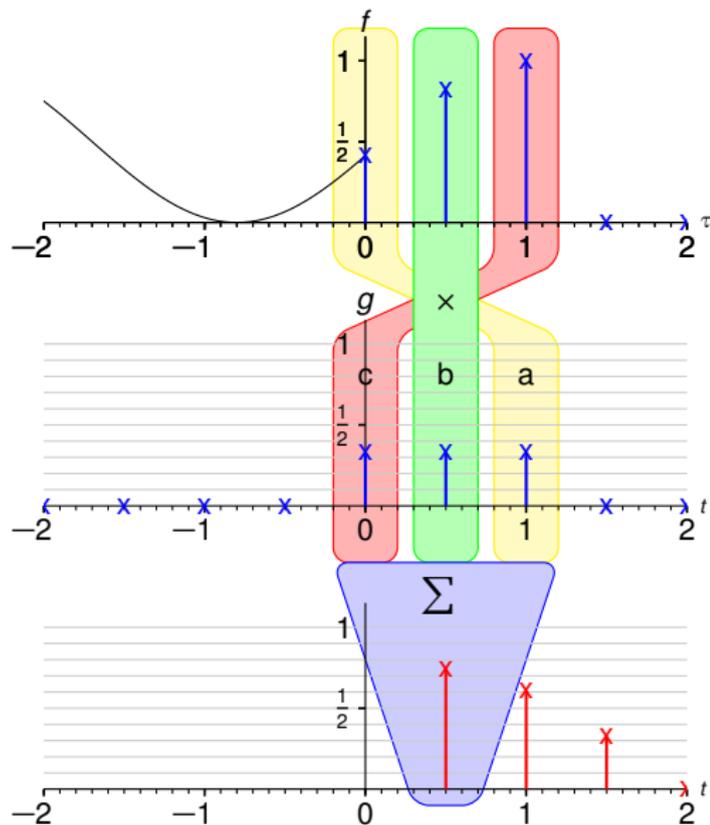
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

Beispiel: Diskrete Faltung



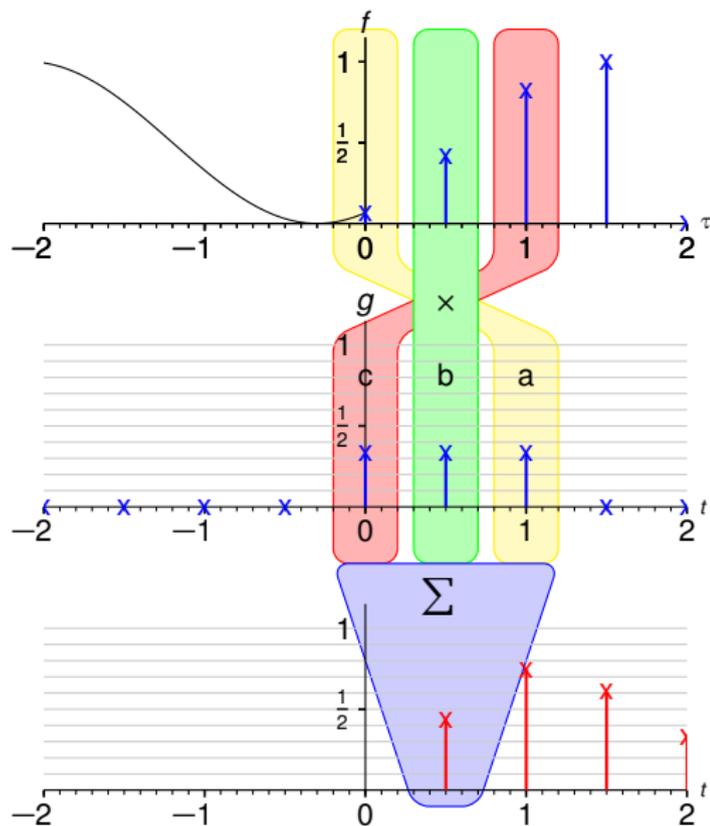
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

Beispiel: Diskrete Faltung



$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

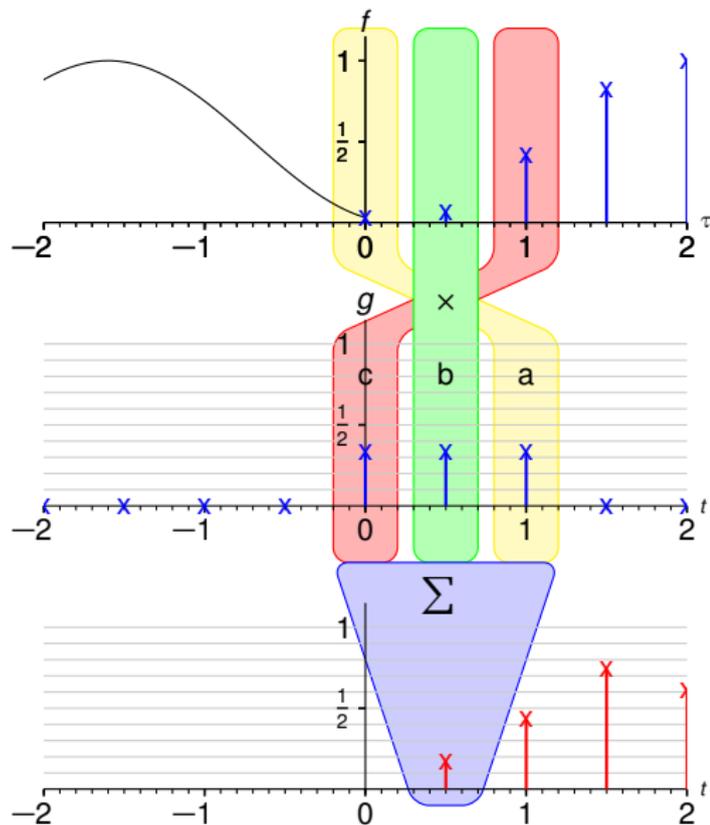
Beispiel: Diskrete Faltung



$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$



Beispiel: Diskrete Faltung



$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$



- Verkürzte Repräsentation eines Datensatzes (Bsp.: Git Commithash)
- Aus Eingabedaten errechnet
- Nutzen: Überprüfung der Datenintegrität
- Anforderungen
 - Stabil
 - Effizient berechenbar
 - Zuverlässige Fehlererkennung
- Beispiel: Quersumme mit Zehner-Restklasse:

$$\text{CHECK}(12345) = (1 + 2 + 3 + 4 + 5) \bmod 10 = 5$$



- Verkürzte Repräsentation eines Datensatzes (Bsp.: Git Commithash)
- Aus Eingabedaten errechnet
- Nutzen: Überprüfung der Datenintegrität
- Anforderungen
 - Stabil
 - Effizient berechenbar
 - Zuverlässige Fehlererkennung
- Beispiel: Quersumme mit Zehner-Restklasse:

$$\text{CHECK}(12345) = (1 + 2 + 3 + 4 + 5) \bmod 10 = 5$$

$$\text{CHECK}(12355) = (1 + 2 + 3 + 5 + 5) \bmod 10 = 6$$

~> Fehlermodell: Schützt vor allen Einzifferfehlern



Exkurs: Prüfsummen

- Verkürzte Repräsentation eines Datensatzes (Bsp.: Git Commithash)
- Aus Eingabedaten errechnet
- Nutzen: Überprüfung der Datenintegrität
- Anforderungen
 - Stabil
 - Effizient berechenbar
 - Zuverlässige Fehlererkennung
- Beispiel: Quersumme mit Zehner-Restklasse:

$$\text{CHECK}(12345) = (1 + 2 + 3 + 4 + 5) \bmod 10 = 5$$

$$\text{CHECK}(12355) = (1 + 2 + 3 + 5 + 5) \bmod 10 = 6$$

~> Fehlermodell: Schützt vor allen Einzifferfehlern

~> jedoch bspw. kein Schutz gegen Vertauschung:

$$\text{CHECK}(12354) = (1 + 2 + 3 + 5 + 4) \bmod 10 = 5$$



Exkurs: Prüfsummen

- Verkürzte Repräsentation eines Datensatzes (Bsp.: Git Commithash)
- Aus Eingabedaten errechnet
- Nutzen: Überprüfung der Datenintegrität
- Anforderungen
 - Stabil
 - Effizient berechenbar
 - Zuverlässige Fehlererkennung

- Beispiel: Quersumme mit Zehner-Restklasse:

$$\text{CHECK}(12345) = (1 + 2 + 3 + 4 + 5) \pmod{10} = 5$$

$$\text{CHECK}(12355) = (1 + 2 + 3 + 5 + 5) \pmod{10} = 6$$

↪ Fehlermodell: Schützt vor allen Einzifferfehlern

↪ jedoch bspw. kein Schutz gegen Vertauschung:

$$\text{CHECK}(12354) = (1 + 2 + 3 + 5 + 4) \pmod{10} = 5$$

- Wichtig: Fehlermodell, welche Arten von Bitfehlern werden erkannt?



- **Einzelnen Filterschritt** implementieren (kein Burst-Filter)
- Verwendung von **Q-Notation**
- Aspekte:
 - Nutzung abstrakter Schnittstellen
 - Einfluss von Schnittstellen auf Verlässlichkeit
 - Entwurfsentscheidungen und -abwägung in der Systemimplementierung

