Verlässliche Systemsoftware

Übungen zur Vorlesung

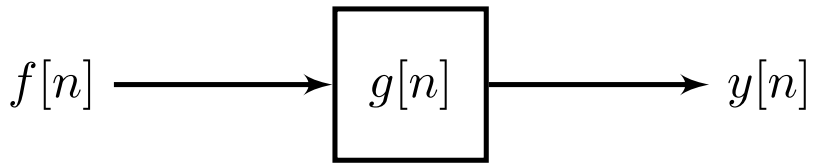
Hinweise zur Aufgabe: Filter

Phillip Raffeck, Simon Schuster, Peter Ulbrich

Technische Universität Dortmund
Lehrstuhl für Informatik 12 (Arbeitsgruppe Systemsoftware)
`https://sys.cs.tu-dortmund.de`

Sommersemester 2021

Hinweise zur Aufgabe Implementierung Filter

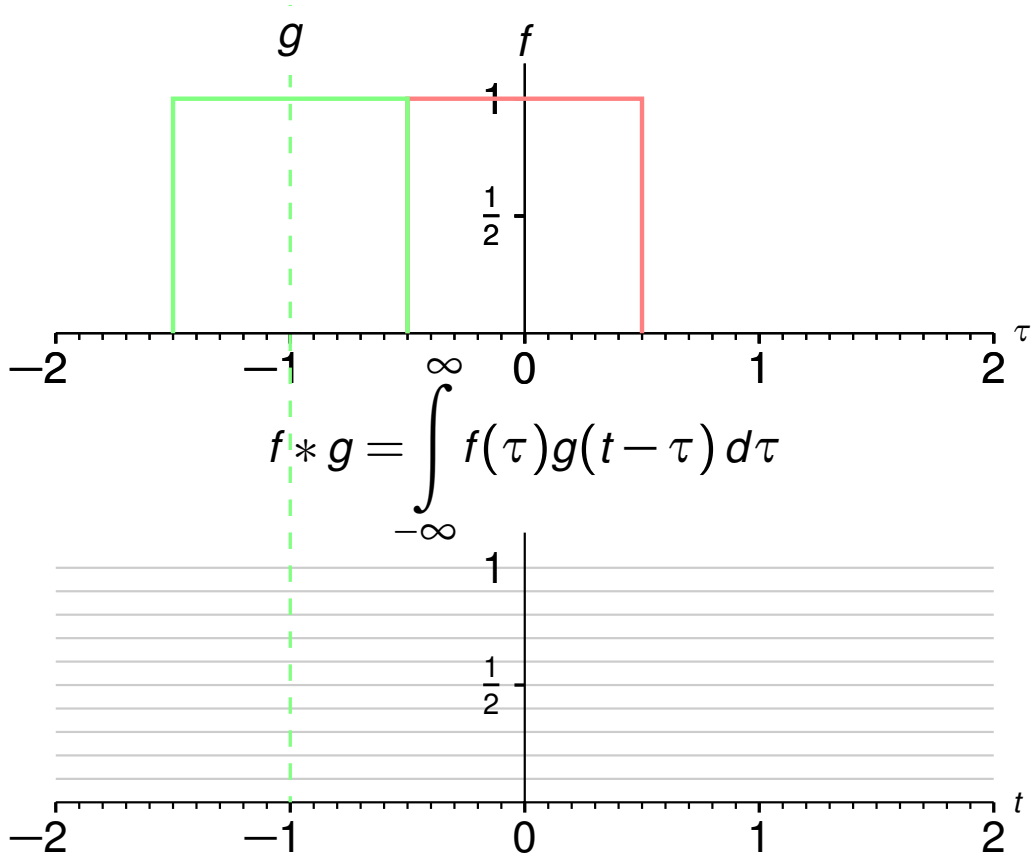


- Objekte identifizieren (z.B. Eingaben)
- Implementierung der Filterung durch **Faltung** (engl. convolution) mit Impulsantwort
  - f Signalwerte, g Filterwerte

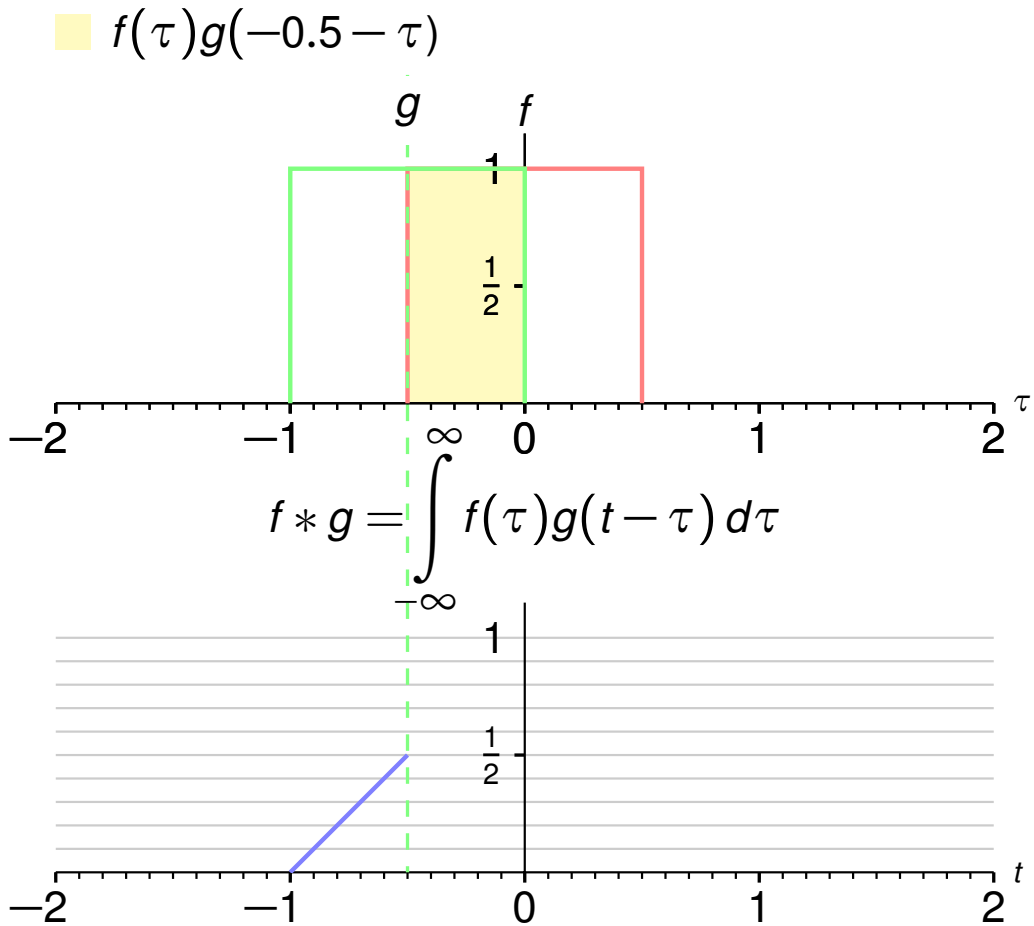
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m] \tag{1}$$

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

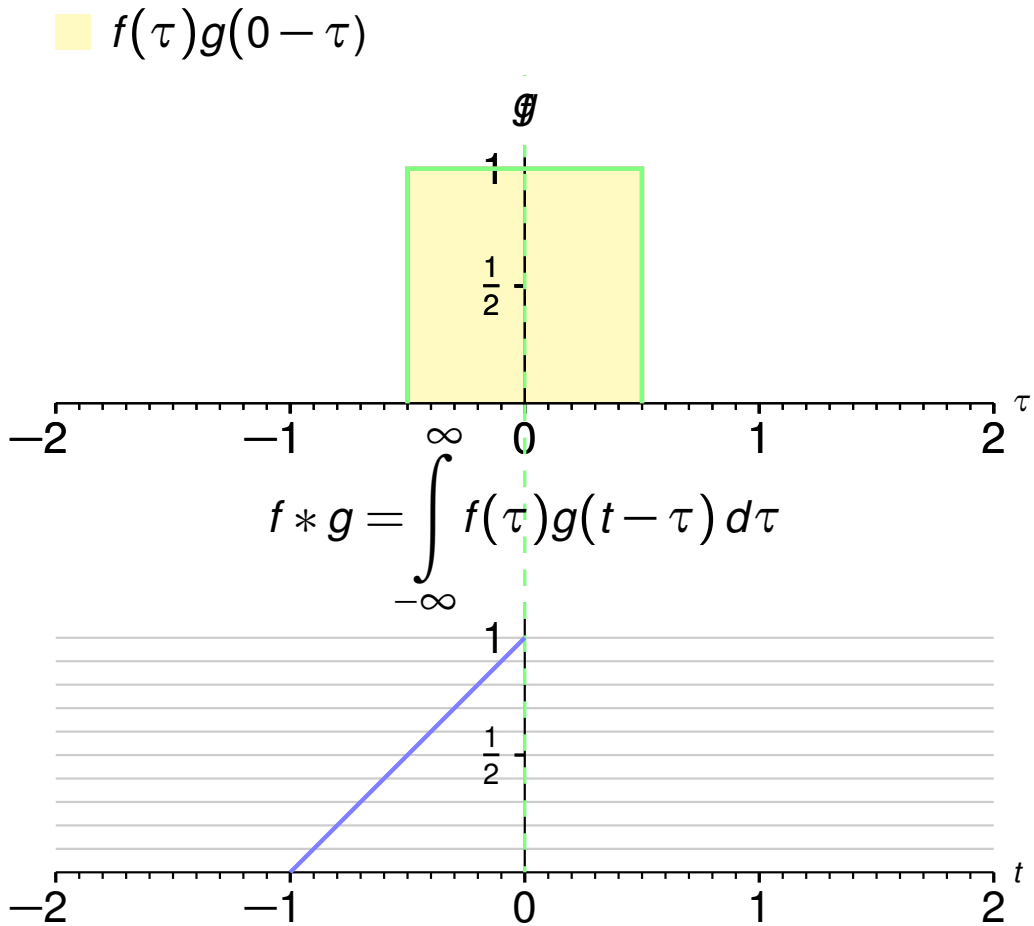
Beispiel: Faltung



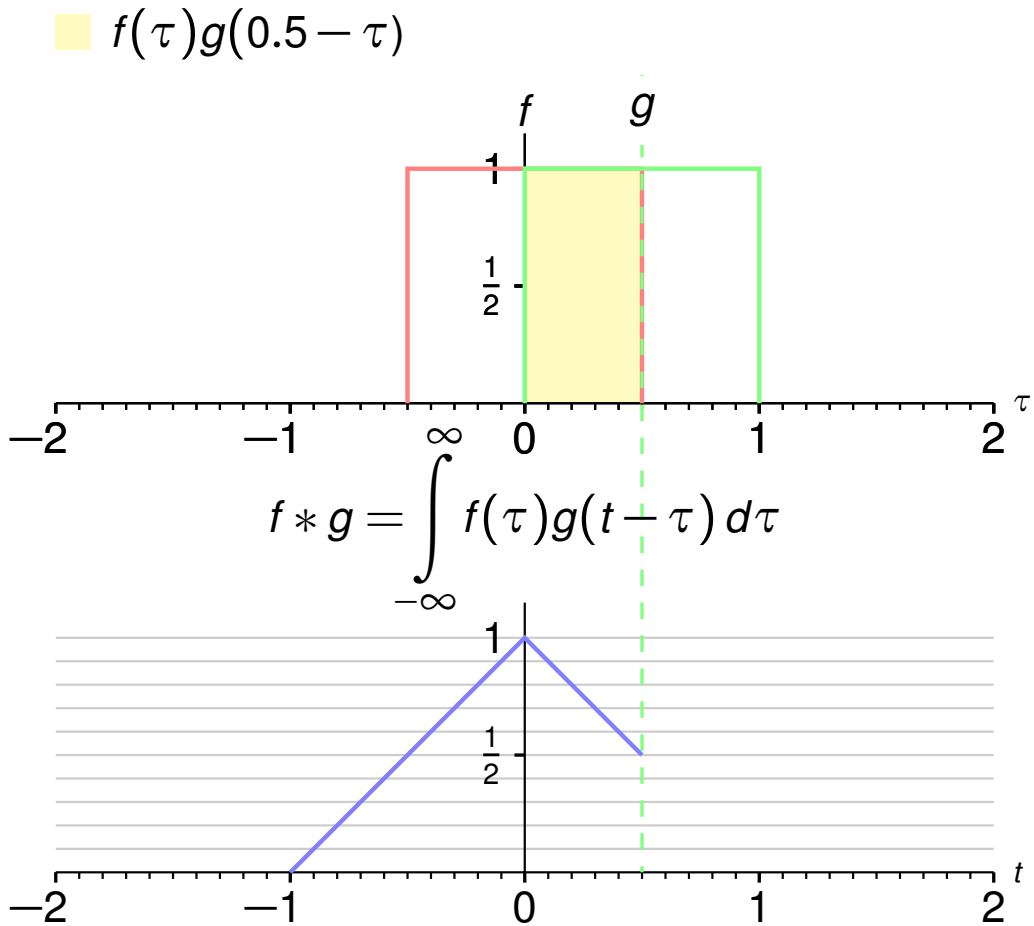
Beispiel: Faltung



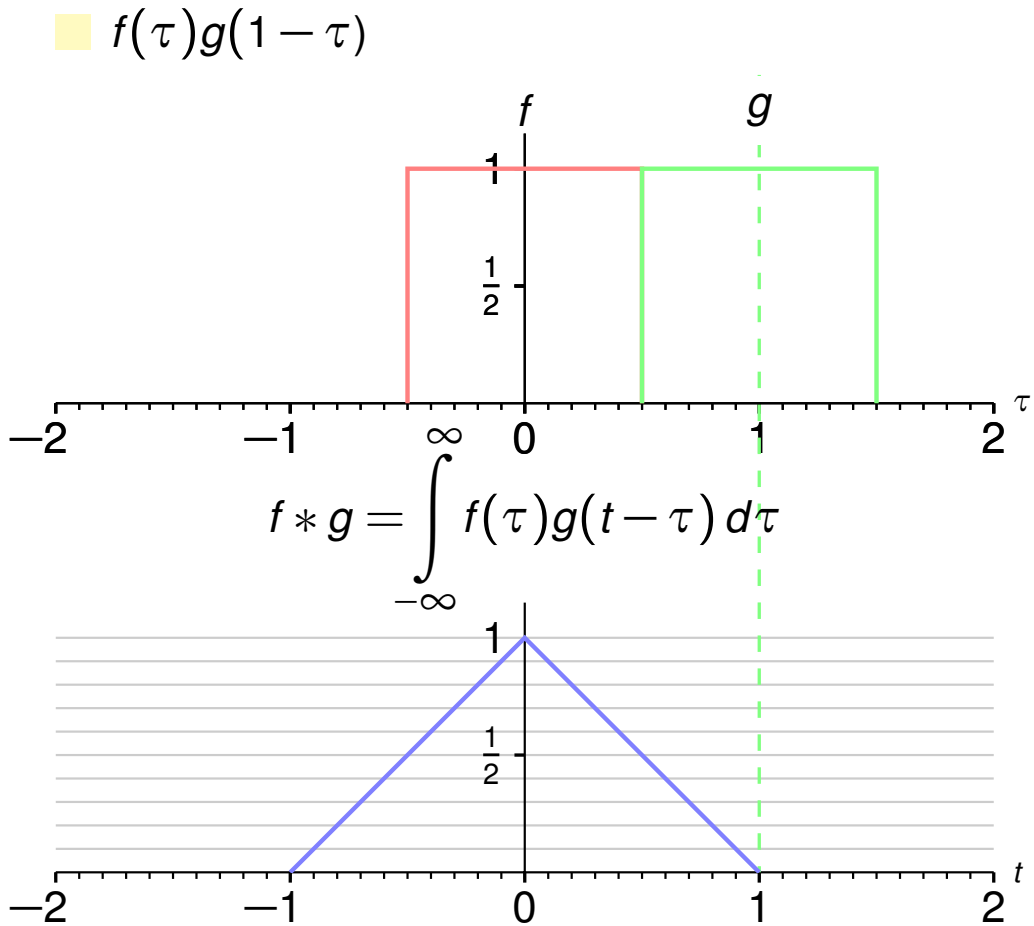
Beispiel: Faltung



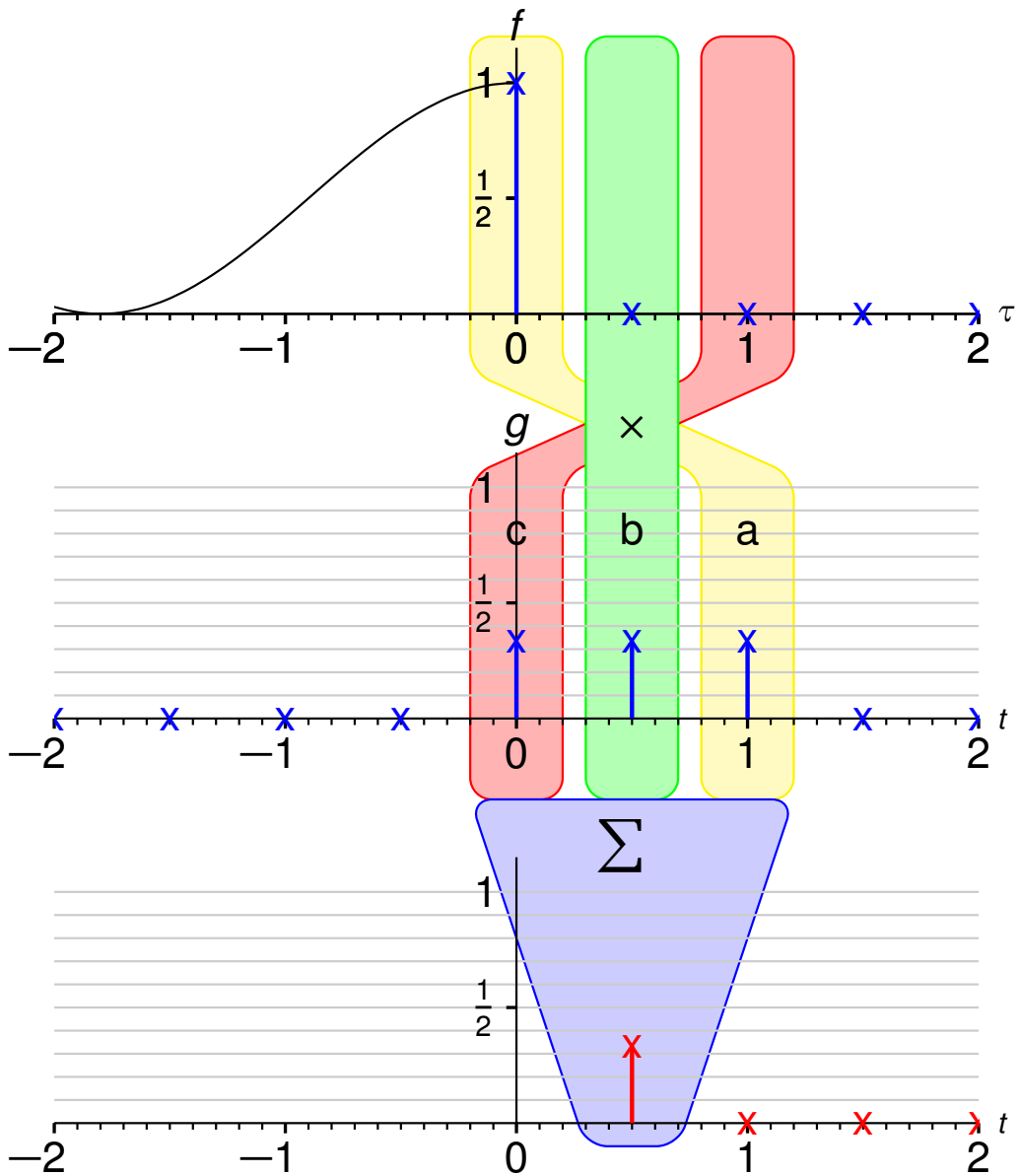
Beispiel: Faltung



Beispiel: Faltung

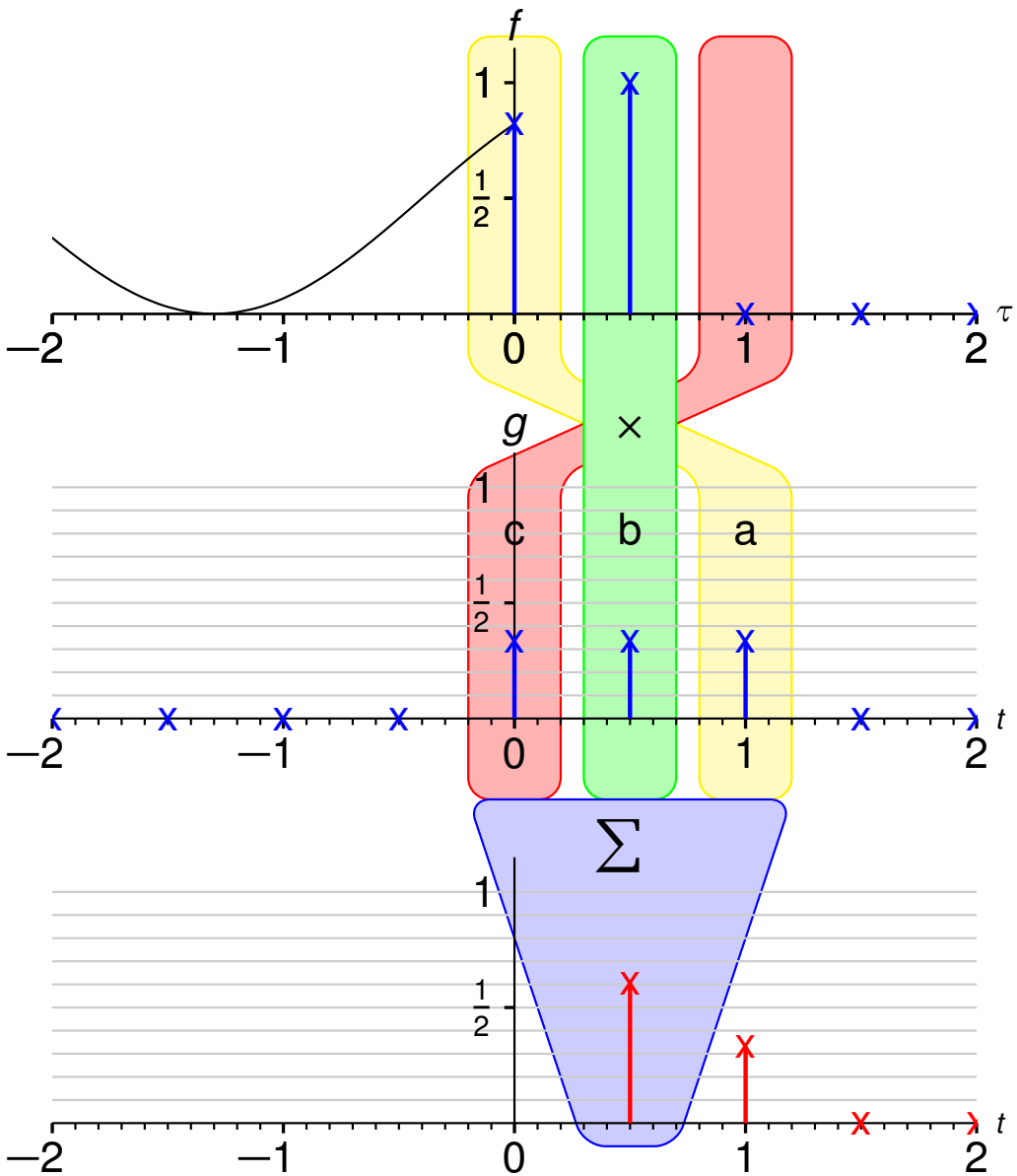


Beispiel: Diskrete Faltung



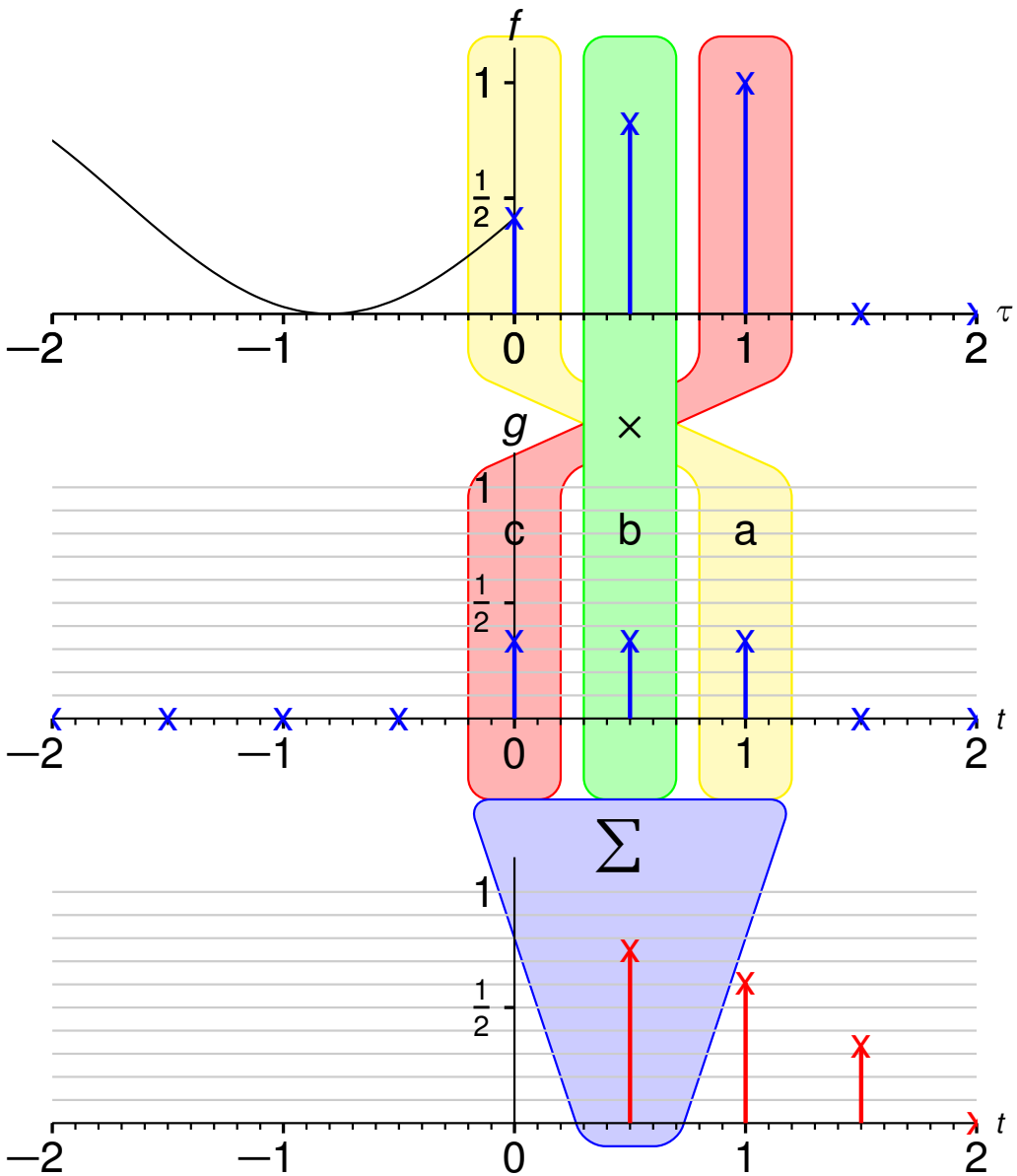
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

Beispiel: Diskrete Faltung



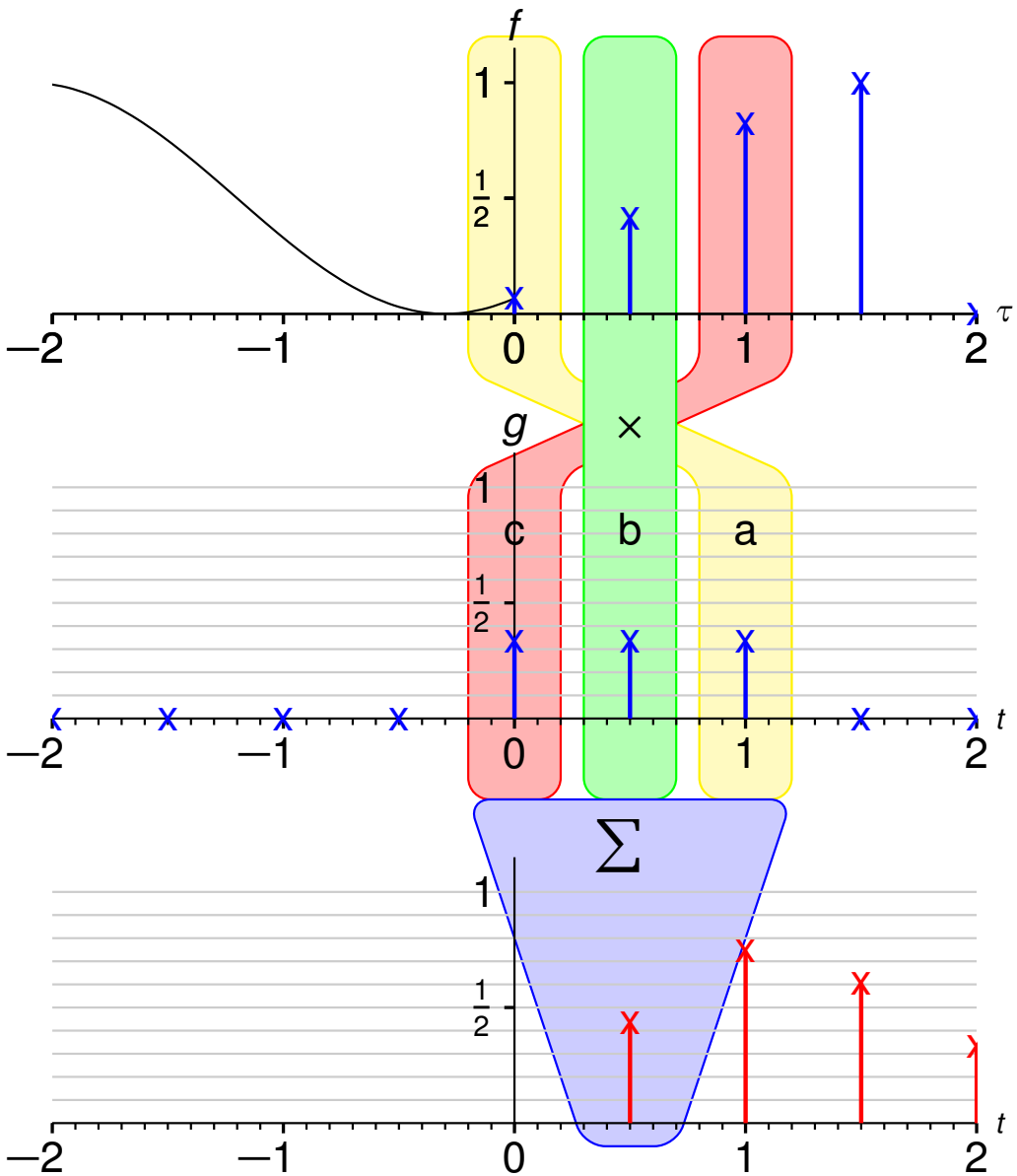
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

Beispiel: Diskrete Faltung



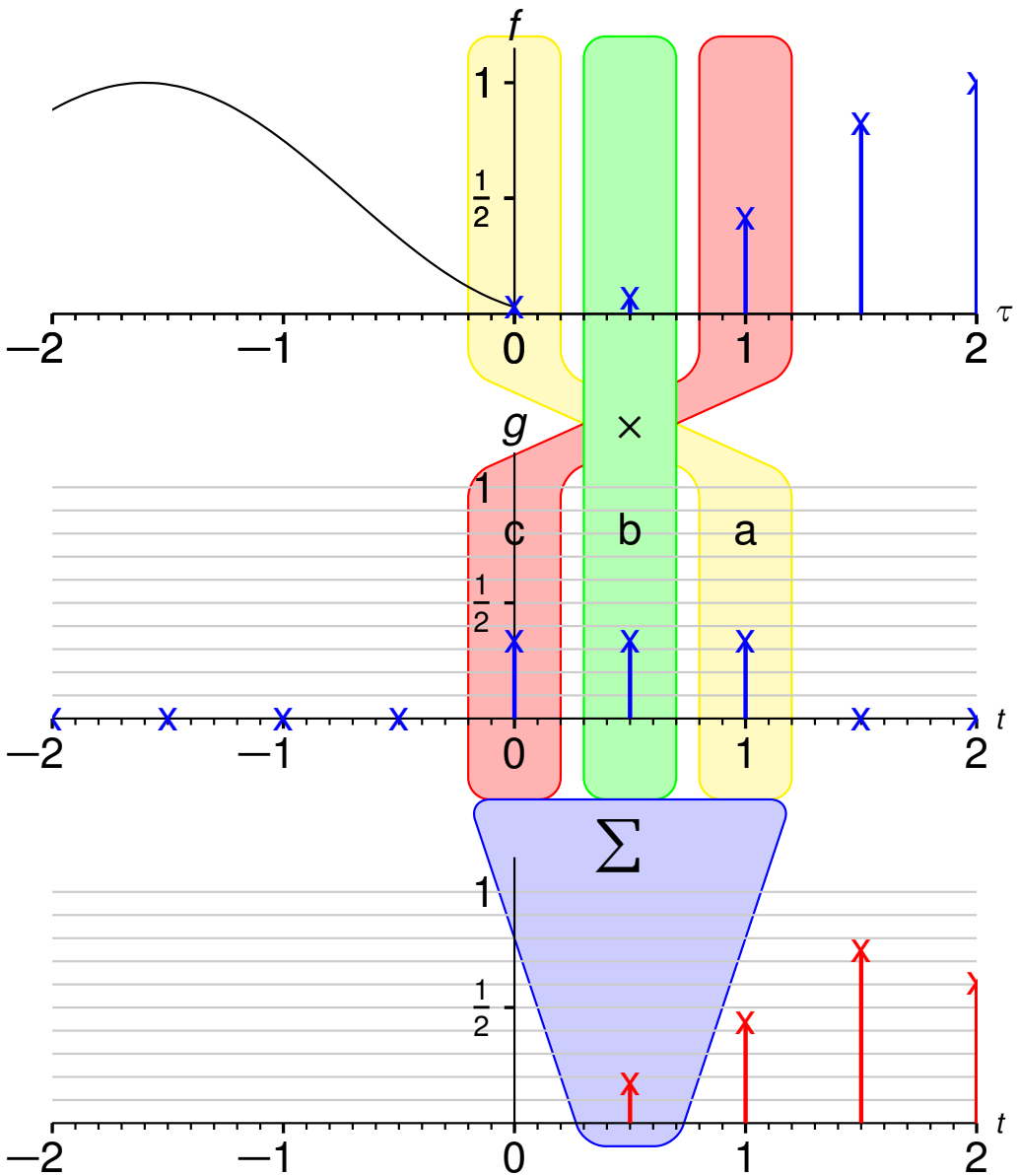
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m] \cdot g[n-m]$$

Beispiel: Diskrete Faltung



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

Beispiel: Diskrete Faltung



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

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:

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

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:

$$\mathrm{CHECK}(12345) = (1+2+3+4+5) \mod 10 = 5$$
$$\mathrm{CHECK}(12355) = (1+2+3+5+5) \mod 10 = 6$$

$\rightsquigarrow$ 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:

$$\mathrm{CHECK}(12345) = (1+2+3+4+5) \mod 10 = 5$$
$$\mathrm{CHECK}(12355) = (1+2+3+5+5) \mod 10 = 6$$

$\rightsquigarrow$ Fehlermodell: Schützt vor allen Einzifferfehlern

$\rightsquigarrow$ jedoch bspw. kein Schutz gegen Vertauschung:

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

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:

$$\mathrm{CHECK}(12345) = (1+2+3+4+5) \mod 10 = 5$$
$$\mathrm{CHECK}(12355) = (1+2+3+5+5) \mod 10 = 6$$

$\rightsquigarrow$ Fehlermodell: Schützt vor allen Einzifferfehlern
$\rightsquigarrow$ jedoch bspw. kein Schutz gegen Vertauschung:

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

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

Hinweise zur Aufgabe

- **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