Numerisches Programmieren (IN0019)

Frank R. Schmidt

Winter Semester 2016/2017

7. Fourier Transformation

ℂ-Vektorräume

Komplexe Zahlen

Die **komplexen Zahlen** $\mathbb{C}$ sind die reellen Zahlen $\mathbb{R}$, die um die **imaginäre Einheit** i adjungiert sind, wobei i eine Nullstelle von $x^2 + 1$ ist.

Jede komplexe Zahl $z \in \mathbb{C}$ lässt sich als $z = a + ib$ darstellen, wobei wir mit $a =: \Re(z)$ den **Realteil** und mit $b =: \Im(z)$ den **Imaginärteil** bezeichnen. Mit $\bar{z} := a - ib$ bezeichnen wir die **komplex konjugierte Zahl** und mit $|z| := \sqrt{a^2 + b^2}$ den **Betrag** von z.

Die komplexen Zahlen sind wie die reellen Zahlen ein **Körper** zusammen mit den folgenden Operationen:

$$\begin{aligned}(a+ib) + (c+id) &= (a+c) + i(b+d)\\ (a+ib)\cdot(c+di) &= (ac-bd) + i(ad+bc)\\ -(a+ib) &= (-a) + i(-b)\\ (a+ib)^{-1} &= \frac{a-ib}{a^2+b^2}\end{aligned}$$

Eulersche Formel

Da $\mathbb{C}$ ein Körper ist, kann man zu jedem $z \in \mathbb{C}$ die Reihe

$$\exp(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!}$$

berechnen und das Ergebnis in seinen Real- und Imaginärteil zerlegen.

Zusammen mit der **Eulerschen Formel**

$$\exp(ib) = \sum_{n=0}^{\infty} (-1)^n \frac{b^{2n}}{(2n)!} + i \sum_{n=0}^{\infty} (-1)^n \frac{b^{2n+1}}{(2n+1)!} = \cos(b) + i\sin(b)$$

erhalten wir

$$\exp(a+ib) = \exp(a)\cdot(\cos(b) + i\sin(b))$$

Polarkoordinaten

Jede komplexe Zahl $z \in \mathbb{C}$ besitzt eine **Polardarstellung**

$$z = |z| \cdot (\cos(\varphi) + i\sin(\varphi)) = |z| \cdot \exp(i\varphi)$$

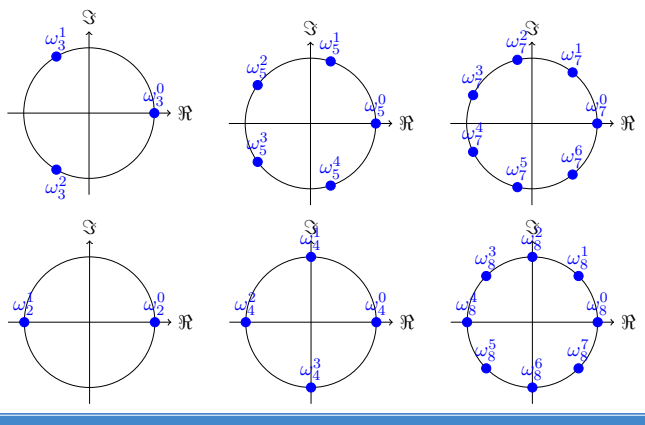
Damit erhalten wir

$$\begin{aligned}\bar{z} &= |z| \cdot \exp(-i\varphi)\\ z_1 \cdot z_2 &= |z_1|\,|z_2| \cdot \exp(i(\varphi_1 + \varphi_2))\\ z\bar{z} &= |z|^2\\ z^n &= |z|^n \cdot \exp(i(n\varphi))\end{aligned}$$

Insbesondere sind die **n-ten Einheitswurzeln**, d.h. die n Lösungen von $x^n = 1$:

$$1 = \omega_n^0, \omega_n^1, \ldots, \omega_n^{n-1} \qquad\qquad \omega_n := \exp\left(i\frac{2\pi}{n}\right)$$

Einheitswurzeln (Beispiele)



ℂ-Vektorraum

Da $\mathbb{C}$ ein Körper ist, können wir **$\mathbb{C}$-Vektorräume** W betrachten.

Jeder $\mathbb{C}$-Vektorraum W ist automatisch auch ein $\mathbb{R}$-Vektorraum, denn jede **$\mathbb{C}$-Skalarmultiplikation** $\lambda \cdot w$ definiert auch eine **$\mathbb{R}$-Skalarmultiplikation** (Wähle $\lambda \in \mathbb{R} \subset \mathbb{C}$). Allerdings gilt

$$\dim_{\mathbb{R}} W = 2 \cdot \dim_{\mathbb{C}} W$$

Interessanterweise kann man aber auch jeden **$\mathbb{R}$-Vektorraum** V in einen **$\mathbb{C}$-Vektorraum** W verwandeln:

$$V = \mathrm{span}_{\mathbb{R}}(v_1, \ldots, v_n) \qquad \Rightarrow \qquad W := \mathrm{span}_{\mathbb{C}}(v_1, \ldots, v_n).$$

mit $\dim_{\mathbb{C}} W = \dim_{\mathbb{R}} V$.

Man schreibt hierfür üblicherweise $W := V \otimes \mathbb{C}$.

Skalarprodukt

Für $\mathbb{C}$-Vektorräume W erfüllt ein Skalarprodukt $\langle \cdot, \cdot \rangle : W \times W \to \mathbb{C}$ folgende Eigenschaften:

$$\begin{aligned}
\langle \alpha x + \beta y, z \rangle &= \bar{\alpha} \langle x, z \rangle + \bar{\beta} \langle y, z \rangle && \text{(Semilinear in der ersten Komponente)} \\
\langle x, \alpha y + \beta z \rangle &= \alpha \langle x, y \rangle + \beta \langle x, z \rangle && \text{(Linear in der zweiten Komponente)} \\
\langle x, y \rangle &= \overline{\langle y, x \rangle} && \text{(Hermitesch)} \\
\langle x, x \rangle &\in \mathbb{R}_0^+ \\
\langle x, x \rangle = 0 &\Leftrightarrow x = 0 && \text{(Positiv-Definit)}
\end{aligned}$$

Ist $V = \operatorname{span}(v_1, \ldots, v_n)$ ein $\mathbb{R}$-Vektorraum mit $a_{ij} = \langle v_i, v_j \rangle$ so wird dadurch ein Skalarprodukt auf $W := V \otimes \mathbb{C}$ definiert.

Sei $w_1 = \sum_{i=1}^n \alpha_i v_i$ und $w_2 = \sum_{i=1}^n \beta_i v_i$, dann ist dieses W-Skalarprodukt

$$\langle w_1, w_2 \rangle = \langle \bar{\alpha}, A\beta \rangle = \bar{\alpha}^\top A \beta$$

Fourier Transformation

Polynom-Interpolation

Im Folgenden betrachten wir das **Interpolationsproblem** für ein Polynom

$$p(z) = \sum_{j=0}^{n-1} a_j z^j \in \Pi_{n-1}$$

mit den n **Einheitswurzeln als Stützstellen**:

$$p(\omega_n^j) = y_j \qquad\qquad j = 0, \ldots, n-1$$

Das führt zum linearen Gleichungssystem mit der **Vandermonde-Matrix** V:

$$V \cdot a = y \qquad\qquad V \in \mathbb{C}^{n \times n}, a, y \in \mathbb{C}^n$$

mit $v_{jk} = \omega_n^{j \cdot k}$ für $j, k = 0, \ldots, n-1$.

Diskrete Fourier Transformation

Bezeichnen wir mit $V^* := \bar{V}^\top$ die **adjungierte Matrix** bzgl. V, so gilt

$$(V^*V)_{jk} = \sum_{m=0}^{n-1} v_{jm}^* v_{mk} = \sum_{m=0}^{n-1} \omega_n^{-(jm)+mk} = \sum_{m=0}^{n-1} \left(\omega_n^{k-j}\right)^m = n \cdot I_{jk}$$

Das heißt wir haben $V^{-1} = \frac{1}{n} V^*$ und es gilt

$$a_k = \frac{1}{n} \sum_{j=0}^{n-1} y_j \bar{\omega}_n^{jk} \qquad\qquad y_k = \sum_{j=0}^{n-1} a_j \omega_n^{jk}$$

Die erste Operation ist die **Diskrete Fourier Transformation (DFT)** und die zweite ist die **inverse DFT (IDFT)**.

In der Literatur werden manchmal andere Faktoren benutzt, d.h. 1 und $\frac{1}{n}$ sind vertauscht oder in beiden Fällen wird $\frac{1}{\sqrt{n}}$ benutzt.

Fourier-Reihe

Trigonometrische Orthonormalbasis

Die Name der Fourier-Transformation hat seinen Ursprung in der **Fourier-Reihe**, die auf folgendem Skalarprodukt beruht:

$$\langle f, g \rangle := \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) g(x) \, \mathrm{d}x \,.$$

Dieses Skalarprodukt kann zur Approximation von Funktionen benutzt werden. Man **projiziert** dann eine Funktion f auf den Vektorraum, der von endlich vielen Funktionen f_i aufgespannt ist.

Man kann zeigen, dass bzgl. des obigen Skalarprodukts die folgenden Funktionen eine **Orthonormalbasis** bilden:

$$\left\{ \frac{1}{\sqrt{2}}, \cos(x), \sin(x), \ldots, \cos(nx), \sin(nx) \right\}$$

Damit erhalten wir also $f(x) \approx a_0 + \sum_{k=1}^n a_k \cos(kx) + b_k \sin(kx)$.

Koeffizienten der Fourier-Reihe

Die Koeffizienten dieser **Fourier-Reihe**

$$f_n(x) = a_0 + \sum_{k=1}^{n} a_k \cos(kx) + b_k \sin(kx)$$

erhält man, indem man f mit den **Basisfunktionen** multipliziert.

Die Fourier-Koeffizienten sind also

$$\langle f(x), \cos(kx) \rangle = a_k \qquad \langle f(x), \sin(kx) \rangle = b_k \qquad \langle f(x), 1 \rangle = \frac{a_0}{2}$$

Diese Darstellung wird benutzt, um hohe Frequenzen in Signalen zu entfernen. Das ist der wesentliche Unterschied zwischen WAV- und MP3-Dateien.

Setzt man $c_k := a_k + i b_k$, so kann man die Notationen vereinfachen. Dazu verwandelt man den $\mathbb{R}$-Vektorraum $C^0([-\pi, \pi], \mathbb{R})$ in den $\mathbb{C}$-Vektorraum $C^0([-\pi, \pi], \mathbb{C})$.

Komplexe Orthonormalbasis

Betrachten wir nun das **Skalarprodukt**

$$\langle f, g \rangle := \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \bar{g}(x) \, \mathrm{d}x,$$

so bilden folgenden Funktionen eine **Orthonormalbasis**:

$$\{\exp(-inx), \ldots, \exp(-ix), 1, \exp(ix), \ldots, \exp(inx)\}$$

Damit erhalten wir also

$$f(x) \approx \sum_{k=-n}^{n} a_k \exp(ikx)$$

mit

$$a_k = \langle f, \exp(ikx) \rangle = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \exp(-ikx) \, \mathrm{d}x \,.$$

Kontinuierliche Fourier-Transformation

Motiviert durch diese Beobachtung, definiert man die kontinuierliche, zyklische **Fourier-Transformation** als

$$\hat{f}(\alpha) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-i\alpha x}\,\mathrm{dx}$$

Benutzt man die Trapezregel als Diskretisierung, so erhält man mit der Notation $x_j = -\pi + \frac{2\pi}{n}j$ und $\omega_n = \exp\left(\frac{2\pi}{n}i\right)$ gerade

$$\hat{f}\left(\frac{2\pi}{n}j\right) \approx \frac{1}{2\pi}\sum_{j=0}^{n-1} f(x_j)\bar{\omega}_n^j \cdot \frac{2\pi}{n}$$

Dies entspricht genau der **diskreten Fourier-Transformation.**

FFT

Motivation der FFT

Die IDFT

$$y_k = \sum_{j=0}^{n-1} a_j\omega_n^{jk}$$

lässt sich als Matrix-Vektor-Multiplikation darstellen, d.h. sie kann in $\mathcal{O}(n^2)$ berechnet werden. Das Ziel der **schnellen Fourier-Transformation (FFT)** besteht darin, diese Berechnung in $\mathcal{O}(n\log(n))$ durchzuführen.

Idee Zerlege die Summen geschickt in zwei Teilsummen halber Länge, aus denen sich die ursprünglichen Summen leicht gewinnen lassen.

Voraussetzung Es wird also in jedem Schritt benötigt, dass n gerade ist, d.h. insgesamt $n = 2^N$.

Zerlegung der IDFT

Es sei $n = 2m$. Dann gilt:

$$\begin{aligned} y_k &= \sum_{j=0}^{2m-1} a_j\omega_n^{jk} \\ &= \sum_{j=0}^{m-1} a_{2j}\omega_n^{2jk} + \sum_{j=0}^{m-1} a_{2j+1}\omega_n^{(2j+1)k} \\ &= \sum_{j=0}^{m-1} a_{2j}\omega_m^{jk} + \omega_n^k \cdot \sum_{j=0}^{m-1} a_{2j+1}\omega_m^{jk} \end{aligned}$$

Die erste Summe ist die m-elementige **IDFT der geraden Koeffizienten.**

Die zweite Summe ist die m-elementige **IDFT der ungeraden Koeffizienten.**

Rekursive IDFT

Angenommen, dass $n = 2m$ und wir haben die IDFT bzgl. m berechnet:

$$\hat{a}_{2k} = \sum_{j=0}^{m-1} a_{2j}\omega_m^{jk} \qquad \hat{a}_{2k+1} = \sum_{j=0}^{m-1} a_{2j+1}\omega_m^{jk}$$

Dann gilt für $k < m$

$$y_k = \sum_{j=0}^{m-1} a_{2j}\omega_m^{jk} + \omega_n^k \cdot \sum_{j=0}^{m-1} a_{2j+1}\omega_m^{jk} = \hat{a}_{2k} + \omega_n^k \cdot \hat{a}_{2k+1}$$

$$y_{k+m} = \sum_{j=0}^{m-1} a_{2j}\omega_m^{j(k+m)} + \omega_n^{k+m} \cdot \sum_{j=0}^{m-1} a_{2(k+m)+1}\omega_m^{j(k+m)} = \hat{a}_{2k} - \omega_n^k \cdot \hat{a}_{2k+1}$$

Diese Operation wird auch **Butterfly-Schritt** genannt.
Falls die IDFT für die beiden m-elementigen Vektoren berechnet wurde, benötigen wir lediglich $\mathcal{O}(n)$ für den **Kombinationsschritt.**

Laufzeit IFFT

Ohne Rekursion benötigen wir jeweils n^2 Operationen ($+$ sowie $\cdot$).

Wird die Rekursion einmal benutzt, ist die Anzahl der Operationen

$$2\left(\frac{n}{2}\right)^2 + n = \frac{1}{2}n^2 + n$$

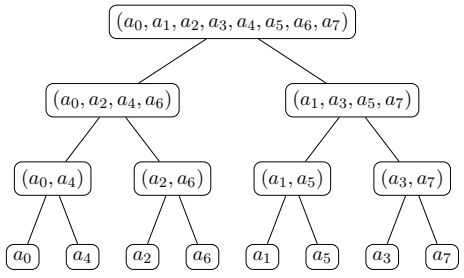
Wird die Rekursion k-mal benutzt, ist die Anzahl der Operationen

$$2^k\left(\frac{n}{2^k}\right)^2 + k \cdot n = \frac{1}{2^k}n^2 + k \cdot n$$

Ist $n = 2^N$, ist die Anzahl der Operationen (bei N Rekursionen)

$$2^N\left(\frac{n}{2^N}\right)^2 + N \cdot n = n + \log(n) \cdot n$$

Aufrufabfolge der Rekursion



Id	0	4	2	6	1	5	3	7
Binär	000	100	010	110	001	101	011	111
Bit-Reversal	000	001	010	011	100	101	110	111
Dezimal	0	1	2	3	4	5	6	7

Iterative IFFT

Um die rekursive IFFT-Berechnung in eine iterative IFFT-Berechnung zu verwandeln, zerlegen wir die Berechnung in drei Phasen.

In der ersten Phase (**Sortierung**) permutieren wir die Einträge a_i derart, dass sie bzgl. ihres **Bit-Reversals** $\pi(i)$ sortiert sind.

In der zweiten Phase (**Kombinationsphase**) werden genau $\log(n)$ Kombinationsschritte durchgeführt.

In der dritten Phase (**Inverse Sortierung**) permutieren wir die Einträge a_i, so dass sie wieder bzgl. i sortiert sind.

Da beim **Bit-Reversal** lediglich paarweise die Einträge ausgetauscht werden, ist die erste Phase mit der dritten Phase identisch.

Das **Bit-Reversal** kann in Linearzeit berechnet werden. Hierfür iterieren wir über alle Einträge i und führen einen Austausch zwischen i und $\pi(i)$ durch genau wenn $\pi(i) > i$.

Faltung

Polynom-Multiplikation

Man kann die diskrete Fourier-Transformation auch benutzen, um zwei Polynome effizient zu multiplizieren. Sei hierzu

$$p(x) = \sum_{k=0}^{n} a_k x^k \qquad q(x) = \sum_{k=0}^{n} b_k x^k$$

Dann gilt für das Produkt $r := p \cdot q \in \Pi_{2n}$ gerade

$$r(x) = \sum_{k=0}^{2n} c_k x^k \qquad c_k = \sum_{j=0}^{k} a_j b_{k-j}$$

Diese Berechnung benötigt $\mathcal{O}(n^2)$ Operationen.

Allerdings wird r eindeutig durch $2n$ **verschiedene Stützstellen** definiert. Das ist die Kernidee, um das Produkt mit Hilfe der FFT zu berechnen.

Polynom-Multiplikation mittels FFT

Es seien die **Koeffizientenvektoren** $a, b \in \mathbb{R}^n$ gegeben, die die Polynome p und q beschreiben. Nun möchte man den **Koeffizientenvektor** $c \in \mathbb{R}^{2n}$ finden, der das Polynom $r = p \cdot q$ beschreibt.

```
1   // Füllen mit Nullen
2   for (j=0; j<n; j++) { a[n+j] = 0; b[n+j] = 0; }
3   // IFFT
4   A = ifft(a); B = ifft(b);
5   // Punktweise Multiplikation
6   for (j=0; j<2*n; j++) {
7     C[j] = A[j]*B[j]; // Komplexe Multiplikation
8   }
9   // FFT
10  c = fft(C);
```

Die Anzahl der **Flops** ist

$$2n + 4n \cdot \log(2n) + 2n + 2n \cdot \log(2n) = \mathcal{O}(n \cdot \log(n))$$

Faltung

Ähnlich wie die Fourier-Transformation hat auch diese Operation ein kontinuierliches Analogon, die sogenannte **Faltung**. Hierzu seien zwei stetige Funktionen $f, g\colon [-\pi, \pi] \to \mathbb{R}$ gegeben, die **periodisch fortgesetzt** werden, d.h.

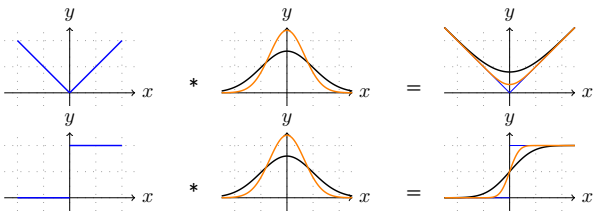
$$f(x + 2k\pi) = f(x) \qquad g(x + 2k\pi) = g(x) \qquad \forall k \in \mathbb{Z}$$

Wir schreiben die Faltung von f und g als

$$\begin{aligned} f * g\colon [-\pi, \pi] &\to \mathbb{R} \\ x &\mapsto \int_{-\pi}^{\pi} f(y) g(x - y)\, \mathrm{d}y \end{aligned}$$

und es gilt $f * g = g * f$.

Faltung (Beispiel)



Sei $g\colon \mathbb{R} \to \mathbb{R}$ eine positive C^∞-Funktion mit $\int g(x)\, \mathrm{d}x = 1$, so ist $f * g$ eine C^∞-Approximation von f. Die Operation nennt man auch **Glättung**.

Fourier-Transformation der Faltung

Für die Fourier-Transformation der Faltung erhalten wir

$$\begin{aligned} \widehat{f * g}(\alpha) &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f * g(x) e^{-i\alpha x}\, \mathrm{d}x \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f(y) g(x - y) e^{-i\alpha y} e^{-i\alpha(x-y)}\, \mathrm{d}x\, \mathrm{d}y \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f(y) g(x) e^{-i\alpha y} e^{-i\alpha x}\, \mathrm{d}x\, \mathrm{d}y \\ &= \frac{1}{2\pi} \left[\int_{-\pi}^{\pi} g(x) e^{-i\alpha x}\, \mathrm{d}x \right] \left[\int_{-\pi}^{\pi} f(y) e^{-i\alpha y}\, \mathrm{d}y \right] \\ &= 2\pi \hat{f}(\alpha) \cdot \hat{g}(\alpha) \end{aligned}$$

Insgesamt verhält sich die **Fourier-Transformation der Faltung** ähnlich zur **diskreten Fourier-Transformation der Polynom-Multiplikation**.

Verallgemeinerungen

Das Konzept der Fourier-Transformation und der Faltung kann man auch für höhere Dimension verallgemeinern. Z.B. gilt für die Dimension $d = 2$ und Funktionen $f, g\colon [-\pi, \pi] \times [-\pi, \pi] \to \mathbb{R}$ gerade

$$\begin{aligned} f * g\colon [-\pi, \pi] \times [-\pi, \pi] &\to \mathbb{R} \\ (x_1, x_2) &\mapsto \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f(y_1, y_2) g(x_1 - y_1, x_2 - y_2)\, \mathrm{d}y_1\, \mathrm{d}y_2 \end{aligned}$$

sowie

$$\hat{f}(\alpha_1, \alpha_2) = \frac{1}{(2\pi)^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} f(y_1, y_2) e^{-\langle \alpha, y \rangle}\, \mathrm{d}y_1\, \mathrm{d}y_2$$

und es gilt

$$\widehat{f * g}(\alpha) = (2\pi)^2 \cdot \hat{f}(\alpha) \cdot \hat{g}(\alpha)$$

Bildglättung (Beispiel)

Laufzeit: 1.94 Sekunden (ohne FFT)
Laufzeit: 0.17 Sekunden (mit FFT)


$I\colon \Omega \to \mathbb{R}^3$


$P\colon \Omega_0 \to \mathbb{R}^3$

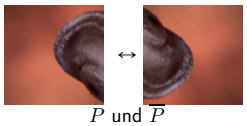

$\overline{P}\colon \Omega_0 \to \mathbb{R}^3$

Wir suchen nach dem Pixelort $x \in \Omega \subset \mathbb{R}^2$ im Bild $I\colon \Omega \to \mathbb{R}^3$, in dem sich das Zentrum des Bildausschnitts $P\colon \Omega_0 \to \mathbb{R}^3$ befindet, d.h. wir sind auf der Suche nach dem Minimum von:

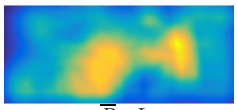
$$\begin{aligned} E(x) &= \int_{\Omega_0} [P(y) - I(x+y)]^2 \,\mathrm{dy} = \int_{\Omega_0} \left[\overline{P}(y) - I(x-y)\right]^2 \mathrm{dy} \\ &= \int_{\Omega_0} \overline{P}(y)^2 - 2\overline{P}(y) I(x-y) + I(x-y)^2 \,\mathrm{dy} \\ &= \mathrm{const} + \left[-2(\overline{P} * I) + I^2 * 1\right](x) \end{aligned}$$

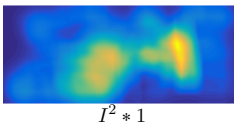
Eingabe:


I

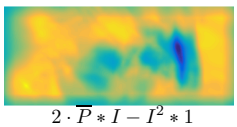

P und $\overline{P}$

FFT:


$\overline{P} * I$


$I^2 * 1$

Ausgabe:


$2 \cdot \overline{P} * I - I^2 * 1$


Ergebnis