FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachbereich Elektrotechnik
und Informationstechnik

Prof. Georg Hoever

SS 2025

4. Praktikum zur Höhere Mathematik 2 für (Wirtschafts-)Informatik

Ziel dieses Praktikums ist die Implementierung einer Fourier-Hin- und Rück-Transformation.

Die originalen Daten und transformierten Werte sollen dabei als `vector<CKomplex>` mit der Template-Klasse `vector` der Standardbibliothek und einer eigenen Klasse `CKomplex` für komplexe Zahlen abgelegt werden.

Hinweis: Die Implementierung der Methoden zu den Aufgabe 1 und 3 sowie die in Aufgabe 2 genannten Funktionen sollen gemeinsam in **einer** `cpp`-Datei enthalten sein.

1. Aufgabe

Damit man komfortabel komplexe Fourierkoeffizienten berechnen kann, soll eine Klasse CKomplex realisiert werden. Implementieren Sie dazu

- private Attribute für den Real- und Imaginärteil,
- einen Konstruktor `CKomplex(double a, double b)` mit zwei Argumenten a, b für die Darstellung von $a + b\mathrm{j}$,
- einen Konstruktor `CKomplex(double phi)` mit einem Argument φ zur Erzeugung von $e^{\mathrm{j}\varphi}$,
- öffentliche Methoden `double re()` und `double im()`, die den Real- und Imaginärteil zurückgeben,
- (überladene) Methoden `+` und `*` für die Addition und Multiplikation von komplexen Zahlen,
- eine weitere (überladene) Methode `*` für die Multiplikation einer `double`-Zahl mit einer komplexen Zahl,
- eine Methode `double abs()`, die den Betrag der komplexen Zahl zurück gibt.

2. Aufgabe

Auf meiner Internetseite habe ich zwei Funktionen

- `vector<CKomplex> werte_einlesen(const std::string dateiname)`
- `werte_ausgeben(const std::string dateiname, vector<CKomplex> werte, double epsilon=-1.0)`

zur Verfügung gestellt, die Vektoren aus komplexen Zahlen aus einer Textdatei einlesen bzw. in eine Textdatei ausgeben können. Dabei werden die Vektoren im sparse-Format gespeichert, d.h. es werden zeilenweise der Index und der zugehörige Wert (Real- und Imaginärteil) abgespeichert, so dass man Null-Einträge sparen kann. Der erste Eintrag in der Datei gibt die Dimension des Vektors an.

Zur Komprimierung der Daten kann beim Ausgeben ein $\varepsilon \geq 0$ gewählt werden. Es werden nur Einträge abgespeichert, die größer als ε sind.

Integrieren Sie die Funktionen in Ihr Projekt und testen Sie das Ein- und Ausgeben an den auf meiner Seite verfügbaren Dateien `Daten_original1.txt` und `Daten_original2.txt`.

3. Aufgabe

Realisieren Sie Methoden, die zu einem Vektor aus komplexen Werten die Fourier-Hin- und Rück-Transformation berechnen. Sie können das auch in einer Funktion realisieren, der Sie als Parameter mitgeben, ob die Hin- oder die Rück-Transformation berechnet werden soll.

4. Aufgabe

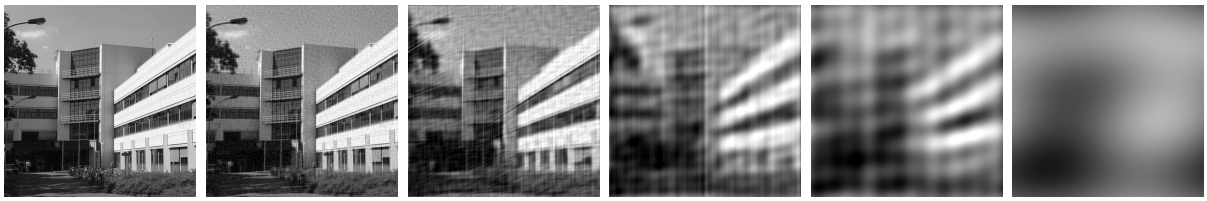
Testen Sie Ihre Methoden und daraus resultierende komprimierte Speichermöglichkeiten wie folgt zum einen mit den Daten aus der Datei `Daten_original1.txt` und zum anderen mit der aus der Datei `Daten_original2.txt`.

- Lesen Sie die Werte aus der Datei ein und führen Sie eine Fouriertransformation der eingelesenen Daten durch.
- Speichern Sie die Daten jeweils
  - mit dem Default-Wert für `epsilon`,
  - mit $\varepsilon = 0.001$, mit $\varepsilon = 0.01$, mit $\varepsilon = 0.1$, mit $\varepsilon = 1.0$

  in einzelnen Dateien ab.
- Vergleichen Sie die Größen der einzelnen Dateien.
- Laden Sie die Werte aus den einzelnen Dateien und führen Sie die jeweilige Rücktransformation durch.
- Berechnen Sie jeweils die maximale Abweichung von zurück-berechneten und originalen Werten. (Tipp: Dazu bietet sich eine eigene Funktion an!)

5. Aufgabe

Ziel der Aufgabe ist, dass Sie ausgehend von einem Bild Ihrer Wahl eine Serie von zunehmend verschwommeneren Bildern ähnlich den folgenden Bildern erzeugen.



Auf meiner Internetseite finden Sie dazu ein Programm „image2array“, das aus einer Bild-Datei die Daten in eine `txt`-Datei transformiert und umgekehrt, so dass Sie Ihre Praktikums-Funktionen nutzen können, um zu experimentieren, was passiert, wenn man kleine Fourierkoeffizienten weglässt.

Zum Programm „image2array“:

> Das Programm steht als Python-Skript sowie als exe-Programm zur Verfügung. Es kann die Bildformate `*.jpg`, `*.png` und `*.bmp` verarbeiten. Bei der Transformation eines Bildes in eine Daten-Datei wird von einem quadratischen Graubild ausgegangen. Nicht-quadratische Formate werden automatisch entsprechend umskaliert, Farbbilder in Graubilder umgewandelt. Bei der Transformation können Sie wählen, mit wieviel Bildpunkten pro Bildzeile gerechnet werden soll; die Auflösung des originalen Bildes wird entsprechend umgerechnet. Durch die Transformation wird im gleichen Verzeichnis, in dem das Bild liegt, eine `txt`-Datei mit gleichem Namen erzeugt, das die Graubild-Daten in einer Form enthält, so dass Sie diese mit Ihren Praktikums-Funktionen einlesen können.
>
> Geben Sie im Programm eine `txt`-Datei entsprechend der Praktikums-Formatierung an, wird durch die Transformation im gleichen Verzeichnis eine `png`-Datei mit gleichem Namen mit angehängtem „_“ erzeugt, indem die angegebenen Realteile als Grauwerte eines rechteckigen Bildes interpretiert werden. Dabei werden die Werte auf ganze Zahlen gerundet, negative Werte auf 0 und Werte über 255 auf 255 gesetzt.

Führen Sie mit Hilfe des Programms „image2array“ folgende Schritte durch:

1. Transformieren Sie ein Bild Ihrer Wahl in eine entsprechende Daten-Datei.

   Hinweis: Falls Sie keine FFT realisiert haben, wählen Sie keine zu große Auflösung, vgl. die Erläuterung unten unter „FFT“.

   Transformieren Sie die Daten mit dem Programm zurück in ein Bild. Sie erhalten das ursprüngliche Bild (ggf. quadratisch umskaliert und als Graubild) in der angegebenen Auflösung.

   Optional: Betrachten Sie die Werte: Die Grauwerte sind Zahlen zwischen 0 und 255. Je nach gewähltem Bild können Sie vielleicht die Struktur Ihres Bildes in den Zahlenwerten erkennen.

2. Berechnen Sie mit Ihrer Praktikums-Funktion die Fourier-Transformation der Daten.

   Optional:

   - Geben Sie die Fourier-Koeffizienten mit dem Default-Wert für `epsilon` aus. Schauen Sie sich die Fourierkoeffizienten an; in den Fourierkoeffizienten wird man nicht viel Struktur erkennen.
   - Lesen Sie die Fourier-Daten ein, berechnen Sie mit Ihrer Praktikums-Funktion die Rücktransformation und geben Sie diese aus. Sie erhalten die ursprünglichen Daten, allerdings mit ein bisschen numerischen Ungenauigkeiten.
   - Transformieren Sie die Daten mit dem Programm „image2array“ in ein Bild. Sie erhalten das ursprüngliche Bild wie oben bei 1..

3. Geben Sie die Fourierkoeffizienten der Daten mit positiven Werten für `epsilon` aus. Da die Fourierkoeffizienten wahrscheinlich betragsmäßig größer sind als bei den Beispiel-Daten zur Aufgabe 4 bieten sich `epsilon`-Werte von 10, 30, 100, 300 oder 1000 an.

   Optional: Schauen Sie sich im Datei-Explorer an, wie klein/groß die Dateien mit den Fourierkoeffizienten sind.

4. Laden Sie die Werte aus den einzelnen Dateien, führen Sie die jeweilige Rücktransformation durch und geben Sie die rücktransformierten Daten mit dem Default-Wert für `epsilon` in einzelne `txt`-Dateien aus.

5. Transformieren Sie die Daten aus den `txt`-Dateien mit dem Programm „image2array“ in Bilder.

FFT: Fast Fourier Transformation

Mit der im Skript, Definition 3.9, genannten Formel kann man nur kleine Auflösungen (bis ca. 64 Pixel pro Zeile) in akzeptabler Zeit transformieren. Die Formel berechnet für jeden der N Werte eine N-fache Summe, hat also eine Laufzeit von $O(N^2)$. Bei Verdoppelung der Auflösung (z.B. von 64 auf 128) wird das Bild 4-mal so groß (von 64^2 zu 128^2 Bildpunkten), so dass die Berechnung $4^2 = 16$ mal so lange braucht. Dauert beispielsweise die Berechnung bei einer Auflösung von 64 eine Sekunde, so sind es bei einer Auflösung 128 ca. 16 Sekunden, bei 256 ca. 4 Minuten und bei 512 mehr als eine Stunde.

Die *Fast Fourier Transformation*, s. Skript, Anhang B.2.2, nutzt geschickt die Gestalt der e-Faktoren in der Formel, um - falls N eine Zweierpotenz ist - durch eine andere Art der Rechnung den Aufwand auf $O(N \cdot \log N)$ zu reduzieren. Wenn Sie diese Rechenvorschrift implementieren, können Sie in akzeptabler Zeit auch die Daten bis zu einer Auflösung von 4096 Fourier-transformieren.