Tipps zur Programmierung von Aufgabe 1.1 c) - e):

- Zur Bewertung der Lösung benötige ich den Quelltext (in C++, inklusive der Beantwortung der gestellten Fragen) und die Fehlerverläufe als (kleine) Bilddatei. Die Matrix-Klasse (sofern unverändert), sowie ausführbare und Objekt- Dateien, die vom Compiler erstellt werden, benötige ich nicht, sie verstopfen nur mein Postfach.

- Die Datenstruktur für die Trainings/Test-Beispiele muss selbst gewählt werden. Möglich ist die Benutzung der Matrix-Klasse.

- Die Konstante π ist als `M_PI` in der `math.h` definiert.

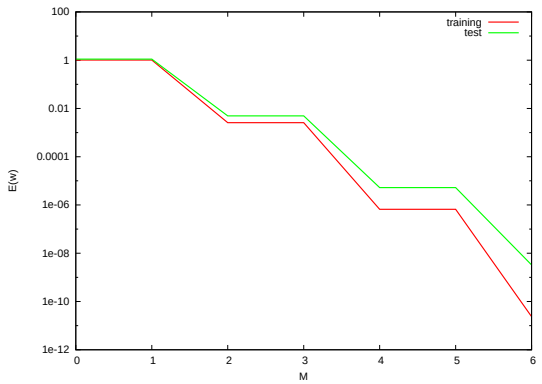
- Die Berechnung der Elemente von $\underline{\underline{A}}$ und $\underline{b}$ aus der Traininmgsmenge geschieht mittels

$$a_{km} = \sum_{p=1}^{P} x_p^{k+m}; \quad b_k = \sum_{p=1}^{P} t_p x_p^k; \qquad 0 \le k, m \le M\,.$$

  Die Dimension der Matrix $\underline{\underline{A}}$ ist $(M+1) \times (M+1)$!

- Um den Restfehler $E(\underline{w})$ zu berechnen, muss zuerst das Gleichungssystem $\underline{\underline{A}}\,\underline{w} = \underline{b}$ gelöst werden.

- Dann sollte für jedes $M = 0, \ldots, M_{max}$ der (Rest-)Fehler der Trainingsmenge und der Fehler der Testmenge berechnet werden, jeweils für die Kosinus- und die Gauss-Beispiele. Diese Fehler sollten in ein Diagramm je Beispielart eingetragen werden (z.B. mit gnuplot, Exel o.ä.) und als Bilddatei mit der Mail mitgeschickt werden. M_{max} sollte ca. $10 - 20$ sein. Da die Fehler sehr klein werden können, sollte man den Fehler mit logarithmischer y-Achse eintragen
  (gnuplot: “set logscale y”). Ein beispielhafter Fehlerverlauf in Abhängigkeit von der Modellkomplexität könnte folgendermaßen aussehen:



(•) Interessant ist auch die Funktion $y(\underline{w}, x)$, die gelernt wurde. Wer will, kann sich die Funktion z.B. an einigen Stützstellen in eine Datei ausgeben lassen, und diese Datei wiederum mit Gnuplot plotten.