Übungen zu Computergrundlagen WS 2013/2014

Übungsblatt 12: Asymptotisches Verhalten und Matplotlib

15.1.2014

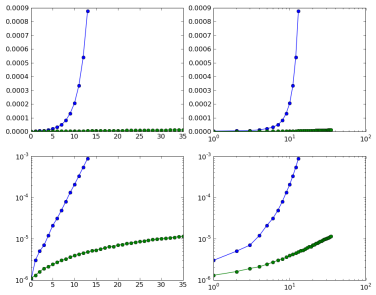
Allgemeine Hinweise

- Abgabetermin für die Lösungen ist
  - **Dienstag, 21.1.2014, 10:00** für die Übungsgruppen am Donnerstag und Freitag
  - **Mittwoch, 22.1.2014, 10:00** für die Übungsgruppen am Montag und Dienstag
- Schickt die Lösungen bitte per Email an Euren Tutor.

Aufgabe 12.1: Aufwand der Berechnung von Fibonacci und Matplotlib (3 Punkte)

Das Skript `/group/cgl/2013/12/plotfib.py` enthält die Funktionen `fib1` und `fib2` vom letzten Übungsblatt und plottet die Fibonaccireihe.

- **12.1.1** Erweitert das Skript wie folgt: (2 Punkte)
  - Es soll die Laufzeiten $t(n)$ messen, die benötigt werden, um die Funktion `fib1(n)` für Werte von $n \leq 17$ bzw. die Funktion `fib2(n)` für Werte von $n \leq 40$ zu berechnen.
  - Es soll mit Hilfe der `matplotlib` einen Plot erstellen, in dem die Laufzeiten $t(n)$ der verschiedenen Funktionen gegen n aufgetragen sind. Dieser soll aus vier Subplots bestehen, in denen jeweils die Plotfunktionen `plot`, `semilogx`, `semilogy` und `loglog` verwendet werden, um beide Kurven darzustellen. Der Plot sollte ungefähr so aussehen wie in nebenstehender Abbildung.



Hinweis: Verwendet die Funktion `time.clock()` zum Messen der Zeiten. Die Aufrufe der Funktion sind zu schnell und `time.clock()` zu ungenau, um die Zeiten einzelner Aufrufe der Funktion zu messen. Messt deswegen die Zeit, die Python benötigt, um `fib1(n)` jeweils 10000 mal und `fib2(n)` jeweils 100000 mal auszuführen, und berechne dann daraus die Zeit für einen einzelnen Funktionsaufruf.

- **12.1.2** Aus den Plots kann man Aussagen über den asymptotischen Aufwand der Funktionen `fib1` und `fib2` treffen. Welchen asymptotischen Aufwand zeigen die Funktionen (exponentiell, polynomial, linear)? (1 Punkt)

Aufgabe 12.2: Konvergenz der numerischen Integration (3 Punkte)

Das Skript `/group/cgl/2013/12/pi.py` enthält die Funktionen `compute_pi1` bis `compute_pi3`. Diese schätzen die Kreiszahl π mit Hilfe der Algorithmen von Übungsblatt 8 ab[1]. Außerdem berechnet das Skript den Fehler der Abschätzungen für jeweils N Schritte (also den Betrag der Differenz von π und der Abschätzung).

- **12.2.1** Erweitert das Skript wie folgt: (1 Punkt)
  - Es soll den Fehler der Abschätzung von π für unterschiedliche Werte von N messen. N soll die Werte der Zweierpotenzen von 1 bis 21 annehmen, also $N = 2^i$ für $i \leq 21$.
  - Der Fehler für jeden Wert von N soll über 10 Läufe gemittelt werden.
- **12.2.2** Erweitert das Skript so, dass es einen doppeltlogarithmischen Plot (`loglog`) erstellen, in dem der (über 10 Läufe gemittelte) Fehler der drei Funktionen über der Anzahl Schritte N aufgetragen ist. (1 Punkt)
- **12.2.3** Aus dem Plot kann man Aussagen über den asymptotischen Genauigkeit der Funktionen treffen. Welche asymptotische Genauigkeit zeigen die Funktionen (exponentiell, polynomial, linear)? (1 Punkt)

Aufgabe 12.3: Geschwindigkeit von Python und C (4 Punkte)

- **12.3.1** Schreibt zwei Pythonskripte, die jeweils die Funktion `compute_pi3` und `compute_pi_numpy` aus dem Skript `/group/cgl/2013/12/pi.py` verwenden, um die Kreiszahl π mit Hilfe von 10^8 Schritten der numerischen Integration anzunähern. (1 Punkt)
- **12.3.2** Schreibt ein Programm in der Programmiersprache C, das dasselbe tut. (2 Punkte)
- **12.3.3** Messt die Laufzeit der beiden Pythonskripte und des C-Programmes mit Hilfe des Unix-Befehls `time`. Wieviel mal schneller als das erste Pythonskript ist das C-Programm? Wie ist der Unterschied beim zweiten Pythonskript? (1 Punkt)

Hinweis: Zum Kompilieren Eures C-Programms verwendet am besten den Befehl

```
gcc -std=c99 -O3 -o pi3 -lm pi3.c
```

[1] Die Funktion `compute_pi_numpy` wird erst in der nächsten Aufgabe wichtig