TMV120-HT10 : Matlab Bonusuppgifter

Varje uppgift ger en bonuspoäng på tentan. Uppgifterna ska redovisas för någon av övningsledarna någon gång innan tentan. Om du vill redovisa något efter läsvecka 6 måste du göra det med din egna dator, ty då kommer vi inte längre att ha reserverade labbtider.

Obs! Man får samarbeta på uppgifterna men varje person måste redovisa dessa bonusuppgifter individuellt. Övningsledarna kommer att ställa frågor för att kontrollera att man förstår båda de program man skrivit och den bakomliggande matematiken.

Dessa uppgifter handlar om ekvationslösning. I en av de obligatoriska uppgifterna har man bekantat sig med Matlab kommandot `fzero`. I följande uppgifter skall man skriva egna program för att implementera ekvationslösningsalgoritmer.

Uppgift 1 : Bisektionsmetoden

Skriv ett program som tar som input ett polynom av grad 3

$$p(x) = ax^3 + bx^2 + cx + d$$

och som hittar en rot till $p(x)$, korrekt till 10 decimalsiffror. Att $p(x)$ har udda grad garanterar existensen av minst en reell rot. Notera att jag söker bara en rot, inte alla (dvs tre st oftast). Använd *bisektionsmetoden*, som du kan läsa om i avsnitt 1.4 av Adams bok. För att starta metoden måste man ha x-värden x_1 och x_2 sådan att

$$p(x_1) < 0, \quad p(x_2) > 0.$$

Man kan välja sådana startvärden utifrån polynomets koefficienter a, b, c, d. Implementera bisektionsprocessen med en lämplig `while` loop t.ex. För att få rätt precision måste du använda `format long`.
Kolla att ditt program funkar genom att mata in slumpmässiga polynom alla vars koefficienter är heltal mellan -5 och $+5$ (det finns 13310 sådana polynom) och kontrollera svaren med antingen `fzero` eller `roots`.

Uppgift 2 : Fixpunktsmetoder

Tag fram en miniräknare, ställ in den till radianer. Knappa in talet noll och tryck på `cos` knappen, igen och igen och igen och ... Notera att talen verkar konvergera mot något. Att de gör det kan bevisas med hjälp av Medelvärdesatsen (kolla avsnitt 4.2 i boken om du är nyfiken). Skriv nu ett Matlab program som beräknar, till 4 decimalsiffror, en lösning till ekvationen

$$\cos x = x. \tag{1}$$

Som i uppgift 1 ovan, ett förslag är att skriva en lämplig `while` loop. Ditt program ska spotta ut både lösningen till (1) plus antalet 'knapptryckningar' som behövdes för att få 4-decimalers precision.
Repetera uppgiften för ekvationerna

$$\cos^n x = x, \quad n = 2, 3, 4, 5.$$

Plotta en graf som visar hur antalet knapptryckningar varierar som en funktion av n. Variera också precisionen (antalet decimalsiffror) och plotta grafer som visar hur antalet knapptryckningar varierar med den angivna precisionen. Notera att du kommer att behöva `format long` i denna uppgift också.