Laboration 1: Skalära olinjära ekvationer och linjära ekvationssystem

1 Målsättning

Lab 1 är tänkt som en förberedelse för föreläsning 2 och 3 som handlar om skalära olinjära ekvationer och linjära ekvationssystem.

Skalära olinjära ekvationer: Vi kommer att få en inblick i hur man hittar en fixpunkt x_f till en skalär olinjär funktion $G(x_f)$, dvs $G(x_f) = x_f$, eller en rot x_0 till en skalär olinjär funktion f, dvs $f(x_0) = 0$, med iterativa metoder.

Linjära ekvationssystem: Vi kommer att lära oss hur vi på ett effektivt sätt kan lösa ett linjärt ekvationssystem med flera obekanta, $Ax = b$.

2 Inledande Laboration och MÖ-uppgifterna

I denna laboration behöver man ingen stor förkunskap i MATLAB men man borde gå igenom introduktionsmaterialet "Snabb genomgång av viktiga MATLAB kommandon" som finns på hemsidan. Om man känner sig osäker så rekommenderas också att lösa några MÖ-uppgifter efter man har gått genom alla uppgifterna i Lab 1. **Man kommer att ha stor nytta av att titta på koden till laborationerna för arbetet med miniprojekten.**

3 Skalära olinjära ekvationer

Kort introduktion

Fixpunktiteration: x_f kallas fixpunkt till funktionen G om $G(x_f) = x_f$. Målet är att hitta en fixpunkt till G med fixpunktiteration. För detta behöver vi en grov gissning x_1 för fixpunkten x_f, $G(x_1) \approx x_f$, som sedan förbättras iterativt:

$$x_1, \quad x_2 = G(x_1), \quad x_3 = G(x_2), \ldots, x_n = G(x_{n-1})$$

Denna iterationsmetod konvergerar, dvs $lim_{n\to\infty} x_n = x_f$, om $|G'(x_f)| < 1$. I praktiken avbryter man metoden efter ett visst antal iterationer. I dessa labborationer avbryts processen om det relativa felet mellan två iterationer är mindre än en satt tolerans:

$$|x_{n+1} - x_n|/|x_n| < tol$$

Rot: En fixpunktiteration kan skrivas om för att hitta en rot x_0 till en ekvation f, $f(x_0) = 0$, om x_0 är en fixpunkt till G, $G(x_0) = x_0$. Man kan nämligen definerar $G(x_0) - x_0 := f(x_0) = 0$. Vi ska lära oss två iterationsalgoritmer för att hitta en rot, Newtons metod och Sekantmetoden.

3.1 Fixpunktmetod

MATLAB-filer: Fixpunkt.m, FixpunktMod.m

1. I Fixpunkt.m vill man veta vilka fixpunkter till $G(x) = 4sin(2x) + \frac{243}{80}$ över intervallet $[0, 8]$ man kan få med fixpunkmetoden. Studera figuren och plotta också G'. Vilka krav gäller för konvergens? Konvergerar den iterativa metoden till alla fixpunkter?

2. Vi modifierar iterationsfunktionen G i den föregående uppgiften till $G(x) = 1.25x - sin(2x) - \frac{243}{320}$ som har samma fixpunkter som $G(x) = 4sin(2x) + \frac{243}{80}$. Kontrollera att denna har verkligen samma fixpunkter. Anpassa Fixpunkt.m för denna funktion. Vilka fixpunkter får vi nu? Testa dem i FixpunktMod.m

3.2 Newtons metod

MATLAB-filer: Newton.m

Newtons metod är en iterationsalgoritm för att hitta en rot x_0 till en olinjär funktion $f : f(x_0) = 0$. Vi börjar igen med en gissning x_1 för en rot och förbättrar gissningen iterativt:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad \text{för} \quad n = 1, \cdots$$

Den iterativa processen avbryts om differensen mellan två iterationer är under en viss tolerans. Då anser man att Newtons metod har konvergerat.

1. I Newton.m formulerar vi om fixpunktiteration i 3.1 till ett problem där vi söker rötter till en motsvarande funktion och löser det med Newtons metod.

$$G(x) = x \Leftrightarrow 4sin(2x) + \frac{243}{80} = x \Leftrightarrow x - 4sin(2x) - \frac{243}{80} = 0$$

Kör programmet och testa de olika rötterna. Hittar vi alla fixpunkter till den ursprungliga ekvationen?

2. Skriv om programmet så att man hittar rötterna till

$$f(x) = x^3 + x^2 - 1.25x - 0.75$$

3.3 Sekantmetoden

MATLAB-filer: Sekant.m

En alternativ för Newtons metod är Sekantmetoden. Fördelen med denna algoritm är att man inte behöver känna till derivatan till f men man måste har två startgissningar x_1 och x_2:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \cdot f(x_n) \quad \text{för} \quad n = 1, \cdots$$

1. I Sekant.m löser vi med Sekantmetoden samma problem som i 3.2. Beskriv varje steg i while-loopen så att ni förstår hur metoden är implementerad.

4 Linjära ekvationssytem

Kort introduktion

Varje linjärt ekvationssystem med n stycken ekvationer och n stycken obekanta kan skrivas på formen $Ax = b$, där A är en nxn matris, x och b är kolonnvektorer av längd n.

4.1 Gausselimination

Gausseliminationen transformerar matrisen A till en övertriangulär matris U dvs en matris där elementen under huvuddiagonalen är noll, $Ax = b \xrightarrow{Gausselimination} Ux = c$. x bestäms nu genom att man startar nerifrån och bestämmer x_n, sätter in detta i näst sista ekvationen och bestämmer x_{n-1} osv. I MATLAB skriver man $x = A \setminus b$ för att lösa systemet med Gausselimination.

1. Exempel: Vi löser följande ekvationssystem:

$$\begin{aligned} 2x_1 + 3x_2 - 5x_3 &= -10 \\ 4x_1 + 8x_2 - 3x_3 &= -19 \\ -6x_1 + x_2 + 4x_3 &= -11 \end{aligned}$$

För att föra över detta till $Ux = c$ skriver vi A och b brevid varandra som $A|b$ och byter rader, adderar resp subtraherar rader med varandra eller multipicerar en rad med ett tal. I $1)$ subtraherar vi första raden multiplicerad med 2 från den andra och adderar första raden multiplicerad med 3 till den sista. I nästa steget $2)$ subtraherar vi den andra raden multiplicerad med 5 från den sista raden.

$$\left[\begin{array}{ccc|c} 2 & 3 & -5 & -10 \\ 4 & 8 & -3 & -19 \\ -6 & 1 & 4 & -11 \end{array}\right] \underbrace{=}_{1} \left[\begin{array}{ccc|c} 2 & 3 & -5 & -10 \\ 0 & 2 & 7 & 1 \\ 0 & 10 & -11 & -41 \end{array}\right] \underbrace{=}_{2} \left[\begin{array}{ccc|c} 2 & 3 & -5 & -10 \\ 0 & 2 & 7 & 1 \\ 0 & 0 & -46 & -46 \end{array}\right]$$

Nu ser vi enkelt att $x_3 = 1$, $x_2 = (1 - 7)/2 = -3$ och $x_1 = (-10 + 5 + 9)/2 = 2$.

2. Lös följande system med Gausselimination på papper, kontrollera sedan lösningen i MATLAB.

$$\begin{bmatrix} 3 & 1 & 6 \\ 2 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 7 \\ 4 \end{bmatrix}$$

4.2 LU-faktorisering

MATLAB-filer: LUCode.m

Om man löser ett ekvationssystem flera gånger med samma matris A men för olika b, så går det att lösa effektivtare med hjälp av två triangulära matriser. Vi formulerar om $Ax = b$ till $LUx = b$ med $A = LU$, där U är en upptriangulär och L en undertriangulär matris. Denna uppdelning av A kallas LU-faktorisering.
Vi inför en ny variabel y och sätter $y = Ux$. Det betyder att man först löser $Ly = b$ för y och sen $Ux = y$. LU-faktorisering i MATLAB erhålls med $[L, U] = lu(A)$.

1. Testa LUCode.m med olika antal ekvationer. CPU-tid är tiden som datorns processor behöver arbeta för att lösa problemet. Vad upptäcker ni?

4.3 Noggrannhet

MATLAB-filer: noggrannhet.m

I detta avsnitt ska vi se att man inte blint kan lita på alla svar som MATLAB ger. I vissa ekvationssystem kan små indatafel (störningar i matrisen A och vektorn b) resultera i stora utdatafel (fel i lösningen x). Sådana störningskänsliga ekvationssystem kallas för illa-konditionerade. Om ett ekvationssystem i motsats är stabilt så kallas det välkonditionerat. Som ett mått på hur störningskänsligt ett system är beräknar man det så kallade konditionstalet. I MATLAB finns för detta kommandot cond(A).

1. Köra noggrannhet.m som visar ett fall för ett illa-konditionerade ekvationssystem.