Linjära ekvationssystem

Standardmetoden att lösa ett (kvadratiskt) ekvationssystem $A\mathbf{x} = \mathbf{b}$ är med hjälp av LU-faktorisering (Gausselimination). Matlab har (åtminstone) tre olika kommandon som gör detta: *mldivide* (MatrixLeftDIVIDE, skrivs $A\backslash\mathbf{b}$), *linsolve* och *lu*. Kolla i hjälpen hur dessa kommandon fungerar. När det gäller kommandot *lu* måste man själv använda den returnerade LU-faktoriseringen (och *mldivide* på två triangulära system) för att lösa ekvationssystemet. En fjärde (mindre bra) metod är att beräkna inversen A^{-1} och sedan göra lämplig multiplikation.

1. Vi ska nu jämföra de olika metoderna för att lösa ett kvadratiskt ekvationssystem.

   (a) Red ut (med papper och penna) hur man använder en LU-faktorisering av A för att hitta lösningen till $A\mathbf{x} = \mathbf{b}$.

   (b) Bilda en 3×3-matris A med slumptal och en 3-vektor $\mathbf{b}$ också med slumptal. Lös sedan systemet $A\mathbf{x} = \mathbf{b}$ med de fyra olika metoderna och kolla att du får samma svar. Kolla också att svaret är rätt! Hur gör man det?

   (c) Vi ska nu jämföra hur lång tid de olika metoderna tar. Med hjälp av paret av kommandon *tic* och *toc* kan man mäta tiden. Detta fungerar bra åtminstone när man har tider på 1/10s och uppåt. (Om man har korta tider kan man köra kommandot flera gånger med en for-loop och ta medeltiden.) Använd slumpmatriser och slumpvektorer och testa storlekar på 1000×1000 och högre. För *LU*-faktorisering kolla dels hur lång tid faktoriseringen tar och dels hur lång tid bakåtsubstitutionerna tar. Kommentarer?

   (d) Ungefär med vilken faktor multipliceras tiden när man dubblar antalet rader och kolumner i matrisen?

   (e) Testa att använda *linsolve* istället för *mldivide* när ni löser de triangulära systemen efter LU-faktorisering. Hur förändras tiden det tar att lösa systemen? Varför?

   (f) När kan man tänka sig att det är effektivare att använda *LU*-faktorisering än *mldivide*?

2. Vi ska nu titta lite kort på ekvationssystem där koefficientmatrisen inte är kvadratisk.

   (a) Skapa en slumpmatris A av storlek 4×3 och lämplig slumpvektor $\mathbf{b}$ och lös ekvationssystemet $A\mathbf{x} = \mathbf{b}$ med *mldivide*. Kolla om det ni fick är en lösning. Kommentarer! Var det oväntat? Vad är det ni fick som svar? Finns det fler lösningar? (Vi kommer att återkomma till detta senare i kursen.)

   (b) Skapa en slumpmatris A av storlek 3×4 och lämplig slumpvektor $\mathbf{b}$ och lös ekvationssystemet $A\mathbf{x} = \mathbf{b}$ med *mldivide*. Kolla om det ni fick är en lösning. Kommentarer! Var det oväntat? Vad är det ni fick som svar? Finns det fler lösningar?

(c) Kommandot *rref* i Matlab ger den reducerade trappstegsformen till en matris. Använd detta för att bestämma alla lösningar i det fall där ni tror att det finns fler än en lösning.

3. Ni ska nu göra egna funktioner för att reducera en matris till trappstegsform.

   (a) Skapa följande tre funktioner för de tre typerna av elementära radoperationer:

   - **changeRows(A,i,j):** Returnerar matrisen man får när man byter plats på rad i och rad j i A.
   - **multiplyRow(A,i,c):** Returnerar matrisen man får när man multiplicerar rad i i A med talet c.
   - **addToRow(A,i,j,c):** Returnerar matrisen man får när man adderar c gånger rad j till rad i i A.

   (b) Använd de tre funktionerna ni just skapat för att lösa uppgifterna 12, 14 och 16 i avsnitt 1.1 i boken. Redovisa hur er startmatris såg ut, hur ni anropade era funktioner, den slutgiltiga trappstegmatrisen samt samtliga lösningar till ekvationssystemen.

Uppgifterna ska redovisas skriftligt till Stefan. Sista inlämningsdag är måndagen den 14 september klockan 12:00. Instruktioner för redovisningen finns på hemsidan. Läs dessa noggrant!