Matthias Jauernig
03IN1

18.11.04

Numerische Mathematik

3. Beleg

1.)

(a)

$$\begin{array}{ccccc|l}
a & c & c & c & c & \cdot\left(-\frac{c}{a}\right) \\
c & a & & & & \\
c & & a & & & \\
c & & & a & & \\
c & & & & a & +
\end{array}$$

$$\begin{array}{ccccc}
a & c & c & c & c \\
 & a-\frac{c^2}{a} & -\frac{c^2}{a} & -\frac{c^2}{a} & -\frac{c^2}{a} \\
 & -\frac{c^2}{a} & a-\frac{c^2}{a} & -\frac{c^2}{a} & -\frac{c^2}{a} \\
 & -\frac{c^2}{a} & -\frac{c^2}{a} & a-\frac{c^2}{a} & -\frac{c^2}{a} \\
 & -\frac{c^2}{a} & -\frac{c^2}{a} & -\frac{c^2}{a} & a-\frac{c^2}{a}
\end{array}$$

→ die Nullen wurden zerstört und durch $-\frac{c^2}{a}$ ersetzt, das spezielle Besetztheitsmuster ging also verloren

(b)

zu finden sind zwei Permutationsmatrizen P_1, P_2 sodass gilt:

$$B = P_1 A P_2 \quad \text{mit} \quad B = \begin{pmatrix} a & & & & c \\ & a & & & c \\ & & a & & c \\ & & & a & c \\ c & c & c & c & a \end{pmatrix}$$

Überlegung: P_1A müsste einer horizontalen, AP_2 einer vertikalen Spiegelung gleich kommen, sodass gilt:

$$P_1A = \begin{pmatrix} c & & & & a \\ c & & & a & \\ c & & a & & \\ c & a & & & \\ a & c & c & c & c \end{pmatrix}, \quad (P_1A)P_2 = \begin{pmatrix} a & & & & c \\ & a & & & c \\ & & a & & c \\ & & & a & c \\ c & c & c & c & a \end{pmatrix} = B$$

$$\rightarrow \underline{P_1 = \begin{pmatrix} & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ 1 & & & & \end{pmatrix}}, \quad P_1A = \begin{pmatrix} c & & & & a \\ c & & & a & \\ c & & a & & \\ c & a & & & \\ a & c & c & c & c \end{pmatrix}$$

$$\rightarrow \underline{P_2 = P_1}: \quad (P_1A)P_2 = \begin{pmatrix} a & & & & c \\ & a & & & c \\ & & a & & c \\ & & & a & c \\ c & c & c & c & a \end{pmatrix} = B$$

(c)

$a = 10, \quad c = 1$

• **B = LR:**

$$\begin{array}{ccccc|l} 10 & & & & 1 & -l_{21}=0 \\ & 10 & & & 1 & -l_{31}=0 \\ & & 10 & & 1 & -l_{41}=0 \\ & & & 10 & 1 & \\ 1 & 1 & 1 & 1 & 10 & -l_{51}=-0.1 \end{array}$$

$$\begin{array}{ccccc|l} 10 & & & & 1 & \\ 0 & 10 & & & 1 & -l_{32}=0 \\ 0 & & 10 & & 1 & \\ 0 & & & 10 & 1 & -l_{42}=0 \\ 0.1 & 1 & 1 & 1 & 9.9 & -l_{52}=-0.1 \end{array}$$

$$\begin{array}{ccccc|l} 10 & & & & 1 & \\ 0 & 10 & & & 1 & \\ 0 & 0 & 10 & & 1 & -l_{43}=0 \\ 0 & 0 & & 10 & 1 & \\ 0.1 & 0.1 & 1 & 1 & 9.8 & -l_{53}=-0.1 \end{array}$$

$$\begin{array}{ccccc|l} 10 & & & & 1 & \\ 0 & 10 & & & 1 & \\ 0 & 0 & 10 & & 1 & \\ 0 & 0 & 0 & 10 & 1 & -l_{54}=-0.1 \\ 0.1 & 0.1 & 0.1 & 1 & 9.7 & \end{array}$$

$$\left.\begin{array}{ccccc} 10 & & & & 1 \\ 0 & 10 & & & 1 \\ 0 & 0 & 10 & & 1 \\ 0 & 0 & 0 & 10 & 1 \\ 0.1 & 0.1 & 0.1 & 0.1 & 9.6 \end{array}\right\} \text{LR-fakt. Matrix } B$$

• **By = P₁b:** $\quad Lz = P_1 b \;\rightarrow\; Ry = z$

$$\rightarrow \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ 0.1 & 0.1 & 0.1 & 0.1 & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \end{pmatrix} = \begin{pmatrix} & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ 1 & & & & \end{pmatrix} \begin{pmatrix} 10.6 \\ 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 10.6 \end{pmatrix}$$

$$\hookrightarrow \quad z = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 9.6 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} 10 & & & & 1 \\ & 10 & & & 1 \\ & & 10 & & 1 \\ & & & 10 & 1 \\ & & & & 9.6 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \\ 9.6 \end{pmatrix} \Rightarrow \underline{y = \begin{pmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0 \\ 1 \end{pmatrix}}$$

• **x = P₂y:**

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} & & & & 1 \\ & & & 1 & \\ & & 1 & & \\ & 1 & & & \\ 1 & & & & \end{pmatrix} \begin{pmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0 \\ 1 \end{pmatrix} \rightarrow \underline{\underline{x = \begin{pmatrix} 1 \\ 0 \\ 0.1 \\ 0.2 \\ 0.3 \end{pmatrix}}}$$

2.) Lösung lin. Gleichungssysteme:

Quelltext:

```c
// Numerik Beleg 3 - Aufgabe 2
// Lösung linearer Gleichungssysteme
// Matthias Jauernig, 2004
/* ----- Includes ------------------------------------------------------------    */
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>

/* ----- Funktions-Deklarationen ---------------------------------------------    */
void LR_DECOMP(const int, double**, int*, bool*);
void L_SOLVE(const int, double**, const int*, double*, double*);
void R_SOLVE(const int, double**, const double*, double*);

/* ----- main() --------------------------------------------------------------    */
int main(void){
	bool Sing=false;
	int N, i, j, *P;
	double **A, **L, **R, *b, *y, *x;

	printf("\n======================================"
	       "\n| Loesung linearer Gleichungssysteme |"
	       "\n======================================\n\n");
	do{
		printf("Dimension des LGS: ");
		scanf("%d",&N);
	}while(N<2 && printf("Dim. muss groesser gleich 2 sein!\n\n"));
	//Speicher gem. der Dimension allokieren
	A=(double**)malloc(N*sizeof(double*));
	L=(double**)malloc(N*sizeof(double*));
	R=(double**)malloc(N*sizeof(double*));
	for(i=0; i<N; i++){
		A[i]=(double*)malloc(N*sizeof(double));
		L[i]=(double*)malloc(N*sizeof(double));
		R[i]=(double*)malloc(N*sizeof(double));
	}
	b=(double*)malloc(N*sizeof(double));
	P=(int*)malloc((N-1)*sizeof(int));
	y=(double*)malloc(N*sizeof(double));
	x=(double*)malloc(N*sizeof(double));

	printf(">> Eingabe von Matrix A\n");
	for(i=0;i<N;i++){
		printf(" - %d. Zeile:\n",i+1);
		for(j=0;j<N; j++){
			printf("  + %d. Element: ",j+1);
			scanf("%lf",&A[i][j]);
		}
	}
	printf("\n>> Eingabe von Vektor b\n");
	for(i=0;i<N;i++){
		printf(" - %d. Element: ",i+1);
		scanf("%lf",&b[i]);
	}

	LR_DECOMP(N,A,P,&Sing); //LR-Faktorisierung von A
	if(Sing){
		printf("Singularitaetstest nicht bestanden - Abbruch\n");
		return 1;
	}

	for(i=0; i<N; i++){ //L und R aus A erzeugen
		for(j=0;j<i;j++){
			L[i][j]=A[i][j];
			R[i][j]=0;
		}
		L[i][j]=1;
		R[i][j]=A[i][j];
		for(j++;j<N;j++){
			L[i][j]=0;
			R[i][j]=A[i][j];
		}
	}

	L_SOLVE(N,L,P,b,y); //Ly=Pb lösen -> y
	R_SOLVE(N,R,y,x);   //Rx=y lösen  -> x

	printf("\n=========================================\n"
	       "<< Ausgabe von Matrix A (LR-faktorisiert)\n");
	for(i=0;i<N;i++){
		for(j=0;j<N;j++)
			printf("%11.7lg  ", A[i][j]);
		printf("\n");
	}
```

```
	printf("\n<< Ausgabe von Vektor P\n");
	for(i=0;i<N-1;i++)
		printf("%d\n",P[i]+1);
	printf("\n<< Ausgabe von Vektor y\n");
	for(i=0;i<N;i++)
		printf("%15.14lg\n",y[i]);
	printf("\n<< Ausgabe des Loesungsvektors x\n");
	for(i=0;i<N;i++)
		printf("%15.14lg\n",x[i]);
	printf("\n\n");
	return 0;
}

/* ----- Funktions-Definitionen -----------------------------------------------    */
void LR_DECOMP(const int N, double **A, int *P, bool *Sing){ //Modul zur LR-Faktorisierung von A
	double masch_eps=0.5E-15, s[N], tmp, Anorm1, spaltsum;
	int i, j, k, max;

	//1-Norm von A berechnen
	Anorm1=0.0;
	for(i=0;i<N;i++){
		spaltsum=0.0;
		for(j=0;j<N;j++)
			spaltsum+=fabs(A[j][i]);
		if(spaltsum>Anorm1)
			Anorm1=spaltsum;
	}

	for(i=0;i<N-1;i++){
		//s berechnen
		max=i;
		for(j=i;j<N;j++){
			s[j]=0.0;
			for(k=i;k<N;k++)
				s[j]+=fabs(A[j][k]);
			s[j]=(1/s[j])*fabs(A[j][i]);
			if(s[j]>s[max])
				max=j;
		}
		//Singularitaetstest
		if(fabs(A[max][i])<masch_eps*Anorm1){
			*Sing=true;
			return;
		}
		//in P eintragen, Pivotzeile tauschen
		P[i]=max;
		if(i!=max)
			for(j=0;j<N;j++){
				tmp=A[i][j];
				A[i][j]=A[max][j];
				A[max][j]=tmp;
			}

		//i+1. Hauptschritt ausfuehren
		for(j=i+1;j<N;j++){
			A[j][i]=A[j][i]/A[i][i];
			for(k=i+1;k<N;k++)
				A[j][k]+=A[i][k]*-A[j][i];
		}
	}
}

void L_SOLVE(const int N, double **L, const int *P, double *b, double *y){ //Modul zum Lösen von Ly=Pb
	int i, j;
	double tmp;
	//Pb berechnen, auf b[] abspeichern
	for(i=0;i<N-1;i++)
		if(P[i]!=i){
			tmp=b[i];
			b[i]=b[P[i]];
			b[P[i]]=tmp;
		}
	//y[] berechnen
	for(i=0; i<N; i++){
		y[i]=b[i];
		for(j=0;j<i;j++)
			y[i]-=L[i][j]*y[j];
		y[i]/=L[i][j];
	}
}

void R_SOLVE(const int N, double **R, const double *y, double *x){ //Modul zum Lösen von Rx=y
	int i, j;
	//x[] berechnen
	for(i=N-1;i>=0;i--){
		x[i]=y[i];
		for(j=i+1;j<N;j++)
			x[i]-=R[i][j]*x[j];
		x[i]/=R[i][i];
	}
}
```

(a) Ergebnisausdruck:

```
======================================
| Loesung linearer Gleichungssysteme |
======================================

Dimension des LGS: 5
>> Eingabe von Matrix A
 - 1. Zeile:
  + 1. Element: 1
  + 2. Element: 0
  + 3. Element: -1
  + 4. Element: -1
  + 5. Element: 0
 - 2. Zeile:
  + 1. Element: 0
  + 2. Element: 1
  + 3. Element: 1
  + 4. Element: 0
  + 5. Element: -1
 - 3. Zeile:
  + 1. Element: 4
  + 2. Element: -5
  + 3. Element: 2
  + 4. Element: 0
  + 5. Element: 0
 - 4. Zeile:
  + 1. Element: 0
  + 2. Element: 0
  + 3. Element: -2
  + 4. Element: 9
  + 5. Element: -12
 - 5. Zeile:
  + 1. Element: 0
  + 2. Element: 5
  + 3. Element: 0
  + 4. Element: 0
  + 5. Element: 12

>> Eingabe von Vektor b
 - 1. Element: 0
 - 2. Element: 0
 - 3. Element: 0
 - 4. Element: 0
 - 5. Element: 50

=========================================
<< Ausgabe von Matrix A (LR-faktorisiert)
          4           -5            2            0            0
          0            1            1            0           -1
       0.25         1.25        -2.75           -1         1.25
          0            0    0.7272727     9.727273    -12.90909
          0            5     1.818182    0.1869159     17.14019

<< Ausgabe von Vektor P
3
2
3
4

<< Ausgabe von Vektor y
0
0
0
0
50

<< Ausgabe des Loesungsvektors x
3.7895310796074
2.9989094874591
-0.081788440567067
3.8713195201745
 2.917121046892
```

(b) Ergebnisausdruck:

```
======================================
| Loesung linearer Gleichungssysteme |
======================================

Dimension des LGS: 5
>> Eingabe von Matrix A
 - 1. Zeile:
  + 1. Element: 10.235
  + 2. Element: -4.56
  + 3. Element: 0
  + 4. Element: -0.035
  + 5. Element: 5.67
 - 2. Zeile:
  + 1. Element: -2.463
  + 2. Element: 1.27
  + 3. Element: 3.97
  + 4. Element: -8.63
  + 5. Element: 1.08
 - 3. Zeile:
  + 1. Element: -6.58
  + 2. Element: 0.86
  + 3. Element: -0.257
  + 4. Element: 9.32
  + 5. Element: -43.6
 - 4. Zeile:
  + 1. Element: 9.83
  + 2. Element: 7.39
  + 3. Element: -17.25
  + 4. Element: 0.036
  + 5. Element: 24.86
 - 5. Zeile:
  + 1. Element: -9.31
  + 2. Element: 34.9
  + 3. Element: 78.56
  + 4. Element: 1.07
  + 5. Element: 65.8

>> Eingabe von Vektor b
 - 1. Element: 8.95
 - 2. Element: 20.54
 - 3. Element: 7.42
 - 4. Element: 5.6
 - 5. Element: 58.43

=========================================
<< Ausgabe von Matrix A (LR-faktorisiert)
     10.235        -4.56            0       -0.035         5.67
  0.9604299     11.76956       -17.25   0.06961505     19.41436
 -0.9096238     2.612852     123.6317    0.8562694     20.23072
 -0.2406448      0.01467   0.03415837    -8.668693     1.468599
  -0.642892   -0.1760123  -0.02663729    -1.076582    -34.41768

<< Ausgabe von Vektor P
1
4
5
4

<< Ausgabe von Vektor y
8.95
-2.9958475818271
 74.39883870317
20.196377094463
36.371421815057

<< Ausgabe des Loesungsvektors x
2.6383625899619
2.6643834462368
0.79208015947959
-2.5088376454102
-1.0567657691375
```

3.) <u>Inverse Matrizen:</u>

Quelltext:

```
// Numerik Beleg 3 - Aufgabe 3
// Berechnung einer inversen Matrix
// Matthias Jauernig, 2004
/* ----- Includes -------------------------------------------------------------    */
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <math.h>

/* ----- Funktions-Deklarationen ----------------------------------------------    */
void LR_DECOMP(const int, double**, int*, bool*);
void L_SOLVE(const int, double**, const int*, double*, double*);
void R_SOLVE(const int, double**, const double*, double*);

/* ----- main() ---------------------------------------------------------------    */
int main(void){
	bool Sing=false;
	int N, i, j, *P;
	double **A, **A_inv, **L, **R, *y, *x, *e;

	printf("\n===================================="
	       "\n| Berechnung einer inversen Matrix |"
	       "\n====================================\n\n");

	do{
		printf("Dimension der n*n-Matrix: ");
		scanf("%d",&N);
	}while(N<2 && printf("Dim. muss groesser gleich 2 sein!\n\n"));
	//Speicher gem. der Dimension allokieren
	A=(double**)malloc(N*sizeof(double*));
	A_inv=(double**)malloc(N*sizeof(double*));
	L=(double**)malloc(N*sizeof(double*));
	R=(double**)malloc(N*sizeof(double*));
	for(i=0; i<N; i++){
		A[i]=(double*)malloc(N*sizeof(double));
		A_inv[i]=(double*)malloc(N*sizeof(double));
		L[i]=(double*)malloc(N*sizeof(double));
		R[i]=(double*)malloc(N*sizeof(double));
	}
	P=(int*)malloc((N-1)*sizeof(int));
	y=(double*)malloc(N*sizeof(double));
	x=(double*)malloc(N*sizeof(double));
	e=(double*)malloc(N*sizeof(double));

	printf(">> Eingabe von Matrix A\n");
	for(i=0;i<N;i++){
		printf(" - %d. Zeile:\n",i+1);
		for(j=0;j<N; j++){
			printf("  + %d. Element: ",j+1);
			scanf("%lf",&A[i][j]);
		}
	}

	LR_DECOMP(N,A,P,&Sing); //LR-Faktorisierung von A
	if(Sing){
		printf("Singularitaetstest nicht bestanden - Abbruch\n");
		return 1;
	}

	for(i=0; i<N; i++){ //L und R aus A erzeugen
		for(j=0;j<i;j++){
			L[i][j]=A[i][j];
			R[i][j]=0;
		}
		L[i][j]=1;
		R[i][j]=A[i][j];
		for(j++;j<N;j++){
			L[i][j]=0;
			R[i][j]=A[i][j];
		}
	}

	//Spalten der inv. Matrix berechnen
	for(i=0;i<N;i++){
		//e mit 0 initialisieren
		for(j=0;j<N;j++)
			e[j]=0.0;
		e[i]=1.0;
		L_SOLVE(N,L,P,e,y); //Ly=Pe^i lösen -> y
		R_SOLVE(N,R,y,x);   //Rx=y lösen    -> x

		for(j=0;j<N;j++)
			A_inv[j][i]=x[j];
	}

	printf("\n=========================================\n"
	       "<< Ausgabe der inversen Matrix A^-1\n");
```

```
	for(i=0;i<N;i++){
		for(j=0;j<N;j++)
			printf("%11.7lg  ", A_inv[i][j]);
		printf("\n");
	}
	printf("\n\n");
	return 0;
}

/* ----- Funktions-Definitionen ----------------------------------------------    */
void LR_DECOMP(const int N, double **A, int *P, bool *Sing){ //Modul zur LR-Faktorisierung von A
	double masch_eps=0.5E-15, s[N], tmp, Anorm1, spaltsum;
	int i, j, k, max;

	//1-Norm von A berechnen
	Anorm1=0.0;
	for(i=0;i<N;i++){
		spaltsum=0.0;
		for(j=0;j<N;j++)
			spaltsum+=fabs(A[j][i]);
		if(spaltsum>Anorm1)
			Anorm1=spaltsum;
	}

	for(i=0;i<N-1;i++){
		//s berechnen
		max=i;
		for(j=i;j<N;j++){
			s[j]=0.0;
			for(k=i;k<N;k++)
				s[j]+=fabs(A[j][k]);
			s[j]=(1/s[j])*fabs(A[j][i]);
			if(s[j]>s[max])
				max=j;
		}
		//Singularitaetstest
		if(fabs(A[max][i])<masch_eps*Anorm1){
			*Sing=true;
			return;
		}
		//in P eintragen, Pivotzeile tauschen
		P[i]=max;
		if(i!=max)
			for(j=0;j<N;j++){
				tmp=A[i][j];
				A[i][j]=A[max][j];
				A[max][j]=tmp;
			}
		//i+1. Hauptschritt ausfuehren
		for(j=i+1;j<N;j++){
			A[j][i]=A[j][i]/A[i][i];
			for(k=i+1;k<N;k++)
				A[j][k]+=A[i][k]*-A[j][i];
		}
	}
}

void L_SOLVE(const int N, double **L, const int *P, double *b, double *y){ //Modul zum Lösen von Ly=Pb
	int i, j;
	double tmp;
	//Pb berechnen, auf b[] abspeichern
	for(i=0;i<N-1;i++)
		if(P[i]!=i){
			tmp=b[i];
			b[i]=b[P[i]];
			b[P[i]]=tmp;
		}
	//y[] berechnen
	for(i=0; i<N; i++){
		y[i]=b[i];
		for(j=0;j<i;j++)
			y[i]-=L[i][j]*y[j];
		y[i]/=L[i][j];
	}
}

void R_SOLVE(const int N, double **R, const double *y, double *x){ //Modul zum Lösen von Rx=y
	int i, j;
	//x[] berechnen
	for(i=N-1;i>=0;i--){
		x[i]=y[i];
		for(j=i+1;j<N;j++)
			x[i]-=R[i][j]*x[j];
		x[i]/=R[i][i];
	}
}
```

zu (2a) Ergebnisausdruck:

```
Dimension des LGS: 5
>> Eingabe von Matrix A
 - 1. Zeile:
  + 1. Element: 1
  + 2. Element: 0
  + 3. Element: -1
  + 4. Element: -1
  + 5. Element: 0
 - 2. Zeile:
  + 1. Element: 0
  + 2. Element: 1
  + 3. Element: 1
  + 4. Element: 0
  + 5. Element: -1
 - 3. Zeile:
  + 1. Element: 4
  + 2. Element: -5
  + 3. Element: 2
  + 4. Element: 0
  + 5. Element: 0
 - 4. Zeile:
  + 1. Element: 0
  + 2. Element: 0
  + 3. Element: -2
  + 4. Element: 9
  + 5. Element: -12
 - 5. Zeile:
  + 1. Element: 0
  + 2. Element: 5
  + 3. Element: 0
  + 4. Element: 0
  + 5. Element: 12

=========================================
<< Ausgabe der inversen Matrix A^-1
  0.4612868    0.2944384    0.1346783   0.05125409   0.07579062
  0.2355507    0.4056707  -0.05888768    0.0261723   0.05997819
 -0.3336968    0.4252999   0.08342421  -0.03707743  -0.001635769
 -0.2050164   -0.1308615   0.05125409   0.08833152   0.07742639
-0.09814613   -0.1690294   0.02453653  -0.01090513   0.05834242
```

zu (2b) Ergebnisausdruck:

```
Dimension des LGS: 5
>> Eingabe von Matrix A
 - 1. Zeile:
  + 1. Element: 10.235
  + 2. Element: -4.56
  + 3. Element: 0
  + 4. Element: -0.035
  + 5. Element: 5.67
 - 2. Zeile:
  + 1. Element: -2.463
  + 2. Element: 1.27
  + 3. Element: 3.97
  + 4. Element: -8.63
  + 5. Element: 1.08
 - 3. Zeile:
  + 1. Element: -6.58
  + 2. Element: 0.86
  + 3. Element: -0.257
  + 4. Element: 9.32
  + 5. Element: -43.6
 - 4. Zeile:
  + 1. Element: 9.83
  + 2. Element: 7.39
  + 3. Element: -17.25
  + 4. Element: 0.036
  + 5. Element: 24.86
 - 5. Zeile:
  + 1. Element: -9.31
  + 2. Element: 34.9
  + 3. Element: 78.56
  + 4. Element: 1.07
  + 5. Element: 65.8

=========================================
<< Ausgabe der inversen Matrix A^-1
  0.1084106   0.04411012   0.04057182    0.0316612  0.004855721
-0.001593844   0.06103793   0.05497453   0.06439203   0.01123435
 0.03118189  0.005954219  0.004788538  -0.02018052  0.008012708
 -0.0194246   -0.1206569   -0.0049223  -0.009522481  0.003990327
-0.02072853  -0.03127992  -0.02905484  -0.005424694  0.0002945293
```

4.)

(a)

· $PA = LR$:

$\boxed{j=1:}$

$s_1 = \frac{1}{9.218} = 0.108483402 \qquad |a_{11}|s_1 = 0.14916\ldots \quad \overset{\text{max},}{p(1)=1}$

$s_2 = \frac{1}{29.47} = 0.33932813 \qquad |a_{21}|s_2 = 0.0909\ldots$

$s_3 = \frac{1}{46.185} = 0.0216520515 \qquad |a_{31}|s_3 = 0.0265\ldots$

$$\begin{array}{ccc}
1.375 & 2.483 & 5.360 \\
2.680 & -1.230 & 25.56 \\
-1.225 & 9.910 & -35.05
\end{array} \qquad \begin{array}{l} -l_{21} = -1.94\overline{90} \\ -l_{31} = 0.8\overline{90} \end{array}$$

$$\begin{array}{c|cc}
1.375 & 2.483 & 5.360 \\
\hline
1.94\overline{90} & -6.069592727 & 15.11287273 \\
-0.8\overline{90} & 12.12212727 & -30.27472727
\end{array} \qquad -l_{32} = 1.997189567$$

$\boxed{j=2:}$

$s_2 = \frac{1}{21.18246546} = 0.0472088578 \qquad |a_{22}|s_2 = 0.2865385401 \quad \text{max}, \ p(2)=2$

$s_3 = \frac{1}{42.39685454} = 0.0235866554 \qquad |a_{32}|s_3 = 0.2859204392$

$$\begin{array}{c|c|c}
1.375 & 2.483 & 5.360 \\
\hline
1.94\overline{90} & -6.069592727 & 15.11287273 \\
-0.8\overline{90} & -1.997189567 & -0.0914555262
\end{array}$$

· $Ly = Pb$: $\qquad \vec{p} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \rightsquigarrow P = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix} = E$

$$\begin{pmatrix} 1 & & \\ 1.94\overline{90} & 1 & \\ -0.8\overline{90} & -1.997189567 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 23.5 \\ -15.76 \\ 2.34 \end{pmatrix} \Rightarrow y = \begin{pmatrix} 23.5 \\ -61.56363615 \\ -99.67788821 \end{pmatrix}$$

· $Rx^* = y$:

$$\begin{pmatrix} 1.375 & 2.483 & 5.360 \\ & -6.069592727 & 15.11287273 \\ & & -0.0914555262 \end{pmatrix} \begin{pmatrix} x_1^* \\ x_2^* \\ x_3^* \end{pmatrix} = \begin{pmatrix} 23.5 \\ -61.56363615 \\ -99.67788821 \end{pmatrix}$$

$$\Rightarrow x^* = \begin{pmatrix} -9150.488134 \\ 2723.933661 \\ 1089.905579 \end{pmatrix}$$

(b)

· $PA = LR$:

$\boxed{j=1:}$ $s_1 = \frac{1}{9.218} = 0.108483402 \quad |a_{11}|s_1 = 0.1491646778$ max, $p(1)=1$

$s_2 = 0.33932813 \quad |a_{21}|s_2 = 0.0909\ldots$

$s_3 = \frac{1}{46.185} = 0.0216520515 \quad |a_{31}|s_3 = 0.026740\ldots$

1.375	2.483	5.360	$-\ell_{21} = -1.94\overline{90}$
2.680	−1.230	25.56	
−1.235	9.910	−35.04	$-\ell_{31} = 0.89\overline{81}$

1.375	2.483	5.360	
1.94$\overline{90}$	−6.069592727	15.11287273	$-\ell_{32} = 0.4999588146$
−0.89$\overline{81}$	12.14018545	−30.22574545	

$\boxed{j=2:}$ $s_2 = 0.04720885578 \quad |a_{22}|s_2 = 0.286538\ldots$

$s_3 = \frac{1}{42.3659309} = 0.02360387718 \quad |a_{32}|s_3 = 0.286555\ldots$ max, $p(2)=3$

1.375	2.483	5.360
−0.89$\overline{81}$	12.14018545	−30.22574545
1.94$\overline{90}$	−0.4999588146	0.0012448634

· $Ly = Pb$:

$$\begin{pmatrix} 1 & & \\ -0.89\overline{81} & 1 & \\ 1.94\overline{90} & -0.4999588146 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 23.50 \\ 2.340 \\ -15.76 \end{pmatrix}$$

$$\Rightarrow y = \begin{pmatrix} 23.50 \\ 23.44\overline{72} \\ -49.84096568 \end{pmatrix}$$

· $Rx = y$: $\quad (R(\Delta x + x^*) = y)$

$$\begin{pmatrix} 1.375 & 2.483 & 5.360 \\ & 12.14018545 & -30.22574545 \\ & & 0.0012448634 \end{pmatrix}\begin{pmatrix} \Delta x_1 + x_1^* \\ \Delta x_2 + x_2^* \\ \Delta x_3 + x_3^* \end{pmatrix} = \begin{pmatrix} 23.50 \\ 23.44\overline{72} \\ -49.84096568 \end{pmatrix}$$

$$\left.\begin{aligned} \Delta x_3 + x_3^* &= -40037.296523 \\ \Delta x_2 + x_2^* &= -99680.00002 \\ \Delta x_1 + x_1^* &= 336093.7086 \end{aligned}\right\} \Delta x = \begin{pmatrix} 345244.1967 \\ -102403.9337 \\ -41127.2021 \end{pmatrix}$$

$$\Rightarrow \frac{\|\Delta x\|_\infty}{\|x^*\|_\infty} = \frac{345244.1967}{9150.488134} = 37.7295934$$

$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} = \frac{0.02}{46.185} = 0.0004330410306$$

$\hookrightarrow$ $\mathrm{cond}_\infty(A) \approx 87127$ $\quad$ (wegen $\frac{\|\Delta x\|_\infty}{\|x^*\|_\infty} \le \mathrm{cond}_\infty(A)\frac{\|\Delta A\|_\infty}{\|A\|_\infty}$

bzw. $\frac{\|\Delta x\|_\infty}{\|x^*\|_\infty} \le \frac{\mathrm{cond}_\infty(A)}{1-\mathrm{cond}_\infty(A)\cdot\frac{\|\Delta A\|_\infty}{\|A\|_\infty}} \cdot \frac{\|\Delta A\|}{\|A\|}$)