Programmbeispiele Matrixvektormultiplikation: $b = Ax$

- C (dynamische Allozierung mit malloc)

```
/* Standardmethode zur Uebergabe einer Matrix (dynamische Allozierung) */

#include <stdio.h>
#include <stdlib.h>

void matvekmult(double **a, double *x, double *b, int m, int n)
{
  int i,j;

  for (i=0; i<m; ++i) {
    b[i] = 0;
    for (j=0; j<n; ++j)
      b[i] += a[i][j]*x[j];
  }
  return;
}

int main(void)
{
  double **a,*x,*b;
  int m,n,i,j;

  printf("m n: ");
  scanf("%d%d",&m,&n);  /* m,n einlesen */
  x = (double*)  malloc(n*sizeof(double));
  b = (double*)  malloc(m*sizeof(double));
  a = (double**) malloc(m*sizeof(double*));
  for (i=0; i<m; ++i) a[i] = (double*) malloc(n*sizeof(double));

  /* Matrixeingabe (a) */
  printf("a:\n");
  for (i=0; i<m; ++i)
    for (j=0; j<n; ++j)
      scanf("%lf",&a[i][j]);

  /* Vektoreingabe (x) */
  printf("\nx:\n");
  for (j=0; j<n; ++j)
    scanf("%lf",&x[j]);

  /* Matrixvektormultiplikation */
  matvekmult(a,x,b,m,n);

  /* Vektorausgabe (b) */
  printf("\nb:\n");
  for (i=0; i<m; ++i)
    printf("%6.2f\n",b[i]);

  return 0;
}
```

- C++ (dynamische Allozierung mit new)

```
#include <iostream>
#include <iomanip>
#include <vector>

using namespace std;

void matvekmult(double **a, double *x, double *b, int m, int n)
{
  int i,j;
  for (i=0; i<m; ++i) {
    b[i] = 0;
    for (j=0; j<n; ++j) b[i] += a[i][j]*x[j];
  }
  return;
}

int main()
{
  int m,n,i,j;
  cout << "m n: "; cin  >> m >> n;

  double *x,*b,**a;
  x = new double[n]; b = new double[m];
  a = new double*[m];
  for (i=0; i<m; ++i) a[i] = new double[n];

  /* Matrixeingabe (a) */
  cout << "a: " << endl;
  for (i=0; i<m; ++i) for (j=0; j<n; ++j) cin >> a[i][j];

  /* Vektoreingabe (x) */
  cout << endl << "x: " << endl;
  for (j=0; j<n; ++j) cin >> x[j];

  /* Matrixvektormultiplikation */
  matvekmult(a,x,b,m,n);

  /* Vektorausgabe (b) */
  cout << endl << "b: " << endl;
  for (i=0; i<m; ++i) cout << fixed << setprecision(2) << setw(6) << b[i] << endl;

  return 0;
}
```

- Java

```
import java.util.*;

public class Matvek4 {
  public static void matvekmult(double a[][], double x[], double b[])
  {
    int i,j,m,n;
    m = b.length;
    n = x.length;
    for (i=0; i<m; ++i) {
      b[i] = 0;
      for (j=0; j<n; ++j)
        b[i] += a[i][j]*x[j];
    }
  }

  public static void main(String[] args)
  {
    int m,n,i,j;
    Scanner scanner = new Scanner(System.in);
    System.out.print("m n: ");
    m = scanner.nextInt();
    n = scanner.nextInt();

    double x[] = new double[n];
    double b[] = new double[m];

    double a[][] = new double[m][];
    for (i=0; i<m; ++i) a[i] = new double[n];
//  Kurzform der letzten 2 Zeilen:
//  double a[][] = new double[m][n];

    /* Matrixeingabe (a) */
    System.out.println("a: ");
    for (i=0; i<m; ++i)
      for (j=0; j<n; ++j)
        a[i][j]=scanner.nextDouble();

    /* Vektoreingabe (x) */
    System.out.println("x: ");
    for (j=0; j<n; ++j)
      x[j] = scanner.nextDouble();

    /* Matrixvektormultiplikation */
    matvekmult(a,x,b);

    /* Vektorausgabe (b) */
    System.out.println("b: ");
    for (i=0; i<m; ++i)
      System.out.printf("%6.2f%n",b[i]);
  }
}
```