**수치해석 레포트**

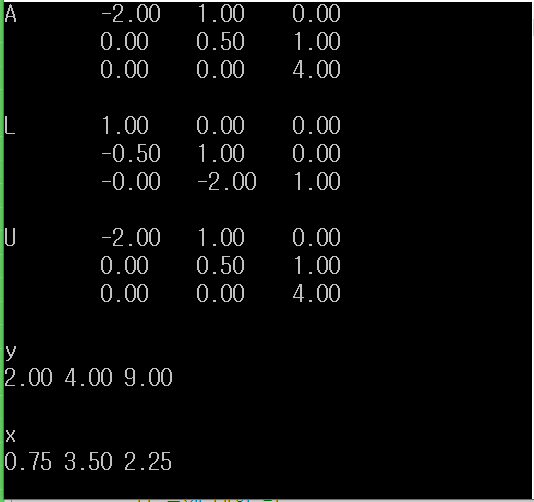
상하행렬 분해 선형방정식 풀이

소프트웨어공학과

2014156005 김동겸

1. 상하행렬분해 순수 가우스 소거법 사용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155 | /\*  수치해석 과제1   상하행렬분해 가우스소거법을 사용하여   Ax = c의 변수벡터 x를 구하는 프로그램   상하행렬분해     LUx = c,     Ly = c,     Ux = y  \*/    #include <stdio.h>  #include <Windows.h>    // 행렬 size  #define size 3    int main()  {      // 디폴트 행렬      double A[size][size] = {          {-2, 1, 0},          {1, 0, 1},          {0, -1, 2}      };      double L[size][size];      double U[size][size];        // 벡터 변수      double x[size];      double y[size];      double c[size] = { 2, 3, 1 };        // 디폴트 값        int i, j, k;        // 초기화      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              if (i == j) {                  // 주대각선에서 L은 1로, U는 A원소로 초기화                  L[i][j] = 1;                  U[i][j] = A[i][j];              }              else if (j > i) {                  L[i][j] = 0;                  U[i][j] = A[i][j];              }              else {                  // 모두 0으로 초기화                  L[i][j] = 0;                  U[i][j] = 0;              }          }            // 변수벡터 초기화          x[i] = 0;          y[i] = 0;      }        // L 행렬, U행렬 계산      for (k = 0; k < size - 1; k++) {    // size가 3이라면, 2단계만 진행                                          // 왜냐하면, 마지막 열의 승수는 구하지 않기 때문          // 피봇 원소가 0이라면 순수 가우스 소거법을          // 수행할 수 없으므로, 프로그램을 종료한다.          if (A[k][k] == 0) {              printf("%.2lf행 %.2lf열 피봇 원소가 0이므로, \                          순수 가우스 소거법을 적용할 수 없습니다.\n", i + 1, j + 1);              system("pause");              return 0;          }          // 행에 대한 반복          for (i = k; i < size; i++) {              if (i == k) continue;    // 피봇 행은 승수를 구하지 않음                // 승수 계산, L행렬 대입              L[i][k] = A[i][k] / A[k][k];                // 열에 대한 반복              for (j = k; j < size; j++) {                  // 피봇 행으로 다른 행 빼기                  A[i][j] = A[i][j] - (L[i][k] \* A[k][j]);                    // U행렬 대입                  if (j >= i)                      U[i][j] = A[i][j];                }    // 열 반복 종료            }    // 행 반복 종료        }    // 가우스 소거법 종료        // 출력      printf("A");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", A[i][j]);          }          printf("\n");      }      printf("\n");        // 출력      printf("L");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", L[i][j]);          }          printf("\n");      }      printf("\n");        // 출력      printf("U");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", U[i][j]);          }          printf("\n");      }      printf("\n");        // Ly = c, y벡터 해 구하기      for (i = 0; i < size; i++) {          for (j = 0; j < i; j++) {              c[i] -= (y[j] \* L[i][j]);        // !! c값 수정됨          }          j = i;          y[i] = c[i] / L[i][j];      }        // 출력      printf("y\n");      for (i = 0; i < size; i++) {          printf("%.2lf ", y[i]);      }      printf("\n\n");        // Ux = y, x벡터 해 구하기      for (i = size - 1; i >= 0;i--){          for (j = size - 1; j > i; j--) {              y[i] -= (x[j] \* U[i][j]);        // !! y값 수정됨          }          j = i;          x[i] = y[i] / U[i][j];      }        // 출력      printf("x\n");      for (i = 0; i < size; i++) {          printf("%.2lf ", x[i]);      }      printf("\n\n");  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |



1. 상하행렬 분해, 부분 피봇팅 가우스 소거법 사용

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200 | /\*  수치해석 과제1  상하행렬분해 부분 피봇팅 가우스소거법을 사용하여  PAx = Pc = d의 변수벡터 x를 구하는 프로그램  상하행렬분해  LUx = c,  Ly = c,  Ux = y  \*/    #include <stdio.h>  #include <Windows.h>    // 행렬 size  #define size 3    void Swap\_double(double \*px, double \*py)  {      double temp = \*px;        \*px = \*py;      \*py = temp;  }  double AbsoluteValue(double a) {      return (a >= 0 ? a : -a);  }    int main()  {      // 디폴트 행렬      double A[size][size] = {          { -1, 1, 2 },          { 1, 2, -1 },          { -2, -2, 2 }      };      // 치환 행렬      double P[size][size];      double L[size][size];      double U[size][size];        // 벡터 변수      double x[size];      double y[size];      double d[size];      double c[size] = { 7, 2, 0 };        // 디폴트 값        int i, j, k;      int pivot\_row;        // 초기화      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              if (i == j) {                  // 주대각선에서 L은 1로, P는 1로 초기화                  L[i][j] = 1;                  P[i][j] = 1;              }              else if (j > i) {                  // 상 행렬을 A원소로 초기화, P는 0                  L[i][j] = 0;                  P[i][j] = 0;              }              else {                  // 모두 0으로 초기화                  L[i][j] = 0;                  P[i][j] = 0;              }          }    // y 반복문 종료            // 변수벡터 초기화          x[i] = 0;          y[i] = 0;          d[i] = 0;        }    // 초기화 종료        // L 행렬, U행렬 계산, 부분 피봇팅 추가      for (k = 0; k < size - 1; k++) {    // size가 3이라면, 2단계만 진행                                          // 왜냐하면, 마지막 열의 승수는 구하지 않기 때문            // 절댓값이 가장 큰 행과 현재 행의 치환 행렬 P를 구하고,          // 두 행을 자리바꿈          pivot\_row = k;          for (i = k; i < size; i++) {              if (AbsoluteValue(A[pivot\_row][k]) < AbsoluteValue(A[i][k]))                  pivot\_row = i;          }            if (k != pivot\_row) {              // 치환 행렬 계산              for (j = 0; j < size; j++) {                  Swap\_double(&P[k][j], &P[pivot\_row][j]);              }                // A 배열 행 자리바꿈              for (j = 0; j < size; j++) {                  Swap\_double(&A[k][j], &A[pivot\_row][j]);              }              // L행렬 피봇팅              for (j = 0; j < k; j++) {                  Swap\_double(&L[k][j], &L[pivot\_row][j]);              }          }            // 행에 대한 반복          for (i = k; i < size; i++) {              if (i == k) continue;    // 피봇 행은 승수를 구하지 않음                                        // 승수 계산, L행렬 대입              L[i][k] = A[i][k] / A[k][k];                // 열에 대한 반복              for (j = k; j < size; j++) {                  // 피봇 행으로 다른 행 빼기                  A[i][j] = A[i][j] - (L[i][k] \* A[k][j]);                }    // 열 반복 종료            }    // 행 반복 종료            // U행렬 초기화          for (i = 0; i < size; i++) {              for (j = 0; j < size; j++) {                  U[i][j] = A[i][j];              }          }        }    // 가우스 소거법 종료        // d벡터 구하기,  d = Pc      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              d[i] += (P[i][j] \* c[j]);          }      }        // 출력      printf("A");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", A[i][j]);          }          printf("\n");      }      printf("\n");        // 출력      printf("L");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", L[i][j]);          }          printf("\n");      }      printf("\n");        // 출력      printf("U");      for (i = 0; i < size; i++) {          for (j = 0; j < size; j++) {              printf("    %.2lf", U[i][j]);          }          printf("\n");      }      printf("\n");        // Ly = d, y벡터 해 구하기      for (i = 0; i < size; i++) {          for (j = 0; j < i; j++) {              d[i] -= (y[j] \* L[i][j]);          }          j = i;          y[i] = d[i] / L[i][j];      }        // 출력      printf("y\n");      for (i = 0; i < size; i++) {          printf("%.2lf ", y[i]);      }      printf("\n\n");        // Ux = y, x벡터 해 구하기      for (i = size - 1; i >= 0; i--) {          for (j = size - 1; j > i; j--) {              y[i] -= (x[j] \* U[i][j]);          }          j = i;          x[i] = y[i] / U[i][j];      }        // 출력      printf("x\n");      for (i = 0; i < size; i++) {          printf("%.2lf ", x[i]);      }      printf("\n\n");  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

