



```
1: #include <iostream>
2: #include <sstream>
3: #include <vector>
4: #include <lapacke.h>
5: using namespace std;
6:
7: int main()
8: {
9:     int n = 10;
10:    unsigned int nodes = n+1;
11:    int rhs = 1;
12:    double a = 1.0;
13:    double alpha = 2.0;
14:    double gB = 1.0;
15:
16:    double d = n + a/3.0/n;
17:    double s = - n + a / 6.0 / n;
18:
19:    cout << d << endl;
20:    cout << s << endl;
21:
22:    //Lapacke overwrites upper/lower diagonal so store twice
23:    vector<double> upperdiagonal(n,s);
24:    vector<double> lowerdiagonal(n,s);
25:    vector<double> diagonal(nodes,2*d);
26:    vector<double> F(nodes,1.0/n);
27:
28:    diagonal[0]=1.0;
29:    diagonal[n]=d+alpha;
30:
31:    upperdiagonal[0] = 0.0;
32:    //
33:
34:    F[0] = 0;
35:    F[n]= 1.0/n/2.0 +alpha*gB;
36:
37:
38:    //Tridiagonal
39:    LAPACKE_dgtsv(LAPACK_COL_MAJOR,nodes,rhs,lowerdiagonal.data(),diagonal.data(),up
perdiagonal.data(),F.data(),nodes);
40:
41:
42:    cout << "Solution u_h at nodes x_1..x_" << n << ":\n";
43:    for (unsigned int i = 0; i < nodes; ++i) {
44:
45:        double xi = double(i) /double(n);
46:        cout << "u_h(" << xi << ") = " << F[i] << "\n";
47:        // F will be overwritten with sol
48:    }
49:
50:    return 0;
51:
52:
53: }
```



```
1:
2: #include <iostream>
3: #include <sstream>
4: #include <vector>
5: #include <cmath>
6: #include <lapacke.h>
7: using namespace std;
8:
9: int n = 10;
10: double threshold = 1.0/sqrt(2);
11: double lambda(double x)
12: {
13:     if(x < threshold)
14:     {
15:         return 1;
16:     }
17:     return 10;
18: }
19:
20: double fillOffDiagonal(unsigned int i)
21: {
22:
23:     double x_l = i/double(n);
24:     double x_u = (i+1)/double(n);
25:
26:     if(x_l < threshold && x_u > threshold)
27:     {
28:         return -n*n*(threshold-x_l+10*(x_u-threshold));
29:     }
30:     return -n*lambda(x_l);
31: }
32:
33: double fillDiagonal(int i)
34: {
35:     double x_l = (i-1)/double(n);
36:     double x_u = (i+1)/double(n);
37:     if(x_l < threshold && x_u > threshold)
38:     {
39:         return n*n*(threshold-x_l+10*(x_u-threshold));
40:     }
41:     return 2*n*lambda(x_l);
42: }
43:
44: int main()
45: {
46:     //Ignored the first Row of K cause we now that u(0)=0
47:
48:     int rhs = 1;
49:     unsigned int nodes = n+1;
50:
51:
52:     //Lapacke overwrites upper/lower diagonal so store twice
53:     vector<double> upperdiagonal(n,0.0);
54:     vector<double> lowerdiagonal(n,0.0);
55:     vector<double> diagonal(nodes,1.0);
56:     vector<double> F(nodes,0.0);
57:
58:     for(int i=0; i < n; i++)
59:     {
60:         upperdiagonal[i] = fillOffDiagonal(i);
61:         lowerdiagonal[i] = fillOffDiagonal(i);
62:         diagonal[i+1] = fillDiagonal(i+1);
63:     }
64:
65:     upperdiagonal[0]=0.0;
66:     lowerdiagonal[n-1]=0.0;
67:     diagonal[n] = 1.0;
68:     F[n] = 1.0;
```

keine globalen Variablen benutzen!

```
69:
70:
71:     //Tridiagonal
72:     LAPACKE_dgtsv(LAPACK_COL_MAJOR,nodes,rhs,lowerdiagonal.data(),diagonal.data(),up
perdiagonal.data(),F.data(),nodes);
73:
74:
75:     cout << "Solution u_h at nodes x_0..x_" << n << ":\n";
76:
77:     for (unsigned int i = 0; i < nodes; ++i) {
78:
79:         double xi = double(i) /double(n);
80:         cout << "u_h(" << xi << ") = " << F[i] << "\n";
81:         // F will be overwritten with sol
82:     }
83:     return 0;
84:
85:
86: }
```



```

1:
2: #include <iostream>
3: #include <sstream>
4: #include <vector>
5: #include <cmath>
6: #include <lapacke.h>
7: using namespace std;
8:
9:
10: int main()
11: { const
12:     unsigned int ns[5] = {10,20,30,40,70};
13:     int p = 70;
14:     int rhs = 1;
15:     for(unsigned int i = 0; i < sizeof(ns)/sizeof(ns[0]); i++)
16:     {
17:         int n=ns[i];
18:         unsigned int nodes = n+1;
19:
20:         vector<double> upperdiagonal(n,-n+p/2.0);
21:         vector<double> lowerdiagonal(n,-n-p/2.0);
22:         vector<double> diagonal(nodes,2*n);
23:         vector<double> F(nodes,0.0);
24:         upperdiagonal[0]=0.0;
25:         lowerdiagonal[n-1]=0.0;
26:         diagonal[0] = 1.0;
27:         diagonal[n] = 1.0;
28:         F[n] = 1.0;
29:
30:         //Tridiagonal
31:         LAPACKE_dgtsv(LAPACK_COL_MAJOR,nodes,rhs,lowerdiagonal.data(),diagonal.data(
),upperdiagonal.data(),F.data(),nodes);
32:
33:
34:         cout << "Solution u_h at nodes x_0..x_" << n << ":\n";
35:
36:         for (unsigned int j = 0; j < nodes; ++j) {
37:
38:             double xi = double(j) /double(n);
39:             cout << "u_h(" << xi << ") = " << F[j] << "\n";
40:             // F will be overwritten with sol
41:         }
42:     }
43:
44:     /*
45:     From Methode der finiten Elemente fuer Ingenieure:
46:     small n approximates pu' not good espicially if n < p/2=35 therefor this oscilla
tion (see plot) occurs.
47:     */
48:     return 0;
49:
50:
51: }

```

*besser: std::array*

*for (auto n : ns)*

*✓*

*✓*